LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

ViewModel and Debugging

Oleh:

Raymond Hariyono NIM 2310817210007

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 Prakitkum ini dikerjakan oleh:

Nama Praktikan : Raymond Hariyono
NIM : 2310817210007

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

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

Soal Praktikum:

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:

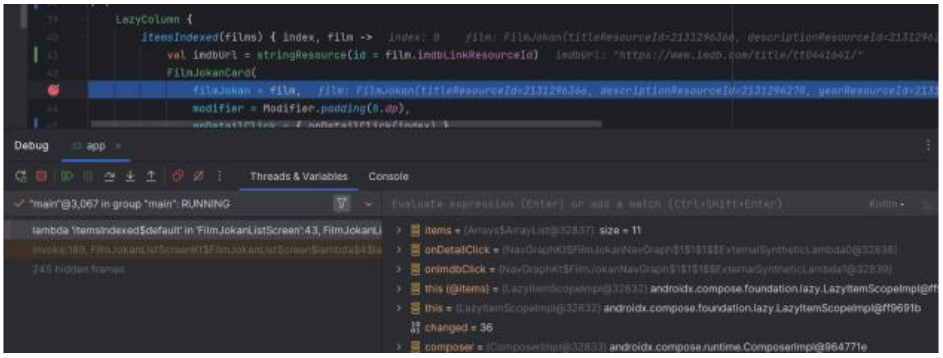
a. Log saat data item masuk ke dalam list

b. Log saat tombol Detail dan tombol Explicit Intent ditekan

c. Log data dari list yang dipilih ketika berpindah ke halaman Detail

2. 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 Aplikasi harus dapat mempertahankan fitur-fitur yang sudah dibuat pada modul sebelumnya.

Berikut adalah contoh debugging dalam Android Studio.



A. Source Code

Source Code XML

item_film.XML

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
3   android:layout_width="match_parent"
4       android:id="@+id/card_view"
5       android:layout_height="wrap_content"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:layout_margin="8dp"
8       app:cardCornerRadius="22dp"
9       app:cardElevation="4dp"
10      android:layout_gravity="center"
11      xmlns:app="http://schemas.android.com/apk/res-auto"
12
13
14  xmlns:android="http://schemas.android.com/apk/res/android">
15
16      <androidx.constraintlayout.widget.ConstraintLayout
17          android:layout_width="match_parent"
18          android:layout_height="wrap_content"
19          android:padding="8dp">
```

```
20
21          <ImageView
22              android:id="@+id/film_image"
23              android:layout_width="150dp"
24              android:layout_height="200dp"
25              android:scaleType="centerCrop"
26
27  app:layout_constraintBottom_toBottomOf="parent"
28              app:layout_constraintEnd_toEndOf="parent"
29              app:layout_constraintHorizontal_bias="0.0"
30              app:layout_constraintStart_toStartOf="parent"
31              app:layout_constraintTop_toTopOf="parent"
32              app:layout_constraintVertical_bias="0.0"
33              tools:src="@tools:sample/avatars" />
34
35          <TextView
36              android:id="@+id/film_title"
37              android:layout_width="120dp"
38              android:layout_height="wrap_content"
39              android:layout_marginStart="12dp"
40              android:textSize="13sp"
41              android:textStyle="bold"
42
43  app:layout_constraintBottom_toBottomOf="parent"
44              app:layout_constraintEnd_toEndOf="parent"
45              app:layout_constraintHorizontal_bias="0.0"
46
47  app:layout_constraintStart_toEndOf="@+id/film_image"
48              app:layout_constraintTop_toTopOf="parent"
49              app:layout_constraintVertical_bias="0.082"
50              tools:text="judul film" />
```

```
51
52          <TextView
53              android:id="@+id/film_year"
54              android:layout_width="wrap_content"
55              android:layout_height="wrap_content"
56              android:layout_marginEnd="20dp"
57              android:textSize="13sp"
58              android:textStyle="bold"
59
60  app:layout_constraintBottom_toBottomOf="parent"
61              app:layout_constraintEnd_toEndOf="parent"
62
63              app:layout_constraintTop_toTopOf="parent"
64              app:layout_constraintVertical_bias="0.083"
65              tools:ignore="MissingConstraints"
66              tools:text="year" />
67
68          <TextView
69              android:id="@+id/film_desc"
70              android:layout_marginStart="10dp"
71              android:layout_width="0dp"
72              android:layout_height="wrap_content"
73              android:textSize="10sp"
74
75  app:layout_constraintBottom_toTopOf="@+id/btn_imdb"
76              app:layout_constraintEnd_toEndOf="parent"
77              app:layout_constraintHorizontal_bias="0.266"
78
79  app:layout_constraintStart_toEndOf="@+id/film_image"
80
81  app:layout_constraintTop_toBottomOf="@+id/film_title"
```

```
            app:layout_constraintVertical_bias="0.2"
            tools:text="hab" />

        <Button
            android:id="@+id/btn_imdb"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toStartOf="@+id/btn_detail"
            app:layout_constraintHorizontal_bias="0.266"

app:layout_constraintStart_toEndOf="@+id/film_image"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.947"
            android:textSize="10sp"
            android:text="@string/btn_imdb" />

        <Button
            android:id="@+id/btn_detail"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="1.0"

app:layout_constraintStart_toEndOf="@+id/film_image"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.947"
```

```
            android:textSize="10sp"
            android:text="@string/btn_detail" />


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

Tabel 1. Source Code file item_film XML Jawaban Soal 1 XML

Fragment_detail.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      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:padding="8dp"
9      tools:context=".DetailFragment">
10
11     <android.widget.ScrollView
12         android:layout_width="match_parent"
13         android:layout_height="match_parent">
14
15         <LinearLayout
16             android:orientation="vertical"
17             android:layout_width="match_parent"
18             android:layout_height="wrap_content"
19             android:padding="8dp"
20             android:gravity="center">
21
22             <ImageView
23                 android:id="@+id/detail_image"
24                 android:layout_width="wrap_content"
```

```
25                 android:layout_height="400dp"
26                 android:scaleType="centerCrop"
27                 tools:src="@tools:sample/avatars"
28                 app:layout_constraintTop_toTopOf="parent"
29
30 app:layout_constraintStart_toStartOf="parent"
31                 app:layout_constraintEnd_toEndOf="parent" />
32
33             <TextView
34                 android:id="@+id/detail_title"
35                 android:layout_width="wrap_content"
36                 android:layout_height="wrap_content"
37                 android:layout_marginTop="16dp"
38                 android:textSize="20sp"
39                 android:textStyle="bold"
40                 app:layout_constraintEnd_toEndOf="parent"
41
42 app:layout_constraintStart_toStartOf="parent"
43                 tools:text="The Lord of the Rings" />
44
45             <TextView
46                 android:id="@+id/detail_year"
47                 android:layout_width="wrap_content"
48                 android:layout_height="wrap_content"
49                 android:layout_marginTop="32dp"
50                 android:textColor="#666"
51                 android:textSize="25sp"
52                 android:textStyle="italic"
53                 app:layout_constraintEnd_toEndOf="parent"
54
55 app:layout_constraintStart_toStartOf="parent"
```

```
56
57  app:layout_constraintTop_toBottomOf="@id/detail_title"
58                  tools:text="2001"                              />
59
60              <TextView
61                  android:id="@+id/detail_desc"
62                  android:layout_width="wrap_content"
63                  android:layout_height="wrap_content"
64                  android:layout_marginVertical="30dp"
65                  android:layout_marginTop="8dp"
66                  android:textSize="14sp"
67
68  app:layout_constraintBottom_toBottomOf="parent"
69                  app:layout_constraintEnd_toEndOf="parent"
70                  app:layout_constraintHorizontal_bias="0.0"
71
72  app:layout_constraintStart_toStartOf="parent"
73
74  app:layout_constraintTop_toBottomOf="@id/detail_year"
75                  app:layout_constraintVertical_bias="0.037"
76                  tools:text="A meek Hobbit from the Shire and
77  eight companions set out on a journey to destroy the powerful
78  One                         Ring..."                           />
          </LinearLayout>


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

Tabel 2. Source Code File fragment_detail Jawaban Soal 1 XML

Fragment_home.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:tools="http://schemas.android.com/tools"
 5      android:layout_width="match_parent"
 6      android:layout_height="match_parent"
 7      xmlns:app="http://schemas.android.com/apk/res-auto"
 8      tools:context=".HomeFragment">
 9
10      <androidx.recyclerview.widget.RecyclerView
11          android:id="@+id/rvFilm"
12          android:layout_width="0dp"
13          android:layout_height="0dp"
14          android:layout_margin="15dp"
15          app:layout_constraintBottom_toBottomOf="parent"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent"
19          />
20
21  </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 3. Source Code File fragment_home.XML Jawaban Soal 1 XML

Activiy_main.XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <FrameLayout
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity"
8      android:id="@+id/frame_container">
9  </FrameLayout>
```

Tabel 4. Source Code File activity_main.XML Jawaban Soal 1 XML.

Film.kt

```
1  package com.example.listxml
2  import android.os.Parcelable
3  import kotlinx.parcelize.Parcelize
4
5  @Parcelize
6  data class Film (
7      val title : String? = null,
8      val year : String? = null,
9      val desc : String? = null,
10     val image : Int = 0,
11     val imdb : String = "",
12     val detail : String = "", // URL for the detail page
13 ): Parcelable
```

Tabel 5 Source Code File Film Jawaban Soal 1 XML

FilmAdapter.kt

```
1  package com.example.listxml.adapter
2
3  import android.view.LayoutInflater
4  import android.view.ViewGroup
5  import androidx.recyclerview.widget.RecyclerView
6  import com.example.listxml.databinding.ItemFilmBinding
7  import com.example.listxml.model.Film
8  import timber.log.Timber
9
10 interface OnFilmClickListener {
11     fun onDetailClicked(film: Film)
12     fun onImdbClicked(imdbUrl: String)
13 }
14
15 class FilmAdapter(
16     private val filmList: ArrayList<Film>,
17     private val listener: OnFilmClickListener
18 ) : RecyclerView.Adapter<FilmAdapter.FilmViewHolder>() {
19
20     inner class FilmViewHolder(val binding: ItemFilmBinding)
21 :
22         RecyclerView.ViewHolder(binding.root)
23
24     override fun onCreateViewHolder(parent: ViewGroup,
25 viewType: Int): FilmViewHolder {
26         val binding =
27 ItemFilmBinding.inflate(LayoutInflater.from(parent.context
28 ), parent, false)
29         return FilmViewHolder(binding)
30     }
```

```
31
32      override fun onBindViewHolder(holder: FilmViewHolder,
33  position: Int) {
34          val film = filmList[position]
35          holder.binding.apply {
36              filmImage.setImageResource(film.image)
37              filmTitle.text = film.title
38              filmYear.text = film.year
39              filmDesc.text = film.desc
40              btnImdb.setOnClickListener {
41                  Timber.d("IMDb button clicked for:
42  ${film.title}")
43                  listener.onImdbClicked(film.imdb)
44              }
45              btnDetail.setOnClickListener {
46                  Timber.d("Detail button clicked for:
47  ${film.title}")
48                  listener.onDetailClicked(film)
49              }
50          }
51      }
52
53
54      override fun getItemCount(): Int = filmList.size
55  }
```

Tabel 6. Source Code File FilmAdapter Jawaban Soal 1 XML

DetailFragment.kt

```
1   package com.example.listxml
2
3   import android.os.Bundle
```

```
4  import androidx.fragment.app.Fragment
5  import android.view.LayoutInflater
6  import android.view.View
7  import android.view.ViewGroup
8  import com.example.listxml.databinding.FragmentDetailBinding
9
10 class DetailFragment : Fragment() {
11
12     private var _binding: FragmentDetailBinding? = null
13     private val binding get() = _binding!!
14
15     private var title: String? = null
16     private var desc: String? = null
17     private var year: String? = null
18     private var image: Int = -1
19
20     override fun onCreate(savedInstanceState: Bundle?) {
21         super.onCreate(savedInstanceState)
22
23         arguments?.let {
24             title = it.getString("EXTRA_TITLE")
25             desc = it.getString("EXTRA_DESC")
26             year = it.getString("EXTRA_YEAR")
27             image = it.getInt("EXTRA_IMAGE")
28         }
29     }
30
31     override fun onCreateView(
32         inflater: LayoutInflater, container: ViewGroup?,
33         savedInstanceState: Bundle?
34     ): View {
```

```
35          _binding  =  FragmentDetailBinding.inflate(inflater,
36 container, false)
37          return binding.root
38      }
39
40      override       fun       onViewCreated(view:       View,
41 savedInstanceState: Bundle?) {
42          super.onViewCreated(view, savedInstanceState)
43
44          binding.detailTitle.text = title
45          binding.detailDesc.text = desc
46          binding.detailYear.text = year
47          binding.detailImage.setImageResource(image)
48      }
49
50      override fun onDestroyView() {
51          super.onDestroyView()
52          _binding = null
53      }
54 }
55
56
57
58
59
```

Tabel 7 Source Code File DetailFragment Jawaban Soal 1 XML.

HomeFragment.kt

```
1   package com.example.listxml
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.fragment.app.viewModels
11 import androidx.lifecycle.lifecycleScope
12 import androidx.recyclerview.widget.LinearLayoutManager
13 import com.example.listxml.adapter.FilmAdapter
14 import com.example.listxml.adapter.OnFilmClickListener
15 import com.example.listxml.databinding.FragmentHomeBinding
16 import com.example.listxml.model.Film
17 import com.example.listxml.viewModel.HomeViewModel
18 import com.example.listxml.viewModel.HomeViewModelFactory
19 import kotlinx.coroutines.flow.collectLatest
20 import kotlinx.coroutines.launch
21 import timber.log.Timber
22
23 class HomeFragment : Fragment() {
24
25     private var _binding: FragmentHomeBinding? = null
26     private val binding get() = _binding!!
27
28     private lateinit var filmAdapter: FilmAdapter
29
30     private val viewModel: HomeViewModel by viewModels {
31         HomeViewModelFactory("Guest")
32     }
33
```

```
34      override fun onCreateView(
35          inflater: LayoutInflater, container: ViewGroup?,
36          savedInstanceState: Bundle?
37      ): View {
38          _binding  =  FragmentHomeBinding.inflate(inflater,
39  container, false)
40          return binding.root
41      }
42
43      override      fun      onViewCreated(view:      View,
44  savedInstanceState: Bundle?) {
45          super.onViewCreated(view, savedInstanceState)
46
47          viewModel.setFilmList(getListFilm())
48
49          viewModel.filmList.observe(viewLifecycleOwner)    {
50  list ->
51              setupRecyclerView(list)
52          }
53
54          viewLifecycleOwner.lifecycleScope.launch {
55              viewModel.selectedFilm.collectLatest { film ->
56                  film?.let {
57                      navigateToDetail(it)
58                      viewModel.clearSelectedFilm()
59                  }
60              }
61          }
62      }
63
64      private fun setupRecyclerView(filmList: List<Film>) {
```

```
65          filmAdapter    =    FilmAdapter(ArrayList(filmList),
66  object : OnFilmClickListener {
67              override fun onDetailClicked(film: Film) {
68                  viewModel.onFilmClicked(film)
69              }
70
71              override fun onImdbClicked(imdbUrl: String) {
72                  val   intent   =   Intent(Intent.ACTION_VIEW,
73  Uri.parse(imdbUrl))
74                  startActivity(intent)
75              }
76          })
77
78          binding.rvFilm.apply {
79              layoutManager = LinearLayoutManager(context)
80              adapter = filmAdapter
81          }
82      }
83
84      private fun navigateToDetail(film: Film) {
85          Timber.d("Navigating  to  DetailFragment  with  film:
86  title=${film.title}, year=${film.year}")
87
88          val detailFragment = DetailFragment().apply {
89              arguments = Bundle().apply {
90                  putString("EXTRA_TITLE", film.title)
91                  putString("EXTRA_DESC", film.desc)
92                  putInt("EXTRA_IMAGE", film.image)
93              }
94          }
95          parentFragmentManager.beginTransaction()
```

```
96              .replace(R.id.frame_container, detailFragment)
97              .addToBackStack(null)
98              .commit()
99      }
100
101
102     private fun getListFilm(): List<Film> {
103         val                    dataTitle                    =
104 resources.getStringArray(R.array.data_filmTitle)
105         val                     dataDesc                    =
106 resources.getStringArray(R.array.data_filmDesc)
107         val                    dataImage                    =
108 resources.obtainTypedArray(R.array.data_filmImage)
109         val                     dataYear                    =
110 resources.getStringArray(R.array.data_filmYear)
111         val                     dataImdb                    =
112 resources.getStringArray(R.array.data_filmImdb)
113
114         val listFilm = mutableListOf<Film>()
115         for (i in dataTitle.indices) {
116             listFilm.add(
117                 Film(
118                     title = dataTitle[i],
119                     year = dataYear[i],
120                     desc = dataDesc[i],
121                     image = dataImage.getResourceId(i, -1),
122                     imdb = dataImdb[i]
123                 )
124             )
125         }
126         dataImage.recycle()
```

```
127         return listFilm
128     }
129
130     override fun onDestroyView() {
131         super.onDestroyView()
132         _binding = null
133     }
134 }
```

Tabel 8. Source Code File HomeFragment Jawaban Soal 1 XML

MainActivity.kt

```
1     package com.example.listxml
2
3     import android.os.Bundle
4     import androidx.appcompat.app.AppCompatActivity
5
6     class MainActivity : AppCompatActivity() {
7         override fun onCreate(savedInstanceState: Bundle?)
8 {
9             super.onCreate(savedInstanceState)
10            setContentView(R.layout.activity_main)
11
12    val fragmentManager = supportFragmentManager
13            val homeFragment = HomeFragment()
14            val                  fragment                  =
15 fragmentManager.findFragmentByTag(HomeFragment::class.java
16 simpleName)
17            if (fragment !is HomeFragment) {
18                fragmentManager
19                    .beginTransaction()
20                    .add(R.id.frame_container,
21 homeFragment, HomeFragment::class.java.simpleName)
22                    .commit()
23            }
24        }
25   }
```

Tabel 9. Source Code File MainActivity Jawaban Soal 1 XML

HomeViewModel.kt

```
1  package com.example.listxml.viewModel
2  import android.app.Application
3  import androidx.lifecycle.AndroidViewModel
4  import androidx.lifecycle.LiveData
5  import androidx.lifecycle.MutableLiveData
6  import androidx.lifecycle.ViewModel
7  import com.example.listxml.model.Film
8  import kotlinx.coroutines.flow.MutableStateFlow
9  import kotlinx.coroutines.flow.StateFlow
10 import timber.log.Timber
11
12 class HomeViewModel(private val userName: String) : ViewModel() {
13
14     private val _filmList = MutableLiveData<List<Film>>()
15     val filmList: LiveData<List<Film>> get() = _filmList
16
17     private val _selectedFilm = MutableStateFlow<Film?>(null)
18     val selectedFilm: StateFlow<Film?> = _selectedFilm
19
20     init {
21         Timber.d("ViewModel initialized for user: $userName")
22     }
23
24     fun setFilmList(list: List<Film>) {
25         list.forEach { film ->
26             Timber.d("Film      added:      title=${film.title},
27 year=${film.year}")
28         }
```

```
29          _filmList.value = list
30      }
31
32      fun onFilmClicked(film: Film) {
33          Timber.d("Film clicked: title=${film.title}")
34          _selectedFilm.value = film
35      }
36
37
38      fun clearSelectedFilm() {
39          _selectedFilm.value = null
40      }
41
42  }
```

Tabel 10. Source Code File HomeViewModel Jawaban Soal 1 XML

HomeViewModelFactory.kt

```
1   package com.example.listxml
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5
6   class FilmViewModelFactory(private val userName: String) :
7   ViewModelProvider.Factory {
8       override fun <T : ViewModel> create(modelClass:
9   Class<T>): T {
10          return FilmViewModel(userName) as T
11      }
12  }
```

Tabel 11. Source Code File HomeViewModelFactory Jawaban Soal 1 XML

Source Code Compose

Films

```
1  package com.example.listxml.data
2
3  data class Films(
4      val title: String,
5      val year: String,
6      val description: String,
7      val image: Int = 0,
8      val imdbUrl: String
9  )
```

Tabel 12. Source Code File Films.kt Jawaban Soal 1 XML.

FilmListItems.kt

```
1  package com.example.listxml
2  import android.content.Intent
3  import android.net.Uri
4  import androidx.compose.foundation.Image
5  import androidx.compose.foundation.layout.Arrangement
6  import androidx.compose.foundation.layout.Column
7  import androidx.compose.foundation.layout.Row
8  import androidx.compose.foundation.layout.Spacer
9  import androidx.compose.foundation.layout.fillMaxWidth
10 import androidx.compose.foundation.layout.padding
11 import androidx.compose.foundation.layout.size
12 import androidx.compose.foundation.layout.width
13 import
14 androidx.compose.foundation.shape.RoundedCornerShape
15 import androidx.compose.material3.Button
16 import androidx.compose.material3.Card
```

```
import androidx.compose.material3.CardDefaults
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.example.listxml.data.Films


@Composable
fun FilmListItems(
    films: Films,
    navController: NavController,
    viewModel: FilmViewModel
) {
    val context = LocalContext.current

    Card(
        modifier = Modifier
            .padding(4.dp)
            .fillMaxWidth(),
        elevation = CardDefaults.cardElevation(4.dp),
```

```
48          shape = RoundedCornerShape(16.dp),
49      ) {
50          Row(modifier = Modifier.padding(8.dp)) {
51              FilmImage(films = films)
52              Spacer(modifier = Modifier.width(8.dp))
53
54              Column(
55                  modifier = Modifier
56                      .weight(1f)
57                      .fillMaxWidth()
58              ) {
59                  Text(
60                      text = films.title,
61                      style = TextStyle(fontWeight =
62  FontWeight.Bold, fontSize = 14.sp)
63                  )
64                  Text(
65                      text = "(${films.year})",
66                      style = TextStyle(fontSize = 10.sp,
67  color = Color.Gray)
68                  )
69
70                  Text(
71                      text = films.description,
72                      style = TextStyle(fontSize = 10.sp),
73                      maxLines = 5,
74                      overflow = TextOverflow.Ellipsis
75                  )
76
77                  Row(
78                      modifier = Modifier
```

```
                        .fillMaxWidth()
                        .padding(top = 30.dp),
                    horizontalArrangement =
Arrangement.Start
                ) {
                    Button(
                        onClick = {
                            viewModel.logImdbClick(films)
                            val intent =
Intent(Intent.ACTION_VIEW, Uri.parse(films.imdbUrl))
                            context.startActivity(intent)
                        },
                        modifier = Modifier
                            .padding(top = 8.dp)
                            .weight(1f),
                        shape = RoundedCornerShape(16.dp),
                    ) {
                        Text("IMDB", style =
TextStyle(fontWeight = FontWeight.Bold, fontSize = 8.sp))
                    }

                    Spacer(modifier =
Modifier.width(5.dp))

                    Button(
                        onClick = {
                            viewModel.selectFilm(films)

viewModel.logDetailClick(films)

navController.navigate("detail/${films.title}")
```

```
110                         },
111                         modifier = Modifier
112                             .padding(top = 8.dp)
113                             .weight(1f),
114                         shape = RoundedCornerShape(16.dp),
115                     ) {
116                         Text("Detail", style =
117 TextStyle(fontWeight = FontWeight.Bold, fontSize = 8.sp))
118                     }
119                 }
120             }
121         }
122     }
123 }
124
125
126 @Composable
127 fun FilmImage(films: Films){
128     Image(
129         painter = painterResource(id = films.image),
130         contentDescription = null,
131         modifier = Modifier
132             .size(150.dp)
133             .padding(16.dp)
134             .clip(RoundedCornerShape(16.dp)),
135         contentScale = ContentScale.Crop,
136     )
137 }
```

Tabel 13. Source Code File FilmListItems.kt Jawaban Soal 1 Compose

DetailContent.kt

```
1  package com.example.listxml
2
3  import androidx.compose.foundation.Image
4  import androidx.compose.foundation.layout.Arrangement
5  import androidx.compose.foundation.layout.Column
6  import androidx.compose.foundation.layout.Row
7  import androidx.compose.foundation.layout.Spacer
8  import androidx.compose.foundation.layout.fillMaxSize
9  import androidx.compose.foundation.layout.fillMaxWidth
10 import androidx.compose.foundation.layout.height
11 import androidx.compose.foundation.layout.padding
12 import androidx.compose.foundation.layout.width
13 import androidx.compose.foundation.shape.RoundedCornerShape
14 import androidx.compose.material3.Text
15 import androidx.compose.runtime.Composable
16 import androidx.compose.ui.Alignment
17 import androidx.compose.ui.Modifier
18 import androidx.compose.ui.draw.clip
19 import androidx.compose.ui.layout.ContentScale
20 import androidx.compose.ui.res.painterResource
21 import androidx.compose.ui.text.TextStyle
22 import androidx.compose.ui.text.font.FontWeight
23 import androidx.compose.ui.unit.dp
24 import androidx.compose.ui.unit.sp
25 import com.example.listxml.data.Films
26
27 @Composable
28 fun DetailContent(films: Films) {
29     Column(
30         modifier = Modifier
```

```
            .fillMaxSize()
            .padding(16.dp)
    ) {
        DetailImageContent(films = films)
        Spacer(modifier = Modifier.height(16.dp))

        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween
        ) {
            Text(
                text = films.title,
                style = TextStyle(
                    fontWeight = FontWeight.Bold,
                    fontSize = 18.sp
                ),
                modifier = Modifier.weight(1f)
            )
            Text(
                text = films.year,
                style = TextStyle(
                    fontWeight = FontWeight.Bold,
                    fontSize = 18.sp

                )
            )
        }

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

        // DESKRIPSI
```

```
62          Text(
63              text = films.description,
64              style = TextStyle(
65                  fontSize = 14.sp
66              ),
67              modifier = Modifier.padding(top = 8.dp)
68          )
69      }
70  }
71
72  @Composable
73  fun DetailImageContent(films: Films) {
74      Column(
75          modifier = Modifier
76              .fillMaxWidth()
77              .padding(16.dp),
78          horizontalAlignment = Alignment.CenterHorizontally,
79          verticalArrangement = Arrangement.Center
80      ) {
81          Image(
82              painter = painterResource(id = films.image),
83              contentDescription = null,
84              modifier = Modifier
85                  .width(250.dp)
86                  .height(250.dp)
87                  .clip(RoundedCornerShape(16.dp)),
88              contentScale = ContentScale.Crop
89          )
90      }
91  }
```

Tabel 14. Source Code File detailContent Jawaban Soal 1 Compose

HomeContent.kt

```
1  package com.example.listxml
2
3  import androidx.compose.foundation.layout.PaddingValues
4  import androidx.compose.foundation.lazy.LazyColumn
5  import androidx.compose.foundation.lazy.items
6  import androidx.compose.runtime.Composable
7  import androidx.compose.runtime.collectAsState
8  import androidx.compose.runtime.getValue
9  import androidx.compose.ui.unit.dp
10 import androidx.navigation.NavController
11
12 @Composable
13 fun HomeContent(navController: NavController, filmViewModel:
14 FilmViewModel) {
15     val films by filmViewModel.filmList.collectAsState()
16
17     LazyColumn(contentPadding = PaddingValues(16.dp)) {
18         items(films) { film ->
19             FilmListItems(films = film, navController =
20 navController, viewModel = filmViewModel)
21         }
22     }
   }
```

Tabel 15. Source Code File HomeContent Jawaban Soal 1 Compose

MainActivity.kt

```
1  package com.example.listxml
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
```

```
import androidx.activity.enableEdgeToEdge
import androidx.compose.runtime.Composable
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import androidx.navigation.navArgument
import com.example.listxml.data.DataProvider
import com.example.listxml.ui.theme.ListXMLTheme
import timber.log.Timber

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Timber.plant(Timber.DebugTree())  //  Logging  aktif
hanya di mode debug

        setContent {
            val              viewModelFactory              =
FilmViewModelFactory("User123")
            val      filmViewModel:      FilmViewModel      =
viewModel(factory = viewModelFactory)

            GhibliFilms(filmViewModel = filmViewModel)
        }
    }

}

@Composable
```

```
fun GhibliFilms(filmViewModel: FilmViewModel) {
    val navController = rememberNavController()

    NavHost(navController = navController, startDestination =
"home") {
        composable("home") {
            HomeContent(navController, filmViewModel)
        }
        composable("detail/{title}") { backStackEntry ->
            val                    title                    =
backStackEntry.arguments?.getString("title")
            val film = filmViewModel.getFilmByTitle(title ?:
"")
            film?.let {
                DetailContent(films = it)
            }
        }
    }
}
```

Tabel 16. Source Code File MainActivity Jawaban Soal 1 Compose

ViewModel.kt

```
package com.example.listxml

import androidx.lifecycle.ViewModel
import com.example.listxml.data.DataProvider
import com.example.listxml.data.Films
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import timber.log.Timber


```

```
class FilmViewModel(private val userName: String) : ViewModel() {

    private val _filmList = MutableStateFlow(DataProvider.filmList)
    val filmList: StateFlow<List<Films>> = _filmList

    private val _selectedFilm = MutableStateFlow<Films?>(null)
    val selectedFilm: StateFlow<Films?> = _selectedFilm

    init {
        Timber.d("[$userName] FilmViewModel initialized with ${_filmList.value.size} items")
    }

    fun selectFilm(film: Films) {
        Timber.d("[$userName] Film selected: ${film.title}")
        _selectedFilm.value = film
    }

    fun logDetailClick(film: Films) {
        Timber.d("[$userName] Detail button clicked for: ${film.title}")
    }

    fun logImdbClick(film: Films) {
        Timber.d("[$userName] IMDB button clicked for: ${film.title}")
    }

```

```
42      fun getFilmByTitle(title: String): Films? {
43          return _filmList.value.find { it.title == title }
44      }
45  }
```

Tabel 17. Source Code File ViewModel Jawaban Soal 1 Compose.

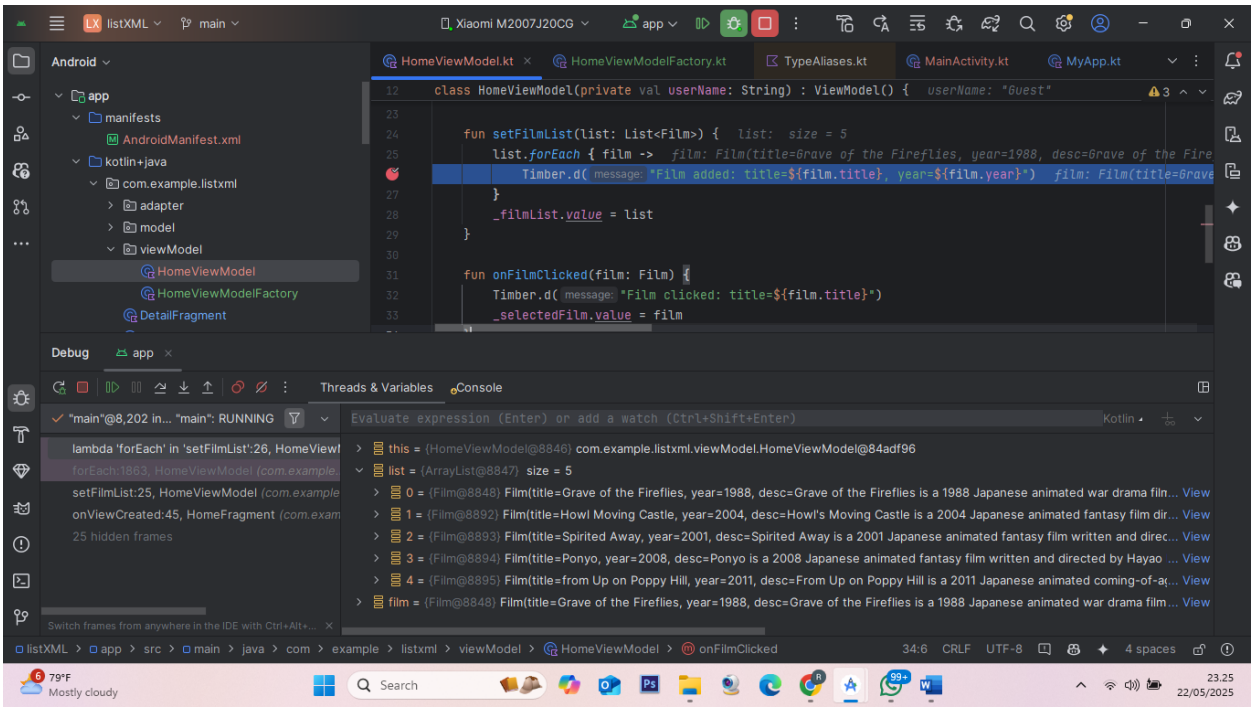
ViewModelFactory.kt

```
1   package com.example.listxml
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5
6   class FilmViewModelFactory(private val userName: String) :
7   ViewModelProvider.Factory {
8       override fun <T : ViewModel> create(modelClass:
9   Class<T>): T {
10          return FilmViewModel(userName) as T
11      }
12  }
```

Tabel 18. Source Code File ViewModelFactory Jawaban Soal 1 Compose

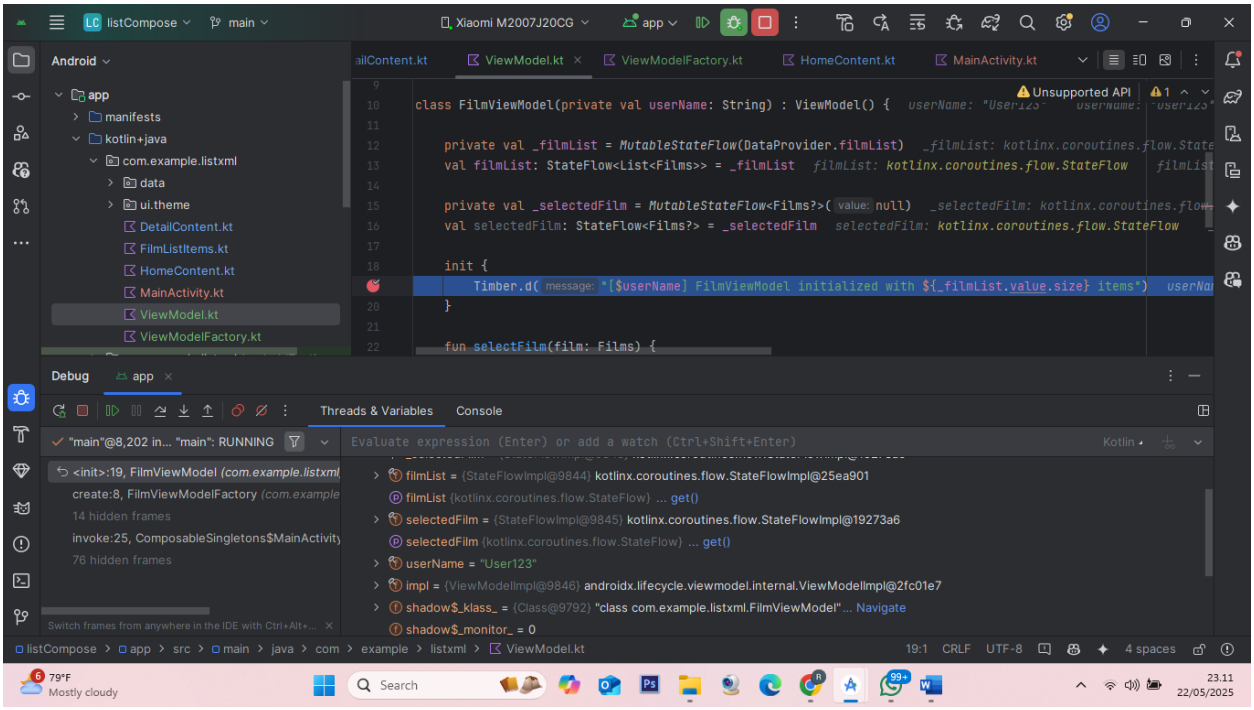
B. Output Program

Output Program XML



Gambar 1. Screenshot Hasil Output Program soal 1 XML

Output Program Compose



Gambar 2. Screenshot Hasil Output Program soal 1 Compose

C. Pembahasan

Step Into

Fungsi: Masuk ke dalam method/fungsi yang dipanggil pada baris saat ini.

Contoh: Jika berada di `listener.onDetailClicked(film)` dan ingin melihat apa yang terjadi di dalam fungsi `onDetailClicked`, tekan Step Into (F7).

Step Over

Fungsi: Melewati fungsi atau method yang dipanggil, tanpa masuk ke dalamnya. Program tetap berjalan ke baris berikutnya.

Contoh: Jika tahu `onDetailClicked()` bekerja dengan baik dan ingin lanjut ke kode berikutnya, tekan Step Over (F8).

Step Out

Fungsi: Keluar dari fungsi saat ini dan kembali ke fungsi pemanggilnya.

Contoh: Jika sudah masuk terlalu dalam ke fungsi lain dan ingin cepat kembali ke tempat pemanggil fungsi, tekan Step Out (Shift + F8).

Pembahasan XML

FilmAdapter.kt

Pada kode FilmAdapter.kt, logging menggunakan Timber digunakan untuk mencatat interaksi pengguna terhadap dua tombol yang terdapat dalam item list RecyclerView, yaitu tombol IMDb (btnImdb) dan tombol Detail (btnDetail). Penggunaan Timber ditulis dengan format Timber.d(...), di mana huruf d menandakan log dengan level debug. Logging ini sangat berguna saat proses pengujian (debugging), karena akan mencetak ke Logcat informasi berupa judul film yang diklik, sehingga memudahkan developer mengetahui apakah aksi klik berhasil dikenali.

Saat tombol IMDb ditekan, akan dicetak pesan log IMDb button clicked for: [judul film], sedangkan ketika tombol Detail ditekan, akan dicetak Detail button clicked for: [judul film]. Pencatatan ini tidak hanya sebagai informasi log biasa, namun juga sebagai indikasi interaksi pengguna yang akan diteruskan ke interface OnFilmClickListener. Interface ini berisi dua fungsi yaitu onImdbClicked() dan onDetailClicked(), yang masing-masing dipanggil ketika tombol yang sesuai ditekan.

Walaupun ViewModel tidak digunakan secara langsung di dalam adapter ini, fungsi onDetailClicked(film) yang dipanggil ketika tombol Detail ditekan biasanya akan diteruskan ke ViewModel oleh Fragment atau Activity yang mengimplementasikan interface tersebut. Misalnya, data film yang dikirim dari adapter akan ditangkap di Fragment, lalu diteruskan ke ViewModel melalui fungsi seperti viewModel.onFilmClicked(film). Dengan demikian, adapter ini tetap berperan dalam menghubungkan UI dengan ViewModel secara tidak langsung melalui interface callback dan logging Timber berfungsi untuk memantau jalannya proses tersebut.

HomeFragment.kt

Pada file ini, digunakan ViewModel HomeViewModel untuk menyimpan dan mengelola data film yang akan ditampilkan pada RecyclerView. ViewModel diinisialisasi menggunakan sintaks by viewModels { HomeViewModelFactory("Guest") }. Sintaks ini digunakan untuk membuat instance ViewModel yang menerima parameter userName, yaitu "Guest" melalui kelas HomeViewModelFactory.

Setelah tampilan dibuat, fungsi viewModel.setFilmList(getListFilm()) dipanggil untuk mengirim data film ke dalam ViewModel. Data yang disimpan di ViewModel tersebut kemudian diamati melalui viewModel.filmList.observe(viewLifecycleOwner). Apabila data berubah, maka fungsi setupRecyclerView(list) akan dipanggil kembali untuk memperbarui daftar film yang ditampilkan.

Selain LiveData, ViewModel ini juga menggunakan StateFlow untuk mengirimkan satu objek film yang diklik. Observer viewLifecycleOwner.lifecycleScope.launch digunakan untuk menjalankan coroutine, dan di dalamnya terdapat viewModel.selectedFilm.collectLatest { ... } yang memantau jika ada film yang dikirim dari ViewModel. Jika film tersedia, maka pengguna akan diarahkan ke halaman detail menggunakan fungsi navigateToDetail(it), dan nilai selectedFilm akan di-reset menggunakan viewModel.clearSelectedFilm() agar tidak terus-menerus memicu navigasi ulang.

Di dalam fungsi setupRecyclerView, ViewModel juga berperan saat tombol detail ditekan. Fungsi viewModel.onFilmClicked(film) dipanggil ketika pengguna mengklik tombol "Detail", dan film tersebut dikirim ke ViewModel agar bisa ditangkap oleh observer collectLatest tadi.

Kemudian, penggunaan Timber terdapat pada fungsi navigateToDetail. Log debug dibuat menggunakan Timber.d(...) yang menampilkan informasi film yang sedang dinavigasi, termasuk title dan year. Fungsi Timber ini digunakan sebagai alternatif dari Log.d(), dengan keunggulan penulisan yang lebih ringkas dan mendukung manajemen log yang lebih baik saat proses debugging.

MainActivity.kt

File ini merupakan entry point aplikasi. Di dalam metode onCreate, fragment manager digunakan untuk menambahkan HomeFragment ke dalam frame_container hanya jika fragment tersebut belum ada. HomeFragment::class.java.simpleName digunakan sebagai tag untuk menghindari penambahan ulang saat orientasi berubah atau aktivitas dibuat ulang.

HomeViewModel.kt

HomeViewModel bertugas menyimpan dan mengelola data film agar tetap bertahan saat rotasi layar atau perubahan konfigurasi. MutableLiveData digunakan untuk menyimpan list film, dan diakses dari luar melalui LiveData agar tidak bisa diubah secara langsung.

Untuk film yang diklik, digunakan MutableStateFlow agar bisa dikonsumsi dengan coroutine pada HomeFragment. Fungsi onFilmClicked digunakan untuk menetapkan film yang sedang dipilih, dan clearSelectedFilm dipanggil untuk menghapus nilai tersebut setelah digunakan agar tidak diproses ulang.

Pada blok init, Timber.d(...) mencatat bahwa ViewModel berhasil dibuat dan mencetak nama pengguna yang diteruskan dari HomeViewModelFactory.

Setiap kali daftar film diatur melalui setFilmList, log tambahan dibuat menggunakan Timber.d untuk setiap film dalam daftar. Ini membantu proses debugging dengan menunjukkan data yang sedang diproses.

HomeViewModelFactory.kt

HomeViewModelFactory digunakan untuk membuat instance HomeViewModel yang memerlukan parameter tambahan userName. Karena ViewModel secara default tidak menerima parameter di konstruktor, maka diperlukan Factory untuk meneruskan parameter tersebut. Pada saat HomeFragment memanggil by viewModels { HomeViewModelFactory("Guest") }, sistem akan memanggil metode create di sini.

Pembahasan Compose

FilmListItems.kt

Di dalam FilmListItems, viewModel dipakai untuk menangani logika ketika tombol diklik. Saat tombol "IMDB" ditekan, fungsi viewModel.logImdbClick(films) dipanggil. Ini berarti bahwa FilmViewModel memiliki fungsi bernama logImdbClick yang tujuannya mencatat log ketika tombol tersebut ditekan, kemungkinan dengan Timber untuk debugging.

Begitu pula dengan tombol "Detail", yang memanggil dua fungsi ViewModel: selectFilm(films) untuk menyimpan film yang dipilih, dan logDetailClick(films) yang digunakan untuk mencatat log saat tombol "Detail" ditekan. Penggunaan Timber tidak ditampilkan langsung dalam file ini, tetapi bisa diasumsikan bahwa logging terjadi di dalam fungsi ViewModel yang dipanggil. Semua aksi user seperti klik tombol tidak ditangani langsung oleh UI, tetapi didelegasikan ke ViewModel, sesuai prinsip pemisahan logika dan tampilan (MVVM).

HomeContent.kt

Pada HomeContent, pengambilan data film dilakukan melalui filmViewModel.filmList.collectAsState(). Ini berarti filmList adalah sebuah StateFlow yang dideklarasikan di dalam ViewModel, dan Compose akan otomatis merender ulang UI ketika data tersebut berubah. Nilai films ini kemudian diteruskan ke fungsi FilmListItems.

ViewModel menjadi satu-satunya sumber data untuk menampilkan daftar film, dan Compose hanya bertugas merender. Tidak ada logika manipulasi data dalam UI — semua sudah dikelola di ViewModel.

MainActivity.kt

Di dalam MainActivity, Timber diinisialisasi pertama kali dengan Timber.plant(Timber.DebugTree()). Pemanggilan ini berfungsi untuk mengaktifkan pencatatan log selama proses debugging. Semua log yang ditulis dengan Timber.d() di kelas lain seperti ViewModel akan muncul di logcat Android Studio jika aplikasi dalam mode debug. Selanjutnya, FilmViewModel dibuat menggunakan viewModel() dengan parameter viewModelFactory, yang diisi dengan string "User123". Ini artinya, ViewModel akan menyimpan nama pengguna tersebut dan mencatat log-log dengan menyertakan nama pengguna itu. Di dalam fungsi setContent, ViewModel ini diteruskan ke composable GhibliFilms.

Composable ini mendefinisikan navigasi dengan NavHost, dan filmViewModel digunakan untuk menyediakan data di HomeContent serta mengambil objek film berdasarkan title untuk DetailContent. ViewModel bertanggung jawab sepenuhnya terhadap data, sementara composable hanya menampilkan data dan menangani navigasi.

ViewModel.kt

FilmViewModel menyimpan dua properti utama dalam bentuk StateFlow: _filmList yang menyimpan daftar film dari DataProvider, dan _selectedFilm yang menyimpan film yang sedang dipilih. StateFlow memungkinkan UI Compose untuk terus mengamati perubahan secara reaktif.

Pada blok init, log pertama dicetak menggunakan Timber untuk mencatat saat ViewModel diinisialisasi. Log tersebut menyebutkan nama user dan jumlah film yang dimuat dari DataProvider. Fungsi selectFilm, logDetailClick, dan logImdbClick masing-masing mencatat log saat pengguna memilih film, mengklik tombol detail, atau tombol IMDB. Semua log ini menyertakan userName, sehingga setiap interaksi pengguna bisa ditelusuri di log dengan lebih mudah. Fungsi getFilmByTitle juga disediakan untuk mengambil objek film berdasarkan judul, biasanya dipakai di halaman detail. ViewModel ini tidak hanya menyimpan state aplikasi, tapi juga memantau aktivitas pengguna melalui Timber secara terstruktur.

ViewModelFactory.kt

FilmViewModelFactory adalah class yang mengimplementasikan ViewModelProvider.Factory. Kelas ini bertugas menyediakan instance dari FilmViewModel dengan parameter tambahan userName, yang tidak bisa langsung disediakan oleh viewModel() tanpa factory. Di dalam fungsi create, ada pemeriksaan apakah kelas yang diminta adalah FilmViewModel, dan jika sesuai, objek baru dikembalikan dengan userName yang sudah diberikan sebelumnya. Penggunaan factory ini penting karena FilmViewModel membutuhkan parameter konstruktor, sedangkan ViewModel default hanya mendukung konstruktor kosong.

SOAL 2

Pembahasan

Application class merupakan kelas dasar yang mewakili keseluruhan siklus hidup aplikasi Android. Kelas ini akan dibuat pertama kali saat aplikasi dijalankan, dan tetap aktif selama aplikasi belum ditutup sepenuhnya. Peran utamanya adalah untuk melakukan inisialisasi terhadap komponen global yang hanya perlu dijalankan satu kali, seperti konfigurasi library logging (contohnya Timber), inisialisasi dependency injection (seperti Hilt atau Dagger), atau pengaturan database. Selain itu, Application class juga bisa digunakan untuk menyimpan data atau status yang bersifat global dan dibutuhkan di berbagai bagian aplikasi.

Untuk menggunakannya, developer cukup membuat turunan dari kelas Application lalu mendaftarkannya di AndroidManifest.xml agar sistem Android mengenali kelas tersebut sebagai titik awal aplikasi. Penggunaan Application class sangat membantu dalam mengelola data global dan membuat struktur aplikasi menjadi lebih rapi dan efisien.

LINK GITHUB

https://github.com/raymondhariyono/mobile-praktikum/tree/main/modul3