LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

ViewModel and Debugging

Oleh:

Harry Pratama Yunus NIM. 2310817210010

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 4

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

Nama Praktikan : Harry Pratama Yunus
NIM : 2310817210010

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

DAFTAR ISI

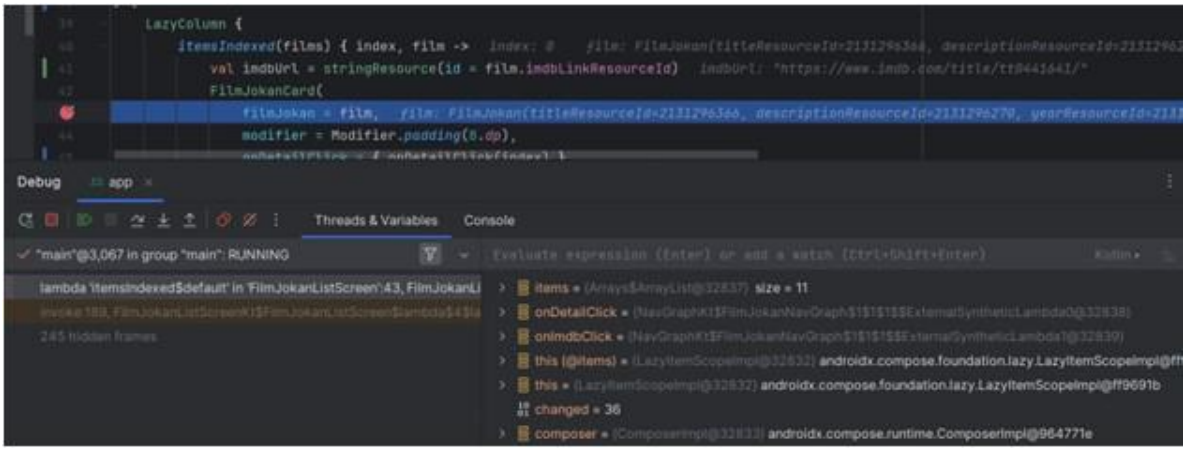
DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

Soal Praktikum:

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



Gambar 1. Contoh Penggunaan Debugger

A. Source Code

XML :

Games.kt

Table 1 source code soal 1 Games.kt (xml)

```
1  package com.pemrogamanmobile.myfavoritegames.data
2
3  import java.io.Serializable
4
5  data class Games(
6      val detailResourceId: Int,
7      val imageResourceId: Int,
```

```
8       val steamLinkResourceId: Int,
9       val yearResourceId: Int,
10      val titleResourceId: Int,
11      val category: String
12  ) : Serializable
```

DataSource.kt

Table 2 source code soal 1 DataSource.kt (xml)

```
1   package com.pemrogamanmobile.myfavoritegames.data
2
3   import com.pemrogamanmobile.myfavoritegames.R
4
5   object DataSource {
6       fun loadGames(): List<Games> = listOf(
7           Games(R.string.detail1, R.drawable.image1, R.string.link1, R.string.year1,
    R.string.title1, "All"),
8           Games(R.string.detail2, R.drawable.image2, R.string.link2, R.string.year2,
    R.string.title2, "All"),
9           Games(R.string.detail3, R.drawable.image3, R.string.link3, R.string.year3,
    R.string.title3, "All"),
10          Games(R.string.detail4, R.drawable.image4, R.string.link4, R.string.year4,
    R.string.title4, "All"),
11          Games(R.string.detail5, R.drawable.image5, R.string.link5, R.string.year5,
    R.string.title5, "All"),
12          Games(R.string.detail6, R.drawable.image6, R.string.link6, R.string.year6,
    R.string.title6, "All"),
13          Games(R.string.detail7, R.drawable.image7, R.string.link7, R.string.year7,
    R.string.title7, "All"),
14          Games(R.string.detail8, R.drawable.image8, R.string.link8, R.string.year8,
    R.string.title8, "All"),
15          Games(R.string.detail9, R.drawable.image9, R.string.link9, R.string.year9,
    R.string.title9, "All"),
16          Games(R.string.detail10, R.drawable.image10, R.string.link10,
    R.string.year10, R.string.title10, "All")
17      )
18  }
```

GamesAdapter.kt

Table 3 source code soal 1 GamesAdapter.kt (xml)

```
1   package com.pemrogamanmobile.myfavoritegames.ui
2
3   import android.annotation.SuppressLint
4   import android.view.LayoutInflater
5   import android.view.ViewGroup
6   import androidx.recyclerview.widget.RecyclerView
7   import com.pemrogamanmobile.myfavoritegames.data.Games
8   import
    com.pemrogamanmobile.myfavoritegames.databinding.ItemGamesBinding
9   import timber.log.Timber
10
11  class GamesAdapter(
12      private val onViewClicked: (Games) -> Unit,
13      private val onDetailClicked: (Games) -> Unit
14  ) : RecyclerView.Adapter<GamesAdapter.ViewHolder>() {
15
16      private val items = mutableListOf<Games>()
17
18      inner class ViewHolder(val binding: ItemGamesBinding) :
    RecyclerView.ViewHolder(binding.root)
19
20      override fun onCreateViewHolder(parent: ViewGroup, viewType:
    Int): ViewHolder {
21          val binding = ItemGamesBinding.inflate(
22              LayoutInflater.from(parent.context), parent, false
23          )
24          return ViewHolder(binding)
25      }
26
27      override fun onBindViewHolder(holder: ViewHolder, position:
    Int) {
28          val item = items[position]
29          with(holder.binding) {
30
31              // tambah padding(entah knp di xmlnya gak mau nambah
    padding ke bawah, mau gak mau ya lewat sini yang ulun tau)
32              val layoutParams = root.layoutParams as
    ViewGroup.MarginLayoutParams
33              layoutParams.bottomMargin = 60
34              root.layoutParams = layoutParams
35
36              imageView.setImageResource(item.imageResourceId)
37              titleText.text =
    holder.itemView.context.getString(item.titleResourceId)
38              yearText.text =
    holder.itemView.context.getString(item.yearResourceId)
```

```
39              detailText.text =
    holder.itemView.context.getString(item.detailResourceId)
40
41              viewButton.setOnClickListener {
42                  val title =
    holder.itemView.context.getString(item.titleResourceId)
43                  Timber.d("View button clicked: $title")
44                  onViewClicked(item)
45              }
46
47              detailButton.setOnClickListener {
48                  val title =
    holder.itemView.context.getString(item.titleResourceId)
49                  Timber.d("Detail button clicked: $title")
50                  onDetailClicked(item)
51              }
52          }
53      }
54
55      override fun getItemCount(): Int = items.size
56
57      @SuppressLint("NotifyDataSetChanged")
58      fun submitList(newItems: List<Games>) {
59          items.clear()
60          items.addAll(newItems)
61          notifyDataSetChanged()
62      }
63  }
```

GamesListFragment.kt

Table 4 source code soal 1 GamesListFragment.kt (xml)

```
1   package com.pemrogamanmobile.myfavoritegames.ui
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.view.LayoutInflater
7   import android.view.View
8   import android.view.ViewGroup
9   import androidx.fragment.app.Fragment
10  import androidx.fragment.app.viewModels
11  import androidx.lifecycle.Lifecycle
12  import androidx.lifecycle.lifecycleScope
13  import androidx.lifecycle.repeatOnLifecycle
14  import androidx.navigation.fragment.findNavController
```

```
15 import com.pemrogamanmobile.myfavoritegames.GamesViewModel
16 import com.pemrogamanmobile.myfavoritegames.GamesViewModelFactory
17 import
   com.pemrogamanmobile.myfavoritegames.databinding.FragmentGamesListBin
18 import kotlinx.coroutines.launch
19 import timber.log.Timber
20
21 class GamesListFragment : Fragment() {
22     private var _binding: FragmentGamesListBinding? = null
23     private val binding get() = _binding!!
24
25     private val viewModel: GamesViewModel by viewModels {
26         GamesViewModelFactory("All") // string pakai All
27     }
28
29     override fun onCreateView(
30         inflater: LayoutInflater, container: ViewGroup?,
   savedInstanceState: Bundle?
31     ) = FragmentGamesListBinding.inflate(inflater, container, false)
32         .also { _binding = it }
33         .root
34
35     override fun onViewCreated(view: View, savedInstanceState: Bundle
   {
36         val adapter = GamesAdapter(
37             onViewClicked = { game ->
38                 val title =
   requireContext().getString(game.titleResourceId)
39                 Timber.d("Launching external app with title: $title")
40
41                 // Intent ke aplikasi lain menggunakan URI
42                 val url =
   requireContext().getString(game.steamLinkResourceId)
43                 Timber.d("Intent to other app, and the game is: $url"
44                 val intent = Intent(Intent.ACTION_VIEW, Uri.parse(url
45                 startActivity(intent)
46             },
47             onDetailClicked = { game ->
48                 val title =
   requireContext().getString(game.titleResourceId)
49                 Timber.d("Navigating to DetailFragment for game:
   $title")
50                 viewModel.onGameClicked(game)
51                 val action =
   GamesListFragmentDirections.actionListToDetail(game)
52                 findNavController().navigate(action)
53             }
```

```
        )
        binding.recyclerView.adapter = adapter

        viewLifecycleOwner.lifecycleScope.launch {

viewLifecycleOwner.lifecycle.repeatOnLifecycle(Lifecycle.State.STARTE
                viewModel.gamesState.collect { games ->
                    Timber.d("Games list updated: $games")
                    adapter.submitList(games)
                }
            }
        }
    }

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

DetailFragment.kt

Table 5 source code soal 1 DetailFragment.kt (xml)

```
1   package com.pemrogamanmobile.myfavoritegames.ui
2
3   import android.os.Bundle
4   import android.view.LayoutInflater
5   import android.view.View
6   import android.view.ViewGroup
7   import androidx.fragment.app.Fragment
8   import androidx.navigation.fragment.navArgs
9   import
        com.pemrogamanmobile.myfavoritegames.databinding.FragmentDetailBinding
10
11  class DetailFragment : Fragment() {
12      private var _binding: FragmentDetailBinding? = null
13      private val binding get() = _binding!!
14
15      private val args: DetailFragmentArgs by navArgs()
16
17      override fun onCreateView(
18          inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?
19      ) = FragmentDetailBinding.inflate(inflater, container, false)
20          .also { _binding = it }
21          .root
22
23      override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
24          val games = args.games
25          with(binding) {
26              detailImageView.setImageResource(games.imageResourceId)
27              detailTitleText.text  = getString(games.titleResourceId)
28              detailYearText.text   = getString(games.yearResourceId)
29              detailDescText.text   = getString(games.detailResourceId)
30          }
31      }
32
33      override fun onDestroyView() {
34          super.onDestroyView()
35          _binding = null
36      }
37  }
```

MainActivity.kt

Table 6 source code soal 1 mainactivity.kt (xml)

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

activity_main.xml

Table 7 source code soal 1 acivity_main.xml (xml)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.fragment.app.FragmentContainerView
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      android:id="@+id/nav_host_fragment"

6  android:name="androidx.navigation.fragment.NavHostFragment"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      app:navGraph="@navigation/nav_graph"
10     app:defaultNavHost="true"/>
```

Fragment_games_list.xml

Table 8 source code soal 1 fragment_games_list.xml (xml)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3    xmlns:android="http://schemas.android.com/apk/res/android"
4    xmlns:app="http://schemas.android.com/apk/res-auto"
5    android:background="#000000"
6    android:layout_width="match_parent"
7    android:layout_height="match_parent">
8
9    <androidx.recyclerview.widget.RecyclerView
10     android:id="@+id/recyclerView"
11     android:contentDescription="@string/games_list_desc"
```

```
12          android:layout_width="0dp"
13          android:layout_height="0dp"
14          android:padding="8dp"
15          app:layoutManager="LinearLayoutManager"
16          app:layout_constraintTop_toTopOf="parent"
17          app:layout_constraintBottom_toBottomOf="parent"
18          app:layout_constraintStart_toStartOf="parent"
19          app:layout_constraintEnd_toEndOf="parent"/>
20  </androidx.constraintlayout.widget.ConstraintLayout>
```

Fragment_detail.xml

Table 9 source code soal 1 fragment_detail.xml (xml)

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       android:background="#131418"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent">
7
8       <LinearLayout
9           android:orientation="vertical"
10          android:padding="16dp"
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content">
13
14          <com.google.android.material.imageview.ShapeableImageView
15              android:id="@+id/detailImageView"
16              android:layout_width="match_parent"
17              android:layout_height="400dp"
18              android:scaleType="centerCrop"

19  app:shapeAppearanceOverlay="@style/ShapeAppearance.App.Rounded6dp" />
20
21          <LinearLayout
22              android:layout_width="match_parent"
23              android:layout_height="wrap_content"
24              android:orientation="horizontal"
25              android:gravity="center_vertical"
26              android:layout_marginTop="12dp">
27
28              <TextView
29                  android:id="@+id/detailTitleText"
30                  android:layout_width="0dp"
31                  android:layout_weight="1"
32                  android:layout_height="wrap_content"
33                  android:textColor="#FFFFFF"
```

```
34              android:textSize="26sp"
35              android:textStyle="bold"/>
36
37          <TextView
38              android:id="@+id/detailYearText"
39              android:layout_width="wrap_content"
40              android:layout_height="wrap_content"
41              android:textColor="#7C7C86"
42              android:textSize="14sp"/>
43      </LinearLayout>
44
45      <TextView
46          android:layout_marginTop="8dp"
47          android:layout_width="wrap_content"
48          android:layout_height="wrap_content"
49          android:text="tentang game ini:"
50          android:textColor="#FFFFFF"
51          android:textSize="14sp"
52          android:textStyle="bold"/>
53
54      <TextView
55          android:id="@+id/detailDescText"
56          android:layout_marginTop="4dp"
57          android:layout_width="match_parent"
58          android:layout_height="wrap_content"
59          android:textColor="#FFFFFF"
60          android:textSize="14sp"/>
61  </LinearLayout>
62  </ScrollView>
```

Item_games.kt

Table 10 source code soal 1 item_games.xml (xml)

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
3       xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       android:layout_width="match_parent"
6       android:layout_height="wrap_content"
7       app:cardCornerRadius="6dp"
8       app:cardBackgroundColor="#343539">
9
10      <LinearLayout
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content"
13          android:orientation="horizontal"
14          android:padding="8dp"
15          android:gravity="center_vertical">
```

```
16
17      <com.google.android.material.imageview.ShapeableImageView
18          android:id="@+id/imageView"
19          android:layout_width="100dp"
20          android:layout_height="140dp"
21          android:layout_marginEnd="12dp"
22          android:scaleType="centerCrop"
23
        app:shapeAppearanceOverlay="@style/ShapeAppearance.App.Rounded6dp"
24          />
25
26      <LinearLayout
27          android:layout_width="0dp"
28          android:layout_weight="1"
29          android:orientation="vertical"
30          android:layout_height="wrap_content">
31
32          <LinearLayout
33              android:layout_width="match_parent"
34              android:layout_height="wrap_content"
35              android:orientation="horizontal"
36              android:gravity="center_vertical">
37
38              <TextView
39                  android:id="@+id/titleText"
40                  android:layout_width="0dp"
41                  android:layout_weight="1"
42                  android:layout_height="wrap_content"
43                  android:textColor="@android:color/white"
44                  android:textSize="20sp"
45                  android:textStyle="bold"/>
46
47              <TextView
48                  android:id="@+id/yearText"
49                  android:layout_width="wrap_content"
50                  android:layout_height="wrap_content"
51                  android:textColor="#7C7C86"
52                  android:textSize="14sp"/>
53          </LinearLayout>
54
55          <TextView
56              android:layout_marginTop="8dp"
57              android:layout_width="wrap_content"
58              android:layout_height="wrap_content"
59              android:text="tentang game ini:"
60              android:textColor="@android:color/white"
61              android:textSize="14sp"
```

```
            android:textStyle="bold"/>

        <TextView
            android:id="@+id/detailText"
            android:layout_marginTop="4dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@android:color/white"
            android:textSize="14sp"/>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:gravity="end"
            android:layout_marginTop="12dp">

            <Button
                android:id="@+id/viewButton"
                android:layout_width="100dp"
                android:layout_height="44dp"
                android:text="View"
                android:backgroundTint="#b0c4ff"
                android:textColor="#000"/>

            <Space                                    android:layout_width="8dp"
android:layout_height="wrap_content"/>

            <Button
                android:id="@+id/detailButton"
                android:layout_width="100dp"
                android:layout_height="44dp"
                android:text="Detail"
                android:backgroundTint="#b0c4ff"
                android:textColor="#000"/>
        </LinearLayout>
    </LinearLayout>
  </LinearLayout>
</androidx.cardview.widget.CardView>
```

strings.xml

Table 11 source code soal 1 strings.xml (xml)

```
 <resources>
     <string name="app_name">My favorite games</string>

     <string name="games_list_desc">List of favorite games</string>
```

```
5
6       <string name="title1">Terraria</string>
7       <string name="title2">Mobile Legends</string>
8       <string name="title3">Firewatch</string>
9       <string name="title4">The Elder Scrolls : Skyrim</string>
10      <string name="title5">Monster Hunter World</string>
11      <string name="title6">Genshin Impact</string>
12      <string name="title7">Delta Force</string>
13      <string name="title8">Magic Chess Go Go</string>
14      <string name="title9">STAR WARS Jedi: Fallen Order</string>
15      <string name="title10">Minecraft</string>
16
17      <string name="year1">2011</string>
18      <string name="year2">2016</string>
19      <string name="year3">2016</string>
20      <string name="year4">2016</string>
21      <string name="year5">2018</string>
22      <string name="year6">2020</string>
23      <string name="year7">2025</string>
24      <string name="year8">2025</string>
25      <string name="year9">2019</string>
26      <string name="year10">2011</string>
27
28      <string name="detail1">Dig, fight, explore, build! Nothing is im
    game. Four Pack also available!</string>
29      <string name="detail2">Bergabung dengan teman Anda di Mobile Leg
    Game MOBA 5v5 terbaru, dan bertarung melawan Player sungguhan! Pi
    Anda dan bentuk tim yang sempurna dengan teman Anda! Matchmaking 10 de
    10 menit. Laning, Jungling, Team Fight, dan menghancurkan Turret,
    dari Game aksi dan MOBA PC berada dalam genggaman Anda! Tunjukkan
30  Anda!</string>
        <string name="detail3">Firewatch is a single-player first-perso
31  where your only emotional lifeline is the person on the other end of
        <string name="detail4">Winner of more than 200 Game of the Ye
    Special Edition brings the epic fantasy to life in stunning det
32  critically acclaimed game and add-ons with all-new features.</string
        <string name="detail5">Welcome to a new world! In Monster Hunte
    series, you can enjoy the ultimate hunting experience, using everyth
33  a new world teeming with surprises and excitement.</string>
        <string name="detail6">Genshin Impact adalah game RPG Open World
    yang menjelajah sebuah dunia fantasi yang teramat luas, kamu dapat me
    karakter berkemampuan dan berkepribadian unik, melawan musuh-musuh
    yang hilang. Di dunia yang luas ini, biarkan rasa keingintahuan
    tersembunyi, sampai akhirnya kamu dapat bertemu kembali dengan saud
34  perjalananmu.</string>
        <string name="detail7">Pertempuran Epic Warfare 24vs24, Rasaka
    mode permainan menarik, dan update yang terus menerus! Warfare memb
```

```
    penuh kekacauan dan kehancuran environment. Baik kalian menembaki
35  menyelamatkan rekan satu tim sebagai medic, pilih role kalian dan ra
        <string name="detail8">Magic Chess: Go Go - Game strategi multip
    Legends: Bang Bang. Dengan gameplay seperti catur, game ini kasual
    saja bersama teman-teman! Di sini, kemenangan bergantung pada stra
    keterampilan pengendalian mikro. Di setiap babak, kamu bisa meng
    meningkatkan Hero, menyusun Sinergi, menggunakan equipment, dan
36  mengalahkan lawan. Kalahkan 7 player lain secara bertahap untuk meme
        <string name="detail9">A galaxy-spanning adventure awaits in Sta
    action-adventure title from Respawn. An abandoned Padawan must comp
37  Force abilities, and master the art of the lightsaber - all
    Empire.</string>
        <string name="detail10">Bangun, jelajahi, bertahan hidup! Main
    Jelajahi dunia terbuka dengan membangun, membuat item, dan bertahan
    Kumpulkan sumber daya, coba bertahan hidup di malam hari, dan ranc
    Jelajahi dan buat item sesuka hati di dunia yang terbuka sepenuhnya
38  membangun kota balok, membuka peternakan, menambang jauh ke bawah
39  sekadar bereksperimen sejauh imajinasi membawamu!</string>
40
41      <string name="link1">https://store.steampowered.com/app/105600/T
42      <string name="link2">https://play.google.com/store/apps/details?
        <string name="link3">https://store.steampowered.com/app/383870/F
43      <string
44  name="link4">https://store.steampowered.com/app/489830/The_Elder_Scr
45      <string name="link5">https://store.steampowered.com/app/582010/M
46      <string name="link6">https://play.google.com/store/apps/details?
47      <string name="link7">https://play.google.com/store/apps/details?
48      <string name="link8">https://play.google.com/store/apps/details?
49      <string name="link9">https://store.steampowered.com/app/1172380/
        <string name="link10">https://play.google.com/store/apps/details
    </resources>
```

stlyes.xml

Table 12 source code soal 1 style.xml (xml)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3    <style name="ShapeAppearance.App.Rounded6dp" parent="">
4      <item name="cornerFamily">rounded</item>
5      <item name="cornerSize">6dp</item>
6    </style>
7  </resources>
```

GamesViewModel.kt

Table 13 source code soal 1 GamesViewModel.kt (xml)

```
1  package com.pemrogamanmobile.myfavoritegames
```

```
2
3   import androidx.lifecycle.ViewModel
4   import
    com.pemrogamanmobile.myfavoritegames.data.DataSource
5   import com.pemrogamanmobile.myfavoritegames.data.Games
6   import kotlinx.coroutines.flow.MutableStateFlow
7   import kotlinx.coroutines.flow.StateFlow
8   import kotlinx.coroutines.flow.update
9   import timber.log.Timber
10
11  class GamesViewModel(
12      private val gameCategory: String
13  ) : ViewModel() {
14
15      // StateFlow untuk menyimpan daftar game
16      private val _gamesState = MutableStateFlow(
17          DataSource.loadGames().filter { it.category ==
    gameCategory }
18      )
19      val gamesState: StateFlow<List<Games>> get() =
    _gamesState
20
21      // StateFlow untuk menyimpan game yang dipilih
22      private val _selectedGame =
    MutableStateFlow<Games?>(null)
23      val selectedGame: StateFlow<Games?> get() =
    _selectedGame
24
25      init {
26          // Log data game yang di-load
27          Timber.d("Games data loaded:
    ${_gamesState.value}")
28      }
29
30      // Fungsi untuk menangani klik pada item game
31      fun onGameClicked(game: Games) {
32          _selectedGame.update { game }
33          Timber.d("Game selected: $game")
34      }
35  }
```

GamesViewModelFactory.kt

Table 14 source code soal 1 GamesViewModelFactory.kt (xml)

```
1   package com.pemrogamanmobile.myfavoritegames
2
```

```
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5
6   class GamesViewModelFactory(
7       private val gameCategory: String
8   ) : ViewModelProvider.Factory {
9       override fun <T : ViewModel>
    create(modelClass: Class<T>): T {
10          if (modelClass.isAssignableFrom
    (GamesViewModel::class.java)) {
11              @Suppress("UNCHECKED_CAST")
12              return GamesViewModel(gameCategory) as T
13          }
14          throw          IllegalArgumentException("Unknown
    ViewModel class")
15      }
16  }
```

MyApplication.kt

Table 15 source code soal 1 MyApplication.kt (xml)

```
1   package com.pemrogamanmobile.myfavoritegames
2
3   import android.app.Application
4   import timber.log.Timber
5   import com.pemrogamanmobile.myfavoritegames.BuildConfig
6
7   class MyApplication : Application() {
8       override fun onCreate() {
9           super.onCreate()
10
11          // Inisialisasi Timber
12          if (BuildConfig.DEBUG) {
13              Timber.plant(Timber.DebugTree())
14          }
15      }
16  }
```

COMPOSE :

MainActivity.kt

Table 16 source code soal 1 mainactivity.kt (compose)

```
1   package com.pemrogamanmobile.myfavoritegames
2
3   import android.os.Bundle
```

```
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.compose.material3.Surface
7   import androidx.compose.runtime.collectAsState
8   import androidx.compose.runtime.getValue
9   import androidx.compose.ui.graphics.Color
10  import androidx.lifecycle.viewmodel.compose.viewModel
11  import androidx.navigation.compose.NavHost
12  import androidx.navigation.compose.composable
13  import androidx.navigation.compose.rememberNavController
14  import
    com.pemrogamanmobile.myfavoritegames.data.Datasource
15  import
    com.pemrogamanmobile.myfavoritegames.ui.theme.MyFavorite
    GamesTheme
16  import androidx.compose.runtime.LaunchedEffect
17  import timber.log.Timber
18
19  class MainActivity : ComponentActivity() {
20      override fun onCreate(savedInstanceState: Bundle?) {
21          super.onCreate(savedInstanceState)
22
23          // Log saat MainActivity dibuat
24          Timber.d("MainActivity created")
25
26          setContent {
27              MyFavoriteGamesTheme {
28                  Surface(color = Color(0xFF131418)) {
29                      val          navController          =
    rememberNavController()
30
31                      // menggunakan parameter String "All"
32                      val gamesViewModel: GamesViewModel =
    viewModel(
33                          factory = GamesViewModelFactory(
34                              gameCategory = "All",
35                              datasource = Datasource()
36                          )
37                      )
38
39                      NavHost(navController              =
    navController, startDestination = "home") {
40                          composable("home") {
41                              Timber.d("Navigated  to  Home
    Screen")
42
43                              HomeScreen(
```

```
44                                      gamesViewModel          =
    gamesViewModel,
45                                      onDetailClick = { game -
    >
46  gamesViewModel.onGameClicked(game) // set selectedGame di
    ViewModel
47
48                                          //  Log  saat  tombol
    Detail ditekan
49                                          Timber.d("Detail
    button clicked untuk game: $game")
50
51  navController.navigate("detail")
52                                      }
53                                  )
54                              }
55
56                              composable("detail") {
57                                  Timber.d("Navigated       to
    Detail Screen")
58
59                                  val     selectedGame      by
    gamesViewModel.selectedGame.collectAsState()
60
61                                  selectedGame?.let { game ->
62                                      //  Log  data  game  yang
    dipilih
63                                      Timber.d("Displaying
    details for game: $game")
64
65                                      DetailScreen(
66                                          titleResId          =
    game.titleResourceId,
67                                          imageResId          =
    game.imageResourceId,
68                                          yearResId           =
    game.yearResourceId,
69                                          detailResId         =
    game.detailResourceId
70                                      )
71                                  } ?: run {
72                                      //   Kalau   tidak   ada
    selectedGame, navigasi balik ke home
                                        Timber.d("No    selected
73  game found, navigating back to Home Screen")
```

```
                                    LaunchedEffect(Unit) {
74
    navController.popBackStack()
75                                  }
76                              }
77                          }
78                      }
79                  }
80              }
81          }
82      }
83  }
84
```

HomeScreen.kt

Table 17 source code soal 1 HomeScreen.kt (compose)

```
1   package com.pemrogamanmobile.myfavoritegames
2
3   import androidx.compose.foundation.layout.*
4   import androidx.compose.foundation.lazy.LazyColumn
5   import androidx.compose.foundation.lazy.items
6   import androidx.compose.runtime.Composable
7   import androidx.compose.runtime.collectAsState
8   import androidx.compose.runtime.getValue
9   import androidx.compose.ui.Modifier
10  import androidx.compose.ui.unit.dp
11  import com.pemrogamanmobile.myfavoritegames.model.Games
12
13  @Composable
14  fun HomeScreen(
15      gamesViewModel: GamesViewModel,
16      onDetailClick: (Games) -> Unit
17  ) {
18      val                     games                       by
    gamesViewModel.gamesState.collectAsState()
19
20      LazyColumn(
21          verticalArrangement                              =
22  Arrangement.spacedBy(8.dp),
            modifier = Modifier.fillMaxSize()
23      ) {
24          items(games) { game ->
25              GameCard(
26                  games = game,
27                  onDetailClick = onDetailClick
```

```
            )
        }
    }
}
```

DetailScreen.kt

Table 18 source code soal 1 DetailScreen.kt (compose)

```
package com.pemrogamanmobile.myfavoritegames

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import
androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.unit.dp
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

@Composable
fun DetailScreen(titleResId: Int, imageResId: Int,
yearResId: Int, detailResId: Int) {
    Surface(
        modifier = Modifier.fillMaxSize(),
        color = Color(0xFF131418)
    ) {

        val scrollState = rememberScrollState()

        Column(
            modifier = Modifier
                .verticalScroll(scrollState)
                .padding(16.dp)
        ) {

```

```
37              Image(
38                  painter = painterResource(imageResId),
39                  contentDescription                          =
    stringResource(detailResId),
40                  modifier = Modifier
41                      .fillMaxWidth()
42                      .height(400.dp)
43                      .clip(RoundedCornerShape(6.dp)),
44                  contentScale = ContentScale.Crop
45              )
46
47              Column(modifier = Modifier.fillMaxWidth()) {
48                  Row(
49                      modifier = Modifier.fillMaxWidth(),
50                      horizontalArrangement                   =
    Arrangement.SpaceBetween
51                  ) {
52                      Text(
53                          text                                =
    stringResource(titleResId),
54                          style   =  TextStyle(fontSize   =
    26.sp,   fontWeight   =   FontWeight.Bold,   color   =
    Color.White),
55                          modifier = Modifier.weight(1f)
56                      )
57                      Text(
58                          text                                =
    stringResource(yearResId),
59                          style   =  TextStyle(fontSize   =
    16.sp, color = Color(0xFF7C7C86)),
60                          modifier                            =
    Modifier.align(Alignment.CenterVertically)
61                      )
62                  }
63
64                  Spacer(modifier = Modifier.height(8.dp))
65
66                  Text(
67                      text = "tentang game ini:",
68                      style = TextStyle(fontSize = 14.sp,
    fontWeight = FontWeight.Bold, color = Color.White)
69                  )
70
71                  Text(
72                      text = stringResource(detailResId),
                        style = TextStyle(fontSize = 14.sp,
73  color = Color.White),
```

```
                        modifier = Modifier.padding(top = 4.dp)
                    )
                }
            }
        }
}

```

GameCard.kt

Table 19 source code soal 1 GameCard.kt (compose)

```
package com.pemrogamanmobile.myfavoritegames

import android.content.Intent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
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.res.stringResource
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.net.toUri
import com.pemrogamanmobile.myfavoritegames.model.Games
import timber.log.Timber

@Composable
fun GameCard(
    games: Games,
    modifier: Modifier = Modifier,
    onDetailClick: (Games) -> Unit
) {
    val context = LocalContext.current

    Card(
```

```
34          modifier = modifier.padding(12.dp),
35          colors = CardDefaults.cardColors(containerColor
    = Color(0xFF343539))
36      ) {
37          Row(modifier = Modifier.fillMaxWidth()) {
38
39              Image(
40                  painter                                 =
    painterResource(games.imageResourceId),
41                  contentDescription                      =
    stringResource(games.titleResourceId),
42                  modifier = Modifier
43                      .padding(start = 12.dp)
44                      .width(100.dp)
45                      .height(140.dp)
46                      .clip(RoundedCornerShape(6.dp))
47                      .align(Alignment.CenterVertically),
48                  contentScale = ContentScale.Crop
49              )
50
51              Column(
52                  modifier = Modifier
53                      .padding(16.dp)
54                      .fillMaxWidth(),
55                  verticalArrangement                     =
    Arrangement.SpaceBetween
56              ) {
57                  Row(
58                      modifier = Modifier.fillMaxWidth(),
59                      horizontalArrangement               =
    Arrangement.SpaceBetween,
60                      verticalAlignment = Alignment.Top
61                  ) {
62                      Text(
63                          text                            =
    context.getString(games.titleResourceId),
64                          style  =  TextStyle(fontSize  =
    26.sp,   fontWeight   =   FontWeight.Bold,   color   =
    Color.White),
65                          modifier = Modifier.weight(1f)
66                      )
67
68                      Spacer(modifier                     =
    Modifier.width(8.dp))
69
70                      Text(
71
```

```
                            text                                    =
72  context.getString(games.yearResourceId),
                            style  =  TextStyle(fontSize  =
73  16.sp, color = Color(0xFF7C7C86)),
                            modifier = Modifier
74
    .wrapContentWidth(Alignment.End)
75
76  .align(Alignment.CenterVertically)
77                      )
78                  }
79
                    Spacer(modifier                                 =
80  Modifier.height(8.dp))
81
                    Column(modifier                                 =
82  Modifier.fillMaxWidth()) {
83                      Text(
84                          text = "tentang game ini:",
                            style  =  TextStyle(fontSize  =
    14.sp,   fontWeight   =   FontWeight.Bold,   color   =
85  Color.White)
86                      )
87                      Text(
                            text                                    =
88  context.getString(games.detailResourceId),
                            style  =  TextStyle(fontSize  =
89  14.sp, color = Color.White),
                            modifier = Modifier.padding(top
90  = 4.dp)
91                      )
92                  }
93
                    Spacer(modifier                                 =
94  Modifier.height(12.dp))
95
96                  Row(
97                      modifier = Modifier.fillMaxWidth(),
                        horizontalArrangement                       =
98  Arrangement.End,
                        verticalAlignment                           =
99  Alignment.CenterVertically
100                 ) {
101                     Button(
102                         onClick = {
                                val          url          =
103 context.getString(games.linkResourceId)
```

```
                                val intent =
104 Intent(Intent.ACTION_VIEW, url.toUri())
                                Timber.d("Explicit Intent
    triggered: Opening URL $url") // Logging tombol "View"
105
106 context.startActivity(intent)
107                         },
                            modifier =
108 Modifier.width(100.dp).height(44.dp),
                            colors =
    ButtonDefaults.buttonColors(containerColor =
109 Color(0xFFb0c4ff), contentColor = Color.White)
110                         ) {
                                Text("View", style =
111 TextStyle(fontSize = 16.sp, color = Color.Black))
112                         }
113
                            Spacer(modifier =
114 Modifier.width(8.dp))
115
116                         Button(
117                             onClick = {
                                Timber.d("Detail button
    clicked for game(dengan judul?):
    ${context.getString(games.titleResourceId)}") //
118 Logging tombol "Detail"
119                             onDetailClick(games)
120                         },
                            modifier =
121 Modifier.width(100.dp).height(44.dp),
                            colors =
    ButtonDefaults.buttonColors(containerColor =
122 Color(0xFFb0c4ff), contentColor = Color.White)
123                         ) {
                                Text("Detail", style =
124 TextStyle(fontSize = 16.sp, color = Color.Black))
125                         }
126                 }
127             }
128         }
129     }
    }
```

Datasource.kt

Table 20 source code soal 1 Datasource.kt (compose)

```
1  package com.pemrogamanmobile.myfavoritegames.data
2
3  import com.pemrogamanmobile.myfavoritegames.R
4  import com.pemrogamanmobile.myfavoritegames.model.Games
5
6  class Datasource {
7      fun loadGames(): List<Games> {
8          return listOf(
9              Games(R.string.detail1,   R.drawable.image1,
   R.string.link1, R.string.year1, R.string.title1, "All"),
10             Games(R.string.detail2,   R.drawable.image2,
   R.string.link2, R.string.year2, R.string.title2, "All"),
11             Games(R.string.detail3,   R.drawable.image3,
   R.string.link3, R.string.year3, R.string.title3, "All"),
12             Games(R.string.detail4,   R.drawable.image4,
   R.string.link4, R.string.year4, R.string.title4, "All"),
13             Games(R.string.detail5,   R.drawable.image5,
   R.string.link5, R.string.year5, R.string.title5, "All"),
14             Games(R.string.detail6,   R.drawable.image6,
   R.string.link6, R.string.year6, R.string.title6, "All"),
15             Games(R.string.detail7,   R.drawable.image7,
   R.string.link7, R.string.year7, R.string.title7, "All"),
16             Games(R.string.detail8,   R.drawable.image8,
   R.string.link8, R.string.year8, R.string.title8, "All"),
17             Games(R.string.detail9,   R.drawable.image9,
   R.string.link9, R.string.year9, R.string.title9, "All"),
18             Games(R.string.detail10, R.drawable.image10,
   R.string.link10,   R.string.year10,   R.string.title10,
   "All")
19         )
20     }
21 }
```

strings.xml

Table 21 source code soal 1 strings.xml (compose)

```
1    <resources>
2        <string name="app_name">my favorite games</string>
3
4        <string name="title1">Terraria</string>
5        <string name="title2">Mobile Legends</string>
6        <string name="title3">Firewatch</string>
7
```

```
    <string name="title4">The Elder Scrolls : Skyrim</string>
    <string name="title5">Monster Hunter World</string>
    <string name="title6">Genshin Impact</string>
    <string name="title7">Delta Force</string>
    <string name="title8">Magic Chess Go Go</string>
    <string name="title9">STAR WARS Jedi: Fallen Order</string>
    <string name="title10">Minecraft</string>

    <string name="year1">2011</string>
    <string name="year2">2016</string>
    <string name="year3">2016</string>
    <string name="year4">2016</string>
    <string name="year5">2018</string>
    <string name="year6">2020</string>
    <string name="year7">2025</string>
    <string name="year8">2025</string>
    <string name="year9">2019</string>
    <string name="year10">2011</string>

    <string name="detail1">Dig, fight, explore, build! Nothing is impossible in this action-packed adventure game. Four Pack also available!</string>
    <string name="detail2">Bergabung dengan teman Anda di Mobile Legends: Bang Bang, Game MOBA 5v5 terbaru, dan bertarung melawan Player sungguhan! Pilih Hero favorit Anda dan bentuk tim yang sempurna dengan teman Anda! Matchmaking 10 detik, pertandingan 10 menit. Laning, Jungling, Team Fight, dan menghancurkan Turret, semua kesenangan dari Game aksi dan MOBA PC berada dalam genggaman Anda! Tunjukkan semangat Esports Anda!</string>
    <string name="detail3">Firewatch is a single-player first-person mystery set in the Wyoming wilderness, where your only emotional lifeline is the person on the other end of a handheld radio.</string>
    <string name="detail4">Winner of more than 200 Game of the Year Awards, The Elder Scrolls V: Skyrim Special Edition brings the epic fantasy to life in stunning detail. The Special Edition includes the critically acclaimed game and add-ons with all-new features.</string>
    <string name="detail5">Welcome to a new world! In Monster Hunter: World, the latest installment in the series, you can enjoy the ultimate hunting experience,
```

using everything at your disposal to hunt monsters in a new world teeming with surprises and excitement.</string>

<string name="detail6">Genshin Impact adalah game RPG Open World terbaru. Kamu adalah seorang pengembara yang menjelajah sebuah dunia fantasi yang teramat luas, kamu dapat menjelajahi tujuh bangsa, bertemu berbagai karakter berkemampuan dan berkepribadian unik, melawan musuh-musuh kuat, dan berkelana mencari saudaramu yang hilang. Di dunia yang luas ini, biarkan rasa keingintahuan membawamu menguak misteri-misteri yang tersembunyi, sampai akhirnya kamu dapat bertemu kembali dengan saudaramu, dan menyaksikan akhir dari kisah perjalananmu.</string>

<string name="detail7">Pertempuran Epic Warfare 24vs24, Rasakan map besar, puluhan senjata, berbagai mode permainan menarik, dan update yang terus menerus! Warfare membawa kalian ke medan perang dinamis yang penuh kekacauan dan kehancuran environment. Baik kalian menembaki musuh di darat, laut, dan udara, atau menyelamatkan rekan satu tim sebagai medic, pilih role kalian dan raih kemenangan!</string>

<string name="detail8">Magic Chess: Go Go - Game strategi multiplayer baru yang terinspirasi dari Mobile Legends: Bang Bang. Dengan gameplay seperti catur, game ini kasual dan mudah dimainkan kapan saja, di mana saja bersama teman-teman! Di sini, kemenangan bergantung pada strategi dan sedikit keberuntungan daripada keterampilan pengendalian mikro. Di setiap babak, kamu bisa mengendalikan Commander untuk membeli dan meningkatkan Hero, menyusun Sinergi, menggunakan equipment, dan menempatkan hero dengan cerdik untuk mengalahkan lawan. Kalahkan 7 player lain secara bertahap untuk memenangkan pertandingan.</string>

<string name="detail9">A galaxy-spanning adventure awaits in Star Wars Jedi: Fallen Order, a 3rd person action-adventure title from Respawn. An abandoned Padawan must complete his training, develop new powerful Force abilities, and master the art of the lightsaber - all while staying one step ahead of the Empire.</string>

<string name="detail10">Bangun, jelajahi, bertahan hidup! Main dengan teman dan buat duniamu sendiri! Jelajahi dunia terbuka dengan membangun, membuat item, dan bertahan hidup di game sandbox membangun terbaik. Kumpulkan sumber daya, coba bertahan hidup di malam hari, dan rancang petualangan epik balok demi balok.

```
    Jelajahi dan buat item sesuka hati di dunia yang
    terbuka sepenuhnya tempat kamu bisa bermain dengan
    teman, membangun kota balok, membuka peternakan,
    menambang jauh ke bawah tanah, menghadapi musuh
    misterius, atau sekadar bereksperimen sejauh imajinasi
    membawamu!</string>

36      <string
    name="link1">https://store.steampowered.com/app/10560
    0/Terraria/</string>
37      <string
    name="link2">https://play.google.com/store/apps/detail
    s?id=com.mobile.legends</string>
38      <string
    name="link3">https://store.steampowered.com/app/383870
    /Firewatch/</string>
39      <string
    name="link4">https://store.steampowered.com/app/489830
    /The_Elder_Scrolls_V_Skyrim_Special_Edition/</string>
40      <string
    name="link5">https://store.steampowered.com/app/582010
    /Monster_Hunter_World/</string>
41      <string
    name="link6">https://play.google.com/store/apps/detail
    s?id=com.miHoYo.GenshinImpact</string>
42      <string
    name="link7">https://play.google.com/store/apps/detail
    s?id=com.garena.game.df</string>
43      <string
    name="link8">https://play.google.com/store/apps/detail
    s?id=com.mobilechess.gp</string>
44      <string
    name="link9">https://store.steampowered.com/app/117238
    0/STAR_WARS_Jedi_Fallen_Order/</string>
45      <string
    name="link10">https://play.google.com/store/apps/detai
    ls?id=com.mojang.minecraftpe</string>
46  </resources>
```

Games.kt

Table 22 source code soal 1 Games.kt (compose)

```
1   package com.pemrogamanmobile.myfavoritegames.model
2
3   import androidx.annotation.DrawableRes
4   import androidx.annotation.StringRes
```

```
5
6   data class Games(
7       @StringRes val detailResourceId: Int,
8       @DrawableRes val imageResourceId: Int,
9       @StringRes val linkResourceId: Int,
10      @StringRes val yearResourceId: Int,
11      @StringRes val titleResourceId: Int,
12      val category: String
13  )
```

GamesViewModel.kt

Table 23 source code soal 1 GamesViewModel.kt (compose)

```
1   package com.pemrogamanmobile.myfavoritegames
2
3   import androidx.lifecycle.ViewModel
4   import kotlinx.coroutines.flow.MutableStateFlow
5   import kotlinx.coroutines.flow.StateFlow
6   import
    com.pemrogamanmobile.myfavoritegames.data.Datasource
7   import com.pemrogamanmobile.myfavoritegames.model.Games
8   import timber.log.Timber
9
10  class GamesViewModel(
11      private val gameCategory: String,
12      private val datasource: Datasource
13  ) : ViewModel() {
14
15      private val _gamesState = MutableStateFlow(
16          datasource.loadGames().filter { it.category ==
    gameCategory }
17      )
18      val gamesState: StateFlow<List<Games>> get() =
    _gamesState
19
20      private val _selectedGame =
    MutableStateFlow<Games?>(null)
21      val selectedGame: StateFlow<Games?> get() =
    _selectedGame
22
23      init {
24          // Log saat data masuk ke list
25          Timber.d("Games data loaded:
    ${_gamesState.value}")
26      }
27
```

```
28      fun onGameClicked(game: Games) {
29          _selectedGame.value = game
30          // Log data game yang dipilih
31          Timber.d("Game selected: $game")
32      }
33  }
```

GamesViewModelFactory.kt

Table 24 source code soal 1 GamesViewModelFactory.kt (compose)

```
1   package com.pemrogamanmobile.myfavoritegames
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5   import
    com.pemrogamanmobile.myfavoritegames.data.Datasource
6
7   class GamesViewModelFactory(
8       private val gameCategory: String,
9       private val datasource: Datasource = Datasource()
10  ) : ViewModelProvider.Factory {
11      override fun <T : ViewModel>
    create(modelClass: Class<T>): T {
12          if (modelClass.isAssignableFrom
    (GamesViewModel::class.java)) {
13              @Suppress("UNCHECKED_CAST")
14              return          GamesViewModel(gameCategory,
    datasource) as T
15          }
16          throw          IllegalArgumentException("Unknown
    ViewModel class")
17      }
18  }
```

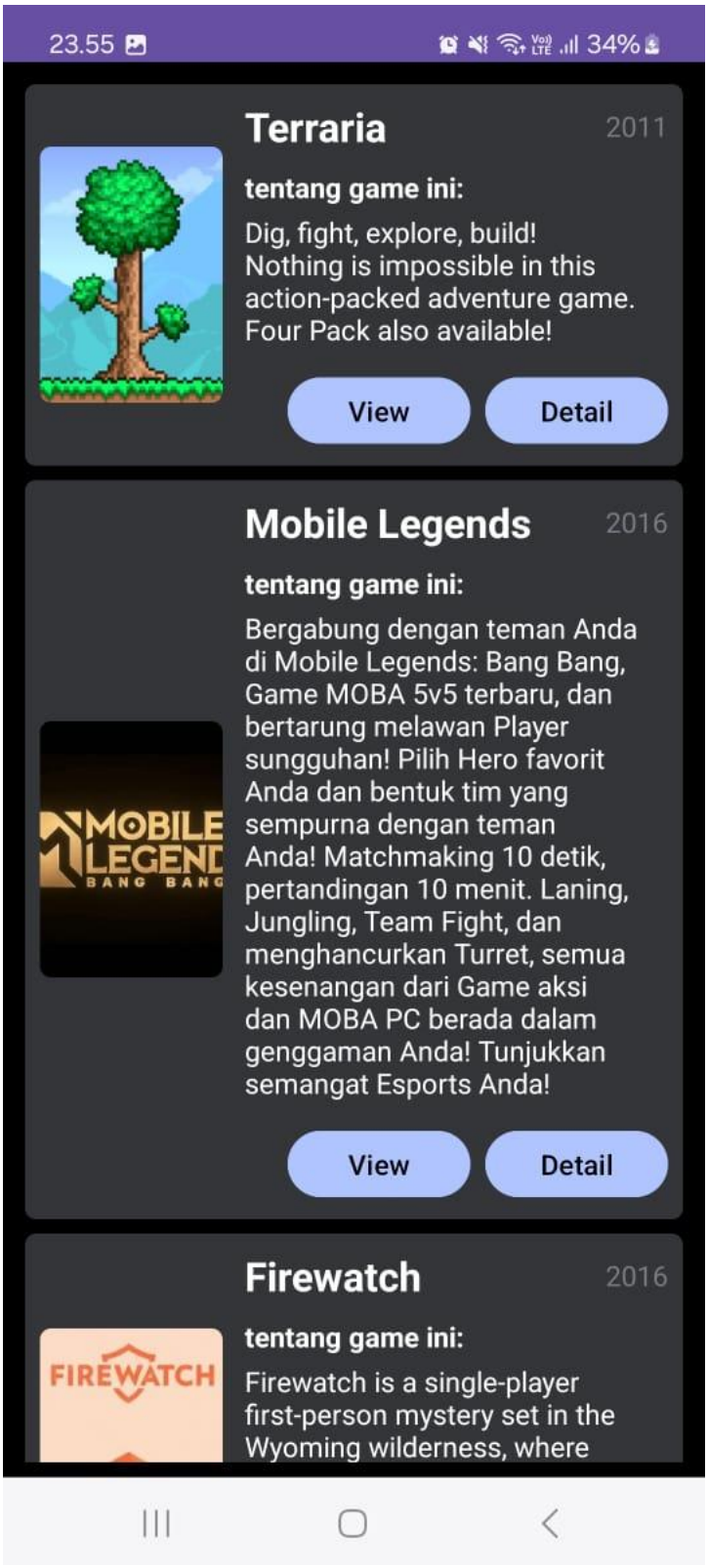
MyFavoriteGamesApplication.kt

Table 25 source code soal 1 MyFavoriteGamesApplication.kt (compose)

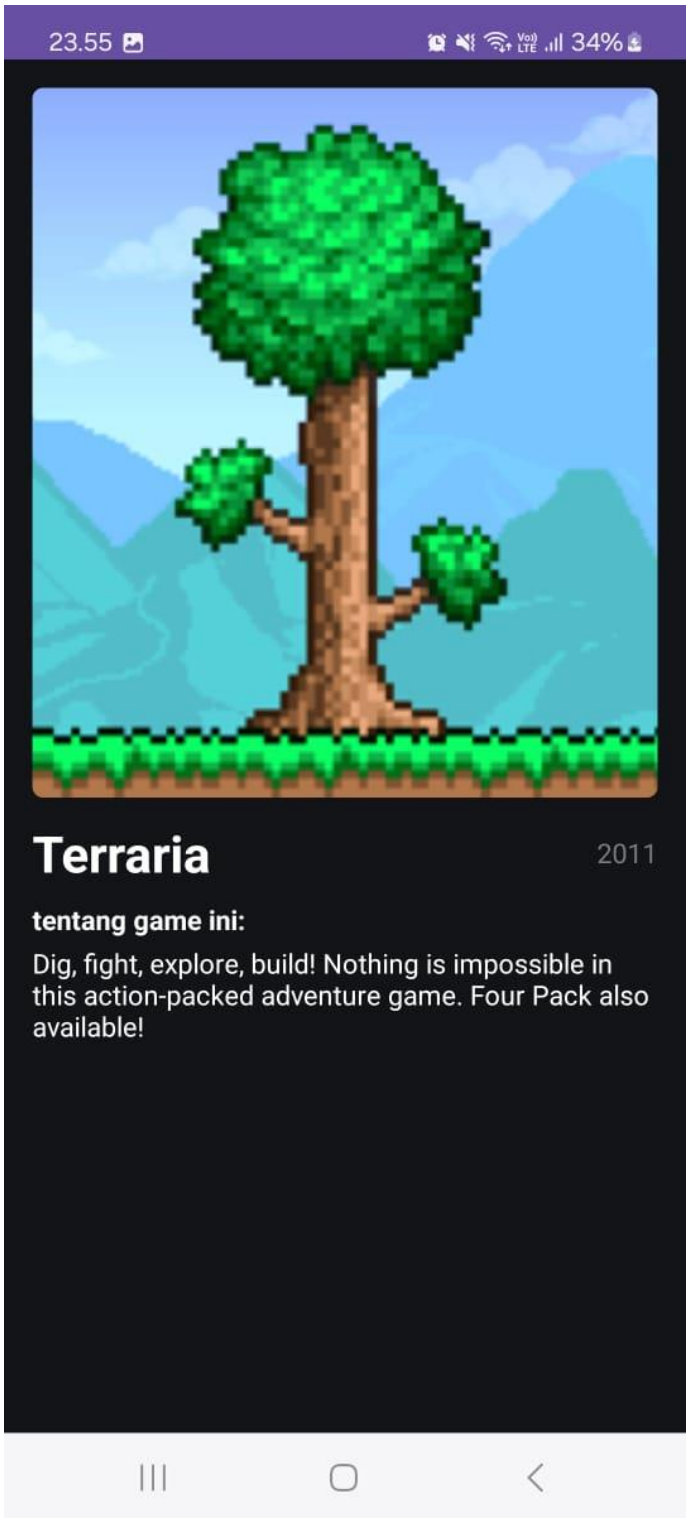
```
1   package com.pemrogamanmobile.myfavoritegames
2
3   import android.app.Application
4   import timber.log.Timber
5
6   class MyFavoriteGamesApplication : Application() {
7       override fun onCreate() {
8           super.onCreate()
```

```
9           if (BuildConfig.DEBUG) {
10              Timber.plant(Timber.DebugTree())
11          }
12      }
13  }
```

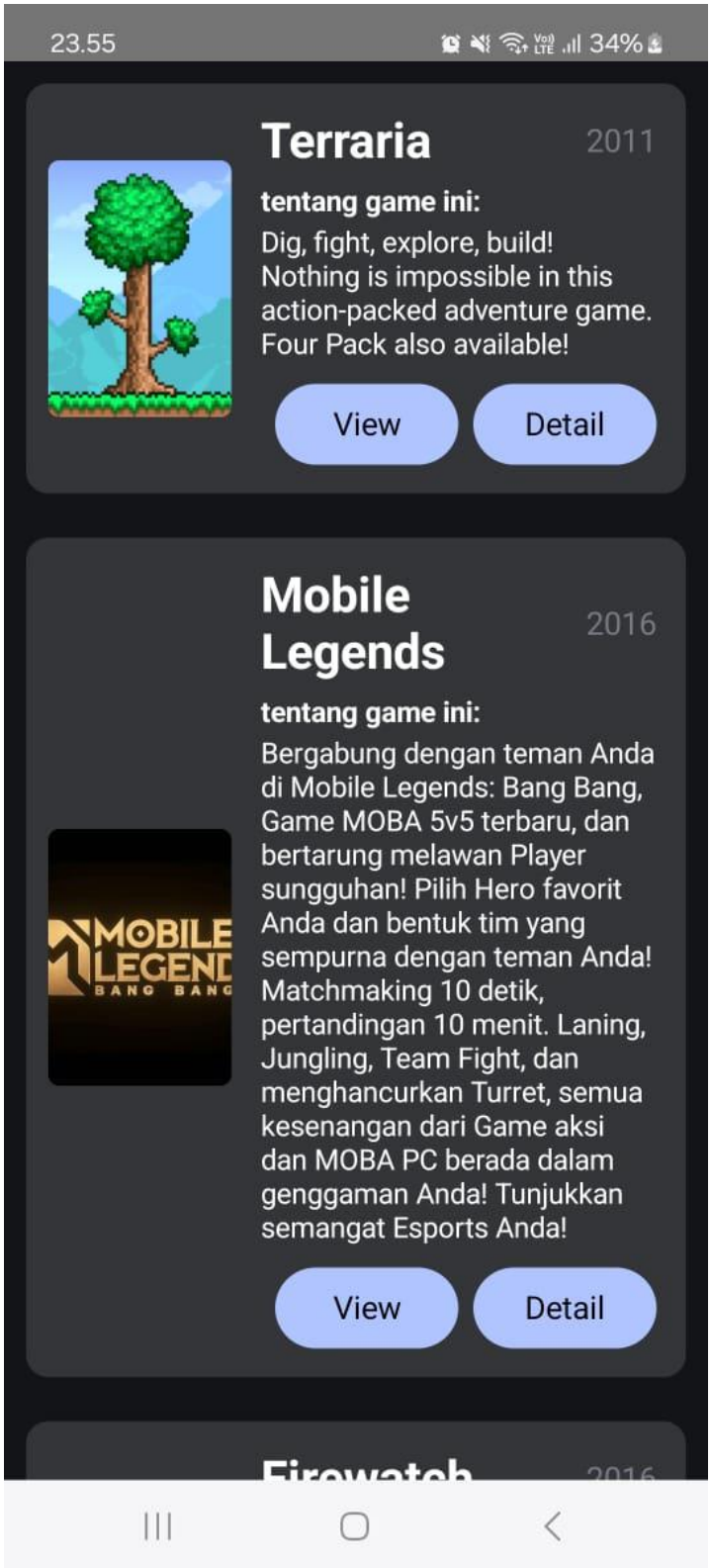
B. Output Program



Gambar 2. Screenshot Hasil Jawaban Soal 1 XML bagian a

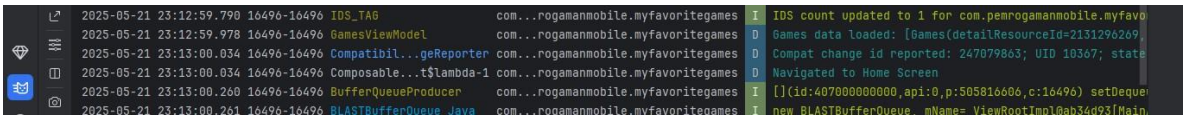


Gambar 3. Screenshot Hasil Jawaban Soal 1 XML bagian b

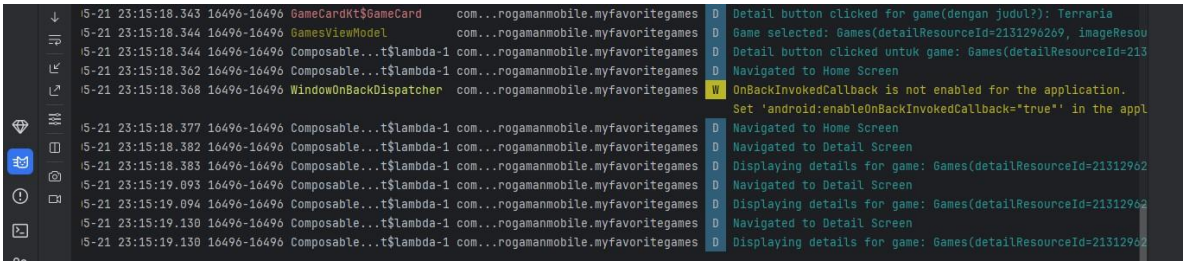


Gambar 4. Screenshot Hasil Jawaban Soal 1 compose bagian a

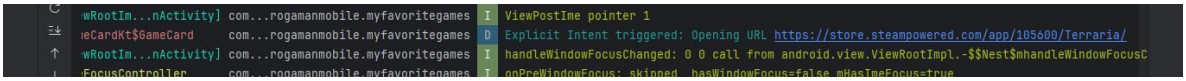
Gambar 5. Screenshot Hasil Jawaban Soal 1 compose bagian b



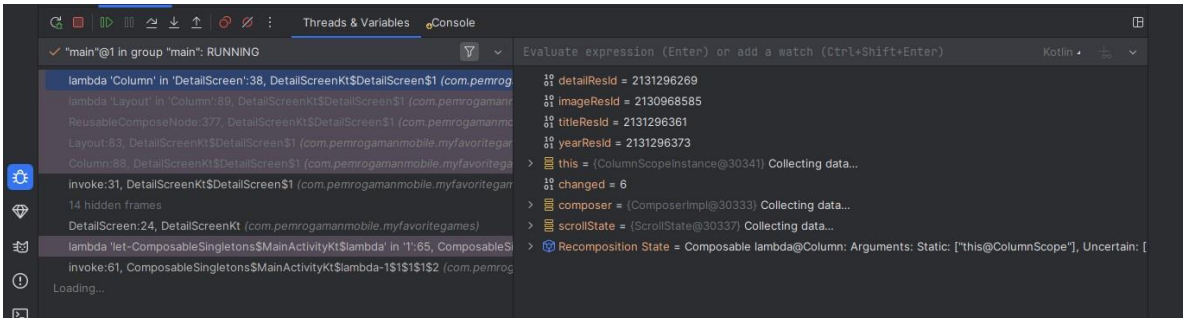
Gambar 6 . Screenshot Hasil Jawaban Soal 1 compose Log saat data item masuk ke dalam list



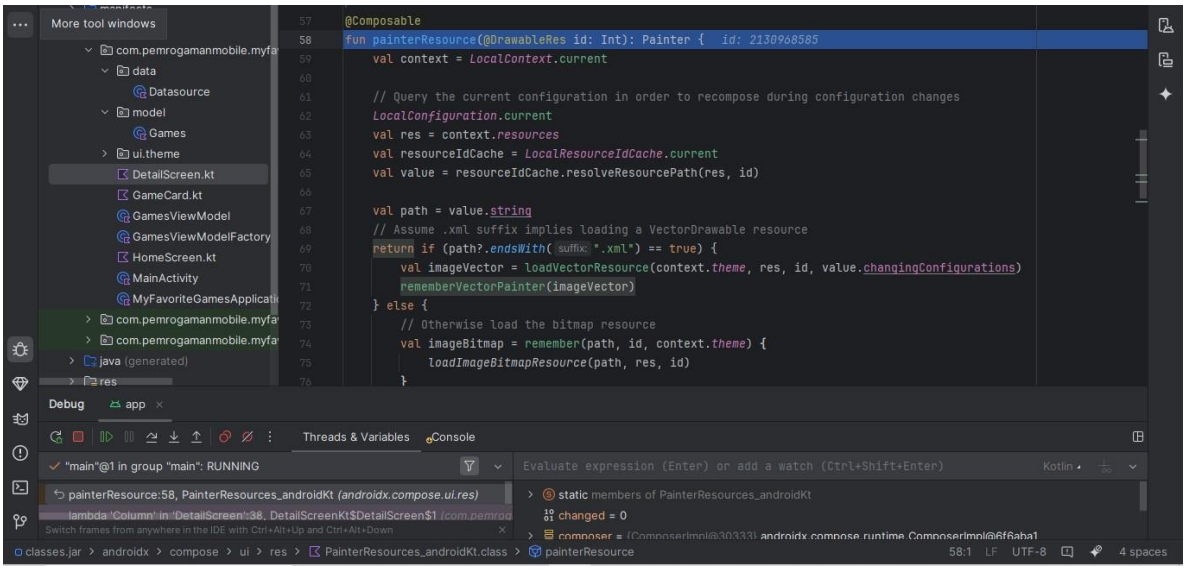
Gambar 7 Screenshot Hasil Jawaban Soal 1 compose Log saat tombol Detail dan Log data dari list yang dipilih ketika berpindah ke halaman Detail



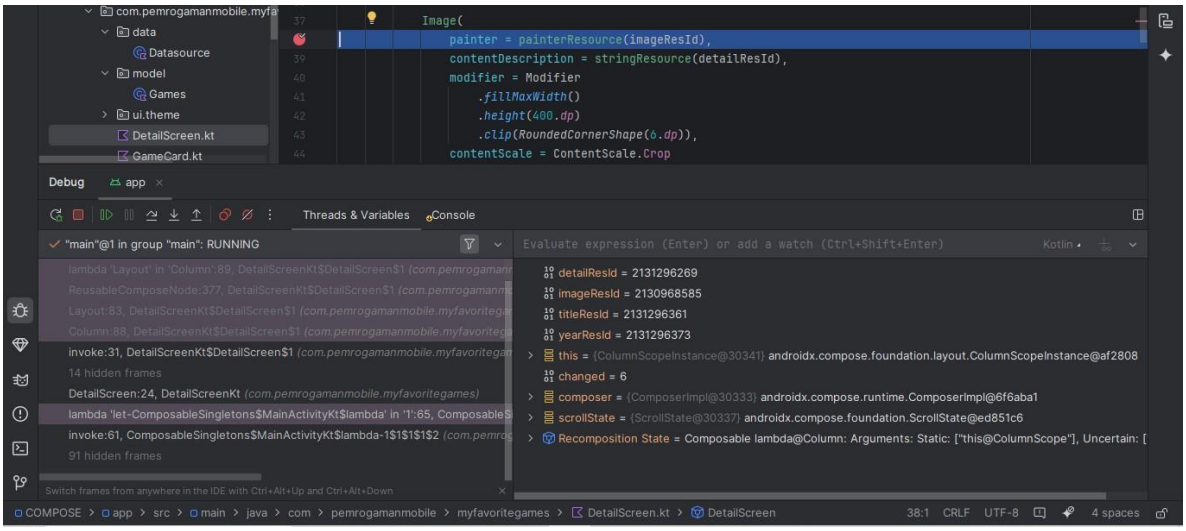
Gambar 8 Screenshot Hasil Jawaban Soal 1 compose Log saat tombol Explicit Intent ditekan



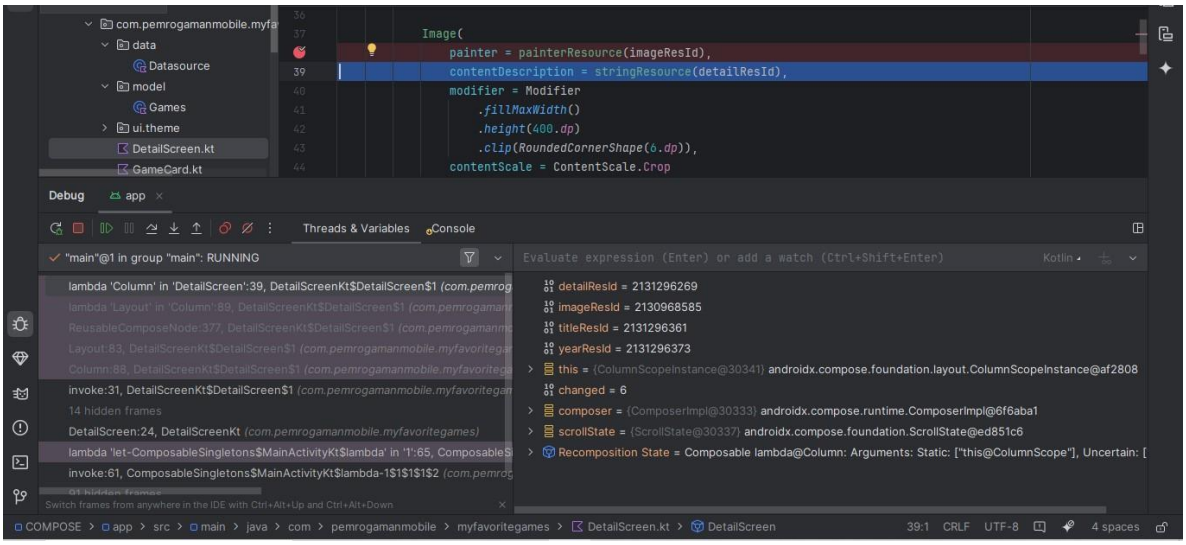
Gambar 9 Screenshot Hasil Jawaban Soal 1 compose debugger



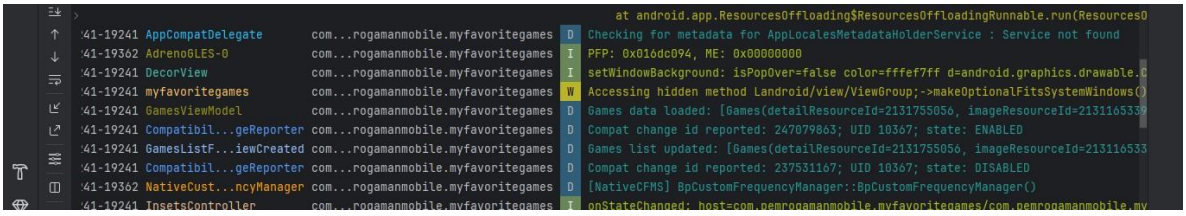
Gambar 10 Screenshot Hasil Jawaban Soal 1 compose debugger step into



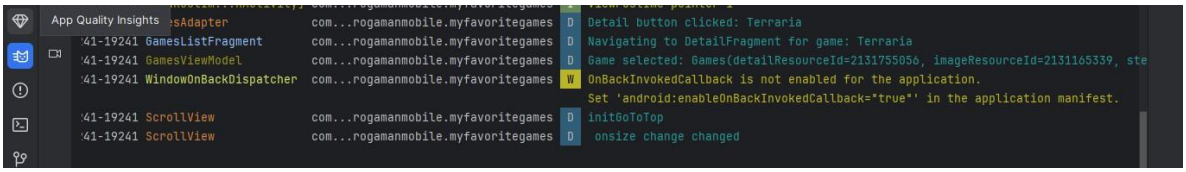
Gambar 11 Screenshot Hasil Jawaban Soal 1 compose debugger step out



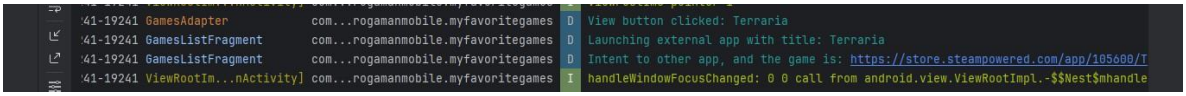
Gambar 12 Screenshot Hasil Jawaban Soal 1 compose debugger step over



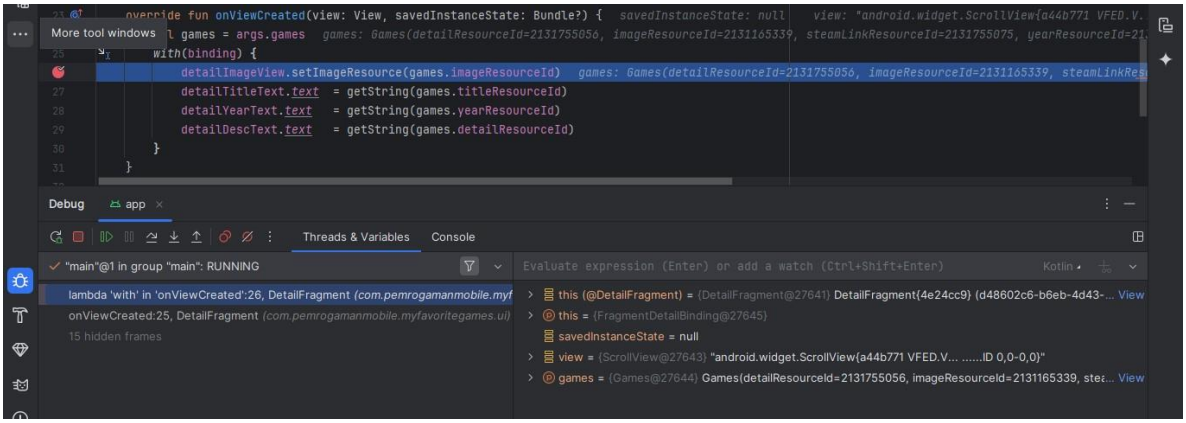
Gambar 13 Screenshot Hasil Jawaban Soal 1 xml Log saat data item masuk ke dalam list



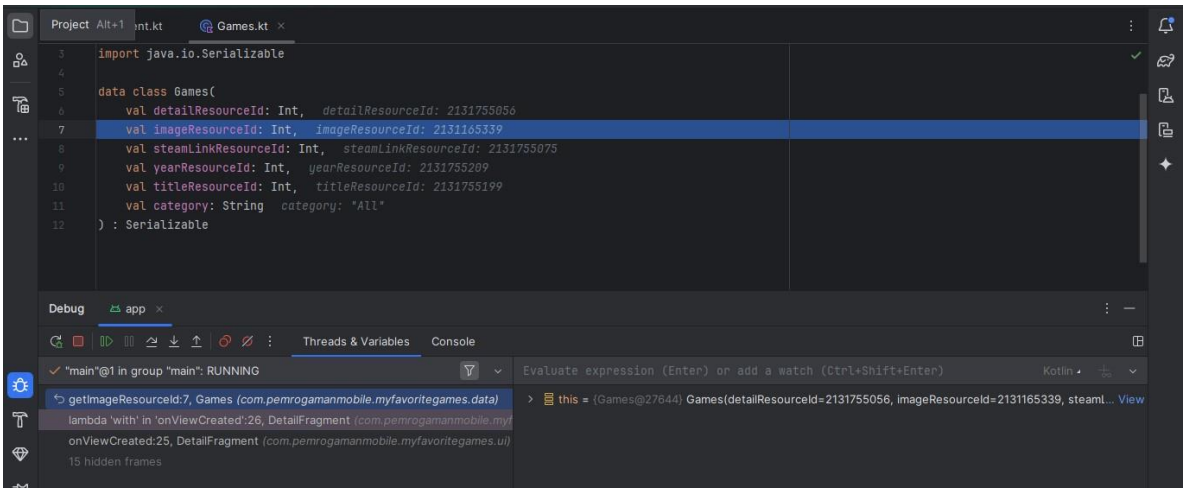
Gambar 14 Screenshot Hasil Jawaban Soal 1 xml Log saat tombol Detail dan Log data dari list yang dipilih ketika berpindah ke halaman Detail



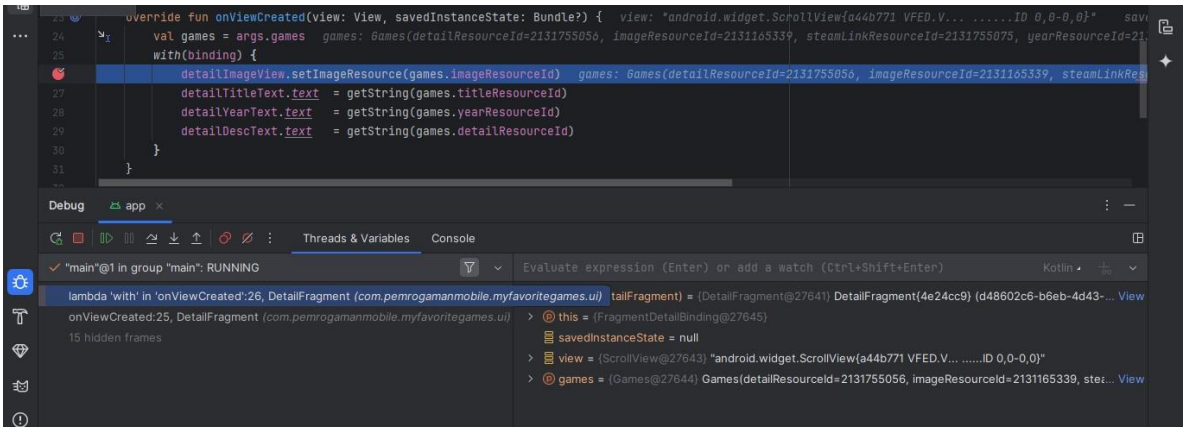
Gambar 15 Screenshot Hasil Jawaban Soal 1 xml Log saat tombol Explicit Intent ditekan



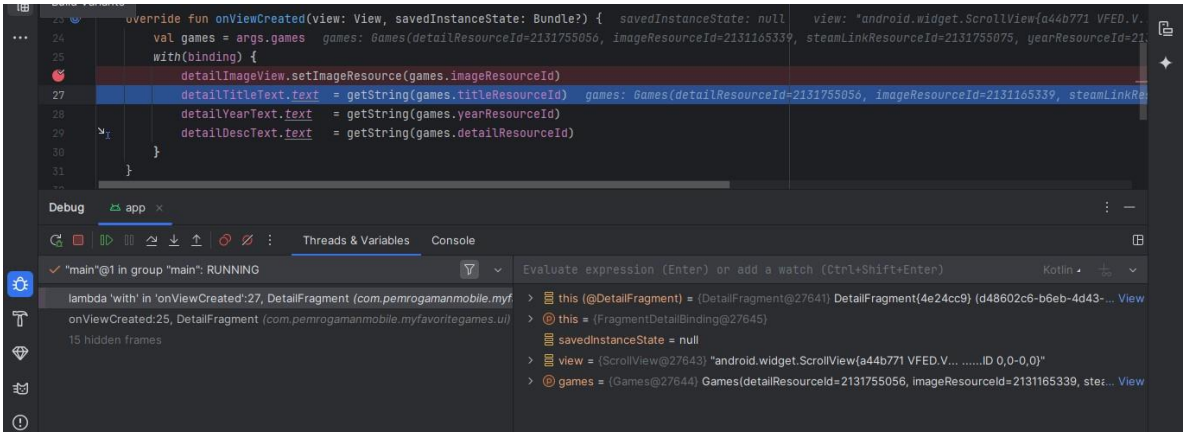
Gambar 16 Screenshot Hasil Jawaban Soal 1 xml debugger



Gambar 17 Screenshot Hasil Jawaban Soal 1 xml step into



Gambar 18 Screenshot Hasil Jawaban Soal 1 xml step out



Gambar 19 Screenshot Hasil Jawaban Soal 1 xml step over

C. Pembahasan

XML :

Games.kt

Pada baris ke [1] terdapat package com.pemrogamanmobile.myfavoritegames.data yang berfungsi sebagai nama paket dari aplikasi dan tempat menyimpan file ini agar terorganisir sesuai struktur proyek Android.

Pada baris ke [3] terdapat import java.io.Serializable yang berfungsi untuk mengimport kelas Serializable, agar objek dari kelas ini bisa dikirim antar activity melalui intent.

Pada baris ke [5] terdapat data class Games(yang berfungsi untuk membuat kelas data bernama Games, yang secara otomatis menyediakan fungsi seperti toString(), equals(), hashCode(), dan copy().

Pada baris ke [6] sampai [11] berfungsi untuk mendeklarasikan properti dalam kelas Games, yaitu detailResourceId, imageResourceId, steamLinkResourceId, yearResourceId, titleResourceId dan category. yang masing-masing bertipe Int dan satu string, dan berfungsi sebagai ID dari resource yang menyimpan informasi detail teks, gambar game, tautan Steam, tahun rilis, judul game dan katagori yang akan digunakan pada aplikasi.

Pada baris ke [12] terdapat) : Serializable yang berfungsi untuk menjadikan kelas Games sebagai objek yang dapat diserialisasi agar bisa dikirim melalui Intent.

Datasource.kt

Pada baris ke [1] terdapat package com.pemrogamanmobile.myfavoritegames.data yang berfungsi sebagai nama paket dari aplikasi dan tempat menyimpan file ini agar terorganisir sesuai struktur proyek Android.

Pada baris ke [3] terdapat import com.pemrogamanmobile.myfavoritegames.R yang berfungsi untuk mengakses resource seperti string dan drawable yang terdapat di dalam file res aplikasi Android ini.

Pada baris ke [5] terdapat object DataSource yang berfungsi untuk mendefinisikan object singleton bernama DataSource, artinya hanya ada satu instance dari objek ini yang akan digunakan sepanjang aplikasi berjalan.

Pada baris ke [6] terdapat fun loadGames(): List<Games> = listOf(yang berfungsi untuk mendeklarasikan fungsi bernama loadGames yang mengembalikan daftar dari objek Games dalam bentuk List.

Pada baris ke [7] sampai [16] berfungsi untuk mengisi list tersebut dengan sepuluh objek Games yang masing-masing berisi enam parameter, yaitu detail, image, link ke Steam, tahun rilis, judul game dan katagori. Keenam parameter ini merujuk pada ID resource yang disimpan dalam file strings.xml dan drawable di folder res dan ada kalimat "All". Setiap baris merepresentasikan satu game favorit yang akan ditampilkan di aplikasi.

GamesAdapter.kt

Pada baris ke [1] terdapat deklarasi package com.pemrogamanmobile.myfavoritegames.ui yang menetapkan namespace tempat file ini berada dalam struktur proyek aplikasi Android

Pada baris ke [3]sampai [9] terdapat rangkaian import yang mencakup android annotation SuppressLint untuk menonaktifkan peringatan Lint tertentu, android view LayoutInflater untuk menghasilkan layout dari XML, android view ViewGroup sebagai wadah view, androidx recyclerview widget RecyclerView untuk membuat daftar geser, kelas Games dari modul data sebagai model item, ItemGamesBinding untuk binding layout item, dan Timber untuk logging terstruktur

Pada baris ke [11] sampai [14] terdapat deklarasi kelas GamesAdapter turunan RecyclerView Adapter dengan dua callback yaitu onViewClicked dan onDetailClicked yang akan dipanggil saat tombol view atau detail di dalam setiap item diklik

Pada baris ke [16] terdapat properti items sebagai mutable list of Games yang menyimpan data daftar game untuk ditampilkan

Pada baris ke [18] terdapat inner class ViewHolder yang menyimpan instance ItemGamesBinding dan mewarisi RecyclerView ViewHolder agar setiap item baris dapat diikat ke view binding

Pada baris ke [20] sampai [25] terdapat metode override onCreateViewHolder yang menggunakan LayoutInflater dari context parent untuk menginflate ItemGamesBinding kemudian mengembalikan instance ViewHolder berisi binding tersebut

Pada baris ke [27] sampai [53] terdapat metode override onBindViewHolder yang mengambil item berdasarkan posisi lalu melakukan konfigurasi view dengan menerapkan margin bawah melalui layoutParams, mengisi imageView dan teks title year dan detail dari resource ID di item, serta mendaftarkan listener pada viewButton dan detailButton untuk mencatat log menggunakan Timber dan memanggil callback onViewClicked atau onDetailClicked

Pada baris ke [55] terdapat override getItemCount yang mengembalikan ukuran list items sebagai jumlah baris yang akan ditampilkan

Pada baris ke [57] sampai [62] terdapat fungsi submitList dengan anotasi SuppressLint NotifyDataSetChanged yang membersihkan konten items lalu menambahkan semua elemen newItems dan memanggil notifyDataSetChanged agar RecyclerView melakukan refresh seluruh tampilan

GamesListFragment.kt

Pada baris ke [1] terdapat deklarasi package com.pemrogamanmobile.myfavoritegames.ui yang menetapkan namespace tempat file ini berada dalam struktur proyek aplikasi Android

Pada baris ke [3] sampai [19] terdapat rangkaian import yang mencakup Intent dan Uri untuk navigasi antar aplikasi, Bundle untuk menyimpan state, LayoutInflater, View, dan ViewGroup untuk menginflate dan mengelola tampilan, Fragment dan viewModels dari Jetpack untuk lifecycle dan dependency injection, Lifecycle, lifecycleScope, dan repeatOnLifecycle dari Lifecycle KTX untuk coroutine yang terikat pada lifecycle, findNavController untuk navigasi antar fragment, GamesViewModel dan GamesViewModelFactory sebagai lapisan presentasi, FragmentGamesListBinding untuk view binding, kotlinx.coroutines.launch untuk coroutine scope, serta Timber untuk logging terstruktur

Pada baris ke [21] terdapat deklarasi class GamesListFragment yang merupakan subclass dari Fragment dan menjadi kontainer UI daftar game

Pada baris ke [22] terdapat properti _binding sebagai nullable FragmentGamesListBinding yang akan menahan referensi binding sampai view dihancurkan

Pada baris ke [23] terdapat properti binding yang memanggil _binding!! untuk memperoleh instance binding non-null ketika view masih valid

Pada baris ke [25] sampai [27] terdapat inisialisasi viewModel dengan delegasi by viewModels menggunakan GamesViewModelFactory dan parameter “All” untuk memfilter data saat pembuatan viewModel

Pada baris ke [29] sampai [33] terdapat override onCreateView yang menginflate layout FragmentGamesListBinding lewat inflater dan container, menyimpan hasil binding ke _binding, lalu mengembalikan root view

Pada baris ke [35] sampai [54] terdapat override onViewCreated yang pertama membuat adapter GamesAdapter dengan dua lambda callback onViewClicked dan onDetailClicked; pada onViewClicked dibuat Intent ACTION_VIEW dengan URI dari resource steamLink, dicatat lognya dengan Timber lalu dipanggil startActivity; pada onDetailClicked dicatat log, viewModel.onGameClicked dipanggil, kemudian navigasi ke DetailFragment melalui NavController dengan actionListToDetail

Pada baris ke [55] terdapat pengaturan adapter pada recyclerView di binding agar menampilkan daftar game

Pada baris ke [57] sampai [64] dijalankan coroutine di viewLifecycleOwner.lifecycleScope yang menggunakan repeatOnLifecycle pada state STARTED untuk mengumpulkan gamesState dari viewModel, mencatat log setiap update, dan memanggil adapter.submitList agar RecyclerView memperbarui tampilannya

Pada baris ke [67] sampai [69] terdapat override onDestroyView yang memanggil super.onDestroyView dan menetapkan _binding ke null untuk mencegah memory leak.

DetailFragment.kt

Pada baris ke [1] terdapat package com.pemrogamanmobile.myfavoritegames.ui yang berfungsi sebagai nama paket dari file ini dan menunjukkan bahwa file ini berada di dalam folder ui.

Pada baris ke [3] sampai [9], berfungsi untuk mengimport beberapa kelas yang digunakan, yaitu Bundle, LayoutInflater, View, ViewGroup, Fragment, navArgs, dan FragmentDetailBinding. Import ini mendukung pembuatan tampilan fragment, navigasi menggunakan Safe Args, dan binding layout XML ke dalam kode.

Pada baris ke [11] terdapat class DetailFragment : Fragment() yang berfungsi untuk mendeklarasikan kelas DetailFragment sebagai subclass dari Fragment, yang merupakan bagian dari tampilan aplikasi.

Pada baris ke [12] terdapat private var _binding: FragmentDetailBinding? = null yang berfungsi untuk mendeklarasikan variabel _binding sebagai objek binding yang akan menghubungkan kode dengan layout fragment. Nilai awalnya adalah null untuk menghindari memory leak.

Pada baris ke [13] terdapat private val binding get() = _binding!! yang berfungsi untuk membuat properti binding yang pasti tidak null saat digunakan, menggunakan not-null assertion (!!).

Pada baris ke [15] terdapat private val args: DetailFragmentArgs by navArgs() yang berfungsi untuk mengambil data dari argument navigasi menggunakan Safe Args, yaitu data Games yang dikirim dari fragment sebelumnya.

Pada baris ke [17] sampai [21] terdapat fungsi onCreateView yang berfungsi untuk meng-inflate layout fragment_detail.xml menggunakan View Binding dan mengembalikan tampilan utama dari fragment.

Pada baris ke [23] terdapat fungsi onViewCreated yang berfungsi untuk mengatur tampilan fragment setelah layout selesai dibuat.

Pada baris ke [24] terdapat val Games = args.Games yang berfungsi untuk mengambil objek Games dari argumen Safe Args yang dikirim melalui navigasi.

Pada baris ke [25] sampai [30] berfungsi untuk mengatur konten layout berdasarkan data dari objek Games, seperti gambar, judul, tahun rilis, dan deskripsi dengan menggunakan getString() berdasarkan ID resource masing-masing.

Pada baris ke [33] sampai [36] terdapat fungsi onDestroyView yang berfungsi untuk menghapus referensi binding saat tampilan fragment dihancurkan, untuk mencegah memory leak dengan mengatur _binding = null.

MainActivity.kt:

Pada baris ke [1] terdapat package com.pemrogamanmobile.myfavoritegames yang berfungsi sebagai nama paket utama dari aplikasi ini dan menunjukkan bahwa file ini berada di bagian utama dari struktur folder aplikasi.

Pada baris ke [3] sampai [5], berfungsi untuk mengimpor beberapa kelas yang akan digunakan, yaitu Bundle dari Android untuk menyimpan data saat pembuatan activity, AppCompatActivity sebagai superclass activity, dan ActivityMainBinding untuk menghubungkan layout XML activity ke dalam kode melalui View Binding.

Pada baris ke [7] terdapat class MainActivity : AppCompatActivity() yang berfungsi untuk mendeklarasikan MainActivity sebagai activity utama aplikasi dan mewarisi perilaku dari AppCompatActivity.

Pada baris ke [8] terdapat private lateinit var binding: ActivityMainBinding yang berfungsi untuk mendeklarasikan variabel binding yang akan diinisialisasi nanti dan digunakan untuk mengakses komponen dalam layout activity melalui View Binding.

Pada baris ke [10] sampai [14] terdapat fungsi onCreate yang merupakan lifecycle method yang dijalankan saat activity pertama kali dibuat.

Pada baris ke [11] terdapat super.onCreate(savedInstanceState) yang berfungsi untuk memanggil implementasi onCreate dari superclass (AppCompatActivity).

Pada baris ke [12] terdapat binding = ActivityMainBinding.inflate(layoutInflater) yang berfungsi untuk menginisialisasi objek binding menggunakan layout inflater agar layout XML bisa digunakan dalam kode.

Pada baris ke [13] terdapat setContentView(binding.root) yang berfungsi untuk menampilkan layout utama activity ke layar dengan menggunakan root dari objek binding.

GamesViewModel.kt

Pada baris ke [1] terdapat deklarasi package com.pemrogamanmobile.myfavoritegames yang menetapkan namespace tempat file ini berada dalam struktur proyek aplikasi Android

Pada baris ke [2] sampai [9] terdapat rangkaian import yang mencakup ViewModel dari Jetpack lifecycle DataSource dan Games dari modul data MutableStateFlow StateFlow dan fungsi update dari kotlinx coroutines flow serta Timber untuk logging terstruktur

Pada baris ke [11] sampai [13] terdapat deklarasi class GamesViewModel dengan parameter privat gameCategory bertipe String dan pewarisan dari ViewModel sebagai lapisan presentasi untuk logika UI

Pada baris ke [16] sampai [23] terdapat deklarasi properti _gamesState sebagai MutableStateFlow yang diinisialisasi dengan daftar game hasil filter category sama dengan gameCategory serta gamesState sebagai StateFlow publik untuk observasi data dan properti _selectedGame sebagai MutableStateFlow nullable Games diinisialisasi null bersama selectedGame sebagai StateFlow publik untuk menyimpan pilihan pengguna

Pada baris ke [25] sampai [28] terdapat blok init yang mencatat log data game yang berhasil dimuat menggunakan Timber dengan nilai _gamesState value

Pada baris ke [31] sampai [34] terdapat fungsi onGameClicked dengan parameter game yang memperbarui nilai _selectedGame menggunakan update dan mencatat log game terpilih melalui Timber

GamesViewModelFactory.kt

Pada baris ke [1] terdapat deklarasi package com.pemrogamanmobile.myfavoritegames yang menetapkan namespace tempat file ini berada dalam struktur proyek aplikasi Android

Pada baris ke [3] sampai [4] terdapat import ViewModel dan ViewModelProvider dari Jetpack Lifecycle yang memungkinkan pembuatan dan penyediaan instance ViewModel secara dinamis

Pada baris ke [6] sampai [8] terdapat deklarasi class GamesViewModelFactory dengan parameter privat gameCategory bertipe String yang mengimplementasi ViewModelProvider Factory sebagai mekanisme injeksi dependensi untuk GamesViewModel

Pada baris ke [9] terdapat override fungsi create generik dengan parameter modelClass bertipe Class T yang mengembalikan instance ViewModel sesuai kelas yang diminta

Pada baris ke [10] sampai [12] terdapat blok kondisi yang memeriksa apakah modelClass dapat diassignable dari GamesViewModel kemudian mereturn instance GamesViewModel dengan gameCategory sambil menonaktifkan peringatan unchecked cast melalui anotasi Suppress

Pada baris ke [14] terdapat perintah throw IllegalArgumentException dengan pesan Unknown ViewModel class yang akan dieksekusi jika kelas ViewModel yang diminta tidak dikenali oleh factory

MyAplication.kt

Pada baris ke [1] terdapat deklarasi package com.pemrogamanmobile.myfavoritegames yang menetapkan namespace tempat file ini berada dalam struktur proyek aplikasi Android

Pada baris ke [3] sampai [5] terdapat import android app Application import Timber log pustaka untuk logging terstruktur dan import BuildConfig yang berisi informasi build agar kelas ini dapat menggunakan flag DEBUG

Pada baris ke [7] sampai [16] terdapat deklarasi kelas MyApplication turunan Application dengan override fungsi onCreate yang memanggil super onCreate lalu memeriksa jika BuildConfig DEBUG bernilai true maka melakukan inisialisasi Timber dengan DebugTree agar log hanya muncul dalam mode debug

activity_main.xml:

Pada baris ke [1], terdapat deklarasi <?xml version="1.0" encoding="utf-8"?> yang berfungsi untuk menyatakan versi XML yang digunakan serta karakter encoding yang dipakai, yaitu UTF-8.

Pada baris ke [2], terdapat elemen FragmentContainerView yang berasal dari pustaka androidx.fragment.app. Elemen ini digunakan sebagai wadah untuk menampilkan fragmen.

Pada baris ke [3], terdapat deklarasi namespace xmlns:android dengan nilai http://schemas.android.com/apk/res/android, yang digunakan untuk mendefinisikan atribut-atribut khusus Android pada elemen-elemen XML.

Pada baris ke [4], terdapat deklarasi namespace tambahan xmlns:app dengan nilai http://schemas.android.com/apk/res-auto, yang memungkinkan penggunaan atribut kustom dari pustaka AndroidX.

Pada baris ke [5], terdapat atribut android:id="@+id/nav_host_fragment", yang menetapkan ID unik untuk elemen FragmentContainerView ini sebagai nav_host_fragment.

Pada baris ke [6], atribut android:name="androidx.navigation.fragment.NavHostFragment" mendefinisikan bahwa elemen ini adalah NavHostFragment, yang merupakan komponen utama untuk navigasi menggunakan Jetpack Navigation.

Pada baris ke [7], atribut android:layout_width="match_parent" digunakan untuk mengatur lebar elemen agar sesuai dengan lebar layar perangkat.

Pada baris ke [8], atribut android:layout_height="match_parent" digunakan untuk mengatur tinggi elemen agar sesuai dengan tinggi layar perangkat.

Pada baris ke [9], atribut app:navGraph="@navigation/nav_graph" menghubungkan elemen ini dengan file navigasi bernama nav_graph, yang berisi konfigurasi alur navigasi aplikasi.

Pada baris ke [10], atribut app:defaultNavHost="true" menetapkan elemen ini sebagai host navigasi utama, sehingga mampu menangani tindakan Back pada perangkat secara otomatis.

fragment_Games_list.xml

Pada baris ke [1], terdapat deklarasi <?xml version="1.0" encoding="utf-8"?> yang berfungsi untuk menyatakan versi XML yang digunakan serta karakter encoding yang dipakai, yaitu UTF-8.

Pada baris ke [2], terdapat elemen root ConstraintLayout dari pustaka androidx.constraintlayout.widget. Elemen ini digunakan sebagai layout utama yang memungkinkan penempatan elemen-elemen anak dengan aturan constraint.

Pada baris ke [3], terdapat deklarasi namespace xmlns:android dengan nilai http://schemas.android.com/apk/res/android, yang digunakan untuk mendefinisikan atribut-atribut khusus Android pada elemen-elemen XML.

Pada baris ke [4], terdapat deklarasi namespace tambahan xmlns:app dengan nilai http://schemas.android.com/apk/res-auto, yang memungkinkan penggunaan atribut kustom dari pustaka AndroidX.

Pada baris ke [5], atribut android:background="#000000" digunakan untuk mengatur latar belakang layout dengan warna hitam.

Pada baris ke [6], atribut android:layout_width="match_parent" digunakan untuk mengatur lebar layout agar sesuai dengan lebar layar perangkat.

Pada baris ke [7], atribut android:layout_height="match_parent" digunakan untuk mengatur tinggi layout agar sesuai dengan tinggi layar perangkat.

Pada baris ke [9], terdapat elemen RecyclerView dari pustaka androidx.recyclerview.widget. Elemen ini digunakan untuk menampilkan daftar data dengan performa tinggi.

Pada baris ke [10], atribut android:id="@+id/recyclerView" menetapkan ID unik untuk elemen ini sebagai recyclerView.

Pada baris ke [11], atribut android:contentDescription="@string/Games_list_desc" memberikan deskripsi konten yang bersifat aksesibilitas untuk RecyclerView, menggunakan nilai dari sumber string bernama Games_list_desc.

Pada baris ke [12] sampai [13], terdapat atribut android:layout_width="0dp" dan android:layout_height="0dp" mengatur lebar dan tinggi elemen ini agar dapat diatur sepenuhnya oleh constraint.

Pada baris ke [14], atribut android:padding="8dp" menambahkan jarak dalam sebesar 8dp di seluruh sisi elemen ini.

Pada baris ke [15], atribut app:layoutManager="LinearLayoutManager" menetapkan tata letak RecyclerView menggunakan LinearLayoutManager, yang menyusun elemen secara vertikal atau horizontal.

Pada baris ke [15] hingga [18], terdapat atribut constraint yang menghubungkan sisi atas, bawah, kiri, dan kanan RecyclerView dengan sisi parent, sehingga elemen ini berada di tengah layout secara proporsional.

fragment_detail.xml

Pada baris ke [1], terdapat deklarasi <?xml version="1.0" encoding="utf-8"?> yang berfungsi untuk menyatakan versi XML yang digunakan serta karakter encoding yang dipakai, yaitu UTF-8.

Pada baris ke [2], terdapat elemen root ScrollView yang berasal dari pustaka bawaan Android. Elemen ini digunakan sebagai wadah untuk memungkinkan konten di dalamnya dapat digulir secara vertikal. deklarasi namespace xmlns:android dengan nilai http://schemas.android.com/apk/res/android digunakan untuk mendefinisikan atribut-atribut khusus Android pada elemen-elemen XML.

Pada baris ke [3], deklarasi namespace tambahan xmlns:app dengan nilai http://schemas.android.com/apk/res-auto memungkinkan penggunaan atribut kustom dari pustaka AndroidX atau Material.

Pada baris ke [4], atribut android:background="#131418" mengatur latar belakang elemen dengan warna abu-abu gelap (#131418).

Pada baris ke [5] dan [6], atribut android:layout_width="match_parent" dan android:layout_height="match_parent" digunakan untuk mengatur elemen agar mengisi seluruh lebar dan tinggi layar perangkat.

Pada baris ke [8] sampai [9], terdapat elemen LinearLayout dengan orientasi vertikal yang berfungsi sebagai kontainer utama untuk menampung elemen-elemen lainnya.

Pada baris ke [10] sampai [12], atribut android:padding="16dp" menambahkan jarak sebesar 16dp di semua sisi elemen ini untuk memberikan ruang antara konten dan tepi layar.

Pada baris ke [14] hingga [19], terdapat elemen ShapeableImageView dari pustaka Material Components. Elemen ini menampilkan gambar dengan ID detailImageView, lebar penuh, tinggi 400dp, skala centerCrop, dan sudut membulat dengan gaya yang diatur oleh @style/ShapeAppearance.App.Rounded6dp.

Pada baris ke [21], terdapat LinearLayout horizontal yang digunakan untuk menempatkan teks secara berdampingan.

Pada baris ke [25], atribut android:gravity="center_vertical" digunakan untuk menyelaraskan konten di dalam elemen secara vertikal di tengah.

Pada baris ke [26], atribut android:layout_marginTop="12dp" menambahkan jarak sebesar 12dp antara elemen ini dan elemen di atasnya.

Pada baris ke [28] sampai [35], terdapat elemen TextView dengan ID detailTitleText, yang digunakan untuk menampilkan judul. Elemen ini menggunakan warna teks putih, ukuran teks 26sp, dan gaya teks tebal (bold).

Pada baris ke [37] hingga [42], terdapat elemen TextView dengan ID detailYearText, yang digunakan untuk menampilkan tahun. Elemen ini menggunakan warna teks abu-abu terang, ukuran teks 14sp, dan tinggi sesuai konten.

Pada baris ke [45] hingga [52], terdapat elemen TextView tanpa ID yang menampilkan teks “tentang game ini:” sebagai label. Elemen ini menggunakan warna teks putih, ukuran teks 14sp, dan gaya teks tebal.

Pada baris ke [54] hingga [60], terdapat elemen TextView dengan ID detailDescText, yang digunakan untuk menampilkan deskripsi detail. Elemen ini menggunakan warna teks putih, ukuran teks 14sp, dan margin atas sebesar 4dp untuk memberi jarak dari elemen di atasnya.

item_Games.xml

Pada baris ke [1], terdapat deklarasi <?xml version="1.0" encoding="utf-8"?> yang berfungsi untuk menyatakan versi XML yang digunakan serta karakter encoding yang dipakai, yaitu UTF-8.

Pada baris ke [2], terdapat elemen root CardView dari pustaka androidx.cardview.widget. Elemen ini digunakan untuk menampilkan konten dalam bentuk kartu dengan sudut membulat dan latar belakang yang dapat disesuaikan.

Pada baris ke [3], deklarasi namespace xmlns:android dengan nilai http://schemas.android.com/apk/res/android digunakan untuk mendefinisikan atribut-atribut khusus Android pada elemen-elemen XML.

Pada baris ke [4], deklarasi namespace tambahan xmlns:app dengan nilai http://schemas.android.com/apk/res-auto memungkinkan penggunaan atribut kustom dari pustaka AndroidX.

Pada baris ke [5] hingga [6], atribut android:layout_width="match_parent" dan android:layout_height="wrap_content" digunakan untuk mengatur elemen agar memenuhi lebar layar perangkat dengan tinggi yang menyesuaikan kontennya.

Pada baris ke [7], atribut android:layout_margin="4dp" menambahkan jarak luar sebesar 4dp di seluruh sisi elemen ini.

Pada baris ke [8], atribut app:cardCornerRadius="6dp" digunakan untuk mengatur radius sudut kartu menjadi 6dp agar tampak membulat.

Pada baris ke [9], atribut app:cardBackgroundColor="#343539" mengatur warna latar belakang kartu dengan warna abu-abu gelap (#343539).

Pada baris ke [11], terdapat elemen LinearLayout dengan orientasi horizontal yang digunakan untuk menata elemen-elemen anak secara berdampingan.

Pada baris ke [15], atribut android:padding="8dp" menambahkan jarak dalam sebesar 8dp di seluruh sisi elemen ini.

Pada baris ke [18] hingga [25], terdapat elemen ShapeableImageView dari pustaka Material Components, yang menampilkan gambar dengan ID imageView. Elemen ini memiliki lebar 100dp, tinggi 140dp, margin kanan 12dp, skala gambar centerCrop, dan sudut membulat sesuai dengan gaya @style/ShapeAppearance.App.Rounded6dp.

Pada baris ke [27] sampai [37], terdapat LinearLayout vertikal yang digunakan untuk menata elemen-elemen teks dan tombol secara bertumpuk di dalamnya.

Pada baris ke [29], atribut android:layout_weight="1" digunakan untuk membuat elemen ini mengisi ruang yang tersisa dalam LinearLayout horizontal induknya.

Pada baris ke [39] hingga [53], terdapat elemen TextView untuk menampilkan judul dan tahun. Teks judul menggunakan ID titleText, warna putih, ukuran teks 20sp, dan gaya tebal, sedangkan teks tahun menggunakan ID yearText, warna abu-abu terang, dan ukuran teks 14sp.

Pada baris ke [56] hingga [63], terdapat elemen TextView tanpa ID yang menampilkan teks "tentang game ini:" sebagai label dengan warna putih, ukuran teks 14sp, dan gaya tebal.

Pada baris ke [65] hingga [71], terdapat elemen TextView dengan ID detailText yang digunakan untuk menampilkan deskripsi tambahan dengan warna putih, ukuran teks 14sp, dan margin atas 4dp.

Pada baris ke [73] hingga [78], terdapat LinearLayout horizontal yang digunakan untuk menempatkan dua tombol di sisi kanan bawah kartu.

Pada baris ke [80] hingga [86], terdapat tombol pertama dengan ID viewButton yang memiliki teks "View", lebar 100dp, tinggi 44dp, warna latar biru terang (#b0c4ff), dan warna teks hitam.

Pada baris ke [88], terdapat elemen Space dengan lebar 8dp yang digunakan untuk memberi jarak antara kedua tombol.

Pada baris ke [90] hingga [96], terdapat tombol kedua dengan ID detailButton yang memiliki teks "Detail", lebar dan tinggi yang sama seperti tombol pertama, warna latar biru terang, dan warna teks hitam.

strings.xml

Pada baris ke [1], terdapat elemen root <resources> yang digunakan untuk menampung semua sumber daya string dalam aplikasi Android.

Pada baris ke [2], terdapat elemen <string> dengan atribut name="app_name" yang menyimpan nama aplikasi, yaitu "My favorite games".

Pada baris ke [4], terdapat elemen <string> dengan atribut name="Games_list_desc" yang berfungsi sebagai deskripsi aksesibilitas untuk daftar permainan favorit, yaitu "List of favorite games".

Pada baris ke [6] sampai [15], terdapat sepuluh elemen <string> dengan atribut name="titleX" yang masing-masing menyimpan judul permainan favorit berdasarkan nomor urut, seperti "Terraria", "Mobile Legends", dan lainnya.

Pada baris ke [17] sampai [26], terdapat sepuluh elemen <string> dengan atribut name="yearX" yang masing-masing menyimpan tahun rilis permainan berdasarkan nomor urut, seperti "2011" untuk year1 dan "2020" untuk year6.

Pada baris ke [28] sampai [37], terdapat sepuluh elemen <string> dengan atribut name="detailX" yang masing-masing menyimpan deskripsi detail permainan berdasarkan nomor urut. Contohnya, detail1 berisi deskripsi "Terraria", dan detail6 berisi deskripsi "Genshin Impact".

Pada baris ke [39] sampai [48], terdapat sepuluh elemen <string> dengan atribut name="linkX" yang masing-masing menyimpan URL terkait permainan berdasarkan nomor urut. Contohnya, link1 berisi URL ke halaman Steam "Terraria", dan link6 berisi URL ke halaman Google Play "Genshin Impact".

styles.xml

Pada baris ke [1], terdapat deklarasi <?xml version="1.0" encoding="utf-8"?> yang berfungsi untuk menyatakan versi XML yang digunakan serta karakter encoding yang dipakai, yaitu UTF-8.

Pada baris ke [2], terdapat elemen root <resources> yang digunakan untuk menampung semua sumber daya gaya (style) dalam aplikasi Android.

Pada baris ke [3], terdapat elemen <style> dengan atribut name="ShapeAppearance.App.Rounded6dp" yang mendefinisikan gaya bernama ShapeAppearance.App.Rounded6dp. Gaya ini digunakan untuk mengatur tampilan sudut elemen tertentu menjadi membulat dengan radius 6dp. atribut parent="" tidak memiliki nilai tertentu, menunjukkan bahwa gaya ini tidak mewarisi properti dari gaya lain.

Pada baris ke [4], terdapat elemen <item> dengan atribut name="cornerFamily" dan nilai rounded, yang menentukan bahwa elemen-elemen yang menggunakan gaya ini akan memiliki sudut membulat.

Pada baris ke [5], terdapat elemen <item> dengan atribut name="cornerSize" dan nilai 6dp, yang menentukan ukuran radius sudut membulat sebesar 6dp.

Pada baris ke [6], elemen </style> menutup definisi gaya ShapeAppearance.App.Rounded6dp.

Pada baris ke [7], elemen </resources> menutup elemen root <resources>.

COMPOSE :
Datasource.kt

Pada baris ke [1] terdapat package com.pemrogamanmobile.myfavoritegames.data yang berfungsi sebagai deklarasi nama paket tempat file ini disimpan dalam struktur proyek aplikasi Android.

Pada baris ke [2] (kosong) hanya berfungsi sebagai pemisah visual agar kode lebih mudah dibaca.

Pada baris ke [3] terdapat import com.pemrogamanmobile.myfavoritegames.R yang berfungsi untuk mengimpor file R, yaitu file auto-generated yang berisi referensi ke semua sumber daya (seperti string, drawable, dan layout) dalam aplikasi.

Pada baris ke [4] terdapat import com.pemrogamanmobile.myfavoritegames.model.Games yang berfungsi untuk mengimpor kelas Games dari package model, yaitu model data yang digunakan untuk merepresentasikan setiap item game favorit.

Pada baris ke [5] (kosong) kembali sebagai pemisah visual agar struktur kode lebih rapi.

Pada baris ke [6] terdapat deklarasi class Datasource { yang berfungsi sebagai sumber data untuk aplikasi ini.

Pada baris ke [7] terdapat fungsi fun loadGames(): List<Games> { yang akan mengembalikan sebuah List<Games>, yaitu daftar data yang berisi objek-objek Games.

Pada baris ke [8] terdapat return listOf(yang berfungsi mengembalikan hasil berupa daftar (List) objek Games.

Pada baris ke [9] hingga [18] setiap item Games diisi menggunakan referensi dari resource R.string, R.drawable, dan R.string lainnya, serta satu argumen literal "All".
Games.kt:

Pada baris ke [1] terdapat package com.pemrogamanmobile.myfavoritegames.model yang berfungsi sebagai deklarasi nama paket tempat file ini disimpan dalam struktur proyek aplikasi Android.

Pada baris ke [3] terdapat import androidx.annotation.DrawableRes yang berfungsi untuk mengimpor anotasi @DrawableRes, yaitu anotasi yang menandai bahwa suatu properti atau parameter harus merujuk pada ID resource drawable.

Pada baris ke [4] terdapat import androidx.annotation.StringRes yang berfungsi untuk mengimpor anotasi @StringRes, yaitu anotasi yang menandai bahwa suatu properti atau parameter harus merujuk pada ID resource string.

Pada baris ke [6] terdapat deklarasi data class Games(yang berfungsi untuk mendefinisikan kelas data Games, yaitu model data yang mewakili setiap entri game favorit beserta atribut-atributnya.

Pada baris ke [7] terdapat @StringRes val detailResourceId: Int, yang berfungsi sebagai properti detailResourceId bertipe Int dan dianotasi @StringRes, menunjukkan bahwa nilai integer ini mengacu pada ID resource string untuk deskripsi detail game.

Pada baris ke [8] terdapat @DrawableRes val imageResourceId: Int, yang berfungsi sebagai properti imageResourceId bertipe Int dan dianotasi @DrawableRes, menunjukkan bahwa nilai integer ini mengacu pada ID resource drawable untuk gambar cover game.

Pada baris ke [9] terdapat @StringRes val linkResourceId: Int, yang berfungsi sebagai properti linkResourceId bertipe Int dan dianotasi @StringRes, menunjukkan bahwa nilai integer ini mengacu pada ID resource string untuk menyimpan tautan terkait game.

Pada baris ke [10] terdapat @StringRes val yearResourceId: Int, yang berfungsi sebagai properti yearResourceId bertipe Int dan dianotasi @StringRes, menunjukkan bahwa nilai integer ini mengacu pada ID resource string untuk menyimpan informasi tahun rilis game.

Pada baris ke [11] terdapat @StringRes val titleResourceId: Int, yang berfungsi sebagai properti titleResourceId bertipe Int dan dianotasi @StringRes, menunjukkan bahwa nilai integer ini mengacu pada ID resource string untuk judul game.

Pada baris ke [12] terdapat val category: String yang berfungsi sebagai properti category bertipe String, yaitu kategori atau tag yang menggambarkan jenis atau pengelompokan game (misalnya "All", "Action", dsb.).

GamesCard.kt

Pada baris ke [1] terdapat package com.pemrogamanmobile.myfavoritegames yang berfungsi sebagai deklarasi nama paket tempat file ini disimpan dalam struktur proyek aplikasi Android.

Pada baris ke [3] sampai [23] terdapat rangkaian import yang berfungsi mengambil berbagai kelas dan fungsi dari Android SDK, Jetpack Compose, serta library tambahan (seperti Timber dan LocalContext) agar komponen UI, pengelolaan resource, penanganan intent, dan pencatatan log dapat digunakan di dalam fungsi GameCard.

Pada baris ke [25] terdapat anotasi @Composable yang menandai bahwa fungsi berikutnya merupakan blok pembentuk UI di Jetpack Compose.

Pada baris ke [26] sampai [30] didefinisikan fungsi GameCard yang menerima tiga parameter: objek games (tipe Games) sebagai data game yang akan ditampilkan, sebuah

modifier (dengan nilai default Modifier) untuk penyesuaian tampilan dari luar, dan onDetailClick (sebuah lambda) yang akan dipanggil saat tombol "Detail" ditekan.

Pada baris ke [31] terdapat inisialisasi val context = LocalContext.current yang mengambil konteks Android saat ini, digunakan untuk memanggil resource (seperti string dan drawable) serta memulai intent.

Pada baris ke [33] sampai [36] terdapat deklarasi Card yang menjadi wadah utama kartu; di sini diterapkan padding 12 dp dan warna latar abu-abu gelap (kode 0xFF343539).

Pada baris ke [37] sampai [49] terdapat Row pertama yang membagi area horizontal menjadi dua bagian: bagian gambar (Image) di sebelah kiri, diatur agar memiliki padding kiri 12 dp, lebar 100 dp, tinggi 140 dp, dan sudut membulat 6 dp; selain itu, gambar diambil dari games.imageResourceId dan di-crop agar memenuhi kotak tanpa merusak rasio aslinya; bagian konten di sebelah kanan, berisi teks dan tombol, diatur agar mengisi lebar yang tersisa secara vertikal.

Pada baris ke [51] sampai [56] terdapat Column yang menumpuk elemen teks vertikal: pada baris ke [57] sampai [77] terdapat Row khusus judul dan tahun rilis, judul diambil dari games.titleResourceId dengan ukuran font 26 sp, tebal, dan warna putih, diberi bobot agar menempati ruang maksimal; tahun rilis diambil dari games.yearResourceId dengan ukuran font 16 sp dan warna abu-abu terang, diselaraskan ke kanan serta vertikal di tengah baris, dan pada baris ke [81] sampai [91] terdapat Column yang menampilkan label "tentang game ini:" dengan gaya font 14 sp tebal dan warna putih, diikuti deskripsi detail game yang diambil dari games.detailResourceId dengan gaya font 14 sp dan warna putih, diberi jarak vertikal 4 dp di atasnya.

Pada baris ke [95] sampai [125] terdapat Row yang menata dua tombol di ujung kanan: tombol "View" (baris ke [100] sampai [111]) akan mengambil URL dari games.linkResourceId, membuat intent ACTION_VIEW untuk membuka browser, mencatat log menggunakan Timber, dan memulai aktivitas; tombol ini memiliki lebar 100 dp, tinggi 44 dp, latar biru muda (0xFFb0c4ff), dan teks "View" berukuran 16 sp berwarna hitam; tombol "Detail" (baris ke [115] sampai [124]) mencatat log dengan Timber berisi judul game, kemudian memanggil callback onDetailClick dengan objek games; ukuran dan warna tombol sama seperti tombol "View", dengan teks "Detail" berukuran 16 sp dan warna hitam.

Pada baris ke [125] sampai [129] seluruh blok tampilan (Row, Card, dan fungsi GameCard) ditutup, menandakan akhir definisi UI untuk satu kartu game.

DetailScreen.kt

Pada baris ke [1] terdapat package com.pemrogamanmobile.myfavoritegames yang berfungsi sebagai penamaan paket dari aplikasi dan sebagai tempat penyimpanan file DetailScreen.kt.

Pada baris ke [3] sampai [20], berfungsi untuk mengimpor berbagai komponen penting dari Jetpack Compose seperti Image, Column, Row, Text, Surface, Modifier, Color, dan lainnya yang digunakan untuk membuat tampilan layar detail game.

Pada baris ke [22] terdapat anotasi @Composable yang menunjukkan bahwa fungsi DetailScreen merupakan fungsi composable, yaitu fungsi yang digunakan untuk membangun UI di Jetpack Compose.

Pada baris ke [23] terdapat deklarasi fungsi DetailScreen dengan parameter titleResId, imageResId, yearResId, dan detailResId bertipe Int, yang digunakan untuk menampilkan detail game berdasarkan resource ID yang diberikan.

Pada baris ke [24] sampai [27] digunakan Surface sebagai wadah utama tampilan yang memenuhi ukuran layar penuh dan memiliki warna latar belakang Color(0xFF131418).

Pada baris ke [31] terdapat Column yang digunakan untuk menyusun komponen secara vertikal, dengan jarak padding sebesar 16dp di sekeliling konten.

Pada baris ke [37] sampai [45] digunakan Image untuk menampilkan gambar game berdasarkan imageResId. Gambar memenuhi lebar penuh (fillMaxWidth), tinggi 400dp, memiliki sudut membulat, dan menggunakan ContentScale.Crop agar gambar menyesuaikan ukuran.

Pada baris ke [47] sampai [76] digunakan Column dan Row untuk menampilkan judul dan tahun rilis game. Judul menggunakan ukuran font 26sp dan tebal dengan warna putih, sedangkan tahun menggunakan ukuran font 16sp dengan warna abu-abu keunguan.

Pada baris ke [64] berfungsi untuk menjadi Spacer untuk memberi jarak vertikal 8dp antara bagian judul dan deskripsi.

Pada baris ke [66] sampai [75] digunakan Text untuk menampilkan label "tentang game ini:" dan isi detail deskripsi dari game. Label ditampilkan dalam font 14sp tebal berwarna putih, sedangkan isi detail ditampilkan dalam font 14sp biasa juga berwarna putih, dengan padding atas sebesar 4dp.

HomeScreen.kt

Pada baris ke [1] terdapat kata kunci package com.pemrogamanmobile.myfavoritegames yang menyatakan nama paket tempat file ini berada dalam struktur proyek aplikasi Android.

Pada baris ke [3] hingga baris ke [5] terdapat baris import androidx.compose.foundation.layout.* import androidx.compose.foundation.lazy.LazyColumn dan import androidx.compose.foundation.lazy.items yang berfungsi mengambil komponen tata letak dan daftar malas dari Jetpack Compose agar dapat digunakan untuk membuat tampilan daftar vertikal dengan elemen-elemen yang dapat dipetakan dari koleksi data.

Pada baris ke [6] hingga baris ke [8] terdapat baris import androidx.compose.runtime.Composable import androidx.compose.runtime.collectAsState dan import androidx.compose.runtime.getValue yang berfungsi mengambil anotasi penanda

fungsi UI serta utilitas untuk mengubah aliran data state flow menjadi objek yang dapat diamati di dalam komposable.

Pada baris ke [9] hingga baris ke [11] terdapat baris import androidx.compose.ui.Modifier import androidx.compose.ui.unit.dp dan import com.pemrogamanmobile.myfavoritegames.model.Games yang berfungsi mengambil objek modifikasi tata letak, satuan ukuran density-independent pixels, dan kelas data Games sebagai model setiap entri game.

Pada baris ke [13] terdapat anotasi Composable yang menandai bahwa fungsi berikutnya merupakan blok pembentuk antarmuka pengguna di Jetpack Compose.

Pada baris ke [14] hingga baris ke [17] terdapat deklarasi fungsi HomeScreen yang menerima dua parameter yaitu gamesViewModel sebagai sumber data dari ViewModel dan onDetailClick sebagai fungsi panggilan balik yang akan dijalankan ketika pengguna memilih detail salah satu game.

Pada baris ke [18] terdapat inisialisasi val games by gamesViewModel.gamesState.collectAsState yang berfungsi mengambil data daftar game dari objek gamesState pada ViewModel lalu mengonversinya menjadi nilai yang dapat dimonitor oleh Compose.

Pada baris ke [20] hingga baris ke [23] terdapat pemanggilan LazyColumn dengan pengaturan verticalArrangement spacedBy 8 dp untuk memberi jarak antar item dan modifier fillMaxSize untuk membuat daftar tersebut mengisi seluruh ruang layar yang tersedia.

Pada baris ke [24] hingga baris ke [29] terdapat blok items yang memetakan setiap elemen dalam variabel games menjadi baris daftar, kemudian memanggil GameCard untuk setiap objek game dengan meneruskan data game dan fungsi onDetailClick sehingga setiap kartu game ditampilkan secara berurutan di dalam daftar.

MainActivity.kt:

Pada baris ke [1] terdapat kata kunci package com.pemrogamanmobile.myfavoritegames yang menyatakan nama paket tempat file ini disimpan dalam struktur proyek aplikasi Android.

Pada baris ke [3] hingga [17] terdapat rangkaian impor yang mengambil berbagai kelas dan fungsi dari Android SDK, Jetpack Compose, ViewModel Compose, Navigation Compose, sumber data Datasource, tema MyFavoriteGamesTheme, LaunchedEffect, serta library Timber untuk logging, sehingga seluruh utilitas tersebut dapat digunakan di dalam MainActivity.

Pada baris ke [19] terdapat deklarasi class MainActivity : ComponentActivity yang menunjukkan bahwa MainActivity adalah kelas utama aplikasi yang mewarisi kemampuan dari ComponentActivity untuk mendukung Jetpack Compose.

Pada baris ke [20] terdapat override fun onCreate(savedInstanceState Bundle?) yang menandai metode onCreate di mana aktivitas diinisialisasi ketika pertama kali dibuat.

Pada baris ke [21] terdapat panggilan super.onCreate(savedInstanceState) yang memastikan siklus hidup aktivitas dijalankan dengan benar sebelum inisialisasi lebih lanjut.

Pada baris ke [23] hingga [24] terdapat komentar dan Timber.d("MainActivity created") yang berfungsi mencatat log debug bahwa MainActivity telah berhasil dibuat.

Pada baris ke [26] hingga [28] terdapat panggilan setContent diikuti oleh MyFavoriteGamesTheme dan Surface color Color 0xFF131418 yang menetapkan konten UI menggunakan Jetpack Compose, membungkus seluruh UI dengan tema khusus, dan memberikan latar abu-abu gelap di permukaan utama.

Pada baris ke [29] terdapat inisialisasi val navController sama dengan rememberNavController yang membuat dan menyimpan objek NavController untuk mengelola navigasi antar layar Compose.

Pada baris ke [31] hingga [37] terdapat komentar dan inisialisasi val gamesViewModel sama dengan viewModel factory GamesViewModelFactory gameCategory sama dengan "All" datasource sama dengan Datasource() yang berarti GamesViewModel dibuat menggunakan GamesViewModelFactory dengan parameter kategori game "All" dan sumber data Datasource.

Pada baris ke [39] terdapat panggilan NavHost navController sama dengan navController startDestination "home" yang membuat host navigasi dengan navController dan menentukan layar awal bernama "home".

Pada baris ke [40] hingga [54] terdapat blok destinasi composable "home" yang mencatat log debug "Navigated to Home Screen", kemudian memanggil HomeScreen gamesViewModel sama dengan gamesViewModel onDetailClick sama dengan lambda game panah yang di dalamnya memanggil gamesViewModel.onGameClicked game untuk menyimpan game yang dipilih, mencatat log debug "Detail button clicked untuk game game", dan memerintahkan navController.navigate "detail" untuk berpindah ke layar detail ketika tombol Detail ditekan.

Pada baris ke [56] hingga [70] terdapat blok destinasi composable "detail" yang mencatat log debug "Navigated to Detail Screen", kemudian menginisialisasi val selectedGame by gamesViewModel.selectedGame.collectAsState untuk mengambil game yang telah dipilih dari ViewModel, lalu jika selectedGame tidak null mencatat log debug "Displaying details for game game" dan memanggil DetailScreen titleResId sama dengan game.titleResourceId imageResId sama dengan game.imageResourceId yearResId sama dengan game.yearResourceId detailResId sama dengan game.detailResourceId untuk menampilkan detail game.

Pada baris ke [71] hingga [77] terdapat blok ?: run yang berarti jika selectedGame null maka mencatat log debug "No selected game found navigating back to Home Screen" dan

menjalankan LaunchedEffect Unit yang memanggil navController.popBackStack untuk kembali ke layar Home.

GamesViewModel.kt

Pada baris ke [1] terdapat package com.pemrogamanmobile.myfavoritegames yang berfungsi sebagai deklarasi nama paket tempat file ini disimpan dalam struktur proyek aplikasi Android.

Pada baris ke [3] sampai [8] terdapat rangkaian import yang mengimpor berbagai kelas dan fungsi: androidx.lifecycle.ViewModel untuk mendefinisikan kelas ViewModel, kotlinx.coroutines.flow.MutableStateFlow dan kotlinx.coroutines.flow.StateFlow untuk mengelola aliran data dalam arsitektur reaktif, com.pemrogamanmobile.myfavoritegames.data.Datasource sebagai sumber data utama, com.pemrogamanmobile.myfavoritegames.model.Games sebagai model data objek game, dan timber.log.Timber untuk mencatat log debug.

Pada baris ke [10] terdapat deklarasi class GamesViewModel(yang menandai dimulainya kelas GamesViewModel, sebuah ViewModel khusus yang mengelola data game sesuai kategori dan berkomunikasi dengan layer UI.

Pada baris ke [11] dan [12] terdapat parameter private val gameCategory: String dan private val datasource: Datasource yang berfungsi menyimpan nilai kategori game yang diinginkan dan instance Datasource untuk mendapatkan daftar game dari sumber data.

Pada baris ke [13] terdapat) : ViewModel() yang menunjukkan bahwa GamesViewModel mewarisi kelas ViewModel agar dapat bertahan selama siklus hidup UI tanpa terpengaruh perubahan konfigurasi.

Pada baris ke [15] terdapat deklarasi private val _gamesState = MutableStateFlow(yang memulai pembuatan objek MutableStateFlow untuk menyimpan dan memancarkan daftar game yang telah difilter sesuai kategori.

Pada baris ke [16] terdapat datasource.loadGames().filter { it.category == gameCategory } yang menjalankan pemanggilan loadGames() pada datasource lalu memfilter hasilnya berdasarkan properti category sama dengan gameCategory, sehingga hanya game yang sesuai kategori yang disimpan dalam _gamesState.

Pada baris ke [18] terdapat val gamesState: StateFlow<List<Games>> get() = _gamesState yang mendefinisikan properti publik gamesState bertipe StateFlow agar UI dapat mengamati daftar game tanpa dapat memodifikasi langsung, sementara _gamesState tetap privat.

Pada baris ke [20] terdapat deklarasi private val _selectedGame = MutableStateFlow<Games?>(null) yang membuat objek MutableStateFlow untuk menyimpan game yang dipilih oleh pengguna, dengan nilai awal null menandakan belum ada game terpilih.

Pada baris ke [21] terdapat val selectedGame: StateFlow<Games?> get() = _selectedGame yang mendefinisikan properti publik selectedGame agar UI dapat mengamati game yang dipilih tetapi tidak dapat mengubahnya langsung.

Pada baris ke [24] terdapat komentar // Log saat data masuk ke list yang menjelaskan bahwa baris berikutnya berfungsi mencatat log saat data game berhasil dimuat.

Pada baris ke [25] terdapat Timber.d("Games data loaded: ${_gamesState.value}") yang berfungsi mencatat log debug berisi daftar game yang telah dimuat dan difilter sesuai kategori.

Pada baris ke [26] terdapat } yang menutup blok inisialisasi init.

Pada baris ke [28] terdapat deklarasi fungsi fun onGameClicked(game: Games) { yang mendefinisikan metode untuk menangani aksi ketika pengguna memilih salah satu game di UI.

Pada baris ke [29] terdapat _selectedGame.value = game yang berfungsi menetapkan nilai game yang dipilih ke dalam _selectedGame, sehingga selectedGame akan memancarkan nilai baru kepada observer.

Pada baris ke [30] terdapat komentar // Log data game yang dipilih yang menjelaskan bahwa baris berikutnya mencatat log terkait game yang baru saja dipilih.

Pada baris ke [31] terdapat Timber.d("Game selected: $game") yang berfungsi mencatat log debug dengan detail objek game yang dipilih..

strings.xml

Pada baris ke [1], terdapat elemen root <resources> yang digunakan untuk menampung semua sumber daya string dalam aplikasi Android.

Pada baris ke [2], terdapat elemen <string> dengan atribut name="app_name" yang menyimpan nama aplikasi, yaitu "My favorite games".

Pada baris ke [4] sampai [13], terdapat sepuluh elemen <string> dengan atribut name="titleX" yang masing-masing menyimpan judul permainan favorit berdasarkan nomor urut, seperti "Terraria", "Mobile Legends", dan lainnya.

Pada baris ke [15] sampai [24], terdapat sepuluh elemen <string> dengan atribut name="yearX" yang masing-masing menyimpan tahun rilis permainan berdasarkan nomor urut, seperti "2011" untuk year1 dan "2020" untuk year6.

Pada baris ke [26] sampai [35], terdapat sepuluh elemen <string> dengan atribut name="detailX" yang masing-masing menyimpan deskripsi detail permainan berdasarkan nomor urut. Contohnya, detail1 berisi deskripsi "Terraria", dan detail6 berisi deskripsi "Genshin Impact".

Pada baris ke [37] sampai [46], terdapat sepuluh elemen <string> dengan atribut name="linkX" yang masing-masing menyimpan URL terkait permainan berdasarkan nomor urut. Contohnya, link1 berisi URL ke halaman Steam "Terraria", dan link6 berisi URL ke halaman Google Play "Genshin Impact".

GamesViewModelFactory.kt

Pada baris ke [1] terdapat deklarasi package com.pemrogamanmobile.myfavoritegames yang berfungsi menyatakan nama paket tempat file ini berada dalam struktur proyek aplikasi Android.

Pada baris ke [3] terdapat import androidx.lifecycle.ViewModel yang berfungsi mengambil kelas ViewModel dari AndroidX Lifecycle, digunakan sebagai basis untuk ViewModel kustom.

Pada baris ke [4] terdapat import androidx.lifecycle.ViewModelProvider yang berfungsi mengambil antarmuka ViewModelProvider.Factory, digunakan untuk membuat instance ViewModel dengan parameter khusus.

Pada baris ke [5] terdapat import com.pemrogamanmobile.myfavoritegames.data.Datasource yang berfungsi mengambil kelas Datasource dari package data, yaitu sumber data untuk daftar game.

Pada baris ke [7] terdapat deklarasi class GamesViewModelFactory yang berfungsi mendefinisikan kelas pabrik untuk ViewModel, menerima parameter kategori game dan sumber data.

Pada baris ke [8] terdapat private val gameCategory: String yang berfungsi menyimpan nilai kategori game yang ingin dimuat ke dalam ViewModel.

Pada baris ke [9] terdapat private val datasource: Datasource = Datasource() yang berfungsi menyimpan instance Datasource dengan nilai default objek baru Datasource.

Pada baris ke [10] terdapat pewarisan dari ViewModelProvider.Factory yang menandakan bahwa kelas ini menyediakan mekanisme pembuatan ViewModel kustom.

Pada baris ke [11] terdapat deklarasi override fun <T : ViewModel> create(modelClass: Class<T>): T yang berfungsi menggantikan metode pembuatan ViewModel, menerima parameter kelas ViewModel yang diminta.

Pada baris ke [12] terdapat kondisi if modelClass.isAssignableFrom(GamesViewModel::class.java) yang berfungsi memeriksa apakah kelas yang diminta dapat dipetakan ke GamesViewModel.

Pada baris ke [13] terdapat anotasi @Suppress("UNCHECKED_CAST") yang berfungsi menonaktifkan peringatan konversi tipe yang tidak aman saat melakukan casting.

Pada baris ke [14] terdapat return GamesViewModel(gameCategory, datasource) as T yang berfungsi membuat instance GamesViewModel dengan parameter kategori dan sumber data, kemudian mengonversinya ke tipe umum T.

Pada baris ke [16] terdapat throw IllegalArgumentException("Unknown ViewModel class") yang berfungsi melempar pengecualian jika kelas ViewModel yang diminta tidak dikenali oleh pabrik.

MyFavoriteGamesApplication.kt

Pada baris ke [1] terdapat deklarasi package com.pemrogamanmobile.myfavoritegames yang menetapkan namespace tempat file ini berada dalam struktur proyek aplikasi Android.

Pada baris ke [3] terdapat perintah import android.app.Application yang berfungsi mengambil kelas Application dari framework Android sebagai kelas induk untuk inisialisasi aplikasi.

Pada baris ke [4] terdapat perintah import timber.log.Timber yang berfungsi mengimpor pustaka Timber untuk logging yang lebih fleksibel dan terstruktur.

Pada baris ke [6] terdapat deklarasi kelas MyFavoriteGamesApplication turunan dari Application yang memungkinkan kita mengoverride perilaku inisialisasi global aplikasi.

Pada baris ke [7] terdapat deklarasi metode override onCreate yang akan dipanggil pertama kali saat instance aplikasi dibuat.

Pada baris ke [8] terdapat pemanggilan super onCreate yang memastikan logika inisialisasi bawaan framework Android tetap dijalankan.

Pada baris ke [9] terdapat blok pengkondisian if BuildConfig DEBUG yang memeriksa apakah aplikasi dalam mode debug agar hanya menjalankan logging lanjutan saat pengembangan.

Pada baris ke [10] terdapat perintah Timber plant DebugTree yang memasang tree bawaan Timber untuk mengeluarkan log ke Logcat selama mode debug.

Penjelasan Debugger :

1. Fungsi Debugger

- Memeriksa Alur Program

Debugger memiliki kemampuan untuk menjalankan program secara terkontrol baris per baris, agar dapat melihat secara detail bagaimana alur eksekusi dapat berjalan. Dengan tujuan suapaya mudah memahami bagian mana yang dieksekusi terlebih dahulu, bagaimana nilai variabel berubah, dan kondisi cabang (if/else) mana yang terpenuhi.

- Menemukan dan Memperbaiki Bug (Errors)

Alih-alih membaca seluruh kode secara manual untuk mencari kesalahannya, debugger dapat membantu untuk menemukan titik (line) di mana program tidak berjalan sesuai harapan. Kita dapat menghentikan eksekusi tepat sebelum munculnya error (misalnya NullPointerException, IndexOutOfBounds, dan sejenisnya) lalu memeriksa tumpukan panggilan (call stack) dan nilai variabel.

- Memantau Nilai Variabel Secara Real-time

Ketika program berhenti di breakpoint, maka isi variabel, baik variabel sederhana (integer, string, dll.) maupun objek kompleks, dapat terlihat., dan memiliki tujuan untuk memastikan apakah nilainya sesuai ekspektasi.

2. Cara Menggunakan Debugger

Untuk menggunakan Debugger maka dapat dilihat sebagai berikut:

1. Memasang Breakpoint
2. Buka file kode sumber yang ingin di periksa.
3. Klik di margin kiri (atau tekan shortcut) pada baris kode di mana ingin di eksekusi program berhenti (contoh: line 42). Breakpoint biasanya ditandai dengan titik merah.
4. Menjalankan Program dalam Mode Debug
5. Selanjutnya, pilih "Debug" dan jalankan.
6. Program akan berjalan hingga menemui breakpoint pertama, lalu menghentikan eksekusi secara otomatis.
7. Periksa Jendela Debugger, setelah eksekusi terhenti, maka terlihat beberapa panel penting yang bisa dibaca seperti variabel.

3. Fitur Step Into, Step Over, dan Step Out

Berikut adalah beberapa dari fitur step:

1. Step Into

Step into dijalankan saat eksekusi berhenti di suatu baris yang memanggil fungsi/metode. Jika pada baris tersebut ada pemanggilan fungsi, maka "Step Into" akan masuk ke baris pertama fungsi tersebut, atau bisa dibilang menelusuri ke dalam implementasi fungsi.

2. Step Over

Step over dijalankan saat eksekusi berhenti di baris yang memanggil fungsi, "Step Over" akan mengeksekusi keseluruhan fungsi tersebut secara instan, lalu berhenti di baris kode berikutnya setelah fungsi selesai dikembalikan.

3. Step Out

Step out digunakan Ketika berada di dalam suatu fungsi (setelah beberapa kali Step Into), "Step Out" akan menjalankan sisa fungsi tersebut hingga kembali ke pemanggilnya, lalu menghentikan eksekusi pada baris di pemanggil.

SOAL 2

Soal Praktikum:

2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

A. Pembahasan

Application class dalam arsitektur Android adalah objek global yang diinisialisasi sekali saat proses aplikasi dijalankan dan bertindak sebagai "entry point" bagi semua komponen seperti Activity, Service, Receiver, dsb. Fungsi utamanya sendiri, meliputi inisialisasi library dan konfigurasi global (misalnya dependency injection, analytics, atau database), menyediakan ApplicationContext yang dapat diakses di seluruh aplikasi, menyimpan variabel atau singleton yang dibutuhkan lintas-komponen, serta menangani event lifecycle tingkat aplikasi seperti onLowMemory() atau onTrimMemory() untuk manajemen cache dan sumber daya.

Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/Harry154git/Pemrogaman_mobile/tree/main/Modul_4