

**LAPORAN PRAKTIKUM  
PEMROGRAMAN MOBILE  
MODUL 4**



**VIEWMODEL AND DEBUGGING**

**Oleh:**

**Muhammad Firas                  NIM. 2210817110014**

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

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

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

Nama Praktikan : Muhammad Firas  
NIM : 2210817110014

Menyetujui,  
Asisten Praktikum

Mengetahui,  
Dosen Penanggung Jawab Praktikum

Muhammad Raka Azwar  
NIM. 2210817210012

Andreyan Rizky Baskara, S.Kom., M.Kom.  
NIP. 19930703 201903 01 011

## **DAFTAR ISI**

LEMBAR PENGESAHAN.....	2
DAFTAR ISI .....	3
DAFTAR GAMBAR.....	4
DAFTAR TABEL .....	5
SOAL.....	6
A.    Source Code .....	7
B.    Output Program.....	31
C.    Pembahasan .....	38
D.    Tautan Git .....	53

## DAFTAR GAMBAR

Gambar 1. Contoh UI List .....	6
Gambar 2. Contoh UI Detail.....	<b>Error! Bookmark not defined.</b>
Gambar 3. Screenshot Hasil Jawaban Soal 1 XML Home Page .....	31
Gambar 4. Screenshot Hasil Jawaban Soal 1 XML Detail Page .....	31
Gambar 5. Screenshot Hasil Jawaban Soal Jetpack Compose Home Page .....	32
Gambar 6. Screenshot Hasil Jawaban Soal 1 Jetpack Compose Detail Page .....	32

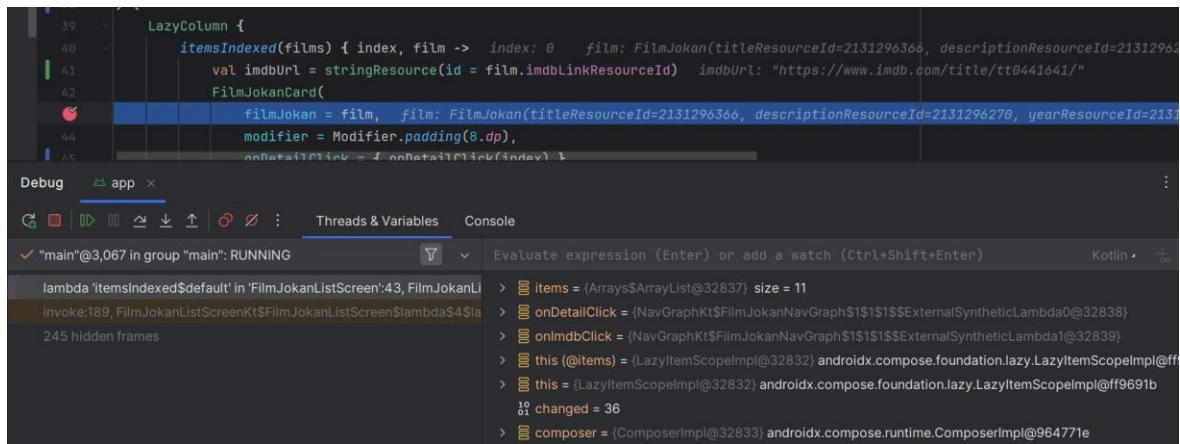
## **DAFTAR TABEL**

Tabel 1. Source Code Jawaban Soal 1 MainActivity XML .....	7
Tabel 2. Source Code Jawaban Soal 1 activity_main XML.....	7
Tabel 3. Source Code Jawaban Soal 1 ListAdapter.kt XML.....	7
Tabel 4. Source Code Jawaban Soal 1 ListItem.kt XML .....	9
Tabel 5. Source Code Jawaban Soal 1 HomeFragment.kt XML .....	11
Tabel 6. Source Code Jawaban Soal 1 DetailFragment.kt XML.....	12
Tabel 7. Source Code Jawaban Soal 1 fragment_detail.xml XML .....	13
Tabel 8. Source Code Jawaban Soal 1 fragment_home.xml XML .....	15
Tabel 9. Source Code Jawaban Soal 1 item_row.xml XML .....	15
Tabel 10. Source Code Jawaban Soal 1 nav_graph1.xml XML.....	17
Tabel 11. Source Code Jawaban Soal 1 MainActivity Jetpack Compose .....	20
Tabel 12. Source Code Jawaban Soal 1 HomeScreen.kt Jetpack Compose .....	20
Tabel 13. Source Code Jawaban Soal 1 DetailScreen.kt Jetpack Compose .....	23
Tabel 14. Source Code Jawaban Soal 1 ListItem.kt Jetpack Compose.....	25
Tabel 15. Source Code Jawaban Soal 1 ListItem.kt Jetpack Compose.....	26

# SOAL

## 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

### XML

#### MainActivity.kt

Tabel 1. Source Code Jawaban Soal 1 MainActivity XML

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

#### activity\_main.xml

Tabel 2. Source Code Jawaban Soal 1 activity\_main XML

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.fragment.app.FragmentContainerView
9         android:id="@+id/nav_host_fragment"
10
11     android:name="androidx.navigation.fragment.NavHostFragment"
12         android:layout_width="0dp"
13         android:layout_height="0dp"
14         app:layout_constraintTop_toTopOf="parent"
15         app:layout_constraintBottom_toBottomOf="parent"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintEnd_toEndOf="parent"
18         app:navGraph="@navigation/nav_graph"
19         app:defaultNavHost="true" />
20 </androidx.constraintlayout.widget.ConstraintLayout>
```

#### ListAdapter.kt

Tabel 3. Source Code Jawaban Soal 1 ListAdapter.kt XML

```
1 package com.example.movieListxml.adapter
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
```

```
5 import androidx.recyclerview.widget.RecyclerView
6 import com.example.movieListxml.databinding.ItemRowBinding
7 import com.example.movieListxml.model.ListItem
8
9 class ListAdapter(
10     private val list: List<ListItem>,
11     private val onDetailClick: (ListItem) -> Unit,
12     private val onOpenLinkClick: (String) -> Unit
13 ) : RecyclerView.Adapter<ListAdapter.ViewHolder>() {
14
15     inner class ViewHolder(private val binding:
16         ItemRowBinding) :
17         RecyclerView.ViewHolder(binding.root) {
18
19         fun bind(item: ListItem) {
20             binding.textViewName.text = item.name
21             binding.textViewRating.text = "★ ${item.rating}"
22             binding.textViewDescription.text =
23                 item.description
24
25             val context = binding.root.context
26             val imageResId = context.resources.getIdentifier(
27                 item.imageResName, "drawable",
28                 context.packageName
29             )
30             binding.imageView.setImageResource(imageResId)
31
32             binding.buttonDetail.setOnClickListener {
33                 onDetailClick(item)
34             }
35
36             binding.buttonIMDb.setOnClickListener {
37                 onOpenLinkClick(item.link)
38             }
39
40         override fun onCreateViewHolder(parent: ViewGroup,
41             viewType: Int): ViewHolder {
42             val binding = ItemRowBinding.inflate(
43                 LayoutInflater.from(parent.context),
44                 parent,
45                 false
46             )
47             return ViewHolder(binding)
48         }
49
50         override fun onBindViewHolder(holder: ViewHolder,
51             position: Int) {
52             holder.bind(list[position])
53         }
54     }
55 }
```

52	override fun getItemCount(): Int = list.size
53	}

## ListItem.kt

Tabel 4. Source Code Jawaban Soal 1 ListItem.kt XML

```

1 package com.example.movieListxml.model
2
3 data class ListItem(
4     val id: Int,
5     val name: String,
6     val rating: Double,
7     val description: String,
8     val imageResName: String, // nama file gambar di drawable
9     (tanpa ekstensi)
10    val link: String
11 )
12
13 // Hardcoded data
14 val listItems = listOf(
15     ListItem(
16         1, "Toy Story", 8.3,
17         "A cowboy doll is profoundly threatened and jealous
when a new spaceman action figure supplants him as top toy in a
boy's bedroom.",
18         "pixar1",
19         "https://www.imdb.com/title/tt0114709/?ref_=ls_t_1"
20     ),
21     ListItem(
22         2, "A Bug's Life", 7.2,
23         "A misfit ant, looking for \"warriors\" to save his
colony from greedy grasshoppers, recruits a group of bugs that
turn out to be an inept circus troupe.",
24         "pixar2",
25         "https://www.imdb.com/title/tt0120623/?ref_=ls_t_2"
26     ),
27     ListItem(
28         3, "Toy Story 2", 7.9,
29         "When Woody is stolen by a toy collector, Buzz and his
friends set out on a rescue mission to save Woody before he
becomes a museum toy.",
30         "pixar3",
31         "https://www.imdb.com/title/tt0120363/?ref_=ls_t_3"
32     ),
33     ListItem(
34         4, "Monsters Inc", 8.1,
35         "In order to power the city, monsters have to scare
children so that they scream. However, the children are toxic
to the monsters...",
         "pixar4",

```

```
36     "https://www.imdb.com/title/tt0198781/?ref_=ls_t_4"
37     ),
38     ListItem(
39         5, "Finding Nemo", 8.2,
40         "After his son is captured in the Great Barrier Reef
41         and taken to Sydney, a timid clownfish sets out on a journey to
42         bring him home.",
43         "pixar5",
44         "https://www.imdb.com/title/tt0266543/?ref_=ls_t_5"
45     ),
46     ListItem(
47         6, "The Incredibles", 8.0,
48         "While trying to lead a quiet suburban life, a family
49         of undercover superheroes are forced into action to save the
50         world.",
51         "pixar6",
52         "https://www.imdb.com/title/tt0317705/?ref_=ls_t_6"
53     ),
54     ListItem(
55         7, "Cars", 7.3,
56         "On the way to the biggest race of his life, a hotshot
57         rookie race car gets stranded in a rundown town.",
58         "pixar7",
59         "https://www.imdb.com/title/tt0317219/?ref_=ls_t_7"
60     ),
61     ListItem(
62         8, "Ratatouille", 8.1,
63         "A rat who can cook makes an unusual alliance with a
64         young kitchen worker at a famous Paris restaurant.",
65         "pixar8",
66         "https://www.imdb.com/title/tt0382932/?ref_=ls_t_8"
67     ),
68     ListItem(
69         9, "WALL-E", 8.4,
70         "A robot responsible for cleaning a waste-covered Earth
71         meets another robot and falls in love with her.",
72         "pixar9",
73         "https://www.imdb.com/title/tt0910970/?ref_=ls_t_9"
74     ),
75     ListItem(
76         10, "Up", 8.3,
77         "78-year-old Carl Fredricksen travels to South America
78         in his house equipped with balloons, inadvertently taking a
79         young stowaway.",
80         "pixar10",
81         "https://www.imdb.com/title/tt1049413/?ref_=ls_t_10"
82     )
83 )
```

## HomeFragment.kt

Tabel 5. Source Code Jawaban Soal 1 HomeFragment.kt XML

```
1 package com.example.movieListxml.ui
2
3 import android.content.Intent
4 import android.net.Uri
5 import android.os.Bundle
6 import android.view.LayoutInflater
7 import android.view.View
8 import android.view.ViewGroup
9 import androidx.fragment.app.Fragment
10 import androidx.recyclerview.widget.LinearLayoutManager
11 import com.example.movieListxml.adapter.ListAdapter
12 import com.example.movieListxml.databinding.FragmentHomeBinding
13 import androidx.navigation.fragment.findNavController
14 import com.example.movieListxml.R
15 import androidx.lifecycle.ViewModelProvider
16 import com.example.movieListxml.ui.viewmodel.MovieViewModel
17 import
18 com.example.movieListxml.ui.viewmodel.MovieViewModelFactory
19 import timber.log.Timber
20
21 class HomeFragment : Fragment() {
22
23     private var _binding: FragmentHomeBinding? = null
24     private val binding get() = _binding!!
25
26     override fun onCreateView(
27         inflater: LayoutInflater, container: ViewGroup?,
28         savedInstanceState: Bundle?
29     ): View {
30         _binding = FragmentHomeBinding.inflate(inflater,
31         container, false)
32         return binding.root
33     }
34
35     override fun onViewCreated(view: View, savedInstanceState:
36     Bundle?) {
37         super.onViewCreated(view, savedInstanceState)
38
39         val factory = MovieViewModelFactory("From
40 HomeFragment")
41         val viewModel = ViewModelProvider(this,
42         MovieViewModelFactory(factory.toString())) [MovieViewModel::clas
43         s.java]
44         val movieList = viewModel.movieList.value
45
46         val adapter = ListAdapter(
47             movieList,
48             onDetailClick = { selectedItem ->
```

```

47         Timber.i("Detail button clicked for ID:
48     ${selectedItem.id}")
49         viewModel.selectItem(selectedItem.id)
50         val bundle = Bundle().apply {
51             putInt("itemId", selectedItem.id)
52         }
53         findNavController().navigate(
54             R.id.action_homeFragment_to_detailFragment,
55             bundle
56         )
57         onOpenLinkClick = { url ->
58             Timber.i("Explicit Intent button clicked.
59 Opening link: $url")
60             val intent = Intent(Intent.ACTION_VIEW)
61             intent.data = Uri.parse(url)
62             startActivity(intent)
63         }
64     )
65
66     binding.recyclerView.layoutManager =
67     LinearLayoutManager(requireContext())
68     binding.recyclerView.adapter = adapter
69 }
70
71 override fun onDestroyView() {
72     super.onDestroyView()
73     _binding = null
74 }
75
76 }
```

## DetailFragment.kt

Tabel 6. Source Code Jawaban Soal 1 DetailFragment.kt XML

```

1 package com.example.movieListxml.ui
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
9 com.example.movieListxml.databinding.FragmentDetailBinding
10 import androidx.lifecycle.ViewModelProvider
11 import com.example.movieListxml.ui.viewmodel.MovieViewModel
12 import
13 com.example.movieListxml.ui.viewmodel.MovieViewModelFactory
14 import timber.log.Timber
15 }
```

```

16 class DetailFragment : Fragment() {
17
18     private var _binding: FragmentDetailBinding? = null
19     private val binding get() = _binding!!
20
21     override fun onCreateView(
22         inflater: LayoutInflater, container: ViewGroup?,
23         savedInstanceState: Bundle?
24     ): View {
25         _binding = FragmentDetailBinding.inflate(inflater,
26         container, false)
27         return binding.root
28     }
29
30     override fun onViewCreated(view: View, savedInstanceState:
31     Bundle?) {
32         super.onViewCreated(view, savedInstanceState)
33
34         val itemId = arguments?.getInt("itemId") ?: return
35         val factory = MovieViewModelFactory("From
36 HomeFragment")
37         val viewModel = ViewModelProvider(this,
38         MovieViewModelFactory(factory.toString()))[MovieViewModel::clas
39         s.java]
40         val item = viewModel.getMovieById(itemId) ?: return
41
42         // Set data ke UI
43         binding.textViewTitle.text = item.name
44         binding.textViewRating.text = "★ ${item.rating}"
45         binding.textViewDescription.text = item.description
46
47         Timber.i("Detail screen loaded with item:
48 ID=${item.id}, Title='${item.name}', Rating=${item.rating}")
49
50         val context = binding.root.context
51         val imageResId = context.resources.getIdentifier(
52             item.imageResName, "drawable", context.packageName
53         )
54         binding.imageView.setImageResource(imageResId)
55     }
56
57     override fun onDestroyView() {
58         super.onDestroyView()
59         _binding = null
60     }
61 }
```

## fragment\_detail.xml

Tabel 7. Source Code Jawaban Soal 1 fragment\_detail.xml XML

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ScrollView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4         android:layout_width="match_parent"
5         android:layout_height="match_parent"
6         android:padding="16dp"
7         android:background="#FFFFFF">
8
9     <LinearLayout
10        android:layout_width="match_parent"
11        android:layout_height="wrap_content"
12        android:orientation="vertical"
13        android:gravity="center_horizontal">
14
15     <ImageView
16         android:id="@+id/imageView"
17         android:layout_width="wrap_content"
18         android:layout_height="wrap_content"
19         android:layout_marginTop="16dp"
20         android:layout_marginBottom="16dp"
21         android:contentDescription="Movie Poster"
22         android:scaleType="centerCrop" />
23
24     <TextView
25         android:id="@+id/textViewTitle"
26         android:layout_width="wrap_content"
27         android:layout_height="wrap_content"
28         android:text="Movie Title"
29         android:textSize="24sp"
30         android:textStyle="bold"
31         android:textColor="#000000"
32         android:layout_marginBottom="8dp" />
33
34     <TextView
35         android:id="@+id/textViewRating"
36         android:layout_width="wrap_content"
37         android:layout_height="wrap_content"
38         android:text="★ 8.4"
39         android:textSize="18sp"
40         android:textColor="#666666"
41         android:layout_marginBottom="16dp" />
42
43     <TextView
44         android:id="@+id/textViewDescription"
45         android:layout_width="match_parent"
46         android:layout_height="wrap_content"
47         android:text="This is the full movie
48         description..."
49         android:textSize="16sp"
50         android:textColor="#333333" />
51
52     </LinearLayout>
53 </ScrollView>
```

## **fragment\_home.xml**

*Tabel 8. Source Code Jawaban Soal 1 fragment\_home.xml XML*

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:padding="16dp">
8
9     <TextView
10        android:id="@+id/textViewTitle"
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13        android:text="Daftar Film"
14        android:textSize="24sp"
15        android:textStyle="bold"
16        android:textColor="#000000"
17        app:layout_constraintTop_toTopOf="parent"
18        app:layout_constraintStart_toStartOf="parent" />
19
20     <androidx.recyclerview.widget.RecyclerView
21         android:id="@+id/recyclerView"
22         android:layout_width="0dp"
23         android:layout_height="0dp"
24         android:clipToPadding="false"
25         android:paddingTop="12dp"
26
27         app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
28         app:layout_constraintTop_toBottomOf="@+id/textViewTitle"
29         app:layout_constraintBottom_toBottomOf="parent"
30         app:layout_constraintStart_toStartOf="parent"
31         app:layout_constraintEnd_toEndOf="parent" />
32 </androidx.constraintlayout.widget.ConstraintLayout>
```

## **Item\_row.xml**

*Tabel 9. Source Code Jawaban Soal 1 item\_row.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     android:layout_width="match_parent"
6     android:layout_height="wrap_content"
7     android:layout_margin="8dp">
```

```
7     app:cardCornerRadius="12dp"
8     app:cardElevation="4dp">
9
10    <LinearLayout
11        android:layout_width="match_parent"
12        android:layout_height="match_parent"
13        android:gravity="center_vertical"
14        android:orientation="horizontal"
15        android:padding="12dp">
16
17        <ImageView
18            android:id="@+id/imageView"
19            android:layout_width="124dp"
20            android:layout_height="208dp"
21            android:layout_marginEnd="12dp"
22            android:contentDescription="Movie Image"
23            android:scaleType="centerCrop" />
24
25        <LinearLayout
26            android:layout_width="236dp"
27            android:layout_height="209dp"
28            android:layout_weight="1"
29            android:orientation="vertical">
30
31            <TextView
32                android:id="@+id/textViewName"
33                android:layout_width="wrap_content"
34                android:layout_height="wrap_content"
35                android:text="Movie Name"
36                android:textColor="#000000"
37                android:textSize="18sp"
38                android:textStyle="bold" />
39
40            <TextView
41                android:id="@+id/textViewRating"
42                android:layout_width="wrap_content"
43                android:layout_height="wrap_content"
44                android:layout_marginTop="4dp"
45                android:text="Rating"
46                android:textColor="#666666" />
47
48            <TextView
49                android:id="@+id/textViewDescription"
50                android:layout_width="match_parent"
51                android:layout_height="wrap_content"
52                android:text="Description"
53                android:textSize="12dp" />
54
55            <LinearLayout
56                android:layout_width="match_parent"
57                android:layout_height="wrap_content"
58                android:layout_marginTop="8dp"
```

```

59         android:orientation="horizontal">
60
61             <Button
62                 android:id="@+id/buttonDetail"
63                 android:layout_width="17dp"
64                 android:layout_height="wrap_content"
65                 android:layout_weight="1"
66                 android:backgroundTint="#9C27B0"
67                 android:text="Lihat Detail"
68                 android:textColor="#FFFFFF" />
69
70             <Button
71                 android:id="@+id/buttonIMDb"
72                 android:layout_width="0dp"
73                 android:layout_height="match_parent"
74                 android:layout_marginStart="8dp"
75                 android:layout_weight="1"
76                 android:backgroundTint="#3F51B5"
77                 android:text="IMDB"
78                 android:textColor="#FFFFFF" />
79         </LinearLayout>
80     </LinearLayout>
81
82 </LinearLayout>
83
84 </androidx.cardview.widget.CardView>

```

## Nav\_graph.xml

Tabel 10. Source Code Jawaban Soal 1 nav\_graph.xml XML

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <navigation
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:id="@+id/nav_graph"
6     app:startDestination="@+id/homeFragment">
7
8     <fragment
9         android:id="@+id/homeFragment"
10        android:name="com.example.movieListxml.ui.HomeFragment"
11        android:label="Home" >
12         <action
13             android:id="@+id/action_homeFragment_to_detailFragment"
14             app:destination="@+id/detailFragment"
15             app:enterAnim="@android:anim/slide_in_left"
16             app:exitAnim="@android:anim/slide_out_right"
17             app:popEnterAnim="@android:anim/slide_in_left"
18             app:popExitAnim="@android:anim/slide_out_right" />
19     </fragment>

```

```

19    <fragment
20        android:id="@+id/detailFragment"
21
22        android:name="com.example.movieListxml.ui.DetailFragment"
23            android:label="Detail" >
24            <argument
25                android:name="itemId"
26                app:argType="integer" />
27        </fragment>
28
29    </navigation>

```

## MovieViewModel.kt

Tabel 11. Source Code Jawaban Soal 1 MovieViewModel.kt XML

```

1 package com.example.movieListxml.ui.viewmodel
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5 import com.example.movieListxml.model.ListItem
6 import com.example.movieListxml.model.listItems
7 import kotlinx.coroutines.flow.MutableStateFlow
8 import kotlinx.coroutines.flow.StateFlow
9 import timber.log.Timber
10
11 class MovieViewModel (private val sourceName: String) :
12 ViewModel() {
13
14     // Movie list
15     private val _movieList = MutableStateFlow(listItems)
16     val movieList: StateFlow<List<ListItem>> get() =
17         _movieList
18
19     init {
20         Timber.i("Movie list initialized with
21             ${listItems.size} items from source: $sourceName")
22     }
23
24     // Selected item (by ID)
25     private val _selectedItemId = MutableStateFlow<Int?>(null)
26     val selectedItemId: StateFlow<Int?> get() =
27         _selectedItemId
28
29     fun selectItem(id: Int) {
30         _selectedItemId.value = id
31     }
32
33     fun getMovieById(id: Int): ListItem? {
34         return _movieList.value.find { it.id == id }
35     }

```

32	}
33	}

## MovieViewModelFactory.kt

Tabel 12. Source Code Jawaban Soal 1 MovieViewModelFactory.kt XML

```
1 package com.example.movieListxml.ui.viewmodel
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5
6 class MovieViewModelFactory(private val sourceName: String) :
7 ViewModelProvider.Factory {
8     override fun <T : ViewModel> create(modelClass: Class<T>):
9 T {
10         if
11             (modelClass.isAssignableFrom(MovieViewModel::class.java)) {
12                 return MovieViewModel(sourceName) as T
13             }
14             throw IllegalArgumentException("Unknown ViewModel
15 class")
16         }
17 }
```

## MovieApp.kt

Tabel 13. Source Code Jawaban Soal 1 MovieApp.kt XML

```
1 package com.example.movieListxml
2
3 import android.app.Application
4 import timber.log.Timber
5
6 class MovieApp : Application() {
7     override fun onCreate() {
8         super.onCreate()
9
10         if (BuildConfig.DEBUG) {
11             Timber.plant(Timber.DebugTree())
12             Timber.i("Timber initialized in MovieApp")
13         }
14     }
15 }
```

## JETPACK COMPOSE

### MainActivity.kt

Tabel 14. Source Code Jawaban Soal 1 MainActivity Jetpack Compose

```
1 package com.example.movieList
2
3 import android.os.Bundle
4 import androidx.activity.ComponentActivity
5 import androidx.activity.compose.setContent
6 import androidx.compose.material3.MaterialTheme
7 import androidx.compose.material3.Surface
8 import androidx.compose.runtime.Composable
9 import androidx.compose.ui.tooling.preview.Preview
10 import androidx.navigation.compose.rememberNavController
11 import com.example.movieList.navigation.AppNavGraph
12 import com.example.movieList.ui.theme.MovieListTheme
13
14 class MainActivity : ComponentActivity() {
15     override fun onCreate(savedInstanceState: Bundle?) {
16         super.onCreate(savedInstanceState)
17         setContent {
18             MovieListTheme {
19                 Surface(color =
20                     MaterialTheme.colorScheme.background) {
21                         val navController = rememberNavController()
22                         AppNavGraph(navController = navController)
23                     }
24                 }
25             }
26         }
27
28 @Composable
29 @Preview(showBackground = true, showSystemUi = true)
30 fun MainPreview() {
31     MovieListTheme {
32         val navController = rememberNavController()
33         AppNavGraph(navController = navController)
34     }
35 }
```

### HomeScreen.kt

Tabel 15. Source Code Jawaban Soal 1 HomeScreen.kt Jetpack Compose

```
1 package com.example.movieList.ui.screen
2
3 import android.content.Intent
4 import android.net.Uri
5 import androidx.compose.foundation.Image
```

```
6 import androidx.compose.foundation.background
7 import androidx.compose.foundation.layout.*
8 import androidx.compose.foundation.lazy.LazyColumn
9 import androidx.compose.foundation.shape.RoundedCornerShape
10 import androidx.compose.material3.*
11 import androidx.compose.runtime.Composable
12 import androidx.compose.ui.Alignment
13 import androidx.compose.ui.Modifier
14 import androidx.compose.ui.draw.clip
15 import androidx.compose.ui.graphics.Color
16 import androidx.compose.ui.layout.ContentScale
17 import androidx.compose.ui.platform.LocalContext
18 import androidx.compose.ui.res.colorResource
19 import androidx.compose.ui.res.painterResource
20 import androidx.compose.ui.text.font.FontWeight
21 import androidx.compose.ui.text.style.TextAlign
22 import androidx.compose.ui.unit.dp
23 import androidx.compose.ui.unit.sp
24 import androidx.navigation.NavController
25 import com.example.movieList.model.listItems
26
27 @Composable
28 fun HomeScreen(navController: NavController) {
29     val context = LocalContext.current
30
31     LazyColumn(
32         contentPadding = PaddingValues(20.dp),
33         verticalArrangement = Arrangement.spacedBy(26.dp),
34     ) {
35         items(listItems.size) { index ->
36             val item = listItems[index]
37
38             Card(
39                 shape = RoundedCornerShape(16.dp),
40                 colors =
41                     CardDefaults.cardColors(containerColor = Color(0xFF141f1f)),
42                     modifier = Modifier.fillMaxWidth()
43             ) {
44                 Row(
45                     modifier = Modifier
46                         .padding(12.dp)
47                         .fillMaxWidth(),
48                     verticalAlignment =
49                         Alignment.CenterVertically
50                 ) {
51                     // Movie Image
52                     Image(
53                         painter = painterResource(
54                             id =
55                             context.resources.getIdentifier(
56                                 item.imageResName,
57                                 "drawable",
58                         )
59                     )
60                 )
61             }
62         }
63     }
64 }
```

```
55                                         context.packageName
56                                         )
57                                         ),
58                                         contentDescription = item.name,
59                                         modifier = Modifier
60                                         .size(width = 90.dp, height =
61                                         130.dp)
62                                         .clip(RoundedCornerShape(10.dp)),
63                                         contentScale = ContentScale.Crop
64                                         )
65
66                                         Spacer(modifier = Modifier.width(12.dp))
67
68                                         // Movie Details
69                                         Column(
70                                         modifier = Modifier.weight(1f)
71                                         ) {
72                                         Row(
73                                         modifier =
74                                         Modifier.fillMaxWidth(),
75                                         horizontalArrangement =
76                                         Arrangement.SpaceBetween
77                                         ) {
78                                         Text(
79                                         text = item.name,
80                                         fontSize = 18.sp,
81                                         fontWeight = FontWeight.Bold,
82                                         color = Color.White
83                                         )
84                                         }
85                                         Spacer(modifier =
86                                         Modifier.height(4.dp))
87
88                                         Text(
89                                         text = "Plot:",
90                                         fontWeight = FontWeight.Bold,
91                                         fontSize = 14.sp,
92                                         color = Color.White
93                                         )
94
95                                         Text(
96                                         text = item.description,
97                                         fontSize = 13.sp,
98                                         color = Color(0xFFE0E0E0),
99                                         maxLines = 4
100                                         )
101                                         Spacer(modifier =
102                                         Modifier.height(8.dp))
103                                         Row(
```

```

101    horizontalArrangement =
102        Arrangement.SpaceEvenly,
103        modifier = Modifier.fillMaxWidth()
104    ) {
105        Button(
106            onClick = {
107                val intent =
108                    Intent(Intent.ACTION_VIEW, Uri.parse(item.link))
109                context.startActivity(intent)
110            },
111            colors =
112                ButtonDefaults.buttonColors(containerColor =
113                    Color(0xFFB3E5FC)),
114                shape = RoundedCornerShape(50)
115            ) {
116                Text("IMDB", color =
117                    Color.Black)
118            }
119        Button(
120            onClick = {
121                navController.navigate("detail/${item.id}")
122            },
123            colors =
124                ButtonDefaults.buttonColors(containerColor =
125                    Color(0xFF9FA8DA)),
126                shape = RoundedCornerShape(50)
127            ) {
128                Text("Detail", color =
129                    Color.Black)
130            }
131        }

```

## DetailScreen.kt

Tabel 16. Source Code Jawaban Soal 1 DetailScreen.kt Jetpack Compose

```

1 package com.example.movieList.ui.screen
2
3 import androidx.compose.foundation.Image
4 import androidx.compose.foundation.background
5 import androidx.compose.foundation.layout.*
6 import androidx.compose.foundation.shape.RoundedCornerShape

```

```
7 import androidx.compose.material3.*  
8 import androidx.compose.material3.Text  
9 import androidx.compose.runtime.Composable  
10 import androidx.compose.ui.Alignment  
11 import androidx.compose.ui.Modifier  
12 import androidx.compose.ui.draw.clip  
13 import androidx.compose.ui.graphics.Color  
14 import androidx.compose.ui.layout.ContentScale  
15 import androidx.compose.ui.platform.LocalContext  
16 import androidx.compose.ui.res.painterResource  
17 import androidx.compose.ui.text.font.FontWeight  
18 import androidx.compose.ui.text.style.TextAlign  
19 import androidx.compose.ui.unit.dp  
20 import androidx.compose.ui.unit.sp  
21 import androidx.navigation.NavController  
22 import com.example.movieList.model.listItems  
23  
24 @Composable  
25 fun DetailScreen(  
26     itemId: Int,  
27     navController: NavController  
28 ) {  
29     val context = LocalContext.current  
30     val item = listItems.find { it.id == itemId }  
31  
32     if (item != null) {  
33         Column(  
34             modifier = Modifier  
35                 .fillMaxSize()  
36                 .background(Color(0xFFFF3EDF))  
37                 .padding(16.dp),  
38             verticalArrangement = Arrangement.Top,  
39             horizontalAlignment = Alignment.CenterHorizontally  
40     ) {  
41         // Image  
42         Image(  
43             painter = painterResource(  
44                 id = context.resources.getIdentifier(  
45                     item.imageResName,  
46                     "drawable",  
47                     context.packageName  
48             )  
49         ),  
50         contentDescription = item.name,  
51         modifier = Modifier  
52             .clip(RoundedCornerShape(12.dp)),  
53             contentScale = ContentScale.Crop  
54     )  
55  
56         Spacer(modifier = Modifier.height(16.dp))  
57  
58         // Name
```

```

59         Text(
60             text = item.name,
61             fontSize = 24.sp,
62             fontWeight = FontWeight.Bold,
63         )
64
65         // Rating
66         Text(
67             text = "★ ${item.rating}",
68             fontSize = 18.sp,
69             color = Color.DarkGray
70         )
71
72         Spacer(modifier = Modifier.height(12.dp))
73
74         // Description
75         Text(
76             text = item.description,
77             fontSize = 16.sp,
78             color = Color.Black,
79             textAlign = TextAlign.Justify
80         )
81
82         Spacer(modifier = Modifier.height(24.dp))
83
84         // Back Button
85         Button(
86             onClick = { navController.popBackStack() },
87             shape = RoundedCornerShape(12.dp),
88             colors =
89             ButtonDefaults.buttonColors(containerColor =
90                 Color(0xFF9C27B0))
91             ) {
92                 Text("Back")
93             }
94         } else {
95             Text("Item not found", color = Color.Red)
96         }

```

## ListItem.kt

Tabel 17. Source Code Jawaban Soal 1 ListItem.kt Jetpack Compose

```

1 package com.example.movieList.navigation
2
3 import androidx.compose.runtime.Composable
4 import androidx.navigation.NavHostController
5 import androidx.navigation.compose.NavHost
6 import androidx.navigation.compose.composable

```

```

7 import com.example.movieList.ui.screen.DetailScreen
8 import com.example.movieList.ui.screen.HomeScreen
9
10 object Routes {
11     const val HOME = "home"
12     const val DETAIL = "detail/{itemId}"
13 }
14
15 @Composable
16 fun AppNavGraph(navController: NavHostController) {
17     NavHost(
18         navController = navController,
19         startDestination = Routes.HOME
20     ) {
21         composable(Routes.HOME) {
22             HomeScreen(navController)
23         }
24         composable("detail/{itemId}") { backStackEntry ->
25             val itemId =
26                 backStackEntry.arguments?.getString("itemId")?.toIntOrNull()
27             if (itemId != null) {
28                 DetailScreen(itemId = itemId, navController =
29                     navController)
30             }
31         }
32     }
33 }

```

## ListItem.kt

Tabel 18. Source Code Jawaban Soal 1 ListItem.kt Jetpack Compose

```

1 package com.example.movieList.model
2
3 data class ListItem(
4     val id: Int,
5     val name: String,
6     val rating: Double,
7     val description: String,
8     val imageResName: String, // just the drawable resource
name, without extension
9     val link: String
10 )
11
12 val listItems = listOf(
13     ListItem(
14         id = 1,
15         name = "Toy Story",
16         rating = 8.3,
17         description = "A cowboy doll is profoundly threatened
and jealous when a new spaceman action figure supplants him as

```

```
18     top toy in a boy's bedroom.",
19         imageResName = "pixar1",
20         link =
21     "https://www.imdb.com/title/tt0114709/?ref_=ls_t_1"
22     ),
23     ListItem(
24         id = 2,
25         name = "A Bug's Life",
26         rating = 7.2,
27         description = "A misfit ant, looking for 'warriors' to
28         save his colony from greedy grasshoppers, recruits a group of
29         bugs that turn out to be an inept circus troupe.",
30         imageResName = "pixar2",
31         link =
32     "https://www.imdb.com/title/tt0120623/?ref_=ls_t_2"
33     ),
34     ListItem(
35         id = 3,
36         name = "Toy Story 2",
37         rating = 7.9,
38         description = "When Woody is stolen by a toy collector,
39         Buzz and his friends set out on a rescue mission to save Woody
40         before he becomes a museum toy property with his roundup gang
41         Jessie, Prospector, and Bullseye.",
42         imageResName = "pixar3",
43         link =
44     "https://www.imdb.com/title/tt0120363/?ref_=ls_t_3"
45     ),
46     ListItem(
47         id = 4,
48         name = "Monster's Inc",
49         rating = 8.1,
50         description = "In order to power the city, monsters
51         have to scare children so that they scream. However, the
52         children are toxic to the monsters, and after a child gets
53         through, two monsters realize things may not be what they
54         think.",
55         imageResName = "pixar4",
56         link =
57     "https://www.imdb.com/title/tt0198781/?ref_=ls_t_4"
58     ),
59     ListItem(
60         id = 5,
61         name = "Finding Nemo",
62         rating = 8.2,
63         description = "After his son is captured in the Great
64         Barrier Reef and taken to Sydney, a timid clownfish sets out on
65         a journey to bring him home.",
66         imageResName = "pixar5",
67         link =
68     "https://www.imdb.com/title/tt0266543/?ref_=ls_t_5"
69     ),
```

```
53     ListItem(
54         id = 6,
55         name = "The Incredibles",
56         rating = 8.0,
57         description = "While trying to lead a quiet suburban
58         life, a family of undercover superheroes are forced into action
59         to save the world.",
60         imageResName = "pixar6",
61         link =
62             "https://www.imdb.com/title/tt0317705/?ref_=ls_t_6"
63     ),
64     ListItem(
65         id = 7,
66         name = "Cars",
67         rating = 7.3,
68         description = "On the way to the biggest race of his
69         life, a hotshot rookie race car gets stranded in a rundown town
70         and learns that winning isn't everything in life.",
71         imageResName = "pixar7",
72         link =
73             "https://www.imdb.com/title/tt0317219/?ref_=ls_t_7"
74     ),
75     ListItem(
76         id = 8,
77         name = "Ratatouille",
78         rating = 8.1,
79         description = "A rat who can cook makes an unusual
80         alliance with a young kitchen worker at a famous Paris
81         restaurant.",
82         imageResName = "pixar8",
83         link =
84             "https://www.imdb.com/title/tt0382932/?ref_=ls_t_8"
85     ),
86     ListItem(
87         id = 9,
88         name = "WALL-E",
89         rating = 8.4,
90         description = "A robot who is responsible for cleaning
91         a waste-covered Earth meets another robot and falls in love
92         with her. Together, they set out on a journey that will alter
93         the fate of mankind.",
94         imageResName = "pixar9",
95         link =
96             "https://www.imdb.com/title/tt0910970/?ref_=ls_t_9"
97     ),
98     ListItem(
99         id = 10,
100        name = "Up",
101        rating = 8.3,
102        description = "78-year-old Carl Fredricksen travels to
103        South America in his house equipped with balloons,
104        inadvertently taking a young stowaway.",
```

```

90     imageResName = "pixar10",
91     link =
92     )
93 )

```

## MovieViewModel.kt

Tabel 19. Source Code Jawaban Soal 1 MovieViewModel.kt XML

```

1 package com.example.movieListxml.ui.viewmodel
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5 import com.example.movieListxml.model.ListItem
6 import com.example.movieListxml.model.listItems
7 import kotlinx.coroutines.flow.MutableStateFlow
8 import kotlinx.coroutines.flow.StateFlow
9 import timber.log.Timber
10
11 class MovieViewModel (private val sourceName: String) :
12 ViewModel() {
13
14     // Movie list
15     private val _movieList = MutableStateFlow(listItems)
16     val movieList: StateFlow<List<ListItem>> get() =
17     _movieList
18
19     init {
20         Timber.i("Movie list initialized with
21 ${listItems.size} items from source: $sourceName")
22     }
23
24     // Selected item (by ID)
25     private val _selectedItemId = MutableStateFlow<Int?>(null)
26     val selectedItemId: StateFlow<Int?> get() =
27     _selectedItemId
28
29     fun selectItem(id: Int) {
30         _selectedItemId.value = id
31     }
32
33     fun getMovieById(id: Int): ListItem? {
34         return _movieList.value.find { it.id == id }
35     }
36
37 }

```

## MovieViewModelFactory.kt

Tabel 20. Source Code Jawaban Soal 1 MovieViewModelFactory.kt XML

```

1 package com.example.movieListxml.ui.viewmodel
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5
6 class MovieViewModelFactory(private val sourceName: String) :
7 ViewModelProvider.Factory {
8     override fun <T : ViewModel> create(modelClass: Class<T>) :
9 T {
10     if
11 (modelClass.isAssignableFrom(MovieViewModel::class.java)) {
12         return MovieViewModel(sourceName) as T
13     }
14     throw IllegalArgumentException("Unknown ViewModel
15 class")
15 }
}

```

## MovieApp.kt

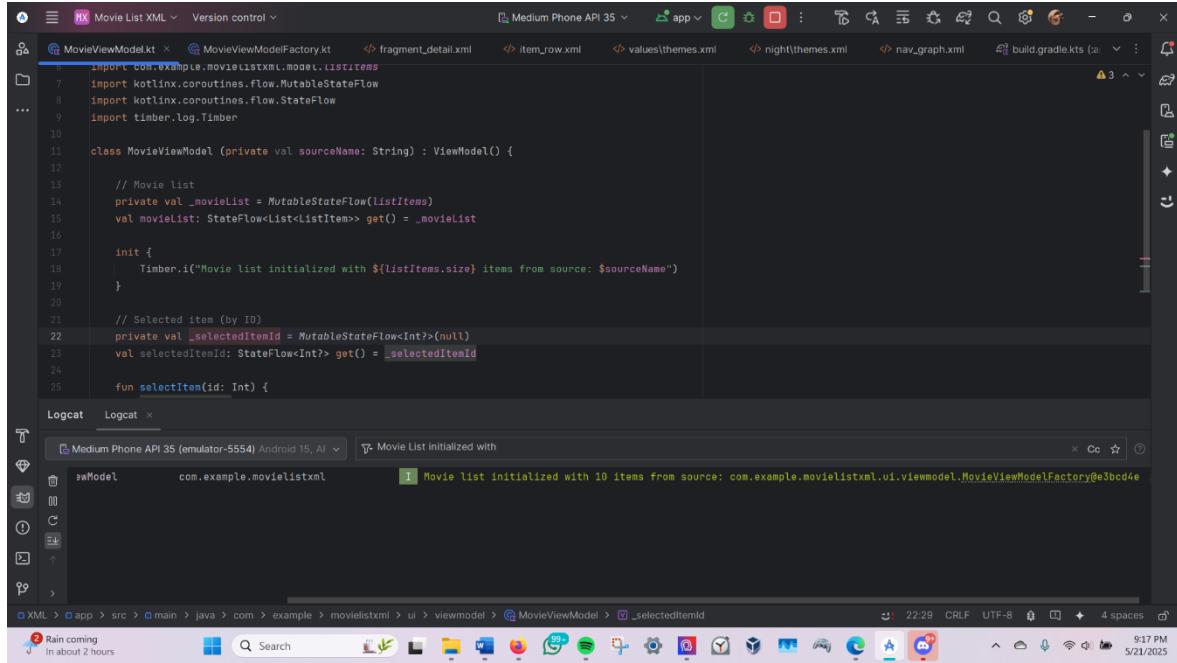
*Tabel 21. Source Code Jawaban Soal 1 MovieApp.kt XML*

```

1 package com.example.movieListxml
2
3 import android.app.Application
4 import timber.log.Timber
5
6 class MovieApp : Application() {
7     override fun onCreate() {
8         super.onCreate()
9
10         if (BuildConfig.DEBUG) {
11             Timber.plant(Timber.DebugTree())
12             Timber.i("Timber initialized in MovieApp")
13         }
14     }
15 }

```

## B. Output Program



The screenshot shows the Android Studio interface with the code editor open to `MovieViewModel.kt`. The code uses Timber for logging when the movie list is initialized:

```
import com.example.movieListxml.movieListItems
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import timber.log.Timber

class MovieViewModel (private val sourceName: String) : ViewModel {
    private val _movieList = MutableStateFlow(listItems)
    val movieList: StateFlow<List<ListItem>> get() = _movieList

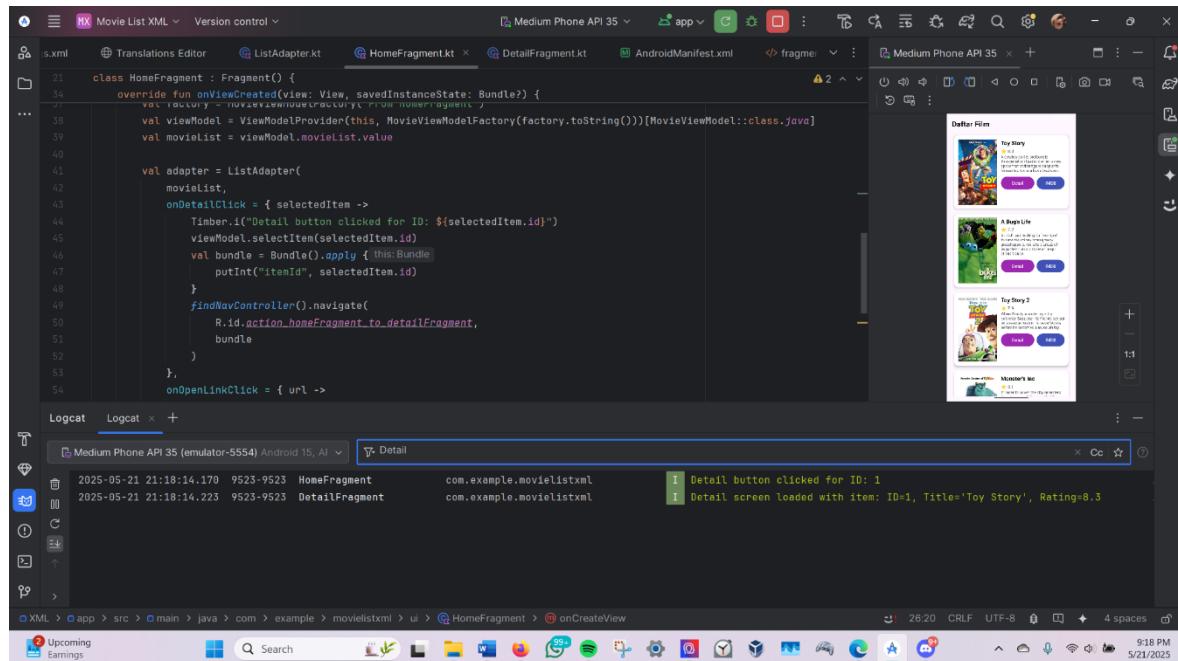
    init {
        Timber.i("Movie list initialized with ${listItems.size} items from source: $sourceName")
    }

    private val _selectedItemId = MutableStateFlow<Int?>(null)
    val selectedItemId: StateFlow<Int?> get() = _selectedItemId

    fun selectItem(id: Int) {
        _selectedItemId.value = id
    }
}
```

The Logcat tab shows the output: "Movie list initialized with 10 items from source: com.example.movieListxml.viewmodel.MovieViewModelFactory@e3bcd4e".

Gambar 2. Screenshot Penggunaan Timber ketika Data Masuk ke List XML



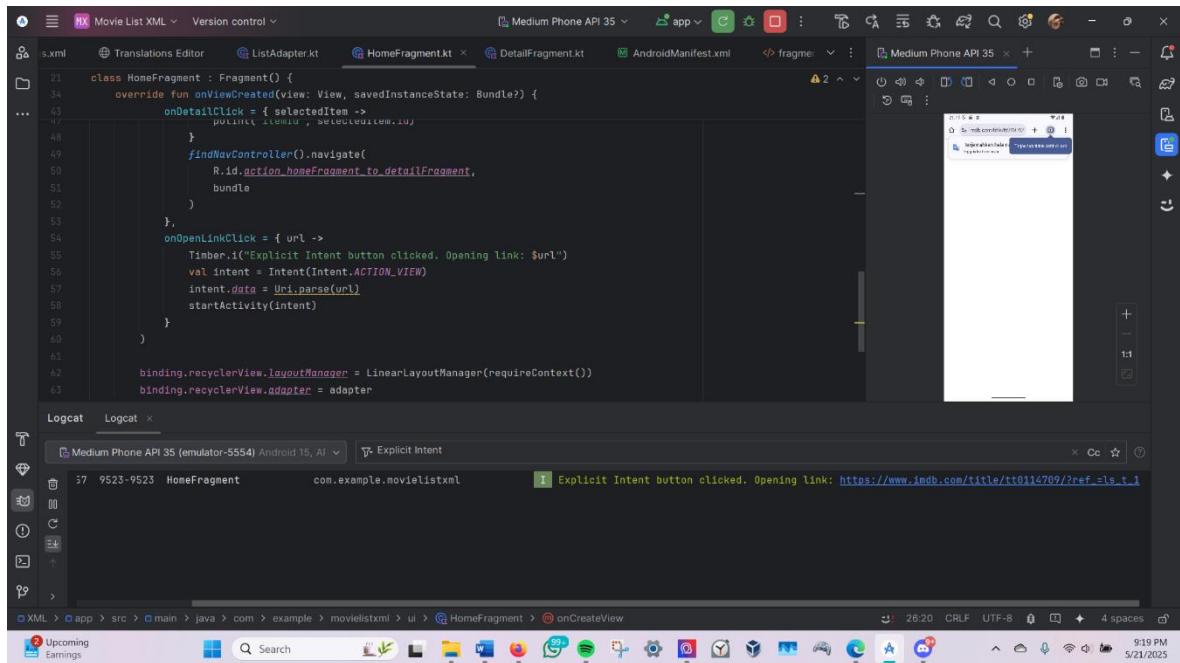
The screenshot shows the Android Studio interface with the code editor open to `HomeFragment.kt`. The code uses Timber for logging when a detail button is clicked:

```
class HomeFragment : Fragment() {
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        val recyclerView = view.findViewById<RecyclerView>(R.id.recyclerview)
        val movieList = viewModel.movieList.value

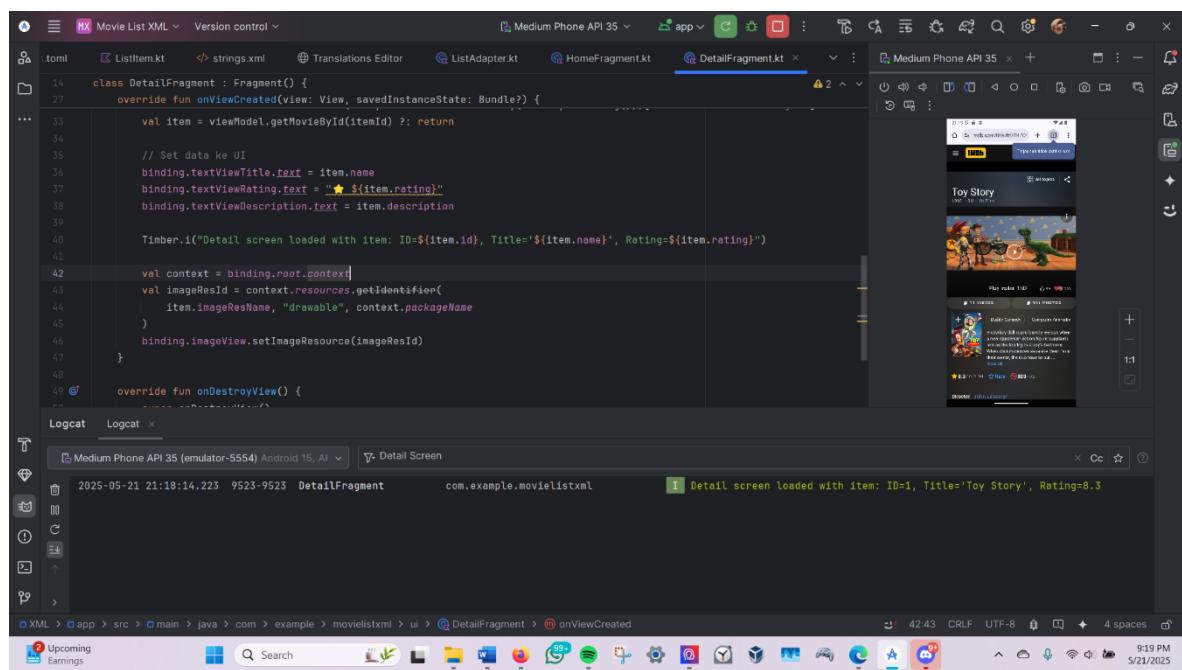
        val adapter = ListAdapter(
            movieList,
            onDetailClick = { selectedItem ->
                Timber.i("Detail button clicked for ID: ${selectedItem.id}")
                viewModel.selectItem(selectedItem.id)
                val bundle = Bundle().apply { this.bundle.putInt("itemId", selectedItem.id) }
                findNavController().navigate(
                    R.id.action_homeFragment_to_detailFragment,
                    bundle
                )
            },
            onOpenLinkClick = { url ->
                ...
            }
        )
        ...
    }
}
```

The Logcat tab shows the output: "Detail button clicked for ID: 1" and "Detail screen loaded with item: ID=1, Title='Toy Story', Rating=8.3".

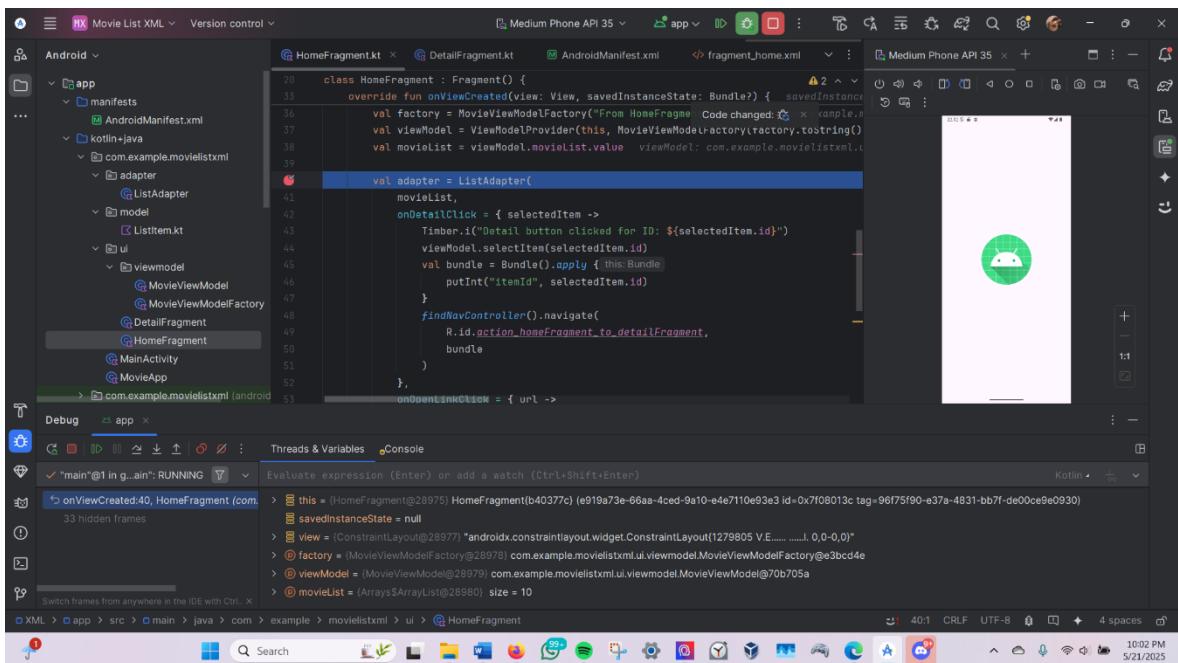
Gambar 3. Screenshot Penggunaan Timber ketika meng-klik Detail XML



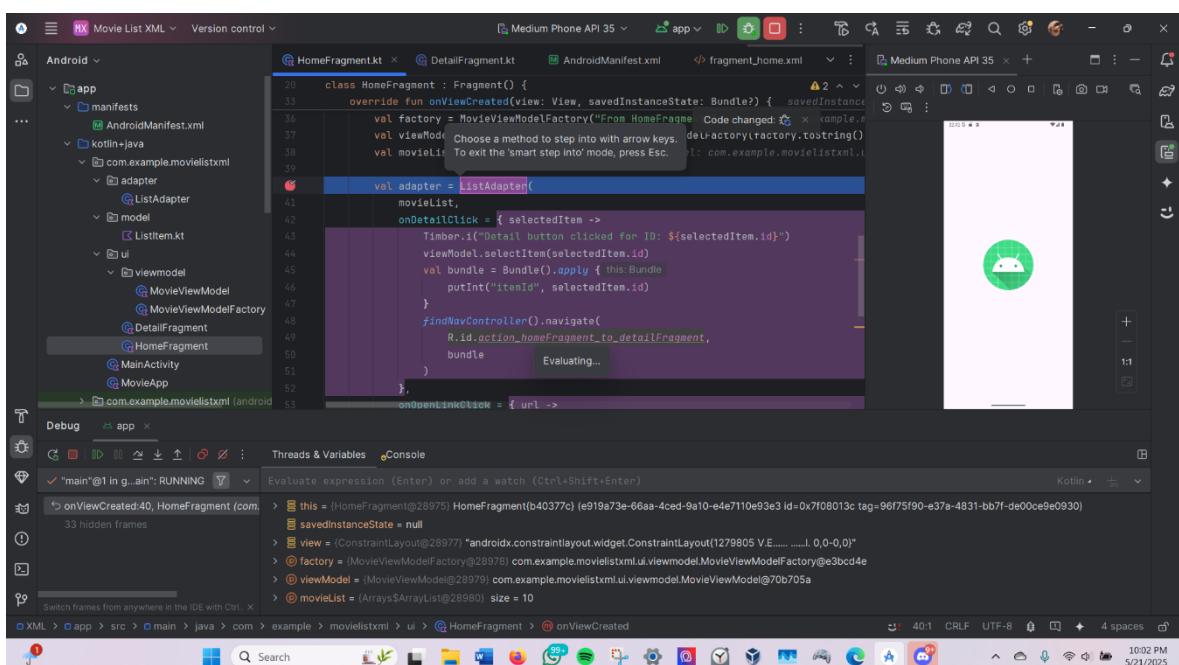
Gambar 4. Screenshot Penggunaan Timber ketika meng-klik tombol IMDB XML



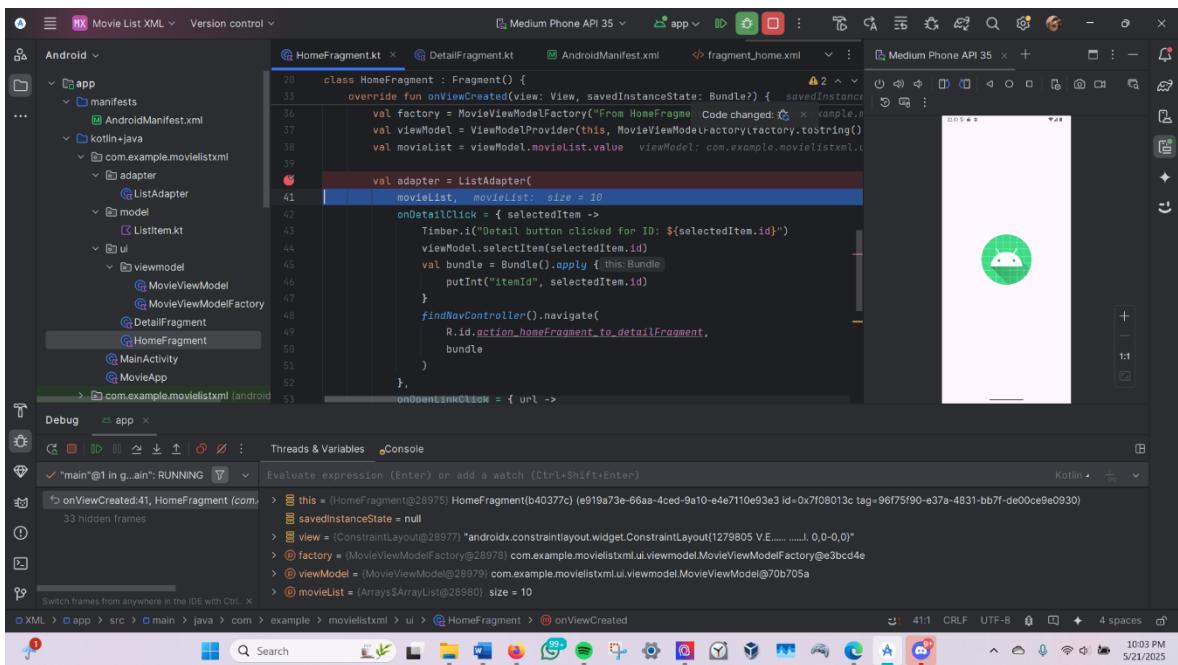
Gambar 5. Screenshot Penggunaan Timber dari List Data di Halaman Detail XML



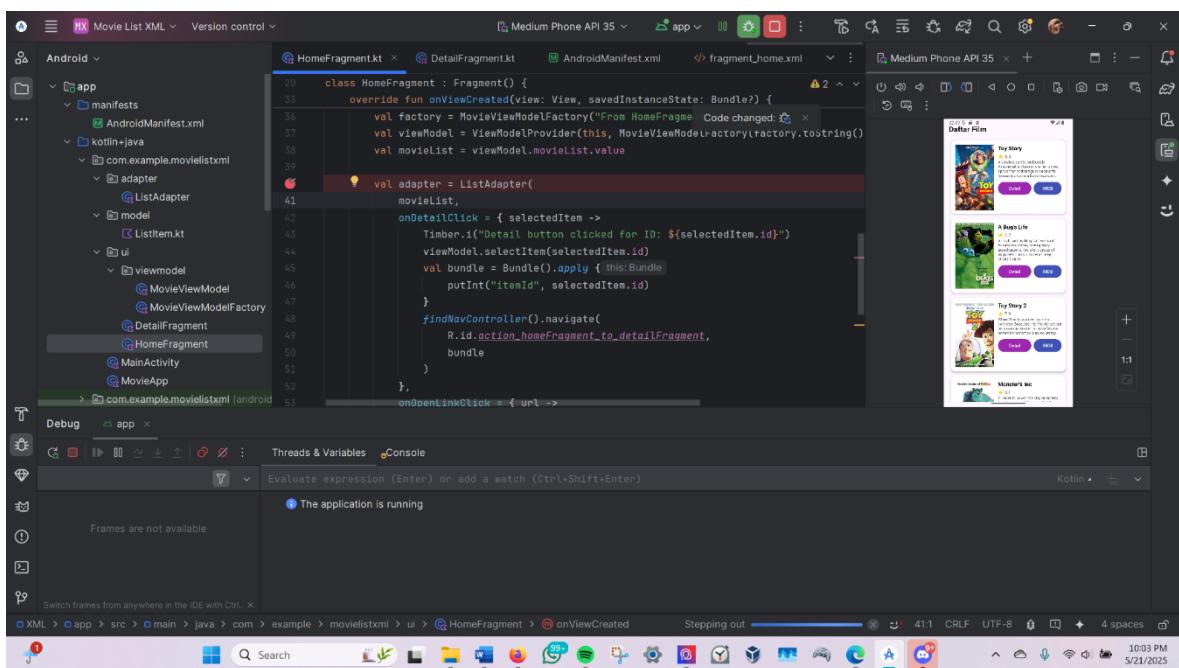
Gambar 6. Screenshot Penggunaan Debugger dengan Breakpoint di HomeFragment.kt XML



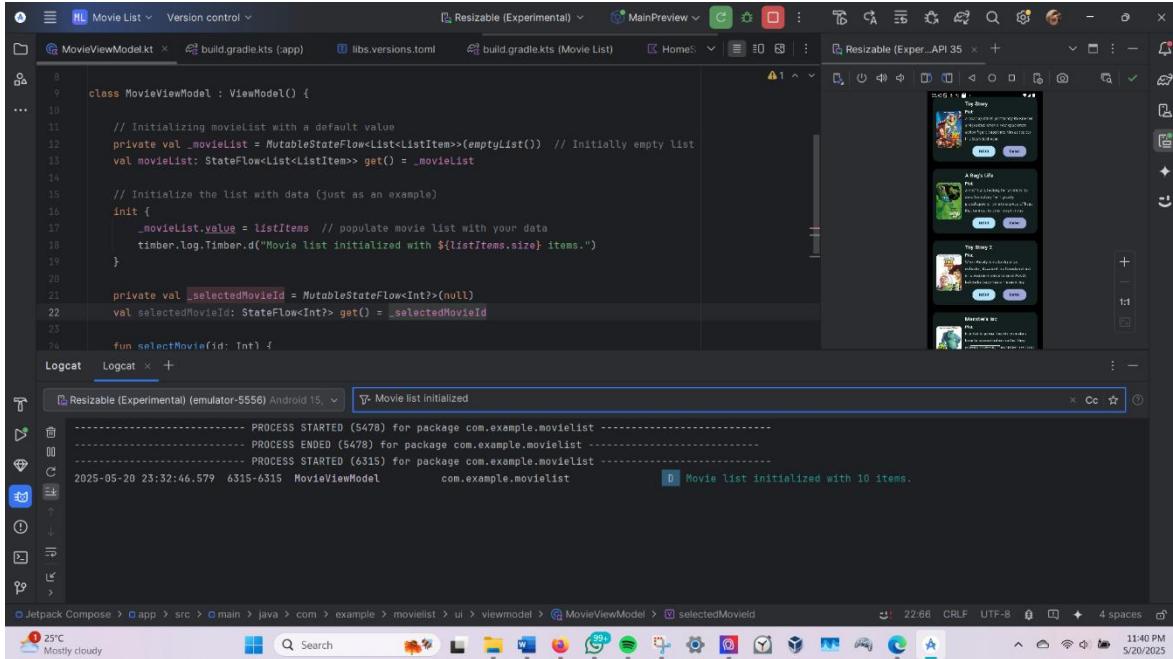
Gambar 7. Screenshot Penggunaan Debugger dengan Step Into XML



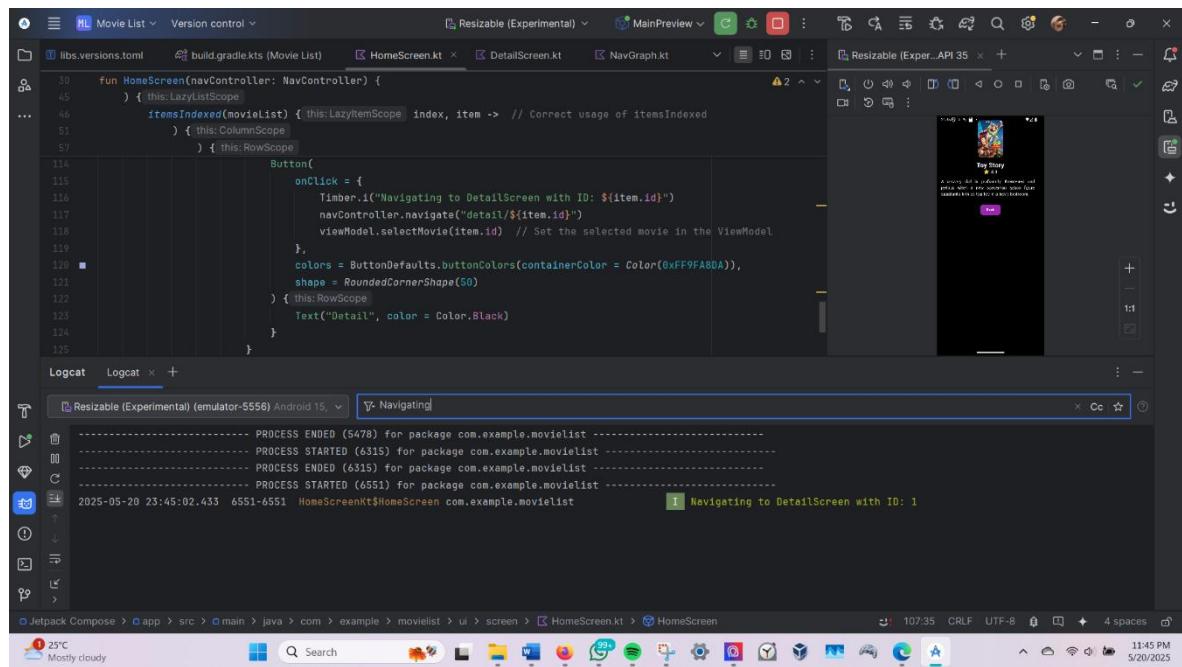
Gambar 8. Screenshot Penggunaan Debugger dengan Step Over XML



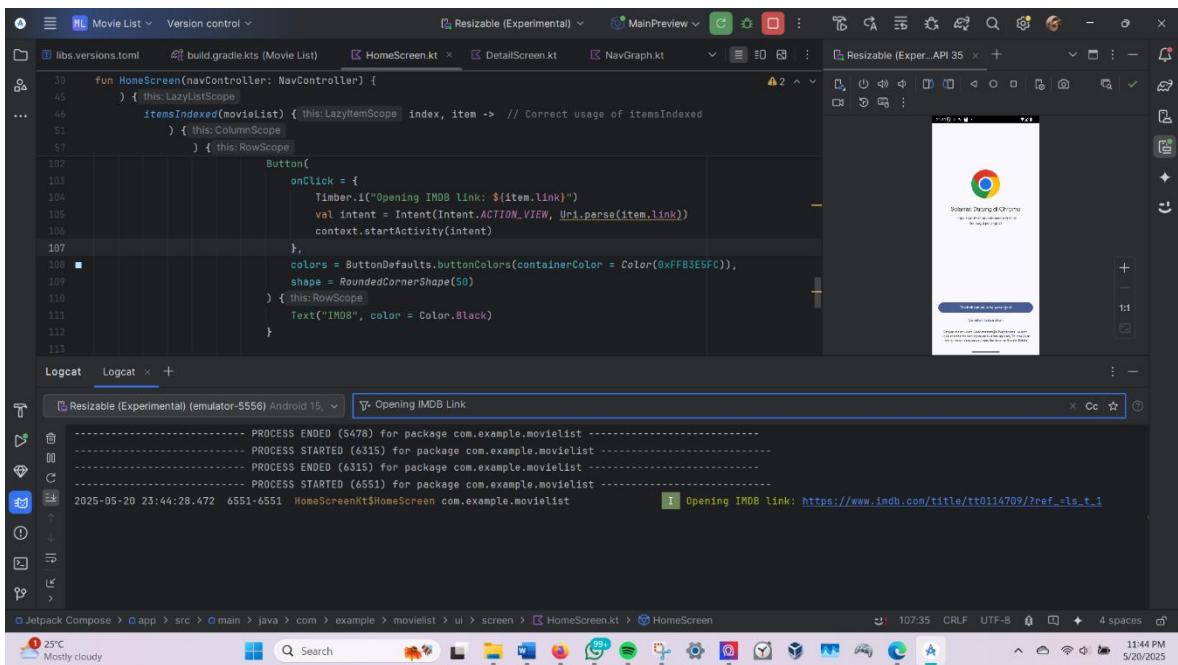
Gambar 9. Screenshot Penggunaan Debugger dengan Step Out XML



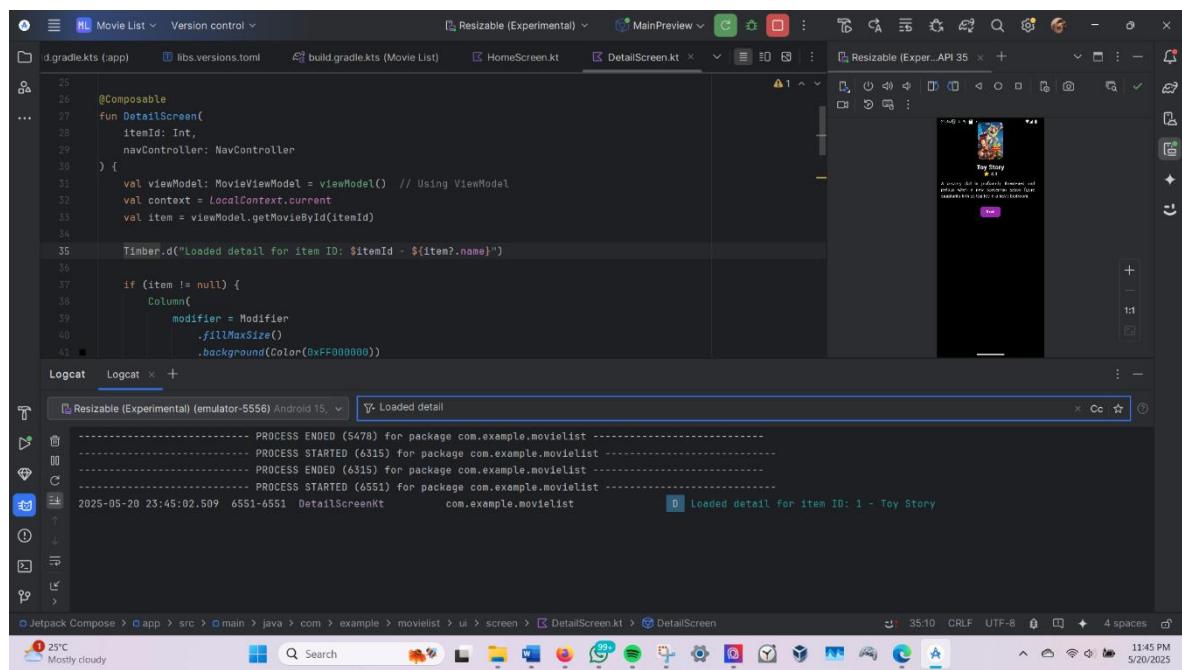
Gambar 10. Screenshot Penggunaan Timber ketika Data Masuk ke List XML



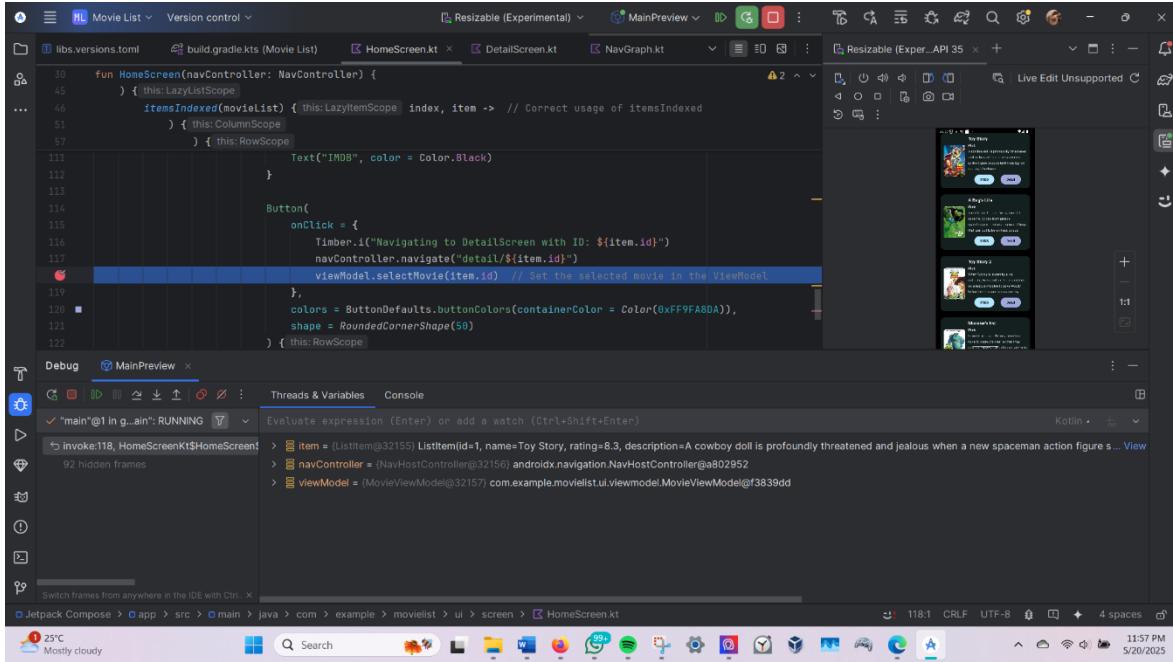
Gambar 11. Screenshot Penggunaan Timber ketika meng-klik Detail



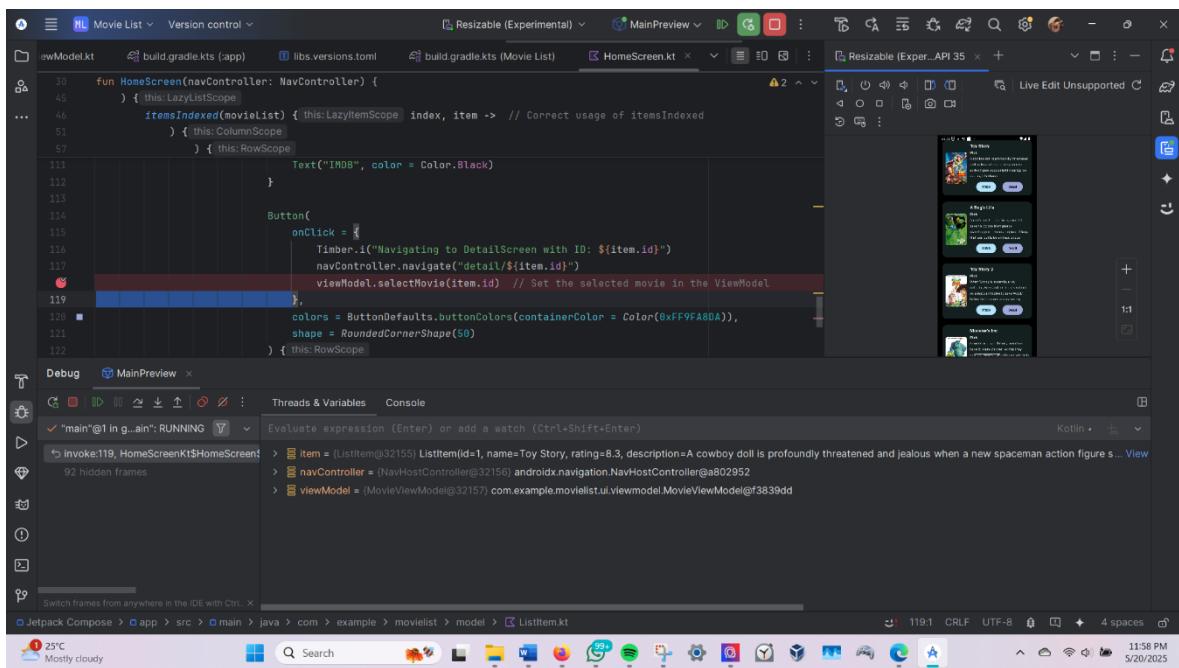
Gambar 12. Screenshot Penggunaan Timber ketika meng-klik tombol IMDB



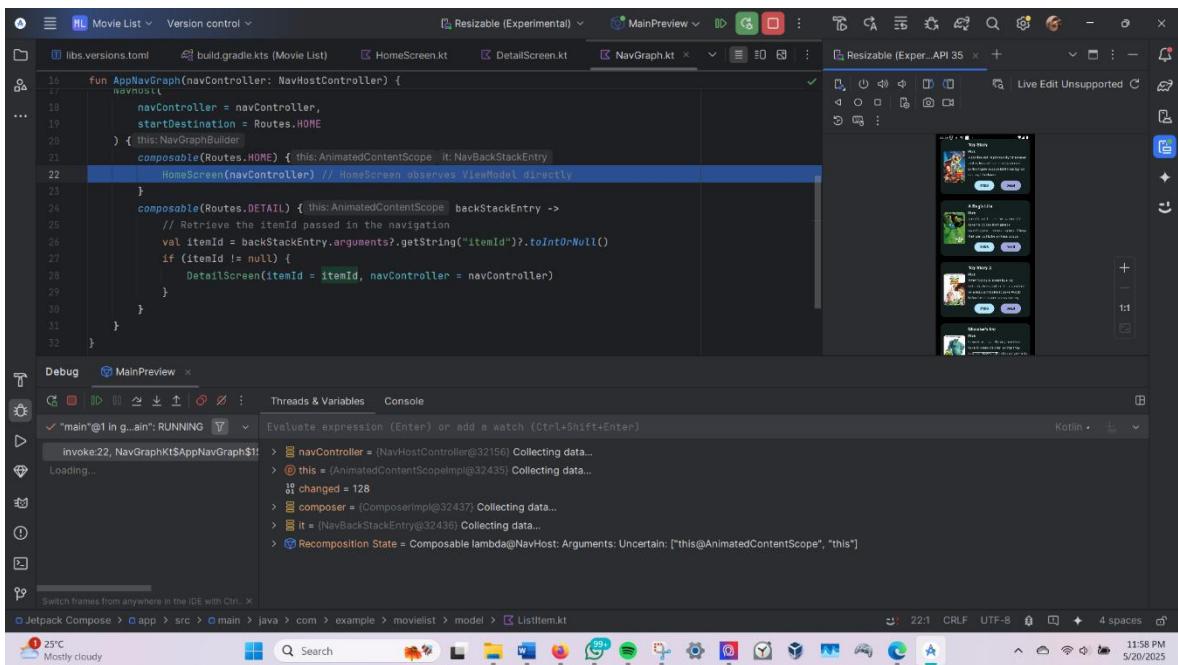
Gambar 13. Screenshot Penggunaan Timber dari List Data di Halaman Detail



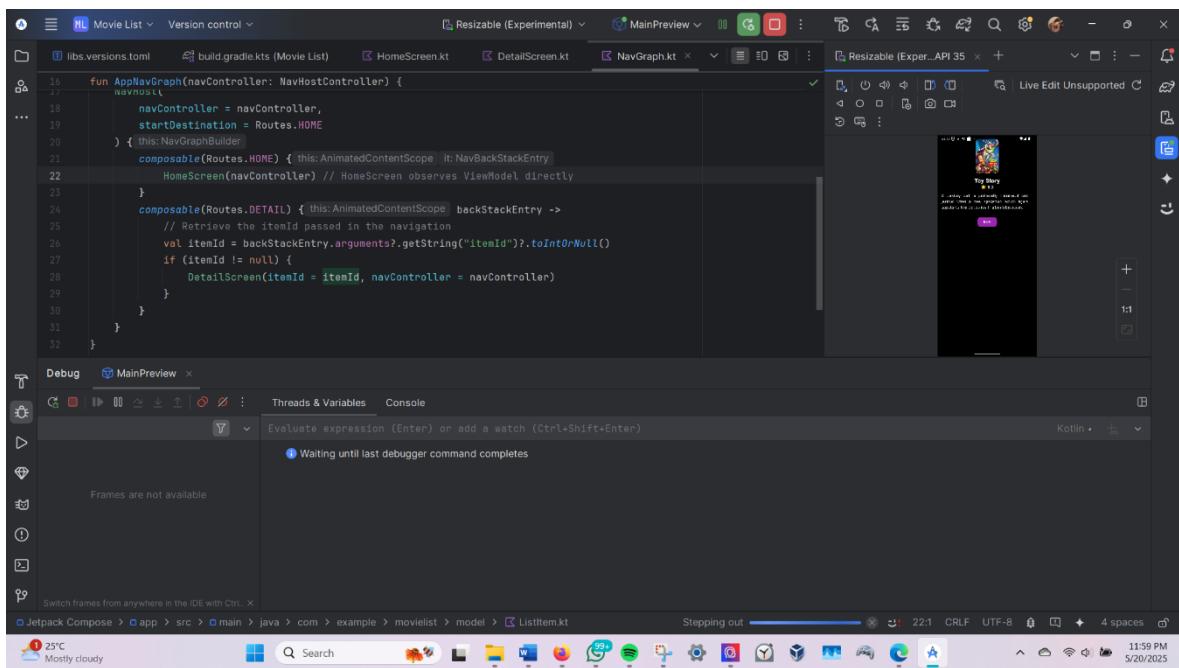
Gambar 14. Screenshot Penggunaan Debugger dengan Breakpoint di HomeFragment.kt



Gambar 15. Screenshot Penggunaan Debugger dengan Step Into



Gambar 16. Screenshot Penggunaan Debugger dengan Step Over



Gambar 17. Screenshot Penggunaan Debugger dengan Step Out

## C. Pembahasan

### 1. Pembahasan Praktikum:

## XML

### MainActivity.kt:

Pada baris [1], package com.example.movieListxml berfungsi untuk mendeklarasikan bahwa file ini termasuk dalam package bernama com.example.movieListxml, yang berguna untuk mengelompokkan file dalam proyek Android. Pada baris [2], import android.os.Bundle digunakan untuk mengimpor kelas Bundle, yaitu objek yang menyimpan data state dari aktivitas ketika terjadi perubahan konfigurasi atau saat aktivitas dijalankan. Pada baris [3], import androidx.appcompat.app.AppCompatActivity digunakan untuk mengimpor kelas AppCompatActivity, yang merupakan superclass untuk activity yang menggunakan fitur-fitur modern Android dari library AndroidX.

Pada baris [5], class MainActivity : AppCompatActivity() adalah deklarasi kelas MainActivity yang merupakan turunan dari AppCompatActivity, artinya kelas ini adalah sebuah activity utama dalam aplikasi dan mewarisi semua fungsionalitas dari AppCompatActivity. Pada baris [6], override fun onCreate(savedInstanceState: Bundle?) adalah deklarasi fungsi onCreate, yang merupakan metode yang dipanggil saat activity pertama kali dibuat. Kata override menunjukkan bahwa fungsi ini menimpa versi dari superclass-nya.

Pada baris [7], super.onCreate(savedInstanceState) berfungsi untuk memanggil implementasi onCreate dari superclass (AppCompatActivity), agar proses inisialisasi bawaan tetap berjalan normal. Pada baris [8], setContentView(R.layout.activity\_main) digunakan untuk menampilkan tampilan UI dari file XML bernama activity\_main.xml sebagai tampilan utama activity ini. Baris ini menghubungkan logika Kotlin dengan layout yang sudah dibuat di folder res/layout.

### activity\_main.xml :

Pada baris [1], <?xml version="1.0" encoding="utf-8"?> digunakan untuk mendeklarasikan bahwa file ini adalah file XML dengan versi 1.0 dan menggunakan encoding UTF-8. Pada baris [2], <androidx.constraintlayout.widget.ConstraintLayout> berfungsi sebagai root layout yang menggunakan ConstraintLayout, yaitu jenis layout fleksibel yang memungkinkan penempatan elemen-elemen UI secara terstruktur menggunakan constraints. Pada baris [3] dan [4], atribut xmlns:android dan xmlns:app mendefinisikan namespace untuk atribut standar Android dan atribut khusus dari AndroidX.

Pada baris [5] dan [6], android:layout\_width="match\_parent" dan android:layout\_height="match\_parent" menyatakan bahwa layout ini akan mengisi seluruh lebar dan tinggi layar. Pada baris [8], <androidx.fragment.app.FragmentContainerView> digunakan untuk menampung fragment dalam tampilan activity.

Pada baris [9], `android:id="@+id/nav_host_fragment"` memberikan ID unik pada FragmentContainerView ini. Pada baris [10], `android:name="androidx.navigation.fragment.NavHostFragment"` menunjukkan bahwa Fragment ini akan berfungsi sebagai NavHost untuk sistem navigasi Jetpack. Pada baris [11–14], layout fragment ini diatur menggunakan constraints agar menempel ke semua sisi parent layout (atas, bawah, kiri, kanan). Pada baris [15], `app:navGraph="@navigation/nav_graph"` menunjukkan file navigasi XML yang digunakan sebagai alur navigasi. Pada baris [16], `app:defaultNavHost="true"` menandakan bahwa fragment ini akan menerima aksi navigasi dari tombol kembali (back button) secara default.

Pada baris [17], `</androidx.fragment.app.FragmentContainerView>` menutup elemen fragment container. Pada baris [18], `</androidx.constraintlayout.widget.ConstraintLayout>` berfungsi untuk menutup tag ConstraintLayout dan menandai akhir dari struktur layout ini.

### **ListAdapter.kt**

Pada baris [1], `package com.example.movieListxml.adapter` mendeklarasikan bahwa file ini berada dalam package `adapter` di proyek `movieListxml`. Pada baris [2–4], beberapa library Android di-*import*, termasuk `LayoutInflater`, `ViewGroup`, `RecyclerView`, serta view binding dan model yang digunakan dalam adapter ini. Pada baris [6], `class ListAdapter` mendefinisikan sebuah adapter khusus untuk `RecyclerView`, yang menerima tiga parameter: `list` berupa data daftar `ListItem`, `onDetailClick` sebagai fungsi callback saat tombol detail ditekan, dan `onOpenLinkClick` sebagai callback saat tombol IMDb ditekan.

Pada baris [10], inner class `ViewHolder` mendefinisikan `ViewHolder` internal yang bertanggung jawab untuk mengelola tampilan item individual menggunakan binding `ItemRowBinding`. Pada baris [12], fungsi `bind` digunakan untuk mengisi data dari objek `ListItem` ke dalam elemen-elemen UI seperti `textviewName`, `textviewRating`, dan `textviewDescription`. Pada baris [16], `getIdentifier` digunakan untuk mengambil ID resource gambar berdasarkan nama yang diberikan oleh `item.imageResName`. Gambar kemudian ditampilkan melalui `imageView`.

Pada baris [21–25], tombol `Detail` dan `IMDb` dihubungkan dengan fungsi callback yang akan dijalankan saat masing-masing tombol ditekan. Pada baris [28], fungsi `onCreateViewHolder` digunakan untuk membuat `ViewHolder` baru dari layout `ItemRowBinding`. Pada baris [34], `onBindViewHolder` bertugas menghubungkan data `ListItem` dengan `ViewHolder` berdasarkan posisi item. Pada baris [37], `getItemCount` mengembalikan jumlah total item dalam daftar, yang akan ditampilkan di `RecyclerView`.

### **ListItem.kt**

Pada baris [1], `package com.example.movieListxml.model` berfungsi untuk menetapkan bahwa file ini berada dalam package `model`, yang merupakan bagian dari struktur proyek

aplikasi Android. Pada baris [3], `data class ListItem` berfungsi untuk mendeklarasikan sebuah kelas data bernama `ListItem` yang digunakan untuk merepresentasikan satu item film dalam bentuk objek. Pada baris [4] hingga [9], terdapat enam properti yang didefinisikan dalam kelas tersebut, yaitu `id` (tipe data Int), `name` (nama film, String), `rating` (nilai film, Double), `description` (deskripsi film, String), `imageResName` (nama file gambar di folder drawable, tanpa ekstensi, String), dan `link` (tautan IMDb ke film, String).

Pada baris [12], `val listItems = listOf(` berfungsi untuk membuat sebuah list berisi beberapa objek `ListItem` secara hardcoded atau ditulis langsung. Pada baris [13] hingga [66], terdapat sepuluh entri `ListItem` yang masing-masing mewakili sebuah film produksi Pixar. Setiap item menyimpan data lengkap seperti ID, judul film (misalnya "Toy Story"), rating (contoh 8.3), deskripsi singkat alur cerita, nama file gambar seperti `pixar1`, dan link IMDb resmi. Data ini biasanya digunakan dalam RecyclerView untuk ditampilkan sebagai daftar film dalam aplikasi Android.

## **HomeFragment.kt**

Pada baris [1], `package com.example.movieListxml.ui` berfungsi untuk menentukan bahwa file ini merupakan bagian dari package `ui`, yang biasanya digunakan untuk menyimpan elemen-elemen tampilan antarmuka pengguna. Pada baris [3–11], dilakukan import berbagai class penting dari Android dan project internal, seperti `Intent`, `Fragment`, `LinearLayoutManager`, `ListAdapter`, dan `FragmentHomeBinding`, yang akan digunakan dalam logika fragment ini.

Pada baris [13], `class HomeFragment : Fragment()` mendefinisikan sebuah kelas `HomeFragment` yang mewarisi dari kelas `Fragment`, menunjukkan bahwa file ini berisi tampilan yang akan menjadi bagian dari UI. Pada baris [15], `_binding` dideklarasikan sebagai variabel nullable yang akan digunakan untuk mengakses elemen UI dengan View Binding. Pada baris [16], properti `binding` disiapkan untuk mengambil nilai `_binding` secara non-null, dengan tanda `!!`.

Pada baris [18–22], fungsi `onCreateView` digunakan untuk menginisialisasi View dari Fragment, di mana `FragmentHomeBinding.inflate()` dipakai untuk menghubungkan layout XML ke Kotlin dengan teknik View Binding, dan hasilnya dikembalikan sebagai tampilan root. Pada baris [23], `onViewCreated()` dijalankan setelah tampilan dibuat, di mana logika utama fragment dituliskan. Di dalamnya, `MovieViewModelFactory` dibuat dengan parameter string dan digunakan untuk menghasilkan objek `MovieViewModel` melalui `ViewModelProvider`. Kemudian data `movieList` diambil dari `viewModel`.

Pada baris [43–44], `binding.recyclerView.layoutManager` diset menggunakan `LinearLayoutManager` untuk membuat daftar tampil secara vertikal, dan adapter-nya diatur ke adapter yang telah dibuat sebelumnya. Pada baris [47–49], `onDestroyView` digunakan untuk membersihkan objek binding dengan menyetelnya ke null agar tidak terjadi memory leak setelah fragment dihancurkan.

## **DetailFragment.kt**

Pada baris [1], package com.example.movieListxml.ui digunakan untuk mendefinisikan bahwa file ini termasuk dalam package ui, yaitu bagian antarmuka pengguna dari aplikasi. Pada baris [3–7], dilakukan import class-class penting seperti Bundle, Fragment, FragmentDetailBinding, dan listItems yang diperlukan dalam fragment ini.

Pada baris [9], class DetailFragment : Fragment() mendefinisikan sebuah kelas bernama DetailFragment yang mewarisi dari Fragment, artinya class ini akan digunakan untuk menampilkan tampilan detail dari sebuah item. Pada baris [11], \_binding dideklarasikan sebagai variabel nullable yang nantinya akan diisi dengan objek binding dari layout. Pada baris [12], binding digunakan untuk mengakses \_binding secara non-null dengan menggunakan operator !!.

Pada baris [14–18], fungsi onCreateView digunakan untuk membuat dan mengembalikan tampilan (View) dari fragment. Di dalamnya, layout FragmentDetailBinding di-inflate, lalu hasilnya dikembalikan sebagai root view. Pada baris [20–36], fungsi onViewCreated akan dijalankan setelah tampilan selesai dibuat. Pada baris [23], onViewCreated() dijalankan setelah tampilan dibuat, di mana logika utama fragment ditulis. Di dalamnya, MovieViewModelFactory dibuat dengan parameter string dan digunakan untuk menghasilkan objek MovieViewModel melalui ViewModelProvider. Kemudian data movieList diambil dari viewModel. Pada baris [25–27], data dari item yang ditemukan ditampilkan ke elemen UI seperti judul film, rating, dan deskripsi menggunakan binding. Pada baris [29–32], dilakukan pencarian resource ID dari gambar berdasarkan imageResName, kemudian gambar tersebut ditampilkan ke dalam imageView. Pada baris [34–36], onDestroyView digunakan untuk menghapus nilai \_binding agar tidak terjadi memory leak saat fragment dihancurkan.

### **Fragment\_detail.xml**

Pada baris [1], <?xml version="1.0" encoding="utf-8"?> digunakan untuk mendefinisikan deklarasi XML dan encoding file, yang merupakan bagian standar dari file layout Android. Pada baris [2], <ScrollView> berfungsi sebagai kontainer scrollable yang memungkinkan isi di dalamnya untuk digulir secara vertikal, terutama jika konten melebihi tinggi layar. Atribut layout\_width dan layout\_height diset ke match\_parent, artinya ScrollView akan mengisi seluruh lebar dan tinggi layar, dengan padding 16dp dan latar belakang berwarna putih.

Pada baris [4], <LinearLayout> digunakan sebagai wadah utama di dalam ScrollView. Layout ini memiliki orientasi vertikal sehingga elemen-elemen UI di dalamnya akan ditampilkan dari atas ke bawah. Atribut gravity="center\_horizontal" memastikan seluruh elemen ditampilkan di tengah secara horizontal.

Pada baris [10], <ImageView> digunakan untuk menampilkan gambar, dalam hal ini biasanya berupa poster film. ID-nya adalah imageView dan memiliki margin atas dan bawah 16dp. Atribut contentDescription memberikan deskripsi untuk aksesibilitas, dan scaleType="centerCrop" memastikan gambar memenuhi area dengan proporsi yang tetap.

Pada baris [18], <TextView> dengan ID `textViewTitle` digunakan untuk menampilkan judul film. Tekstanya diset ke "Movie Title", dengan ukuran font 24sp, teks ditebalkan, berwarna hitam, dan memiliki margin bawah 8dp.

Pada baris [25], <TextView> dengan ID `textViewRating` digunakan untuk menampilkan rating film, seperti "★ 8.4", dengan ukuran font 18sp, warna abu-abu, dan margin bawah 16dp.

Pada baris [32], <TextView> dengan ID `textViewDescription` digunakan untuk menampilkan deskripsi lengkap film. Lebarnya diset `match_parent` agar memenuhi area layar, tingginya `wrap_content`, dan teks berwarna abu gelap dengan ukuran 16sp.

### **Fragment\_home.xml**

Pada baris [1], `<?xml version="1.0" encoding="utf-8"?>` berfungsi untuk menyatakan bahwa file ini adalah dokumen XML dengan encoding UTF-8. Pada baris [2], tag `<androidx.constraintlayout.widget.ConstraintLayout>` membuka layout utama dengan jenis ConstraintLayout, yang memungkinkan penempatan elemen UI dengan cara menghubungkannya ke elemen lain atau ke parent-nya. Atribut `layout_width` dan `layout_height` diset ke `match_parent` agar layout mengisi seluruh layar, dengan padding sebesar 16dp.

Pada baris [7], <TextView> digunakan untuk menampilkan teks "Daftar Film" sebagai judul halaman. Ukuran teks diset ke 24sp, gaya huruf ditebalkan, dan warna teks hitam. TextView ini diposisikan di bagian atas layar menggunakan `app:layout_constraintTop_toTopOf="parent"` dan disejajarkan ke kiri dengan `app:layout_constraintStart_toStartOf="parent"`.

Pada baris [17], <`androidx.recyclerview.widget.RecyclerView`> digunakan untuk menampilkan daftar film dalam bentuk scrollable list. Atribut `layout_width` dan `layout_height` diset ke 0dp karena ukuran akan ditentukan oleh constraint. RecyclerView ini diatur menggunakan LinearLayoutManager agar menampilkan daftar secara vertikal. Posisi RecyclerView ditetapkan di bawah TextView judul menggunakan `layout_constraintTop_toBottomOf="@+id/textViewTitle"`, serta menempel ke sisi kiri, kanan, dan bawah parent layout.

Pada baris [26], tag `</androidx.constraintlayout.widget.ConstraintLayout>` digunakan untuk menutup layout ConstraintLayout dan menandakan akhir dari struktur layout.

### **Item\_row.xml**

Pada baris [1], `<?xml version="1.0" encoding="utf-8"?>` berfungsi untuk menyatakan bahwa file ini adalah dokumen XML dengan encoding UTF-8. Pada baris [2], tag `<androidx.cardview.widget.CardView>` digunakan untuk membuat wadah berbentuk kartu dengan sudut membulat dan bayangan (elevation). Atribut `cardCornerRadius`

digunakan untuk mengatur radius sudut sebesar 12dp, dan `cardElevation` memberi efek mengambang dengan bayangan sebesar 4dp.

Pada baris [8], `<LinearLayout>` digunakan untuk mengatur isi kartu secara horizontal dengan `orientation="horizontal"` dan `gravity="center_vertical"` agar elemen-elemen di dalamnya sejajar secara vertikal di tengah. Layout ini diberi padding sebesar 12dp agar kontennya tidak mepet ke tepi.

Pada baris [14], `<ImageView>` digunakan untuk menampilkan gambar film dengan ukuran 124dp x 208dp, dan diberi `scaleType="centerCrop"` agar gambar memenuhi ruang yang tersedia secara proporsional. Atribut `layout_marginEnd="12dp"` memberi jarak antara gambar dan elemen selanjutnya.

Pada baris [22], `LinearLayout` baru dibuka dengan orientasi vertikal untuk menampung teks-teks informasi film. Lebarnya diset 236dp, tingginya 209dp, dan diberi `layout_weight="1"` agar fleksibel jika ruang memungkinkan.

Pada baris [28], `<TextView>` digunakan untuk menampilkan nama film dengan teks berukuran 18sp dan huruf tebal berwarna hitam. Di bawahnya, pada baris [34], `<TextView>` kedua menampilkan rating film dengan margin atas 4dp dan warna abu-abu.

Pada baris [40], `<TextView>` ketiga digunakan untuk menampilkan deskripsi film dengan ukuran teks kecil 12sp.

Pada baris [46], `LinearLayout` baru dibuka untuk menampilkan dua tombol secara horizontal. Pada baris [48], tombol pertama `<Button>` digunakan untuk aksi "Lihat Detail", dengan latar ungu (#9C27B0) dan teks putih. Pada baris [54], tombol kedua `<Button>` digunakan untuk menuju ke IMDB, dengan warna latar biru (#3F51B5) dan teks putih, serta diberi margin kiri 8dp agar tidak terlalu rapat dengan tombol pertama.

Pada baris [61], tag `</LinearLayout>` digunakan untuk menutup struktur layout tombol, dan diikuti oleh penutupan `LinearLayout` konten serta penutupan tag utama `<androidx.cardview.widget.CardView>` di baris [66], menandai akhir dari struktur tampilan kartu film.

## Nav\_graph.xml

Pada baris [1], `<?xml version="1.0" encoding="utf-8"?>` digunakan untuk mendefinisikan bahwa file ini adalah dokumen XML dengan encoding UTF-8. Pada baris [2], tag `<navigation>` berfungsi sebagai container utama untuk komponen Navigation di Android, dengan atribut `app:startDestination` yang menunjukkan bahwa tampilan awal aplikasi akan diarahkan ke `homeFragment`.

Pada baris [6], `<fragment>` digunakan untuk mendeklarasikan fragment tujuan bernama `homeFragment` yang merujuk ke kelas `HomeFragment` di package aplikasi. Pada baris [7], atribut `android:label="Home"` digunakan untuk memberi label tampilan fragment tersebut.

Pada baris [8], <action> digunakan untuk mendefinisikan transisi dari homeFragment menuju detailFragment, dengan ID action\_homeFragment\_to\_detailFragment. Atribut app:destination menunjukkan tujuan navigasi, sedangkan app:enterAnim, app:exitAnim, app:popEnterAnim, dan app:popExitAnim digunakan untuk mengatur animasi saat berpindah fragment.

Pada baris [15], fragment kedua dengan ID detailFragment didefinisikan, yang merujuk pada kelas DetailFragment dan diberi label "Detail". Pada baris [17], <argument> digunakan untuk mendeklarasikan bahwa fragment ini menerima argumen bernama itemId dengan tipe data integer, yang dapat digunakan untuk mengirim data dari fragment sebelumnya.

Pada baris [19], tag </navigation> menutup struktur navigasi, menandakan akhir dari definisi Graph Navigation di aplikasi.

## MovieViewModel.kt

Pada baris [1], package com.example.movieListxml.ui.viewmodel digunakan untuk mendeklarasikan nama paket tempat class MovieViewModel berada, yang membantu mengorganisasi kode dalam proyek Android. Pada baris [2] dan [3], terdapat import androidx.lifecycle.ViewModel dan androidx.lifecycle.ViewModelProvider, yang digunakan untuk mengelola dan menyediakan instance ViewModel dalam arsitektur MVVM Android. Pada baris [4] dan [5], diimpor class ListItem dan listItems dari package model, yang merepresentasikan data film yang akan ditampilkan dalam aplikasi. Pada baris [6], impor kotlinx.coroutines.flow.MutableStateFlow dan StateFlow memungkinkan ViewModel untuk mengelola dan mengamati perubahan data secara reaktif. Pada baris [7], timber.log.Timber diimpor sebagai alat logging yang efisien untuk debugging.

Pada baris [9], class MovieViewModel(private val sourceName: String) : ViewModel() berfungsi untuk mendefinisikan class ViewModel dengan parameter konstruktor sourceName, dan mewarisi class ViewModel dari Android Jetpack, yang membuat class ini bertanggung jawab atas logika dan data tampilan. Pada baris [11], terdapat deklarasi private val \_movieList = MutableStateFlow(listItems) yang membuat variabel privat \_movieList untuk menyimpan daftar film secara reaktif. Pada baris [12], val movieList: StateFlow<List<ListItem>> get() = \_movieList adalah getter publik agar UI bisa membaca daftar film tanpa bisa mengubahnya.

Pada baris [14], terdapat blok init yang merupakan initializer otomatis ketika ViewModel dibuat. Di dalamnya, pada baris [15], fungsi Timber.i() digunakan untuk mencatat log informasi bahwa daftar film telah diinisialisasi, termasuk jumlah item dan nama sumber datanya.

Pada baris [18], private val \_selectedItemId = MutableStateFlow<Int?>(null) digunakan untuk menyimpan ID item yang sedang dipilih oleh pengguna, dengan nilai awal null menandakan belum ada yang dipilih. Pada baris [19], val selectedItemId:

`StateFlow<Int?> get() = _selectedItemId` adalah properti publik yang hanya bisa dibaca untuk mengamati ID item terpilih.

Pada baris [21], `fun selectItem(id: Int)` mendefinisikan fungsi yang memungkinkan pengguna memilih sebuah film berdasarkan ID-nya. Pada baris [22], `_selectedItemId.value = id` menetapkan ID film yang dipilih ke dalam `MutableStateFlow`, sehingga UI bisa merespons perubahan ini.

Pada baris [25], fungsi `getMovieById(id: Int): ListItem?` digunakan untuk mencari data film berdasarkan ID. Pada baris [26], `return _movieList.value.find { it.id == id }` mencari dan mengembalikan objek `ListItem` yang memiliki ID yang sesuai. Jika tidak ditemukan, fungsi ini akan mengembalikan `null`.

## MovieViewModelFactory.kt

Pada baris [1], `package com.example.movieListxml.ui.viewmodel` digunakan untuk mendefinisikan package tempat class `MovieViewModelFactory` berada, yang merupakan bagian dari struktur proyek Android agar file-file tertata rapi sesuai fungsinya. Pada baris [2] dan [3], terdapat `import androidx.lifecycle.ViewModel` dan `ViewModelProvider`, yang diperlukan karena class ini berfungsi untuk menyediakan instance `ViewModel`, sesuai dengan pola arsitektur MVVM pada Android.

Pada baris [5], class `MovieViewModelFactory` dideklarasikan sebagai class turunan dari `ViewModelProvider.Factory`, yang artinya class ini bertanggung jawab untuk membuat instance `ViewModel` dengan parameter tertentu. Parameter `sourceName` bertipe `String` dan diberikan melalui konstruktor, digunakan nantinya oleh `MovieViewModel`.

Pada baris [6], override function `create()` didefinisikan untuk menggantikan implementasi bawaan dari `ViewModelProvider.Factory`. Fungsi ini menerima parameter `modelClass` yang menunjukkan jenis `ViewModel` yang ingin dibuat.

Pada baris [7], terdapat percabangan `if` yang memeriksa apakah `modelClass` merupakan subclass dari `MovieViewModel`. Jika ya, maka pada baris [8], `return MovieViewModel(sourceName) as T` akan membuat dan mengembalikan instance `MovieViewModel` dengan parameter `sourceName` yang telah diberikan. Ini memungkinkan `ViewModel` menerima parameter, sesuatu yang tidak bisa dilakukan secara langsung dengan `ViewModel` default.

Jika `modelClass` tidak sesuai, maka pada baris [10], baris `throw IllegalArgumentException("Unknown ViewModel class")` akan memunculkan error, untuk memastikan hanya class `MovieViewModel` yang bisa dibuat menggunakan factory ini. Dengan demikian, `MovieViewModelFactory` berguna saat kita perlu menginisialisasi `ViewModel` yang memiliki konstruktor dengan argumen, seperti `sourceName` pada contoh ini.

## **MovieApp.kt**

Pada baris [1], package com.example.movieListxml berfungsi untuk mendefinisikan lokasi package dari class MovieApp, agar sesuai dengan struktur proyek Android. Pada baris [2] dan [3], import android.app.Application dan import timber.log.Timber digunakan untuk mengimpor class Application yang menjadi superclass dari MovieApp, serta library Timber yang digunakan untuk logging.

Pada baris [5], class MovieApp : Application() mendefinisikan class MovieApp sebagai turunan dari class Application, yang berarti kode di dalamnya akan dijalankan saat aplikasi pertama kali dibuka. Pada baris [6], override fun onCreate() digunakan untuk menimpa method onCreate() bawaan dari class Application.

Pada baris [7], super.onCreate() dipanggil agar inisialisasi bawaan dari Application tetap berjalan. Pada baris [9], terdapat pengecekan if (BuildConfig.DEBUG) yang berfungsi untuk memastikan bahwa blok kode di dalamnya hanya dijalankan saat mode debug, bukan pada rilis aplikasi.

Pada baris [10], Timber.plant(Timber.DebugTree()) berfungsi untuk menginisialisasi Timber dengan DebugTree, sehingga kita bisa menampilkan log saat proses debugging. Pada baris [11], Timber.i("Timber initialized in MovieApp") mencetak log informasi bahwa Timber berhasil diinisialisasi.

## **JETPACK COMPOSE**

### **MainActivity.kt :**

Pada baris [1], package com.example.movieList digunakan untuk mendefinisikan bahwa file ini berada dalam package com.example.movieList. Pada baris [2–8], dilakukan import berbagai library yang dibutuhkan seperti ComponentActivity, setContent, elemen dari MaterialTheme, Surface, hingga Navigation.

Pada baris [10], kelas MainActivity didefinisikan dan mewarisi ComponentActivity yang merupakan komponen utama dalam aplikasi berbasis Jetpack Compose. Pada baris [11], fungsi onCreate() dioverride untuk menyiapkan tampilan awal aplikasi. Pada baris [12], super.onCreate(savedInstanceState) memanggil implementasi dari superclass. Pada baris [13], setContent digunakan untuk menentukan isi UI dengan menggunakan Jetpack Compose.

Pada baris [14], MovieListTheme digunakan untuk menerapkan tema aplikasi. Pada baris [15], Surface digunakan sebagai container dasar dengan warna latar belakang dari MaterialTheme. Pada baris [16], rememberNavController() dipanggil untuk membuat dan menyimpan instance NavController yang digunakan untuk navigasi. Pada baris [17], AppNavGraph(navController = navController) memanggil fungsi navigasi utama aplikasi yang mendefinisikan alur navigasi antar layar.

Pada baris [21], anotasi `@Composable` menandai fungsi `MainPreview` sebagai fungsi komposabel untuk menampilkan preview UI. Pada baris [22], anotasi `@Preview` digunakan agar fungsi ini dapat dipreview di Android Studio dengan tampilan UI dan sistem. Pada baris [23–26], isi fungsi preview sama dengan `setContent` di `MainActivity`, yaitu menggunakan tema, membuat `navController`, dan menjalankan navigasi melalui `AppNavGraph`.

### **ListItem.kt**

Pada baris [1], `package com.example.movieList.model` digunakan untuk menentukan bahwa file ini berada dalam package `model` dari proyek `com.example.movieList`. Pada baris [3], `data class ListItem` digunakan untuk membuat kelas data `ListItem` yang secara otomatis menyediakan fungsi seperti `toString()`, `equals()`, dan `copy()` untuk menyimpan data terkait item film. Pada baris [4–9], didefinisikan enam properti utama dari setiap `ListItem`, yaitu `id` (nomor unik), `name` (judul film), `rating` (nilai rating film), `description` (ringkasan cerita), `imageResName` (nama file gambar di resource drawable), dan `link` (tautan ke IMDb).

Pada baris [11], `val listItems = listOf(...)` digunakan untuk membuat sebuah list berisi banyak objek `ListItem`. Setiap objek `ListItem` dibuat menggunakan konstruktor dengan nilai-nilai spesifik untuk masing-masing properti. Pada baris [12–20], didefinisikan item pertama yaitu "Toy Story", lengkap dengan deskripsi dan tautan IMDb-nya. Pada baris [21–29], item kedua "A Bug's Life" ditambahkan. Baris-baris berikutnya mengandung item ketiga hingga kesepuluh, seperti "Toy Story 2", "Monster's Inc", "Finding Nemo", dan seterusnya.

Setiap `ListItem` memiliki struktur data yang sama, yaitu ID numerik, judul, rating numerik, deskripsi dalam bentuk string panjang, nama gambar, dan URL. Kode ini digunakan sebagai data dummy atau sumber data untuk ditampilkan dalam aplikasi film list menggunakan Jetpack Compose atau navigasi lainnya.

### **NavGraph.kt**

Pada baris [1], `package com.example.movieList.navigation` digunakan untuk menunjukkan bahwa file ini berada dalam package `navigation` dari proyek `com.example.movieList`. Pada baris [3–6], dilakukan impor terhadap beberapa komponen Compose dan Navigation yang digunakan untuk membuat navigasi antar layar, seperti `Composable`, `NavController`, `NavHost`, dan `composable`. Pada baris [7–8], diimporkan dua composable yaitu `DetailScreen` dan `HomeScreen` yang nantinya akan digunakan dalam navigasi aplikasi.

Pada baris [10], dideklarasikan object `Routes` yang berisi konstanta `HOME` dan `DETAIL`, yang masing-masing mewakili rute untuk tampilan beranda ("home") dan tampilan detail ("detail/{itemId}"). Baris [14], `@Composable fun AppNavGraph` adalah fungsi utama yang mendefinisikan struktur navigasi aplikasi, dengan parameter `navController` bertipe `NavController`. Pada baris [15–17], `NavHost` digunakan untuk menyusun navigasi, dengan `startDestination` mengarah ke `Routes.HOME`.

Pada baris [18–20], `composable(Routes.HOME)` digunakan untuk menampilkan `HomeScreen` ketika rute adalah "home", dan `navController` diteruskan ke `HomeScreen`. Pada baris [21–25], `composable("detail/{itemId}")` digunakan untuk menangani navigasi ke layar detail berdasarkan `itemId` yang dikirim dari layar sebelumnya. `backStackEntry.arguments?.getString("itemId")?.toIntOrNull()` digunakan untuk mengambil dan mengubah `itemId` dari string menjadi integer. Jika berhasil, maka `DetailScreen` akan ditampilkan dengan `itemId` dan `navController` sebagai argumen.

## HomeScreen.kt

Pada baris [1], `package com.example.movieList.ui.screen` menunjukkan bahwa file ini merupakan bagian dari package `ui.screen` dalam proyek bernama `com.example.movieList`. Pada baris [3–21], berbagai library dari Jetpack Compose dan Android SDK diimpor, termasuk elemen UI seperti `Image`, `Text`, `Row`, `Column`, `Button`, dan fungsi terkait warna, ukuran, dan navigasi. Pada baris [22], `com.example.movieList.model.listItems` diimpor, yang berisi daftar film yang akan ditampilkan.

Pada baris [24], didefinisikan fungsi `composable` `HomeScreen` dengan parameter `navController` untuk menangani navigasi. Pada baris [25], variabel `context` diinisialisasi menggunakan `LocalContext.current` untuk keperluan akses resource dan intent.

Pada baris [27–30], digunakan `LazyColumn` untuk membuat daftar vertikal scrollable, dengan padding konten sebesar 20dp dan jarak antar item sebesar 26dp, serta latar belakang berwarna hitam. Pada baris [31], fungsi `items` dipanggil sebanyak jumlah elemen `listItems`, dan setiap iterasi mendapatkan indeks yang digunakan untuk mengakses elemen `item`.

Pada baris [33–36], sebuah `Card` dibuat dengan sudut membulat 16dp dan warna latar tertentu, serta mengisi lebar penuh layar. Pada baris [37–41], di dalam `Card` digunakan `Row` untuk menata gambar film dan informasi film secara horizontal.

Pada baris [43–51], komponen `Image` digunakan untuk menampilkan poster film. Resource gambar diambil menggunakan `context.resources.getIdentifier` berdasarkan nama gambar dalam `item.imageResName`. Gambar diatur agar memiliki ukuran 90x130dp, sudut membulat 10dp, dan skala isi `Crop`.

Pada baris [53], digunakan `Spacer` untuk memberi jarak antara gambar dan teks. Pada baris [56–83], sebuah `Column` digunakan untuk menampung detail film. Pada baris [57–63], ditampilkan nama film menggunakan `Text` dengan ukuran font 18sp dan tebal. Pada baris [65–67], label "Plot:" ditampilkan dengan gaya tebal. Pada baris [69–72], deskripsi film ditampilkan dengan batas maksimal 4 baris dan warna abu terang.

Pada baris [74], `Spacer` kembali digunakan untuk memberi jarak sebelum tombol. Pada baris [76–91], dua buah tombol ditampilkan sejajar secara horizontal. Pada baris [77–82], tombol pertama berlabel "IMDb" akan membuka tautan IMDb dari film dengan

`Intent.ACTION_VIEW`. Pada baris [84–89], tombol kedua berlabel "Detail" akan menavigasi ke layar detail dengan mengirimkan `item.id` ke rute "detail/{itemId}".

## DetailScreen.kt

Pada baris [1], `package com.example.movieList.ui.screen` menunjukkan bahwa file ini merupakan bagian dari package `ui.screen` dalam proyek `com.example.movieList`. Pada baris [3–19], berbagai library dari Jetpack Compose diimpor untuk membangun tampilan UI, seperti `Image`, `Column`, `Text`, `Button`, dan properti tambahan seperti warna, ukuran, dan `alignment`. Pada baris [20], `listItems` diimpor dari model sebagai sumber data daftar film.

Pada baris [22], didefinisikan fungsi composable `DetailScreen` yang menerima parameter `itemId` bertipe `Int` dan `navController` untuk navigasi antar layar. Pada baris [24], `context` diinisialisasi menggunakan `LocalContext.current` untuk keperluan akses resource aplikasi. Pada baris [25], variabel `item` diisi dengan data film yang memiliki `id` sesuai `itemId` yang dikirimkan.

Pada baris [27], dicek apakah `item` tidak bernilai `null`. Jika ya, maka akan ditampilkan detailnya. Pada baris [28–35], digunakan `Column` untuk menata elemen secara vertikal dengan `padding 16dp`, latar belakang warna ungu muda, dan isi ditata ke tengah horizontal. Pada baris [36–44], komponen `Image` digunakan untuk menampilkan gambar poster film. Resource gambar diambil berdasarkan `imageResName` dari item yang ditemukan, dan diberi sudut membulat 12dp dengan skala isi `Crop`.

Pada baris [46], `Spacer` digunakan untuk memberi jarak 16dp sebelum elemen berikutnya. Pada baris [49–52], nama film ditampilkan dengan ukuran font 24sp dan tebal. Pada baris [55–58], rating film ditampilkan dengan simbol bintang "★", ukuran font 18sp, dan warna abu-abu gelap.

Pada baris [60], `Spacer` kembali digunakan untuk memberi jarak vertikal. Pada baris [63–67], deskripsi film ditampilkan dalam `Text` dengan ukuran font 16sp, warna hitam, dan perataan teks rata kanan-kiri (*justify*). Pada baris [69], `Spacer` ditambahkan lagi untuk memberi jarak sebelum tombol.

Pada baris [72–76], tombol "Back" ditampilkan. Saat diklik, tombol ini akan memanggil fungsi `navController.popBackStack()` untuk kembali ke layar sebelumnya. Tombol diberi warna ungu dan sudut membulat 12dp. Pada baris [77], label teks "Back" ditampilkan di dalam tombol. Pada baris [79], jika `item` tidak ditemukan (bernilai `null`), maka teks "Item not found" akan ditampilkan berwarna merah.

## MovieViewModel.kt

Pada baris [1], `package com.example.movieListxml.ui.viewmodel` berfungsi untuk mendefinisikan lokasi package tempat `MovieViewModel` berada. Pada baris [2] hingga [7], dilakukan import terhadap berbagai class seperti `ViewModel`, `ViewModelProvider`,

`MutableStateFlow`, `StateFlow`, `model ListItem`, serta `Timber` untuk logging, yang dibutuhkan dalam class ini.

Pada baris [9], class `MovieViewModel`(private val `sourceName: String`) : `ViewModel()` mendeklarasikan kelas `MovieViewModel` yang merupakan turunan dari `ViewModel` dan memiliki parameter `sourceName` untuk mencatat sumber data. Pada baris [11], `_movieList` dideklarasikan sebagai `MutableStateFlow` yang berisi data awal daftar film dari `listItems`. Pada baris [12], properti publik `movieList` menyediakan akses ke data `StateFlow` agar bisa diamati oleh UI tanpa bisa diubah dari luar.

Pada baris [14], blok `init` digunakan untuk menampilkan log jumlah item film yang diinisialisasi beserta sumbernya, dengan bantuan `Timber`.

Pada baris [17], `_selectedItemId` menyimpan ID dari film yang dipilih dalam bentuk `MutableStateFlow`. Baris [18] menyediakan akses publik melalui `selectedItemId`.

Pada baris [20], fungsi `selectItem()` digunakan untuk menetapkan ID film yang dipilih. Sedangkan pada baris [23], fungsi `getMovieById()` memungkinkan pengambilan satu objek `ListItem` berdasarkan ID dengan mencarinya dari daftar `movieList`.

## **MovieViewModelFactory.kt**

Pada baris [1], package `com.example.movieListxml.ui.viewmodel` digunakan untuk menyatakan bahwa file ini berada di dalam package `ui.viewmodel` dari proyek `movieListxml`. Pada baris [2] dan [3], dilakukan import terhadap `ViewModel` dan `ViewModelProvider`, yang diperlukan untuk membuat dan mengelola lifecycle dari `ViewModel`.

Pada baris [5], class `MovieViewModelFactory`(private val `sourceName: String`) mendefinisikan sebuah class `MovieViewModelFactory` yang menerima parameter `sourceName` dan mengimplementasikan interface `ViewModelProvider.Factory`. Pada baris [6], fungsi `create` dioverride untuk menyediakan objek `ViewModel` sesuai kebutuhan.

Pada baris [7], dilakukan pengecekan apakah class yang diminta adalah `MovieViewModel`. Jika benar, maka pada baris [8] akan dibuat dan dikembalikan instance dari `MovieViewModel` dengan `sourceName` yang telah diberikan. Pada baris [10], jika class yang diminta bukan `MovieViewModel`, maka akan dilemparkan exception `IllegalArgumentException` dengan pesan “Unknown ViewModel class”.

## **MovieApp.kt**

Pada baris [1], package `com.example.movieListxml` berfungsi untuk mendefinisikan lokasi package dari class `MovieApp`, agar sesuai dengan struktur proyek Android. Pada baris [2] dan [3], import `android.app.Application` dan import `timber.log.Timber`

digunakan untuk mengimpor class Application yang menjadi superclass dari MovieApp, serta library Timber yang digunakan untuk logging.

Pada baris [5], `class MovieApp : Application()` mendefinisikan class MovieApp sebagai turunan dari class Application, yang berarti kode di dalamnya akan dijalankan saat aplikasi pertama kali dibuka. Pada baris [6], `override fun onCreate()` digunakan untuk menimpa method `onCreate()` bawaan dari class Application.

Pada baris [7], `super.onCreate()` dipanggil agar inisialisasi bawaan dari Application tetap berjalan. Pada baris [9], terdapat pengecekan `if (BuildConfig.DEBUG)` yang berfungsi untuk memastikan bahwa blok kode di dalamnya hanya dijalankan saat mode debug, bukan pada rilis aplikasi.

Pada baris [10], `Timber.plant(Timber.DebugTree())` berfungsi untuk menginisialisasi Timber dengan DebugTree, sehingga kita bisa menampilkan log saat proses debugging. Pada baris [11], `Timber.i("Timber initialized in MovieApp")` mencetak log informasi bahwa Timber berhasil diinisialisasi.

### **Penjelasan mengenai Debugger, dan fitur Step Into, Step Over, dan Step Out**

Jadi disini, debugger adalah alat bantu dalam lingkungan pengembangan perangkat lunak (IDE) yang digunakan untuk menganalisis dan menemukan bug dalam kode program secara lebih mendalam dan sistematis. Dengan debugger, seorang developer dapat menjalankan program secara bertahap, memantau nilai variabel secara langsung, dan melihat bagaimana alur eksekusi kode berjalan. Ini sangat membantu untuk mengetahui di bagian mana program tidak bekerja sesuai yang diharapkan.

Pada penggunaan debugger, pertama-tama kita harus menempatkan breakpoint, yaitu tanda yang menunjukkan di baris mana program harus berhenti sementara saat dijalankan. Breakpoint ini biasanya dipasang dengan mengklik sisi kiri baris kode di IDE seperti Android Studio, IntelliJ, atau Visual Studio Code. Setelah breakpoint ditetapkan, kita bisa menjalankan program dalam mode debug. Ketika eksekusi program mencapai breakpoint, proses akan berhenti dan kita dapat mulai menganalisis kondisi program saat itu, seperti memeriksa isi variabel, stack trace, dan lain-lain.

Terdapat beberapa fitur penting dalam debugger, yaitu Step Into, Step Over, dan Step Out. Step Into digunakan untuk masuk ke dalam fungsi yang sedang dipanggil pada baris tersebut, sehingga kita bisa melihat baris per baris isi fungsi tersebut. Ini berguna saat ingin menelusuri lebih dalam logika internal suatu fungsi. Step Over digunakan untuk menjalankan baris kode saat ini dan langsung melanjutkan ke baris berikutnya tanpa masuk ke dalam fungsi yang dipanggil. Ini cocok jika kita tidak perlu melihat detail fungsi dan hanya ingin melanjutkan alur utama. Sementara itu, Step Out digunakan untuk keluar dari fungsi saat ini dan kembali ke pemanggilnya, biasanya digunakan saat kita sudah selesai menelusuri sebuah fungsi dan ingin melanjutkan eksekusi di tingkat yang lebih tinggi.

## **2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya**

Jadi disini, Application class adalah sebuah kelas khusus yang digunakan untuk menyimpan dan mengelola state global dari seluruh aplikasi. Kelas ini merupakan turunan langsung dari kelas `android.app.Application`, dan hanya akan dibuat sekali selama siklus hidup aplikasi, yaitu saat aplikasi pertama kali dijalankan. Karena hanya diinisialisasi satu kali, Application class digunakan untuk mengatur konfigurasi global, menginisialisasi library pihak ketiga seperti Timber, Retrofit, atau Room, serta menyimpan objek atau data yang perlu diakses dari banyak bagian aplikasi.

Fungsi utama dari Application class adalah untuk menyediakan titik awal (entry point) global sebelum komponen lain seperti Activity atau Service dijalankan. Di dalam method `onCreate()` miliknya, kita bisa menuliskan logika inisialisasi seperti membuat instance ViewModel, menyiapkan dependency injection, mengatur log, atau konfigurasi lainnya yang perlu tersedia secara menyeluruh selama aplikasi berjalan. Untuk dapat menggunakan Application class, kita harus mendeklarasikannya di file `AndroidManifest.xml` melalui atribut `android:name`.

Contohnya ada di penjelasan kode juga diatas, untuk XML dan Jetpack Compose

#### **D. Tautan Git**

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/MuhammadFiras/Praktikum-Mobile/tree/main>