LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

VIEWMODEL AND DEBUGGING

Oleh:

Damarjati Suryo Laksono **NIM. 2310817210014**

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 : Damarjati Suryo Laksono

NIM : 2310817210014

Menyetujui,	Mengetahui,
Asisten Praktikum	Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar	Muti`a Maulida S.Kom M.T.I
NIM. 2210817210026	NIP. 19881027 201903 20 13

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

\

SOAL 1

Soal Praktikum:

1. Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:
    a. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
    b. Gunakan ViewModelFactory dalam pembuatan ViewModel
    c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
    d. gunakan logging untuk 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.

A. Source Code

1. MainActivity.kt

```
1  package com.example.myrecylerview
2
3  import android.os.Bundle
4  import android.widget.Toast
5  import androidx.appcompat.app.AppCompatActivity
6  import androidx.appcompat.app.AppCompatDelegate
7  import
   androidx.core.splashscreen.SplashScreen.Companion.installSpla
   shScreen
```

```
8  import
   com.example.myrecylerview.databinding.ActivityMainBinding
9
10 class MainActivity : AppCompatActivity() {
11
12     private lateinit var binding: ActivityMainBinding
13
14     override fun onCreate(savedInstanceState: Bundle?) {
15
16 AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_
   NIGHT_NO)
17         installSplashScreen()
18
19         super.onCreate(savedInstanceState)
20         binding = ActivityMainBinding.inflate(layoutInflater)
           setContentView(binding.root)
21
22         binding.btnMenu.setOnClickListener {
23             Toast.makeText(this, "This Button is used for
24 Decoration", Toast.LENGTH_SHORT).show()
           }
25     }
26 }
```

Tabel 1 Source Code MainActivity.kt

2. DetailFragment.kt

```
1  package com.example.myrecylerview
2
3  import android.content.Intent
4  import android.os.Bundle
5  import android.view.LayoutInflater
6  import android.view.View
7  import android.view.ViewGroup
```

```
8  import androidx.fragment.app.Fragment
9  import androidx.navigation.fragment.findNavController
   import
10 com.example.myrecylerview.databinding.FragmentDetailBinding
11
12 class DetailFragment : Fragment() {
13
14     private var _binding: FragmentDetailBinding? = null
15     private val binding get() = _binding!!
16
17     override fun onCreateView(
18         inflater: LayoutInflater, container: ViewGroup?,
19         savedInstanceState: Bundle?
20     ): View {
           _binding = FragmentDetailBinding.inflate(inflater,
21 container, false)
22         return binding.root
23     }
24
25     override fun onViewCreated(view: View,
   savedInstanceState: Bundle?) {
26         super.onViewCreated(view, savedInstanceState)
27
28         val args =
   DetailFragmentArgs.fromBundle(requireArguments())
29         val photo = args.extraPhoto
30         val link = args.extraLink
31         val detail = args.extraDetail
32
33         binding.detailImage.setImageResource(photo)
34         binding.detailDescription.text = detail
35
36         binding.btnBack.setOnClickListener {
37             findNavController().navigateUp()
```

```
38          }
39
40          binding.btnShare.setOnClickListener {
41              val shareText = buildString {
42                  append("Check this out!\n\n$detail\n\nMore
    info: $link")
43              }
44
45              val shareIntent =
    Intent(Intent.ACTION_SEND).apply {
46                  type = "text/plain"
47                  putExtra(Intent.EXTRA_TEXT, shareText)
48              }
49
50              startActivity(Intent.createChooser(shareIntent,
    "Share via"))
51          }
52      }
53
54      override fun onDestroyView() {
55          super.onDestroyView()
56          _binding = null
57      }
58  }
```

Tabel 2 Source Code DetailFragment.kt

3. MyAdapter.kt

```
1   package com.example.myrecylerview
2
3   import android.view.LayoutInflater
4   import android.view.ViewGroup
5   import androidx.recyclerview.widget.RecyclerView
6   import com.example.myrecylerview.databinding.ItemLayoutBinding
```

```
7
8   class MyAdapter(
9       private val onYTClick: (String) -> Unit,
10      private val onDetailClick: (String, Int, String) -> Unit
11  ) : RecyclerView.Adapter<MyAdapter.ListViewHolder>() {
12
13      private val items = ArrayList<MyData>()
14
15      fun submitList(newList: List<MyData>) {
16          items.clear()
17          items.addAll(newList)
18          notifyDataSetChanged()
19      }
20
21      class ListViewHolder(val binding: ItemLayoutBinding) :
        RecyclerView.ViewHolder(binding.root)
22
23      override fun onCreateViewHolder(parent: ViewGroup,
        viewType: Int): ListViewHolder {
24          val binding =
        ItemLayoutBinding.inflate(LayoutInflater.from(parent.context),
        parent, false)
25          return ListViewHolder(binding)
26      }
27
28      override fun getItemCount(): Int = items.size
29
30      override fun onBindViewHolder(holder: ListViewHolder,
        position: Int) {
31          val (name, link, photo, detail, subtext) =
        items[position]
32          with(holder.binding) {
33              textTitle.text = name
34              textDescription.text = subtext
```

```
35              itemImage.setImageResource(photo)
36              btnWebsite.setOnClickListener { onYTClick(link) }
37              btnDetails.setOnClickListener {
    onDetailClick(detail, photo, link) }
38          }
39      }
40  }
```

Tabel 3 Source Code MyAdapter.kt

4. MyFragment.kt

```
1   package com.example.myrecylerview
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.lifecycle.ViewModelProvider
11  import androidx.lifecycle.lifecycleScope
12  import androidx.navigation.fragment.findNavController
13  import androidx.recyclerview.widget.LinearLayoutManager
14  import
    com.example.myrecylerview.databinding.FragmentMyBinding
15
16
17  class MyFragment : Fragment() {
18
19      private var _binding: FragmentMyBinding? = null
20      private val binding get() = _binding!!
21
```

```
22      private lateinit var viewModel: MyViewModel
23      private lateinit var characterAdapter: MyAdapter
24
25      override fun onCreateView(
26          inflater: LayoutInflater, container: ViewGroup?,
27          savedInstanceState: Bundle?
28      ): View {
29          _binding = FragmentMyBinding.inflate(inflater,
    container, false)
30          return binding.root
31      }
32
33      override fun onViewCreated(view: View,
    savedInstanceState: Bundle?) {
34          super.onViewCreated(view, savedInstanceState)
35
36          viewModel = ViewModelProvider(
37              this,
38              MyViewModelFactory(requireActivity().application)
            )[MyViewModel::class.java]
39
40          characterAdapter = MyAdapter(
41              onYTClick = { link -> viewModel.onYTClick(link)
42      },
43              onDetailClick = { detail, photo, link ->
    viewModel.onDetailClick(detail, photo, link) }
44          )
45
46          binding.rvCharacter.apply {
47              layoutManager =
    LinearLayoutManager(requireContext())
48              adapter = characterAdapter
49              setHasFixedSize(true)
50          }
```

```
51
52          lifecycleScope.launchWhenStarted {
53              viewModel.characterList.collect { list ->
54                  characterAdapter.submitList(list)
55              }
56          }
57
58          lifecycleScope.launchWhenStarted {
59              viewModel.uiEvent.collect { event ->
60                  when (event) {
61                      is MyViewModel.UiEvent.OpenYouTube -> {
62                          val intent =
63  Intent(Intent.ACTION_VIEW, Uri.parse(event.link))
                            startActivity(intent)
64                      }
65                      is MyViewModel.UiEvent.NavigateToDetail -
66  > {
                            val action = MyFragmentDirections
67
68  .actionMyFragmentToDetailFragment(event.photo, event.link,
69  event.detail)
                            findNavController().navigate(action)
70                      }
71                      null -> Unit
72                  }
73                  viewModel.clearEvent()
74              }
75          }
76
77          viewModel.loadCharacterList()
78      }
79
80      override fun onDestroyView() {
81          super.onDestroyView()
```

```
82          _binding = null
83      }
84  }
```

Tabel 4 Source Code MyFragment.kt

5. MyData.kt

```
1   package com.example.myrecylerview
2
3   import android.os.Parcelable
4   import kotlinx.parcelize.Parcelize
5
6   @Parcelize
7   data class MyData(
8       val name: String,
9       val link: String,
10      val photo: Int,
11      val detail: String,
12      val subtext: String
13  ):Parcelable
```

Tabel 5 Source Code MyData.kt

6. MyViewModel.kt

```
1   package com.example.myrecylerview
2
3   import android.app.Application
4   import android.util.Log
5   import androidx.lifecycle.AndroidViewModel
6   import kotlinx.coroutines.flow.MutableStateFlow
7   import kotlinx.coroutines.flow.StateFlow
8
9   class MyViewModel(application: Application) :
    AndroidViewModel(application) {
```

```
10
11      private val _characterList =
12  MutableStateFlow<List<MyData>>(emptyList())
        val characterList: StateFlow<List<MyData>> =
13  _characterList

14      private val _uiEvent = MutableStateFlow<UiEvent?>(null)
15      val uiEvent: StateFlow<UiEvent?> = _uiEvent
16
17      fun loadCharacterList() {
18          val context = getApplication<Application>()
19          val resources = context.resources
20
21          val dataName =
    resources.getStringArray(R.array.data_name)
22          val dataLink =
    resources.getStringArray(R.array.data_link)
23          val dataSubtext =
    resources.getStringArray(R.array.data_subtext)
24          val dataPhoto =
    resources.obtainTypedArray(R.array.data_photo)
25          val dataDetail =
    resources.getStringArray(R.array.data_detail)
26
27          val listCharacter = List(dataName.size) { i ->
28              MyData(
29                  name = dataName[i],
30                  link = dataLink[i],
31                  subtext = dataSubtext[i],
32                  detail = dataDetail[i],
33                  photo = dataPhoto.getResourceId(i, -1)
34              )
35          }
36
```

```
        dataPhoto.recycle()
        _characterList.value = listCharacter

        Log.d("MyViewModel", "Loaded ${listCharacter.size}
characters: $listCharacter")
    }

    fun onYTClick(link: String) {
        Log.d("MyViewModel", "YouTube button clicked with
link: $link")
        _uiEvent.value = UiEvent.OpenYouTube(link)
    }

    fun onDetailClick(detail: String, photo: Int, link:
String) {
        Log.d("MyViewModel", "Detail button clicked with data
-> detail: $detail, photo: $photo, link: $link")
        _uiEvent.value = UiEvent.NavigateToDetail(photo,
link, detail)
    }

    fun clearEvent() {
        _uiEvent.value = null
    }

    sealed class UiEvent {
        data class OpenYouTube(val link: String) : UiEvent()
        data class NavigateToDetail(val photo: Int, val link:
String, val detail: String) : UiEvent()
    }
}
```

Tabel 6 Source Code MyViewModel.kt

7. MyViewModelFactory.kt

```
1  package com.example.myrecylerview
2
3  import android.app.Application
4  import androidx.lifecycle.ViewModel
5  import androidx.lifecycle.ViewModelProvider
6
7  class MyViewModelFactory(private val application:
   Application) : ViewModelProvider.Factory {
8      override fun <T : ViewModel> create(modelClass:
   Class<T>): T {
9          if
   (modelClass.isAssignableFrom(MyViewModel::class.java)) {
10             return MyViewModel(application) as T
11         }
12         throw IllegalArgumentException("Unknown ViewModel
   class")
13     }
14
15 }
```

Tabel 7 Source Code MyViewModelFactory.kt

8. activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:background="@android:color/white"
8      tools:context=".MainActivity">
9
```

```
10      <com.google.android.material.appbar.MaterialToolbar
11          android:id="@+id/toolbar"
12          android:layout_width="match_parent"
13          android:layout_height="?attr/actionBarSize"
14          android:background="@android:color/white"
15          android:elevation="4dp"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent"
19          app:title="Owi Core"
20          app:titleTextColor="@android:color/black"
21
22  app:titleTextAppearance="@style/TextAppearance.MaterialCompone
23  nts.Headline6">
24
25          <ImageView
26              android:id="@+id/btn_menu"
27              android:layout_width="24dp"
28              android:layout_height="24dp"
29              android:layout_gravity="end"
30              android:layout_marginEnd="16dp"
31              android:src="@drawable/ic_more_vert"
32              app:tint="@android:color/black" />
33
34      </com.google.android.material.appbar.MaterialToolbar>
35
36      <androidx.fragment.app.FragmentContainerView
37          android:id="@+id/nav_host_fragment"
38
39  android:name="androidx.navigation.fragment.NavHostFragment"
40          android:layout_width="0dp"
41          android:layout_height="0dp"
42          app:layout_constraintTop_toBottomOf="@id/toolbar"
43          app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:navGraph="@navigation/nav_graph"
        app:defaultNavHost="true" />

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

Tabel 8 Source Code activity_main.xml

9. fragment_detail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.core.widget.NestedScrollView

xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true">

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

        <ImageButton
            android:id="@+id/btn_back"
            android:layout_width="48dp"
            android:layout_height="48dp"

android:background="?selectableItemBackgroundBorderless"
            android:src="@drawable/ic_arrow_back"
            app:layout_constraintStart_toStartOf="parent"
```

```
            app:layout_constraintTop_toTopOf="parent"
            app:tint="@android:color/black" />


<com.google.android.material.imageview.ShapeableImageView
            android:id="@+id/detail_image"
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:scaleType="centerCrop"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0"
            android:layout_marginTop="32dp"
            android:elevation="4dp"

android:transitionName="shared_image_container"/>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="24dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/detail_image"
            android:layout_marginTop="24dp">

            <androidx.cardview.widget.CardView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                app:cardCornerRadius="16dp"
                app:cardElevation="4dp"
```

```
57
58  app:cardBackgroundColor="@android:color/white"
59                  android:layout_marginBottom="16dp">
60
61                  <TextView
62                      android:id="@+id/detail_description"
63                      android:layout_width="match_parent"
64                      android:layout_height="wrap_content"
65                      android:textSize="16sp"
66                      android:lineSpacingMultiplier="1.2"
67                      android:textColor="@android:color/black"
68                      android:padding="24dp"
69                      tools:text="Lorem ipsum dolor sit amet,
70  consectetur adipiscing elit. Sed do eiusmod tempor
71  incididunt ut labore et dolore magna aliqua. Ut enim ad
72  minim veniam, quis nostrud exercitation ullamco laboris nisi
73  ut aliquip ex ea commodo consequat."/>
74
75              </androidx.cardview.widget.CardView>
76
77              <LinearLayout
78                  android:layout_width="match_parent"
79                  android:layout_height="wrap_content"
80                  android:orientation="horizontal"
81                  android:gravity="center_horizontal"
82                  android:spacing="16dp">
83
84
85  <com.google.android.material.button.MaterialButton
86                      android:id="@+id/btn_share"
87
88  style="@style/Widget.Material3.Button.OutlinedButton"
89                      android:layout_width="match_parent"
90                      android:layout_height="48dp"
```

```
91                      android:text="Share"
92                      app:icon="@drawable/ic_share"
93                      app:iconTint="@null"
94                      app:cornerRadius="8dp"/>
95
96              </LinearLayout>
97
98          </LinearLayout>
99
100     </androidx.constraintlayout.widget.ConstraintLayout>
101
102 </androidx.core.widget.NestedScrollView>
```

Tabel 9 Source Code fragment_detail.xml

10. fragment_my.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       tools:context=".MyFragment">
8
9
10      <androidx.recyclerview.widget.RecyclerView
11          android:id="@+id/rv_character"
12          android:layout_width="0dp"
13          android:layout_height="0dp"
14          app:layout_constraintBottom_toBottomOf="parent"
15          app:layout_constraintEnd_toEndOf="parent"
16          app:layout_constraintStart_toStartOf="parent"
```

```
17          app:layout_constraintTop_toTopOf="parent" />
18  </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 10 Source Code fragment_my.xml

11. item_layout.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView

3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="wrap_content"
8       android:layout_margin="2dp"
9       app:cardCornerRadius="12dp"
10      app:cardElevation="4dp"
11      app:cardUseCompatPadding="true">
12
13      <androidx.constraintlayout.widget.ConstraintLayout
14          android:layout_width="match_parent"
15          android:layout_height="wrap_content"
16          android:padding="16dp">
17
18          <ImageView
19              android:src="@drawable/pakdhe_1"
20              android:id="@+id/item_image"
21              android:layout_width="64dp"
22              android:layout_height="64dp"
23              android:scaleType="centerCrop"
24              app:layout_constraintBottom_toBottomOf="parent"
25              app:layout_constraintStart_toStartOf="parent"
26              app:layout_constraintTop_toTopOf="parent"
```

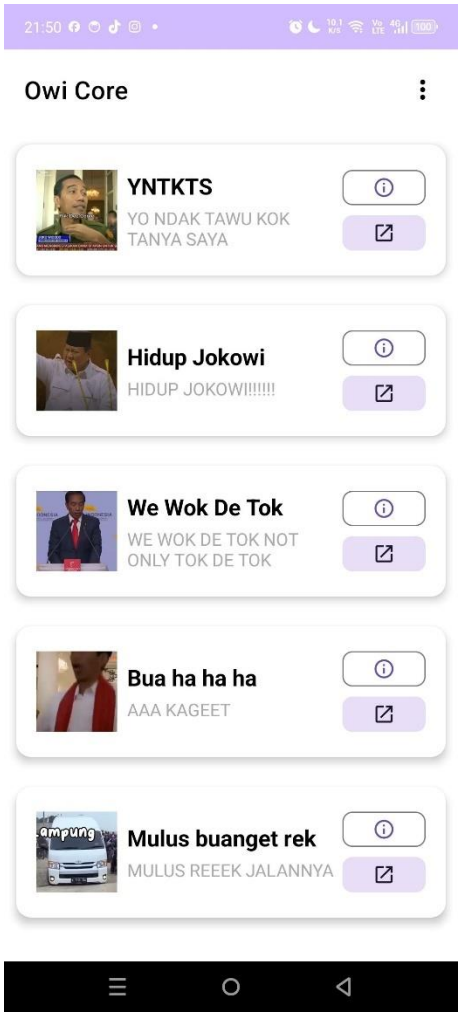
```
27              android:layout_marginEnd="16dp"/>
28
29          <LinearLayout
30              android:layout_width="0dp"
31              android:layout_height="wrap_content"
32              android:orientation="vertical"
33
34  app:layout_constraintBottom_toBottomOf="@id/item_image"
35
36  app:layout_constraintEnd_toStartOf="@+id/button_group"
37
38  app:layout_constraintStart_toEndOf="@id/item_image"
39
40  app:layout_constraintTop_toTopOf="@id/item_image">
41
42              <TextView
43                  android:id="@+id/text_title"
44                  android:layout_width="match_parent"
45                  android:layout_height="wrap_content"
46                  android:layout_marginStart="8dp"
47                  android:ellipsize="end"
48                  android:maxLines="1"
49                  android:textColor="@android:color/black"
50                  android:textSize="18sp"
51                  android:textStyle="bold"
52                  tools:text="Main Title" />
53
54              <TextView
55                  android:id="@+id/text_description"
56                  android:layout_width="match_parent"
57                  android:layout_height="wrap_content"
58                  android:layout_marginStart="8dp"
59                  android:layout_marginTop="4dp"
60                  android:ellipsize="end"
```

```
                android:maxLines="2"

android:textColor="@android:color/darker_gray"
                android:textSize="14sp"
                tools:text="Secondary description text that
can span multiple lines" />
        </LinearLayout>

        <LinearLayout
            android:id="@+id/button_group"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:gravity="end"
            android:spacing="8dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent">


<com.google.android.material.button.MaterialButton
                android:id="@+id/btn_details"

style="@style/Widget.Material3.Button.OutlinedButton"
                android:layout_width="wrap_content"
                android:layout_height="36dp"
                android:minWidth="64dp"
                app:icon="@drawable/ic_info_outline"
                app:iconPadding="0dp"
                app:iconGravity="textStart"
                app:cornerRadius="8dp"/>


<com.google.android.material.button.MaterialButton
```

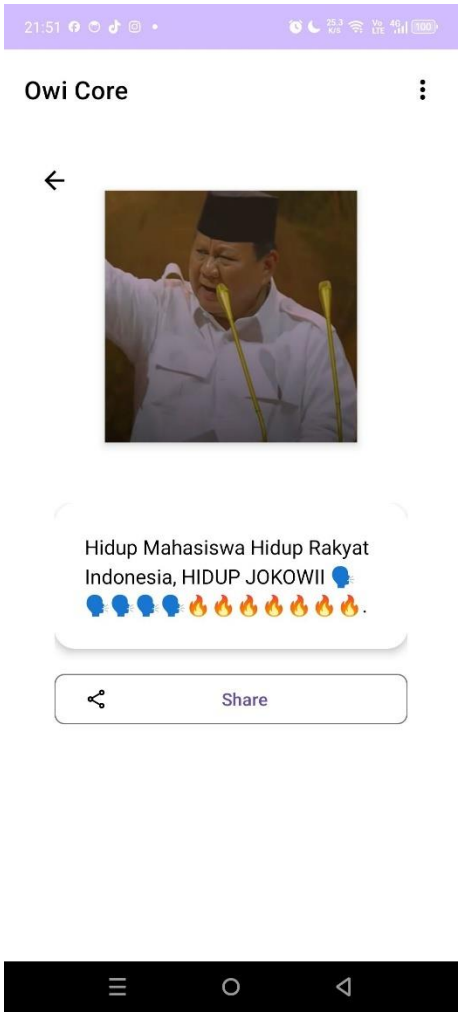
```
95                  android:id="@+id/btn_website"
96
97  style="@style/Widget.Material3.Button.TonalButton"
98                  android:layout_width="wrap_content"
99                  android:layout_height="36dp"
100                 android:minWidth="64dp"
101                 app:icon="@drawable/ic_open_in_browser"
102                 app:iconPadding="0dp"
103                 app:iconGravity="textStart"
104                 app:cornerRadius="8dp"/>
105         </LinearLayout>
106
107     </androidx.constraintlayout.widget.ConstraintLayout>
108 </androidx.cardview.widget.CardView>
```

Tabel 11 Source Code item_layout.xml

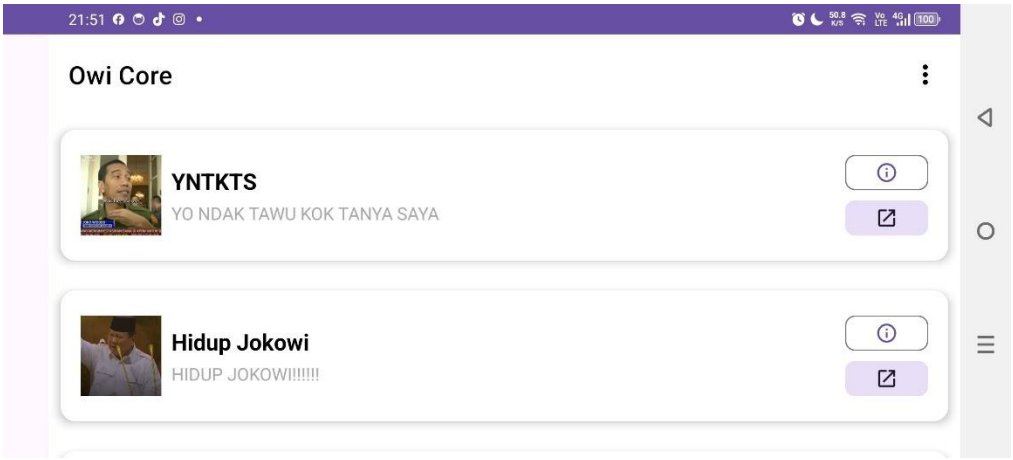
B. Output Program



Gambar 1 Screenshot Aplikasi



Gambar 2 Screenshot Hasil Aplikasi

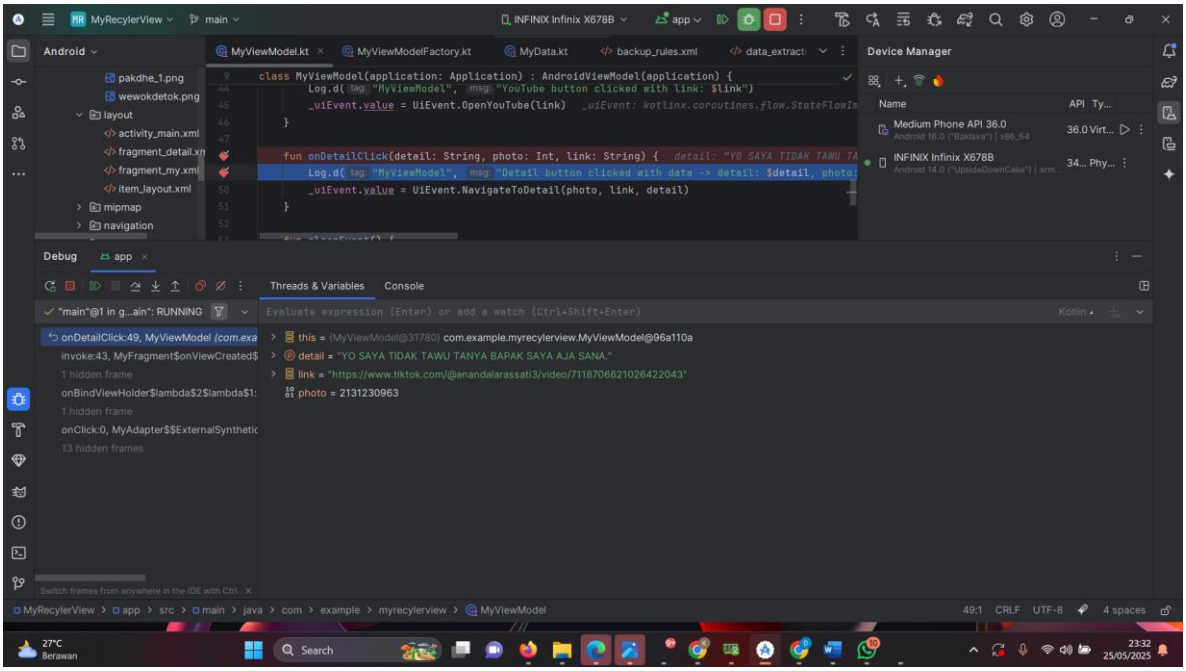


Gambar 3 Screenshot Hasil Aplikasi

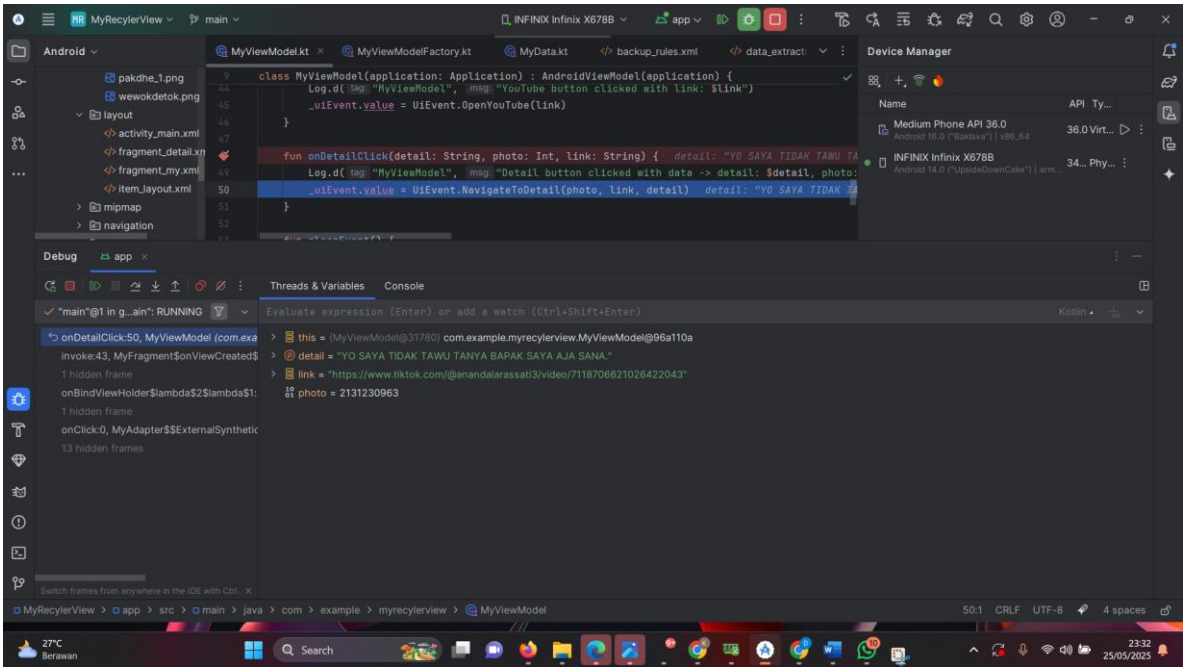
Gambar 4 Screenshot Logcat Loading Data

Gambar 5 Screenshot Info Button Click

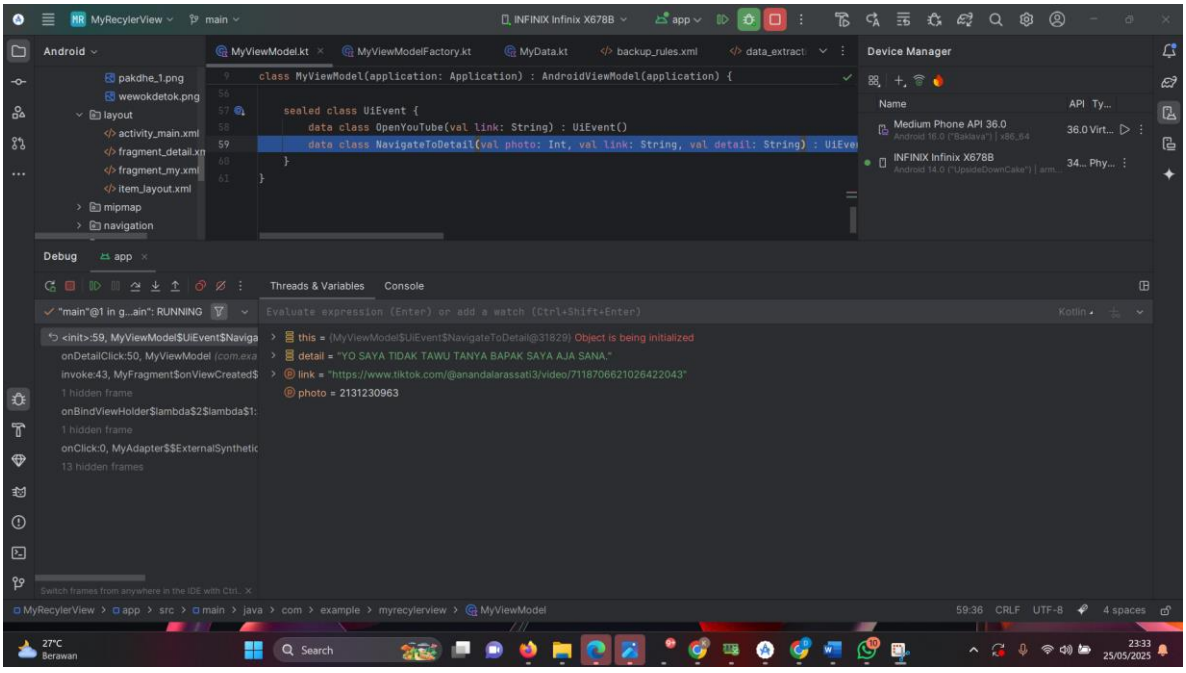
Gambar 6 Screenshot Youtube Button Cick(Link)



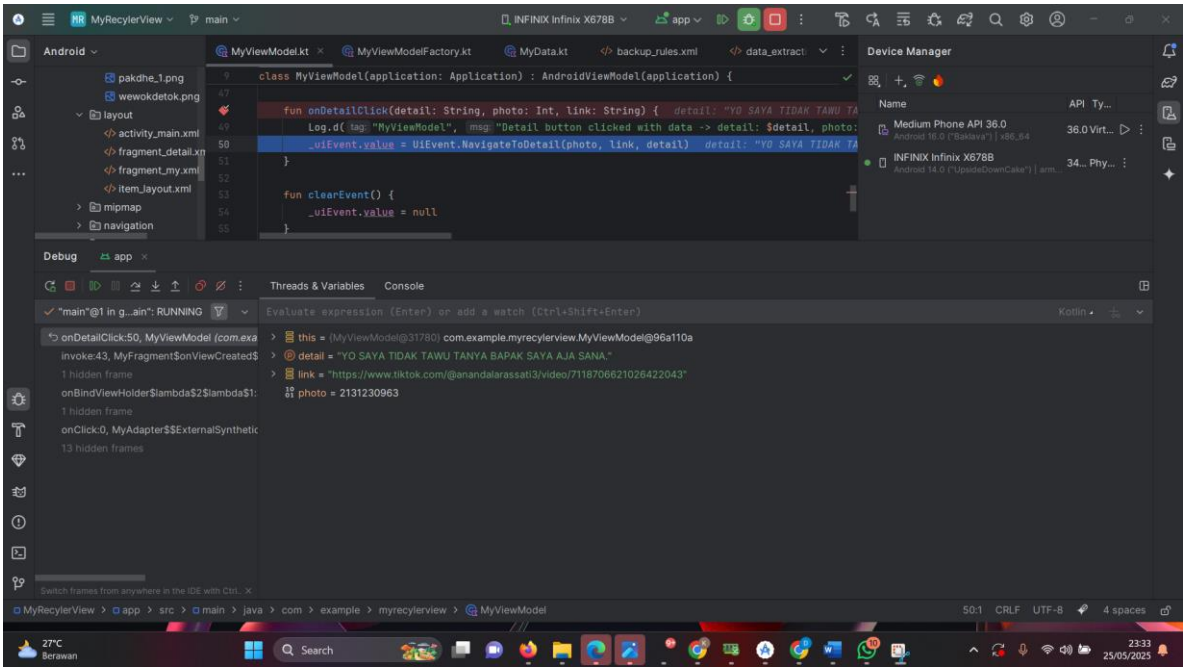
Gambar 7 Screenshot Debugger (ONYTCLICK)



Gambar 8 Screenshot Debugger(Step Over)



Gambar 9 Screenshot Debugger(Step Into)



Gambar 10 Screenshot Debugger(Step Out)

C. Pembahasan

1. MainActivity.kt

Pada modul kali ini MainActivity hanya digunakan sebagai wadah untuk fragment-fragment yang ada, sehingga kode pada MainActivity berfokus hanya pada instalasi splash screen, dan juga karena start destination sudah di set pada MyFragment, jadi secara kesuluran MainActivtiy ada hanya sebagai pelengkap dari android manifest itu sendiri.

2. DetailFragment.kt

Overall kegunaan fragment ini hanya sebagai penunjuk detail saja, pada filenya logic yang digunakan hanya menerima data yang di passing oleh adapter seperti pada line [28]-[31] , dan melakukan pergantian image atau ui dari fragment_detail seperti pada line [33]-[38] saja, dan juga beberapa button untuk share, dan kembali pada line [40]-[50].

3. MyAdapter.kt

Oke, Logic dari RecyclerView dimulai pada file ini, dari baris [9] hingga [12], di mana terdapat inisialisasi beberapa parameter, yaitu listCharacter, onYTClick, dan onDetailClick. Parameter listCharacter merupakan data utama yang akan ditampilkan dalam RecyclerView. Parameter onYTClick dan onDetailClick adalah fungsi lambda yang akan dieksekusi ketika tombol tertentu ditekan.

Selanjutnya, pada baris [14] hingga [22], terdapat struktur utama dari RecyclerView Adapter. Pada bagian ini, kelas ListViewHolder digunakan untuk mendefinisikan bahwa setiap elemen yang ditampilkan merupakan bagian dari RecyclerView. Kelas ini menerima parameter berupa ItemLayoutBinding, yang menghubungkan layout XML dengan kode Kotlin.

Fungsi onCreateViewHolder digunakan untuk membuat tampilan item menggunakan LayoutInflater, dengan menghubungkannya ke ItemLayoutBinding. Ini berarti adapter akan membuat tampilan sesuai template yang sudah didefinisikan di file XML.

Fungsi getItemCount mengembalikan jumlah item yang ada dalam listCharacter, yang berarti menyesuaikan jumlah data yang akan ditampilkan dalam daftar.

Pada bagian akhir, yaitu baris [23] sampai selesai, terdapat fungsi onBindViewHolder. Fungsi ini bertanggung jawab untuk binding data ke dalam tampilan. Artinya, setiap elemen seperti textTitle, textDescription, dan itemImage akan diisi dengan data dari objek yang sesuai di dalam listCharacter. Selain itu, fungsi ini juga menetapkan aksi ketika tombol btnWebsite dan btnDetails ditekan, dengan menjalankan fungsi onYTClick dan onDetailClick yang sudah didefinisikan sebelumnya.

4. MyFragment.kt

Oke, pada file ini, MyFragment merupakan fragment pertama yang akan dijumpai saat membuka aplikasi, karena sudah diatur di nav_graph. Pada baris 17 hingga 21 merupakan template (sering disebut boilerplate) untuk inisialisasi binding dan adapter.

Selanjutnya, pada baris 23 sampai 32 terdapat fungsi onCreateView. Pertama, fungsi ini akan melakukan clear terhadap list yang sudah ada. Setelah itu, data baru akan dimasukkan ke dalam listCharacter, dan kemudian akan memanggil fungsi setupRecyclerView.

Pada baris 34 sampai 52 terdapat logika dari tombol-tombol yang ada. Pada onYTClick, jika tombol ditekan, maka akan dilakukan navigasi ke link yang telah disediakan menggunakan Intent. Sementara itu, pada onDetailClick, data akan dipassing ke DetailFragment menggunakan MyFragmentDirections.

Kemudian, terdapat binding.rvCharacter yang digunakan untuk menerapkan RecyclerView, dengan menggunakan LinearLayoutManager dan menetapkan adapter.

Pada baris 55 hingga 74, terdapat fungsi getListCharacter yang digunakan untuk menjalankan RecyclerView itu sendiri. Fungsi ini mengambil data dari resources, seperti array nama, link, subtext, photo, dan detail, lalu dimasukkan ke dalam ArrayList bertipe MyData. Setelah selesai, TypedArray akan di-recycle.

Terakhir, pada fungsi onDestroyView, dilakukan pembersihan terhadap objek binding agar tidak terjadi memory leak.

5. MyData.kt

Wadah untuk melakukan pendefinisian tipe variable yang digunakan, serta penyimpanan data itu sendiri

6. MyViewModel.kt

MyViewModel adalah turunan dari AndroidViewModel yang mengelola data dan event UI secara terpisah dari fragment. Di dalamnya, terdapat dua MutableStateFlow: satu untuk menyimpan daftar karakter (_characterList) dan satu lagi untuk event UI tunggal (_uiEvent), masing-masing diekspos sebagai StateFlow hanya-baca agar fragment dapat mengamati perubahan tanpa memodifikasi langsung.

Fungsi loadCharacterList() mengambil konteks aplikasi untuk membaca resource array—nama, link YouTube, subteks, detail, dan foto—kemudian membangun List<MyData> dengan memetakan tiap indeks ke objek MyData yang sesuai, setelah itu me-recycle() TypedArray foto untuk mencegah memory leak dan mengubah nilai _characterList, sehingga UI ter-update secara reaktif.

Ketika tombol YouTube ditekan, onYTClick(link) mencatat log dan memancarkan UiEvent.OpenYouTube(link), sementara onDetailClick(detail, photo, link) mencatat log serupa lalu memancarkan UiEvent.NavigateToDetail(photo, link, detail) untuk menginstruksi fragment agar menavigasi ke detail screen melalui Safe Args. Setelah event ditangani, clearEvent() dipanggil untuk mengatur ulang _uiEvent menjadi null, mencegah pengulangan event saat re-collect berlangsung. Dengan pola ini, MyViewModel menjaga UI tetap bersih dari logika pemrosesan data dan navigasi, memanfaatkan StateFlow yang lifecycle-aware untuk memudahkan pengujian dan menghindari memory leak.

7. MyViewModelFactory.kt

MyViewModelFactory mengimplementasikan ViewModelProvider.Factory untuk menyediakan instans MyViewModel yang memerlukan Application sebagai parameter konstruktor. Ketika create() dipanggil, factory memeriksa apakah modelClass dapat di-assign ke MyViewModel::class.java; jika ya, ia mengembalikan objek baru MyViewModel(application) setelah melakukan cast ke tipe generik T. Jika tidak cocok, factory melempar IllegalArgumentException, sehingga mencegah pembuatan ViewModel yang tak terduga. Kehadiran factory ini memungkinkan fragment atau activity untuk menggunakan ViewModelProvider(this, MyViewModelFactory(requireActivity().application)) sehingga MyViewModel dapat menggunakan context aplikasi dengan benar, sekaligus menjaga integritas tipe ViewModel.

8. activity_main.xml

File ini merupakan layout utama yang digunakan oleh MainActivity. Layout ini menggunakan ConstraintLayout sebagai elemen root, yang memungkinkan setiap

elemen diatur secara fleksibel berdasarkan hubungan antar elemen atau terhadap parent-nya.

Pertama, terdapat MaterialToolbar dari com.google.android.material.appbar.MaterialToolbar, yang berfungsi sebagai app bar atau toolbar utama. Toolbar ini memiliki lebar match_parent dan tinggi berdasarkan atribut ?attr/actionBarSize. Latar belakangnya berwarna putih dan diberi elevation sebesar 4dp agar terlihat bayangan (shadow) pada perangkat dengan material design. Judul dari toolbar ini adalah "Owi Core", dengan warna teks hitam dan gaya teks sesuai dengan TextAppearance.MaterialComponents.Headline6.

Di dalam MaterialToolbar, terdapat satu buah ImageView dengan id btn_menu. Elemen ini memiliki ukuran lebar dan tinggi 24dp, dengan posisi gravitasi ke end (kanan), serta diberi margin end sebesar 16dp. Ikon yang digunakan diambil dari drawable/ic_more_vert, dan diberi warna menggunakan atribut tint hitam.

Setelah itu, terdapat FragmentContainerView dari androidx.fragment.app.FragmentContainerView, yang digunakan sebagai wadah untuk menampilkan fragment yang aktif. Elemen ini memiliki id nav_host_fragment dan menggunakan NavHostFragment sebagai kelas utama yang menjalankan sistem navigasi. Ukuran lebar dan tingginya diatur menggunakan 0dp karena akan disesuaikan oleh constraint. Komponen ini berada di bawah toolbar, dan dikaitkan ke semua sisi parent untuk memenuhi seluruh layar yang tersisa. Properti app:navGraph menunjukkan nav_graph yang digunakan untuk mengatur alur navigasi antar fragment, dan app:defaultNavHost disetel ke true agar fragment ini bertindak sebagai host utama navigasi.

9. fragment_detail.xml

Layout ini menggunakan NestedScrollView dari androidx.core.widget.NestedScrollView sebagai elemen utama. Elemen ini memungkinkan konten yang lebih panjang dari layar untuk digulir secara vertikal. Properti fillViewport disetel ke true agar isi konten mengisi seluruh area tampilan saat belum bisa digulir.

Di dalamnya terdapat ConstraintLayout yang berfungsi untuk mengatur posisi setiap elemen UI secara fleksibel. Layout ini memiliki padding sebesar 16dp dan tinggi wrap_content, sehingga akan menyesuaikan dengan isi yang dimuat.

Elemen pertama adalah ImageButton dengan id btn_back. Tombol ini berfungsi sebagai tombol kembali. Lebar dan tingginya 48dp, tidak memiliki latar belakang tetap (borderless) karena menggunakan ?selectableItemBackgroundBorderless, dan menggunakan ikon dari drawable/ic_arrow_back. Tombol ini diletakkan di kiri atas dengan menggunakan layout_constraintStart_toStartOf dan layout_constraintTop_toTopOf parent. Warna ikonnya diatur menggunakan app:tint hitam.

Berikutnya adalah ShapeableImageView dari com.google.android.material.imageview.ShapeableImageView yang berfungsi untuk menampilkan gambar utama atau gambar detail. Elemen ini memiliki ukuran 200dp x 200dp, dengan scaleType centerCrop agar gambar terpotong sesuai proporsi tampilan. Gambar ini ditempatkan di tengah horizontal dan di bagian atas layout, dengan margin atas sebesar 32dp dan bayangan sebesar 4dp melalui elevation. Selain itu, elemen ini diberi properti transitionName yaitu shared_image_container untuk mendukung animasi transisi antar fragment.

Setelah itu terdapat LinearLayout yang orientasinya vertikal dan posisinya berada di bawah detail_image. Elemen ini memiliki padding sebesar 24dp dan margin atas 24dp. Di dalamnya terdapat dua elemen utama: CardView dan LinearLayout horizontal.

Pertama, CardView dari androidx.cardview.widget.CardView digunakan untuk menampilkan deskripsi dalam bentuk TextView. Kartu ini memiliki sudut membulat sebesar 16dp, bayangan sebesar 4dp, dan latar belakang putih. Di dalamnya, TextView dengan id detail_description menampilkan teks deskripsi

dengan ukuran huruf 16sp, jarak antar baris sebesar 1.2, warna teks hitam, dan padding 24dp di semua sisi.

Kedua, terdapat LinearLayout horizontal yang digunakan untuk menempatkan tombol di bagian bawah konten. Elemen ini memiliki orientasi horizontal dan properti gravity disetel ke center_horizontal untuk memusatkan isinya. Properti spacing diatur sebesar 16dp, walaupun atribut ini tidak berlaku langsung pada LinearLayout standar dan biasanya perlu ditangani melalui margin antar elemen secara manual atau menggunakan spacing dari library tambahan.

Di dalamnya terdapat MaterialButton dari com.google.android.material.button.MaterialButton dengan id btn_share. Tombol ini menggunakan gaya Widget.Material3.Button.OutlinedButton, memiliki lebar match_parent, tinggi 48dp, dan menampilkan teks "Share" serta ikon dari drawable/ic_share. Ikon tidak diberi warna tambahan karena iconTint disetel ke null. Sudut tombol dibulatkan dengan cornerRadius sebesar 8dp.

Secara keseluruhan, layout ini dirancang untuk menampilkan halaman detail dengan gambar utama di atas, deskripsi di tengah, dan tombol aksi di bagian bawah, semuanya dalam struktur yang dapat digulir secara vertikal.

10. fragment_my.xml

File layout ini menggunakan ConstraintLayout dari androidx.constraintlayout.widget.ConstraintLayout sebagai elemen utama. Layout ini memiliki lebar dan tinggi match_parent, artinya akan mengisi seluruh ruang dari layar. Properti tools:context diatur ke .MyFragment, yang memberi petunjuk kepada Android Studio bahwa layout ini digunakan oleh kelas MyFragment. Atribut ini hanya berfungsi saat preview, tidak memengaruhi saat runtime.

Di dalamnya hanya terdapat satu komponen utama yaitu RecyclerView dari androidx.recyclerview.widget.RecyclerView dengan id rv_character. Komponen ini

bertugas untuk menampilkan daftar data dalam bentuk list yang bisa digulir. Ukuran lebar dan tingginya diatur ke 0dp karena akan diatur menggunakan constraint. Properti layout_constraintTop_toTopOf, layout_constraintBottom_toBottomOf, layout_constraintStart_toStartOf, dan layout_constraintEnd_toEndOf digunakan untuk mengaitkan semua sisi RecyclerView ke parent-nya, yaitu ConstraintLayout, sehingga elemen ini akan memenuhi seluruh ruang yang tersedia.

Layout ini bersifat minimal dan hanya bertugas menyediakan wadah untuk menampilkan daftar item dalam fragment. Semua logika pemrosesan data dan penanganan klik dilakukan di kelas MyFragment.

11. Item_layout.xml

Layout ini menggunakan CardView dari androidx.cardview.widget.CardView sebagai elemen utama, yang berfungsi untuk membungkus setiap item dalam daftar agar memiliki tampilan seperti kartu. Atribut layout_width diatur ke match_parent dan layout_height ke wrap_content, artinya kartu akan mengisi lebar penuh tapi tingginya akan menyesuaikan kontennya. Properti cardCornerRadius diatur ke 12dp agar memiliki sudut melengkung, cardElevation sebesar 4dp untuk memberi efek bayangan, dan cardUseCompatPadding digunakan agar padding kompatibel dengan perangkat lama.

Di dalam CardView terdapat ConstraintLayout, yang bertugas untuk mengatur posisi elemen-elemen di dalam kartu secara fleksibel. Layout ini memiliki padding 16dp agar konten tidak menempel langsung ke tepi kartu.

Elemen pertama adalah ImageView dengan id item_image. Gambar ini memiliki ukuran tetap 64dp x 64dp, dengan scaleType diatur ke centerCrop agar gambar memenuhi area tanpa merusak rasio. Gambar ini diposisikan secara vertikal di tengah dan di sisi kiri kartu menggunakan constraint ke parent.

Berikutnya adalah LinearLayout yang menampung dua TextView, yaitu text_title dan text_description. Layout ini diletakkan di sebelah kanan gambar menggunakan constraint ke id item_image, dan di sebelah kiri tombol-tombol menggunakan constraint ke button_group. Layout ini disusun secara vertikal.

text_title berfungsi menampilkan judul item, memiliki ukuran teks 18sp, maksimal 1 baris dan akan terpotong dengan ellipsize jika terlalu panjang. text_description menampilkan deskripsi tambahan dengan ukuran teks 14sp, maksimal 2 baris.

Terakhir, terdapat LinearLayout lain dengan id button_group yang menampung dua tombol dari MaterialButton. Layout ini disusun secara vertikal, dan diletakkan di paling kanan kartu. Tombol pertama adalah btn_details, menggunakan style OutlinedButton dan menampilkan ikon ic_info_outline. Tombol kedua adalah btn_website, menggunakan style TonalButton dengan ikon ic_open_in_browser. Kedua tombol ini memiliki tinggi tetap 36dp dan lebar minimum 64dp. Ikon berada di sebelah kiri teks tombol sesuai dengan iconGravity yang diatur ke textStart.

Secara keseluruhan, layout ini dirancang untuk menampilkan item dalam bentuk kartu yang berisi gambar, judul, deskripsi, dan dua tombol aksi.

12. Penjelasan debugger.

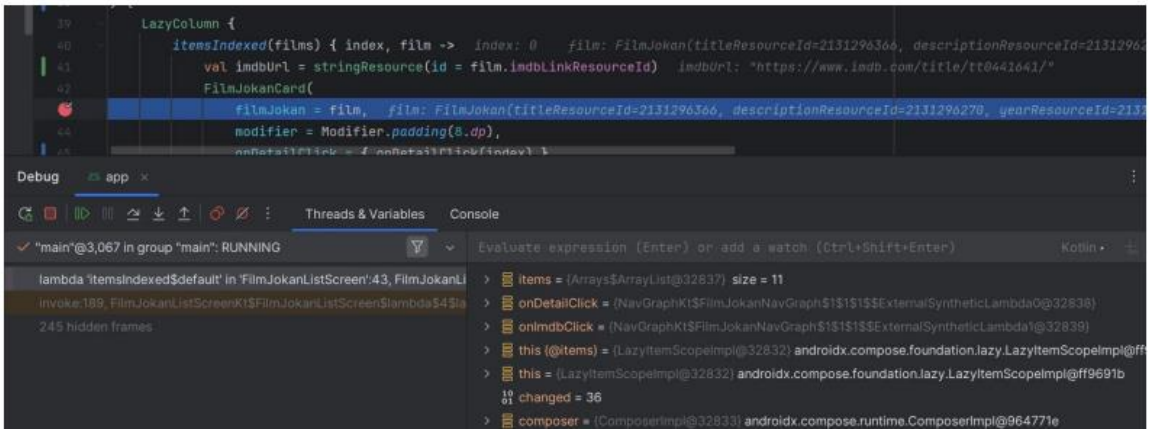
Debugger di Android Studio adalah alat interaktif yang memungkinkan Anda menghentikan eksekusi aplikasi pada titik yang telah ditentukan (breakpoint) untuk memeriksa nilai variabel, tumpukan panggilan (call stack), dan ekspresi yang dipantau (watches) secara real time, sehingga memudahkan proses identifikasi dan perbaikan bug. Untuk menggunakannya, pertama-tama Anda menempatkan breakpoint dengan mengklik margin kiri di baris kode yang ingin dipantau, lalu menjalankan aplikasi dalam mode Debug; eksekusi akan berhenti tepat di breakpoint tersebut. Setelah pause, Anda dapat memeriksa panel Variables untuk melihat nilai-nilai saat ini, memodifikasi watches, dan mengamati alur program. Tombol “Step

Into” (F7) akan memasuki implementasi fungsi atau metode yang dipanggil pada baris saat ini, memungkinkan Anda menelusuri internal fungsi tersebut, sedangkan “Step Over” (F8) akan menjalankan seluruh baris termasuk pemanggilan fungsi tanpa masuk ke dalamnya, kemudian berhenti di baris berikutnya, cocok digunakan ketika Anda percaya fungsi tersebut sudah benar. Jika Anda sudah berada di dalam sebuah fungsi dan ingin cepat kembali ke pemanggilnya, gunakan “Step Out” (Shift + F8), yang akan menyelesaikan eksekusi sisa fungsi lalu menghentikan debug tepat setelah fungsi itu dipanggil. Dengan kombinasi breakpoint, inspeksi variabel, dan kontrol alur eksekusi melalui Step Into, Step Over, dan Step Out, debugger membantu Anda memahami perilaku aplikasi dan menemukan kesalahan secara lebih efisien dibandingkan hanya mengandalkan logging.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

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



Gambar 11 Contoh Penggunaan Debugger

Jawab:

Application class di Android adalah komponen tingkat atas yang mewakili proses aplikasi dan dibuat sebelum komponen lain (Activity, Service, dll.) diinisialisasi; dengan mewarisi Application dan mendaftarkannya di AndroidManifest.xml, Anda dapat menyediakan titik pusat untuk inisialisasi dependensi (misalnya DI framework seperti Hilt atau Dagger), konfigurasi global (seperti logging, analytics, atau error reporting), serta menyimpan state atau objek singleton yang perlu bertahan selama lifecycle aplikasi. Karena instance Application hanya dibuat sekali sepanjang masa hidup proses, ia ideal untuk menyimpan resource bersama, context aplikasi yang aman dari memory leak, dan menjalankan kode setup sebelum UI tampil, sehingga membantu menjaga kebersihan arsitektur dengan memisahkan logika inisialisasi global dari Activity atau Fragment.

Tautan Git

https://github.com/au290/College-Work/tree/725c863263eb03d95289b2e719318e73a0883cf5/Semester-4/Pemrogaman-Mobile/Modul-4