LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

ViewModel and Debugging

Oleh:

Muhammad Adh-Dhiya'Us Salim NIM. 2310817210022

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE
MODUL 4

Laporan Praktikum Pemrograman Mobile Modul 4: ViewModel and Debugging 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

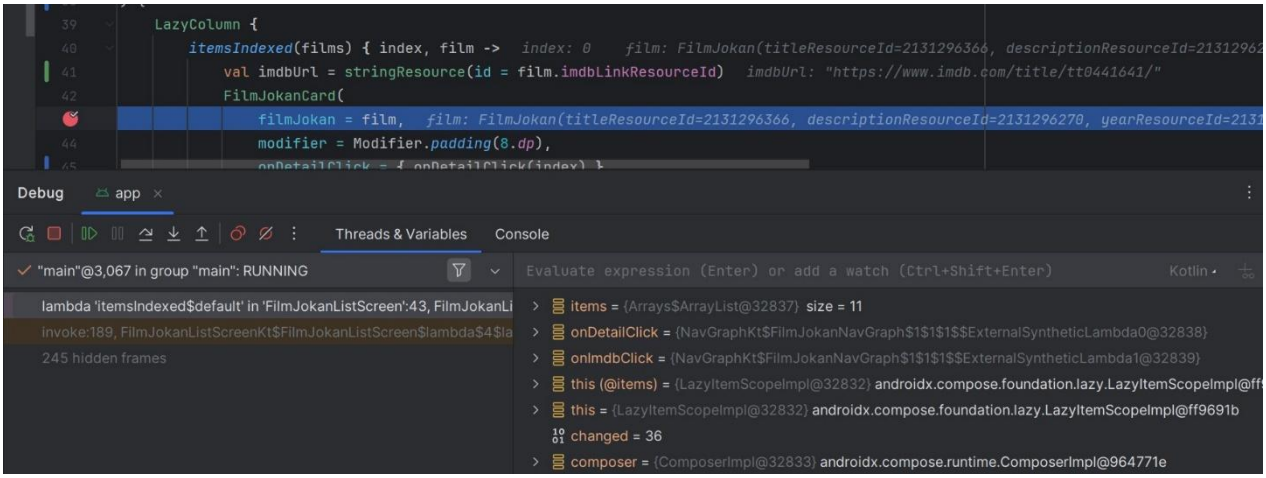
DAFTAR TABEL

SOAL 1

Soal Praktikum:

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

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



Gambar 1 Contoh Penggunaan Debugger

A. Source Code

Data/MovieRepository/ Compose

```
1  package com.example.modul4.data
2
3  import com.example.modul4.R
4  import com.example.modul4.model.Movie
5  import timber.log.Timber
6
7  object MovieRepository {
8      fun getMovies(): List<Movie> {
9          val movies = listOf(
10             Movie(
11                 title = "The Uncanny Counter",
12                 year = "2020",
13                 plot = "Para Counter yang karyawan toko mi di siang
14 hari dan pemburu iblis di malam hari, memakai berbagai kemampuan
15 khusus untuk memburu roh-roh jahat yang mengincar manusia.",
16                 imdb = "https://www.imdb.com/title/tt13273826/",
17                 posterResId = R.drawable.dukun
18             ),
19             Movie(
20                 title = "Sweet Home",
21                 year = "2020",
22                 plot = "Saat banyak manusia berubah menjadi monster
23 ganas dan dunia terpuruk ke dalam teror, sekelompok penyintas berjuang
24 untuk hidup—dan mempertahankan kemanusiaan mereka..",
25                 imdb =
26 "https://www.imdb.com/title/tt11612120/?ref_=nv_sr_srsg_0_tt_8_nm_0_
27 in_0_q_Sweet%2520Home",
28                 posterResId = R.drawable.rm
29             ),
30             Movie(
31                 title = "Money Heist Korea",
32                 year = "2022",
33                 plot = "Pencuri menguasai gedung percetakan uang
34 milik Korea bersatu dan menawan banyak sandera. Polisi harus
35 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",
                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
            )
        )
        Timber.d("Data  film  berhasil  dimuat.  Total:  ${movies.size}
film.")
        return                                                  movies
    }

}
```

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

```
package com.example.scrollablelist_modul3.ui

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
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.modul4.ui.screens
2
3   import                                        android.content.Intent
4   import                                               android.net.Uri
5   import                              androidx.compose.foundation.Image
6   import                            androidx.compose.foundation.layout.*
7   import                    androidx.compose.foundation.lazy.LazyColumn
8   import                         androidx.compose.foundation.lazy.items
9   import      androidx.compose.foundation.shape.RoundedCornerShape
    import                                    androidx.compose.material3.*
    import                                      androidx.compose.runtime.*
    import                                    androidx.compose.ui.Modifier
    import                                   androidx.compose.ui.draw.clip
    import                    androidx.compose.ui.platform.LocalContext
    import                      androidx.compose.ui.res.painterResource
    import                                    androidx.compose.ui.unit.dp
    import                               androidx.navigation.NavController
    import                           com.example.modul4.navigation.Screen
    import                  com.example.modul4.viewmodel.MovieViewModel
    import          com.example.modul4.viewmodel.SharedMovieViewModel
    import                                               timber.log.Timber

    @OptIn(ExperimentalMaterial3Api::class)
    @Composable
    fun                                                 MovieListScreen(
        navController:                                    NavController,
        movieViewModel:                                  MovieViewModel,
        sharedViewModel:                               SharedMovieViewModel
    )                                                                    {
        val          context          =          LocalContext.current
        val   movieList   by   movieViewModel.movies.collectAsState()

        Scaffold(
            topBar                              =                        {
                TopAppBar(title   =   {   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)) {
                        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))

                        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         =         {
                                            Timber.d("Tombol
Explicit  Intent  (IMDB)  ditekan  untuk  film:  ${movie.title}")
                                            val      intent      =
Intent(Intent.ACTION_VIEW,                Uri.parse(movie.imdb))

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

                                    Button(
                                        onClick         =         {
                                            Timber.d("Tombol  Detail
ditekan.")
                                            Timber.i("Data     yang
dipilih       untuk       detail:       Judul=${movie.title}")


movieViewModel.onMovieClick(movie)

sharedViewModel.selectMovie(movie)

navController.navigate(Screen.Detail.route)
                                        }
                                    )                                    {
                                        Text("Detail")
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Tabel 6 Source Code Soal 1 - Compose

Viewmodel / MovieViewModel / Compose

```
1  package                              com.example.modul4.viewmodel
2
3  import                               androidx.lifecycle.ViewModel
4  import                    com.example.modul4.data.MovieRepository
5  import                  kotlinx.coroutines.flow.MutableStateFlow
6  import                         kotlinx.coroutines.flow.StateFlow
7  import                            com.example.modul4.model.Movie
8  import                                        timber.log.Timber
9
   class MovieViewModel(private val userId: String) : ViewModel()
```

```
{
    private val _movies = MutableStateFlow<List<Movie>>(emptyList())
    val movies: StateFlow<List<Movie>> get() = _movies

    private val _selectedMovie = MutableStateFlow<Movie?>(null)
    val selectedMovie: StateFlow<Movie?> get() = _selectedMovie

    init {
        val movieList = MovieRepository.getMovies()
        _movies.value = movieList
        Timber.d("User $userId - Loaded movie list with ${movieList.size} items")
    }

    fun onMovieClick(movie: Movie) {
        _selectedMovie.value = movie
        Timber.d("User $userId - Movie clicked: ${movie.title}")
    }
}
```

Tabel 7 Source Code Soal 1 – Compose

Viewmodel / MovieViewModelFactory / Compose

```
1  package com.example.modul4.viewmodel
2
3  import androidx.lifecycle.ViewModel
4  import androidx.lifecycle.ViewModelProvider
5
6  class MovieViewModelFactory(private val userId: String) :
7  ViewModelProvider.Factory {
8      @Suppress("UNCHECKED_CAST")
9      override fun <T : ViewModel> create(modelClass: Class<T>):
   T {
          if
   (modelClass.isAssignableFrom(MovieViewModel::class.java)) {
              return MovieViewModel(userId) as T
          }
          throw IllegalArgumentException("Unknown ViewModel
   class")
      }
  }
```

Tabel 8 Source Code Soal 1 - Compose

Viewmodel / SharedViewModel / Compose

```
1  package com.example.modul4.viewmodel
2
3  import androidx.compose.runtime.getValue
4  import androidx.compose.runtime.mutableStateOf
5  import androidx.compose.runtime.setValue
6  import androidx.lifecycle.ViewModel
7  import com.example.modul4.model.Movie
8
9  class SharedMovieViewModel : ViewModel() {
       var selectedMovie by mutableStateOf<Movie?>(null)
           private set
```

```
    fun selectMovie(movie: Movie) {
        selectedMovie = movie
    }
}
```

Tabel 9 Source Code Soal 1 - Compose

MainActivity / Compose

```
package com.example.scrollablelist_modul3

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import com.example.scrollablelist_modul3.navigation.AppNavigation
import
com.example.scrollablelist_modul3.ui.theme.ScrollableListModul3Theme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ScrollableListModul3Theme {
                AppNavigation()
            }
        }
    }
}
```

Tabel 10 Source Code Soal 1 - Compose

MyApplication.kt / Compose

```
package com.example.modul4

import android.app.Application
import com.example.modul4.BuildConfig
import timber.log.Timber

class MyApplication : Application() {
    override fun onCreate() {
        super.onCreate()
        if (BuildConfig.DEBUG) {
            Timber.plant(Timber.DebugTree())
        }
    }
}
```

Tabel 11 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 12 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 13 Source Code Soal 1 - XML

Ui.theme / DetailFragment / XML

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

import                                      android.content.Intent
import                                             android.net.Uri
import                                          android.os.Bundle
import                                          android.view.View
import                                androidx.fragment.app.Fragment
import                      androidx.fragment.app.activityViewModels
import                          androidx.navigation.fragment.navArgs
import                                              com.example.m3xml.R
import                                        com.example.m3xml.data.Film
import       com.example.m3xml.databinding.FragmentDetailBinding
import                    com.example.m3xml.viewmodel.FilmViewModel
import                com.example.m3xml.viewmodel.ViewModelFactory
import                                          timber.log.Timber

class  DetailFragment  :  Fragment(R.layout.fragment_detail)  {

```

```
19      private var _binding: FragmentDetailBinding? = null
20      private val binding get() = _binding!!
21      private val viewModel: FilmViewModel by activityViewModels
22  {
23          ViewModelFactory("Ini dari DetailFragment")
24      }
25      private val args: DetailFragmentArgs by navArgs()
26
27      override fun onViewCreated(view: View, savedInstanceState:
28  Bundle?) {
29          super.onViewCreated(view, savedInstanceState)
30          _binding = FragmentDetailBinding.bind(view)
31
32          val filmId = args.filmId
33
34          val film: Film? = viewModel.getFilmById(filmId)
35
36          if (film != null) {
37              Timber.d("Membuka halaman detail untuk film:
38  ID=${film.id}, Nama='${film.title}'")
39
40  binding.ivDetailPoster.setImageResource(film.poster)
41              binding.tvDetailTitle.text = "${film.title}
42  (${film.year})"
43              binding.tvDetailPlot.text = film.plot
44              binding.ivDetailPoster.setOnClickListener {
45                  Timber.d("Poster '${film.title}' diklik,
46  membuka URL IMDb.")
47                  val openUrlIntent = Intent(Intent.ACTION_VIEW,
48  Uri.parse(film.imdbUrl))
49                  startActivity(openUrlIntent)
              }
          } else {
              Timber.e("Film dengan ID $filmId tidak ditemukan.")
          }
      }

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

Tabel 14 Source Code Soal 1 - XML

Ui.theme / FilmAdapter / XML

```
1   package com.example.m3xml.ui.theme
2
3   import android.view.LayoutInflater
4   import android.view.ViewGroup
5   import androidx.recyclerview.widget.DiffUtil
6   import androidx.recyclerview.widget.ListAdapter
7   import androidx.recyclerview.widget.RecyclerView
8   import com.example.m3xml.data.Film
9   import com.example.m3xml.databinding.ListItemBinding
```

```
10
11  class                                                FilmAdapter(
12      private     val     onDetailClick:     (Film)     ->     Unit,
13      private      val      onImdbClick:      (Film)      ->      Unit
14  )                        :                          ListAdapter<Film,
15  FilmAdapter.FilmViewHolder>(FilmDiffCallback())                {
16
17      override fun onCreateViewHolder(parent: ViewGroup, viewType:
18  Int):                        FilmViewHolder                        {
19          val                          binding                         =
20  ListItemBinding.inflate(LayoutInflater.from(parent.context),
21  parent,                                                    false)
22          return                               FilmViewHolder(binding)
23      }
24
25      override    fun    onBindViewHolder(holder:    FilmViewHolder,
26  position:                             Int)                            {
27          val           film           =           getItem(position)
28          holder.bind(film)
29      }
30
31      inner     class     FilmViewHolder(private     val     binding:
32  ListItemBinding)                                                    :
33          RecyclerView.ViewHolder(binding.root)                       {
34
35          fun              bind(film:              Film)              {
36              binding.ivItemPoster.setImageResource(film.poster)
37              binding.tvItemTitle.text         =         film.title
38
39              binding.buttonItemDetail.setOnClickListener        {
40                  onDetailClick(film)
41              }
42
43              binding.buttonItemImdb.setOnClickListener          {
44                  onImdbClick(film)
45              }
46          }
47      }
48  }
49
    class    FilmDiffCallback    :    DiffUtil.ItemCallback<Film>()    {
        override fun areItemsTheSame(oldItem: Film, newItem: Film):
    Boolean                                                            {
            return          oldItem.id          ==          newItem.id
        }

        override   fun   areContentsTheSame(oldItem:   Film,   newItem:
    Film):                              Boolean                        {
            return            oldItem            ==            newItem
        }
    }
```

Tabel 15 Source Code Soal 1 - XML

Ui.theme / ListFragment / XML

```
1  package com.example.m3xml.ui.theme
2
3  import android.content.Intent
4  import android.net.Uri
5  import android.os.Bundle
6  import android.view.View
7  import androidx.fragment.app.Fragment
8  import androidx.fragment.app.activityViewModels
9  import androidx.lifecycle.Lifecycle
10 import androidx.lifecycle.lifecycleScope
11 import androidx.lifecycle.repeatOnLifecycle
12 import androidx.navigation.fragment.findNavController
13 import androidx.recyclerview.widget.LinearLayoutManager
14 import com.example.m3xml.R
15 import com.example.m3xml.databinding.FragmentListBinding
16 import com.example.m3xml.viewmodel.FilmViewModel
17 import com.example.m3xml.viewmodel.ViewModelFactory
18 import kotlinx.coroutines.launch
19 import timber.log.Timber
20
21 class ListFragment : Fragment(R.layout.fragment_list) {
22
23     private var _binding: FragmentListBinding? = null
24     private val binding get() = _binding!!
25
26     private val viewModel: FilmViewModel by activityViewModels {
27         ViewModelFactory("Ini dari ListFragment")
28     }
29
30     override fun onViewCreated(view: View, savedInstanceState:
31 Bundle?) {
32         super.onViewCreated(view, savedInstanceState)
33         _binding = FragmentListBinding.bind(view)
34
35         val filmAdapter = FilmAdapter(
36             onDetailClick = { film ->
37                 Timber.d("Tombol detail untuk film
38 '${film.title}' ditekan.")
39                 viewModel.onFilmClicked(film)
40             },
41             onImdbClick = { film ->
42                 Timber.d("Tombol IMDb untuk film '${film.title}'
43 ditekan.")
44                 val openUrlIntent = Intent(Intent.ACTION_VIEW,
45 Uri.parse(film.imdbUrl))
46                 startActivity(openUrlIntent)
47             }
48         )
49
         binding.recyclerView.apply {
             layoutManager = LinearLayoutManager(context)
             adapter = filmAdapter
         }
```

```
        viewLifecycleOwner.lifecycleScope.launch {
            repeatOnLifecycle(Lifecycle.State.STARTED) {
                launch {
                    viewModel.filmList.collect { filmList ->
                        filmAdapter.submitList(filmList)
                    }
                }
                launch {
                    viewModel.navigateToDetail.collect { film ->
                        film?.let {
                            val action = ListFragmentDirections.actionListFragmentToDetailFragment(it.id)
                            findNavController().navigate(action)

viewModel.onNavigateToDetailComplete()
                        }
                    }
                }
            }
        }
    }

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

Tabel 16 Source Code Soal 1 - XML

Viewmodel / FilmViewModel / XML

```
1  package com.example.m3xml.viewmodel
2
3  import androidx.lifecycle.ViewModel
4  import com.example.m3xml.data.DataSource
5  import com.example.m3xml.data.Film
6  import kotlinx.coroutines.flow.MutableStateFlow
7  import kotlinx.coroutines.flow.StateFlow
8  import kotlinx.coroutines.flow.asStateFlow
9  import kotlinx.coroutines.flow.update
10 import timber.log.Timber
11
12 class FilmViewModel(private val someString: String) :
13 ViewModel() {
14     private val _filmList =
15 MutableStateFlow<List<Film>>(emptyList())
16     val filmList: StateFlow<List<Film>> = _filmList
17
18     private val _navigateToDetail =
19 MutableStateFlow<Film?>(null)
20     val navigateToDetail: StateFlow<Film?> =
    _navigateToDetail.asStateFlow()

    init {
        Timber.i("FilmViewModel created with string:
    $someString")
```

```
        loadFilms()
    }

    private fun loadFilms() {
        _filmList.value = DataSource.getFilms()
        Timber.i("Data film berhasil dimuat. Jumlah data:
${_filmList.value.size}")
    }
    fun getFilmById(id: Int): Film? {
        return _filmList.value.find { it.id == id }
    }

    fun onFilmClicked(film: Film) {
        _navigateToDetail.update { film }
    }

    fun onNavigateToDetailComplete() {
        _navigateToDetail.update { null }
    }
}
```

Tabel 17 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 18 Source Code Soal 1 – XML

MyApllication.kt / XML

```
1  package com.example.m3xml
2
3  import android.app.Application
4  import timber.log.Timber
5
6  class MyApplication : Application() {
7
8      override fun onCreate() {
9          super.onCreate()
10         if (BuildConfig.DEBUG) {
```

```
            Timber.plant(Timber.DebugTree())
        }
    }
}
```

Tabel 19 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 20 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 21 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"
```

```
2           app:layout_constraintBottom_toBottomOf="parent"
2           app:layout_constraintEnd_toEndOf="parent"
2           app:layout_constraintStart_toStartOf="parent"
3           app:layout_constraintTop_toBottomOf="@id/tv_header_title"
2           tools:listitem="@layout/list_item"                          />
4
2   </androidx.constraintlayout.widget.ConstraintLayout>
5
```

Tabel 22 Source Code Soal 1 - XML

Layout / list_item.xml / XML

```
1   <?xml              version="1.0"              encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="wrap_content"
8       android:layout_marginStart="8dp"
9       android:layout_marginTop="4dp"
10      android:layout_marginEnd="8dp"
11      android:layout_marginBottom="4dp"
12      app:cardCornerRadius="8dp"
13      app:cardElevation="2dp">
14
15      <androidx.constraintlayout.widget.ConstraintLayout
16          android:layout_width="match_parent"
17          android:layout_height="wrap_content"
18          android:paddingBottom="8dp">
19
20          <ImageView
21              android:id="@+id/iv_item_poster"
22              android:layout_width="100dp"
23              android:layout_height="150dp"
24              android:scaleType="centerCrop"
25              app:layout_constraintBottom_toBottomOf="parent"
26              app:layout_constraintStart_toStartOf="parent"
27              app:layout_constraintTop_toTopOf="parent"
28              tools:srcCompat="@tools:sample/avatars"          />
29
30          <TextView
31              android:id="@+id/tv_item_title"
32              android:layout_width="0dp"
33              android:layout_height="wrap_content"
34              android:layout_marginStart="16dp"
35              android:layout_marginTop="8dp"
36              android:layout_marginEnd="16dp"
37
38  android:textAppearance="?attr/textAppearanceTitleMedium"
39              android:textStyle="bold"
40              app:layout_constraintEnd_toEndOf="parent"
41
42  app:layout_constraintStart_toEndOf="@id/iv_item_poster"
43              app:layout_constraintTop_toTopOf="parent"
```

```
            tools:text="Judul Film yang Sangat Panjang Sekali"
/>

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:orientation="horizontal"

app:layout_constraintEnd_toEndOf="@id/tv_item_title"

app:layout_constraintStart_toStartOf="@id/tv_item_title"

app:layout_constraintTop_toBottomOf="@id/tv_item_title">

            <Button
                android:id="@+id/button_item_detail"
                style="?attr/materialButtonOutlinedStyle"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginEnd="4dp"
                android:layout_weight="1"
                android:text="Detail"                          />

            <Button
                android:id="@+id/button_item_imdb"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginStart="4dp"
                android:layout_weight="1"
                android:text="IMDb"                            />

        </LinearLayout>

    </androidx.constraintlayout.widget.ConstraintLayout>

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

Tabel 23 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"
```

```
13          tools:layout="@layout/fragment_list"                    >
14          <action
15
16  android:id="@+id/action_listFragment_to_detailFragment"
17              app:destination="@id/detailFragment"               />
18      </fragment>
19      <fragment
20          android:id="@+id/detailFragment"
21
22  android:name="com.example.m3xml.ui.theme.DetailFragment"
23          android:label="fragment_detail"
24          tools:layout="@layout/fragment_detail"                  >
25          <argument
26              android:name="filmId"
27              app:argType="integer"                              />
28      </fragment>
29  </navigation>
```

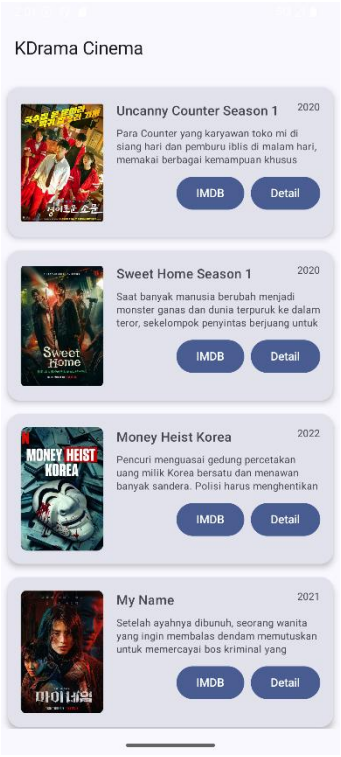
Tabel 24 Source Code Soal 1 - XML

Values/ themes.xml / XML

```
1   <resources      xmlns:tools="http://schemas.android.com/tools">
2       <style                                  name="Base.Theme.M3XML"
3   parent="Theme.Material3.DayNight.NoActionBar">
4       </style>
5
6       <style   name="Theme.M3XML"   parent="Base.Theme.M3XML"   />
7
8       <style                                  name="roundedImageView">
9           <item                name="cornerFamily">rounded</item>
10          <item                     name="cornerSize">16dp</item>
11      </style>
12
13  </resources>
```

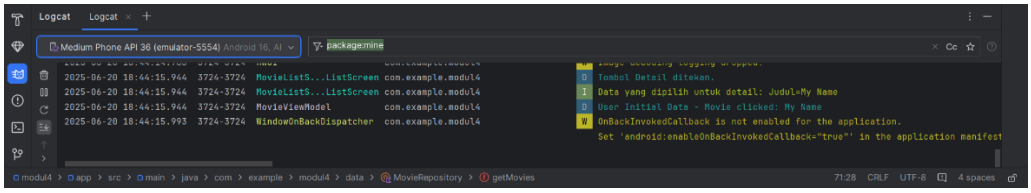
Tabel 25 Source Code Soal 1 - XML

B. Output Program

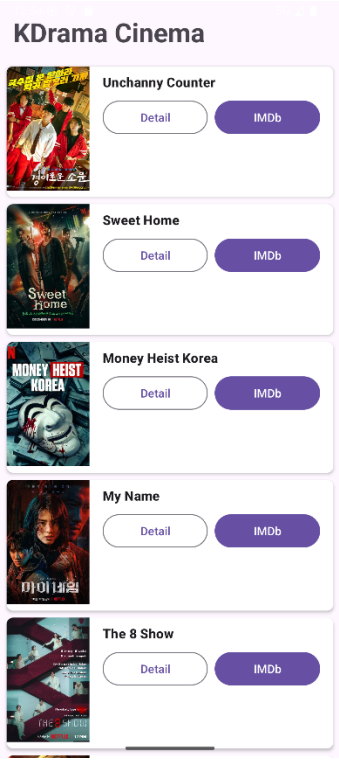


Gambar 2 Screenshot Hasil Jawaban Soal 1 - Compose

Gambar 3 Screenshot Hasil Jawaban Soal 1 - Compose

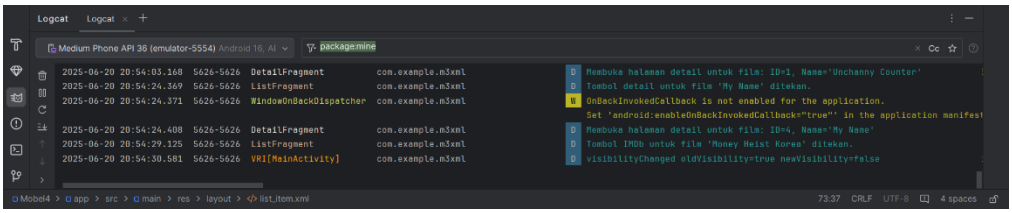


Gambar 4 Screenshot Penggunaan Timber Soal 1 - Compose



Gambar 5 Screenshot Hasil Jawaban Soal 1 - XML

Gambar 6 Screenshot Hasil Jawaban Soal 1 - XML



Gambar 7 Screenshot Penggunaan Timber 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 bertanggung jawab untuk menyimpan, mengelola, dan menyediakan data yang dibutuhkan oleh UI, serta bertahan dari perubahan konfigurasi seperti rotasi layar.

- Pada blok init, sebuah log Timber.i(...) ditempatkan untuk mencatat peristiwa pembuatan ViewModel, memenuhi syarat logging (poin d.a).
- Pada metode loadFilms, setelah data berhasil diambil dari DataSource, log Timber.i(...) kedua dicatat untuk menandakan bahwa data telah berhasil dimuat ke dalam list (poin d.a).
- Sebuah fungsi publik baru, getFilmById(id: Int), ditambahkan. Fungsi ini berperan penting untuk menyediakan data objek Film yang lengkap kepada DetailFragment hanya dengan menggunakan ID sebagai parameter pencarian.

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

Kelas ini berfungsi sebagai jembatan antara sumber data (list film) dan RecyclerView yang menampilkannya.

- Konstruktor kelas FilmAdapter dimodifikasi untuk menerima dua parameter fungsi (lambda), yaitu onDetailClick dan onImdbClick. Ini memungkinkan penanganan aksi klik yang berbeda untuk setiap tombol.
- Di dalam ViewHolder, pada metode bind, data dari objek Film diikat ke View yang sesuai menggunakan ID yang benar (iv_item_poster, tv_item_title).
- Sebuah setOnClickListener dipasang pada button_item_detail. Ketika diklik, listener ini akan memanggil fungsi onDetailClick yang telah diteruskan dari ListFragment.
- Secara serupa, setOnClickListener juga dipasang pada button_item_imdb yang akan memanggil fungsi onImdbClick.

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

Fragment ini bertindak sebagai controller untuk UI yang menampilkan daftar film.

- Saat membuat FilmAdapter, dua fungsi lambda didefinisikan dan diteruskan sebagai parameter.
- Lambda untuk onDetailClick berisi dua aksi: pertama, mencatat log peristiwa dengan Timber.d("Tombol detail ... ditekan.") (memenuhi syarat d.b); kedua, memanggil metode pada ViewModel untuk memicu event navigasi.
- Lambda untuk onImdbClick juga mencatat log dengan Timber.d("Tombol IMDb ... ditekan.") dan kemudian membuat sebuah Intent dengan ACTION_VIEW untuk membuka URL IMDb di browser eksternal.
- RecyclerView dikonfigurasi menggunakan LinearLayoutManager untuk memastikan item ditampilkan dalam format daftar vertikal sesuai desain awal.

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

Fragment ini bertindak sebagai controller untuk UI yang menampilkan rincian dari satu film yang dipilih.

- Pertama, Fragment mengambil filmid (sebuah Integer) dari argumen navigasi yang diterima.
- filmid tersebut kemudian digunakan untuk memanggil metode viewModel.getFilmById() guna mendapatkan objek Film yang utuh.
- Setelah objek Film berhasil didapatkan, sebuah log dicatat menggunakan Timber.d("Membuka halaman detail untuk film: ..."), yang memenuhi syarat logging (poin d.c).

- Selanjutnya, data dari objek Film (poster, judul, tahun, plot) ditampilkan ke komponen-komponen View yang sesuai.
- Terakhir, setOnClickListener dipasang pada button_share. Ketika diklik, listener ini akan mencatat log peristiwa dengan Timber.d("Tombol Explicit Intent (Share) ditekan ...") (memenuhi syarat d.b) dan kemudian membuat serta meluncurkan Intent dengan ACTION_SEND untuk berbagi informasi film.

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).

13. MyApplication.kt

Ini adalah implementasi dari kelas Application kustom yang berfungsi sebagai titik masuk tunggal untuk inisialisasi level aplikasi.

- Kelas MyApplication merupakan turunan dari kelas android.app.Application.
- Di dalam metode onCreate(), yang merupakan lifecycle callback pertama saat aplikasi dimulai, dilakukan pengecekan kondisi menggunakan if (BuildConfig.DEBUG). Variabel BuildConfig.DEBUG adalah flag yang secara otomatis bernilai true selama pengembangan (build variant "debug") dan false saat aplikasi dibuat untuk rilis.
- Di dalam blok kondisional tersebut, metode Timber.plant(Timber.DebugTree()) dipanggil. Perintah ini menginisialisasi Timber agar aktif dan mencetak log ke Logcat, namun hanya pada build "debug". Hal ini memastikan bahwa pada versi rilis, semua panggilan logging tidak akan dieksekusi, sehingga aplikasi menjadi lebih bersih dan aman.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/Adiyaus/Praktikum-PemrogramanMobile.git

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya
Aplikasi harus dapat mempertahankan fitur-fitur yang sudah dibuat pada modul sebelumnya. Berikut adalah contoh debugging dalam Android Studio.

A. Jawaban

Dalam arsitektur aplikasi Android, Application class adalah sebuah kelas dasar (base class) yang esensial dan berfungsi sebagai titik masuk utama serta *global singleton instance* untuk seluruh aplikasi. Sebuah instance dari kelas ini akan dibuat secara otomatis oleh sistem Android ketika proses aplikasi pertama kali dimulai. Instance ini akan tetap ada selama siklus hidup aplikasi (*application lifecycle*) berlangsung, bahkan ketika komponen lain seperti Activity dihancurkan dan dibuat ulang akibat perubahan konfigurasi atau navigasi pengguna.

Fungsi dan Peran Utama Application Class

Application class memiliki beberapa fungsi strategis dalam pengembangan aplikasi Android, yang utamanya berpusat pada manajemen data dan inisialisasi global.

1. **Manajemen State Global (Global State Management)** Fungsi primer dari Application class adalah untuk mengelola state atau data yang bersifat global dan perlu diakses oleh berbagai komponen aplikasi. Hal ini memungkinkan Activity, Service, atau BroadcastReceiver untuk mengakses data bersama tanpa perlu melewatkannya secara eksplisit melalui mekanisme seperti Intent. Dengan demikian, konsistensi data di seluruh aplikasi dapat terjaga. Contoh data global yang lazim disimpan di sini mencakup status autentikasi pengguna, konfigurasi aplikasi, atau instance tunggal dari objek utilitas seperti klien jaringan (misalnya, Retrofit) atau *database helper* (misalnya, Room).
2. **Inisialisasi Tunggal (One-Time Initialization)** Metode onCreate() dari Application class merupakan lokasi yang ideal untuk melakukan proses inisialisasi yang hanya perlu dijalankan satu kali selama siklus hidup aplikasi. Ini sangat efisien untuk menginisialisasi *Software Development Kits* (SDK) pihak ketiga, seperti library untuk analisis (contoh: Firebase Analytics), *dependency injection* (contoh: Hilt, Koin), atau konfigurasi *crash reporting*. Menempatkan logika ini di Application class memastikan bahwa komponen-komponen tersebut siap digunakan sejak awal dan tidak diinisialisasi berulang kali setiap kali Activity dibuat.
3. **Reaksi terhadap Peristiwa Tingkat Aplikasi** Kelas ini dapat digunakan untuk merespons peristiwa tingkat sistem yang memengaruhi seluruh aplikasi. Melalui *override* metode seperti onLowMemory(), aplikasi dapat secara terpusat mengelola sumber daya dengan membersihkan *cache* atau objek yang tidak lagi esensial ketika sistem kehabisan memori. Demikian pula, metode onConfigurationChanged() dapat digunakan untuk menangani perubahan konfigurasi global, seperti perubahan lokal atau bahasa perangkat.

Praktik yang Harus Dihindari

Meskipun sangat berguna, terdapat praktik penggunaan Application class yang harus dihindari. Sangat tidak disarankan untuk menyimpan referensi ke Context dari komponen dengan siklus hidup yang

lebih pendek, seperti Activity Context, di dalam Application class. Praktik ini dapat menyebabkan kebocoran memori (*memory leak*). Hal ini terjadi karena Application class yang memiliki siklus hidup panjang akan terus menahan referensi ke Activity, sehingga *Garbage Collector* tidak dapat membersihkan objek Activity tersebut dari memori meskipun sudah tidak lagi ditampilkan kepada pengguna.

Implementasi Konseptual

Untuk memanfaatkan fungsionalitas ini, pengembang perlu membuat sebuah kelas baru yang merupakan turunan (*subclass*) dari android.app.Application. Selanjutnya, kelas kustom tersebut harus dideklarasikan dalam file AndroidManifest.xml melalui atribut android:name pada tag <application>. Langkah ini menginstruksikan sistem Android untuk menggunakan kelas kustom tersebut sebagai objek aplikasi utama, bukan kelas Application bawaan.

Kesimpulan

Secara keseluruhan, Application class memegang peranan krusial dalam arsitektur aplikasi Android sebagai wadah terpusat untuk state global dan logika inisialisasi. Penggunaannya yang tepat dapat meningkatkan modularitas, menyederhanakan manajemen data antar komponen, dan memastikan konsistensi fitur di seluruh aplikasi, yang pada akhirnya menghasilkan arsitektur yang lebih solid dan mudah dikelola.