LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

VIEWMODEL AND DEBUGGING

Oleh:

Alysa Armelia **NIM. 2310817120009**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 4

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

Nama Praktikan : Alysa Armelia
NIM : 2310817120009

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

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

Soal Praktikum:

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

A. Source Code

1. MainActivity.kt

```
1   package com.example.londondestination
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5   import
    com.example.londondestination.databinding.ActivityMainB
    inding
6
7   class MainActivity : AppCompatActivity() {
8       private lateinit var binding: ActivityMainBinding
9       override fun onCreate(savedInstanceState: Bundle?)
    {
10
11          super.onCreate(savedInstanceState)
```

```
12          binding =
13  ActivityMainBinding.inflate(layoutInflater)
            setContentView(binding.root)
14      }
15  }
```

Tabel 1. 1 Source Code MainActivity

2. FragmentGuweh.kt

```
1   package com.example.londondestination
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
    com.example.londondestination.databinding.DetailFragmen
    tBinding
9
10
11  class FragmentGuweh : Fragment() {
12
13      private var _binding: DetailFragmentBinding? = null
14      private val binding get() = _binding!!
15
16      override fun onCreateView(
17          inflater: LayoutInflater, container:
    ViewGroup?,
18          savedInstanceState: Bundle?
19      ): View {
20          _binding =
    DetailFragmentBinding.inflate(inflater, container,
    false)
21          return binding.root
22      }
23
24      override fun onViewCreated(view: View,
    savedInstanceState: Bundle?) {
25          super.onViewCreated(view, savedInstanceState)
26
27          val imageResId =
    arguments?.getInt("imageResId") ?:
    R.drawable.ic_launcher_background
28          val nama = arguments?.getString("nama") ?:
    "Nama tidak tersedia"
29          val deskripsi =
```

```
    arguments?.getString("deskripsi") ?: "Deskripsi tidak
    tersedia"
30
31
32  binding.detailImage.setImageResource(imageResId)
33          binding.detailTitle.text = nama
34          binding.detailDescription.text = deskripsi
35      }
36
37      override fun onDestroyView() {
38          super.onDestroyView()
39          _binding = null
40      }
41      companion object {
            fun newInstance(imageResId: Int, nama: String,
    deskripsi: String): FragmentGuweh {
42              val fragment = FragmentGuweh()
43              val args = Bundle()
44              args.putInt("imageResId", imageResId)
45              args.putString("nama", nama)
46              args.putString("deskripsi", deskripsi)
47              fragment.arguments = args
48              return fragment
49          }
50      }
51  }
```

Tabel 1. 2 Source Code FragmentGuweh

3. HomeFragment.kt

```
1   package com.example.londondestination
2
3   import android.os.Bundle
4   import android.util.Log
5   import android.view.LayoutInflater
6   import android.view.View
7   import android.view.ViewGroup
8   import androidx.fragment.app.Fragment
9   import androidx.fragment.app.viewModels
10  import androidx.lifecycle.lifecycleScope
11  import androidx.navigation.fragment.findNavController
12  import androidx.core.os.bundleOf
13  import androidx.recyclerview.widget.LinearLayoutManager
14  import
    com.example.londondestination.databinding.HomeFragmentB
    inding
15  import kotlinx.coroutines.flow.collectLatest
```

```
16 import kotlinx.coroutines.launch
17
18 class HomeFragment : Fragment() {
19
20     private var _binding: HomeFragmentBinding? = null
21     private val binding get() = _binding!!
22
23     private lateinit var adapter: MyAdapter
24
25     private val viewModel: HomeViewModel by viewModels
   {
26         HomeViewModelFactory()
27     }
28
29     override fun onCreateView(inflater: LayoutInflater,
   container: ViewGroup?, savedInstanceState: Bundle?):
   View {
30         _binding =
   HomeFragmentBinding.inflate(inflater, container, false)
31         return binding.root
32     }
33
34     override fun onViewCreated(view: View,
   savedInstanceState: Bundle?) {
35         super.onViewCreated(view, savedInstanceState)
36
37         adapter = MyAdapter(emptyList()) { selectedItem
   ->
38             viewModel.onItemClicked(selectedItem)
39         }
40
41         binding.rvCharacter.layoutManager =
   LinearLayoutManager(requireContext())
42         binding.rvCharacter.adapter = adapter
43
44         viewLifecycleOwner.lifecycleScope.launch {
45             viewModel.destinationList.collectLatest {
   list ->
46                 adapter = MyAdapter(list) {
   selectedItem ->
47
48 viewModel.onItemClicked(selectedItem)
49                 }
50                 binding.rvCharacter.adapter = adapter
51             }
52         }
53
54         viewLifecycleOwner.lifecycleScope.launch {
```

```
55              viewModel.selectedItem.collectLatest { item ->
56                  item?.let {
57                      Log.d("HomeFragment", "Navigasi ke DetailFragment untuk: ${it.nama}, Tahun: ${it.year}, Deskripsi: ${it.description}")
58
59                      val bundle = bundleOf(
60                          "imageResId" to it.image,
61                          "nama" to it.nama,
62                          "deskripsi" to it.description
63                      ) findNavController().navigate(R.id.action_HomeFragment_to_detailFragment, bundle)
64                      viewModel.resetSelectedItem()
65                  }
66              }
67          }
68
69      }
70
71      override fun onDestroyView() {
72          super.onDestroyView()
73          _binding = null
74      }
75  }
```

Tabel 1. 3 Source Code HomeFragment

4. HomeViewModel.kt

```
1   package com.example.londondestination
2
3   import android.util.Log
4   import androidx.lifecycle.ViewModel
5   import kotlinx.coroutines.flow.MutableStateFlow
6   import kotlinx.coroutines.flow.StateFlow
7   import kotlinx.coroutines.flow.asStateFlow
8
9   class HomeViewModel : ViewModel() {
10
11      private val _destinationList = MutableStateFlow<List<MyData>>(emptyList())
12      val destinationList: StateFlow<List<MyData>> = _destinationList.asStateFlow()
13
14      private val _selectedItem = MutableStateFlow<MyData?>(null)
```

```
15      val selectedItem: StateFlow<MyData?> =
        _selectedItem.asStateFlow()
16
17      init {
18          val data = listOf(
19              MyData(
20                  nama = "Natural History Museum",
21                  description = "Natural History Museum
        di London adalah destinasi wajib bagi pecinta sains,
        sejarah alam, dan keluarga dengan anak-anak. Museum
        ini memiliki lebih dari 80 juta spesimen, mencakup
        zoologi, paleontologi, botani, dan geologi. Daya tarik
        utama termasuk rangka dinosaurus, seperti Diplodocus
        dan model animatronik Tyrannosaurus Rex. Selain itu,
        pengunjung bisa melihat koleksi batuan langka,
        meteorit, serta kristal dan permata.\n" +
22                          "\n" + "Di Earth Galleries,
        ada pameran interaktif tentang geologi bumi, gunung
        berapi, dan gempa bumi. Museum ini juga memiliki zona
        edukasi yang menarik untuk anak-anak. Dibangun dengan
        arsitektur Romanesque bergaya Victoria, bangunannya
        memukau dengan interior yang megah. Yang menarik,
        museum ini gratis untuk dikunjungi, memberikan
        pengalaman yang tak terlupakan bagi semua
        pengunjung.",
23                  descriptionsingkat = "Museum sejarah
        tentang alam",
24                  year = 1881,
25                  image =
        R.drawable.naturalhistorymuseum,
26                  link =
        "https://en.wikipedia.org/wiki/Natural_History_Museum,
        _London"
27              ),
28              MyData(
29                  nama = "London Eye",
30                  description = "London Eye adalah
        kincir raksasa setinggi 135 meter di tepi Sungai
        Thames, dan menjadi salah satu ikon paling terkenal di
        London. Dari dalam kapsul kacanya, pengunjung bisa
        menikmati panorama kota yang menakjubkan, termasuk
        pemandangan Big Ben, Gedung Parlemen, dan Sungai
        Thames. Pada hari cerah, jarak pandang bisa mencapai
        hampir 40 kilometer.\n" +
31                          "\n" + "Waktu terbaik untuk
        naik adalah saat senja, ketika cahaya kota mulai
        menyala dan langit berwarna keemasan karena
        menciptakan suasana romantis dan tenang. Setiap
```

```
      putaran berlangsung sekitar 30 menit, memberikan cukup
      waktu untuk mengagumi pemandangan, mengambil foto,
      atau sekadar menikmati momen dari ketinggian. London
      Eye bukan hanya atraksi wisata, tetapi juga simbol
      kebanggaan kota yang terus memikat baik turis maupun
      warga lokal.",
32                  descriptionsingkat = "Komedi putar
      raksasa London",
33                  year = 2000,
34                  image = R.drawable.londoneye,
35                  link =
      "https://en.wikipedia.org/wiki/London_Eye"
36              ),
37              MyData(
38                  nama = "British Museum",
39                  description = "British Museum adalah
      destinasi luar biasa bagi pecinta sejarah dan
      arkeologi, dengan koleksi global yang mencakup ribuan
      tahun peradaban manusia. Begitu masuk, suasana elegan
      langsung membawa pengunjung seakan menjelajahi masa
      lalu dari mumi Mesir, patung Yunani, artefak
      Mesopotamia, hingga keramik Tiongkok dan seni Islam.
      Batu Rosetta adalah salah satu sorotan utama, bersama
      Patung Ramses II dan reruntuhan Kuil Parthenon yang
      ikonik.\n" +
40                              "\n" + "Setiap koleksi
      disertai penjelasan informatif yang memudahkan
      pengunjung memahami konteks sejarahnya. Great Court,
      dengan atap kaca yang terang dan desain modern,
      menjadi pusat bangunan yang menawan dan nyaman untuk
      bersantai. British Museum bukan sekadar tempat melihat
      artefak, tapi juga ruang kontemplatif yang
      menghubungkan kita dengan jejak panjang umat manusia
      dan semuanya bisa dinikmati tanpa biaya masuk.",
41                  descriptionsingkat = "Museum koleksi
      dunia",
42                  year = 1753,
43                  image = R.drawable.british_museum,
44                  link =
      "https://en.wikipedia.org/wiki/British_Museum"
45              ),
46              MyData(
47                  nama = "Big Ben",
48                  description = "Big Ben adalah ikon tak
      tergantikan London, berdiri megah di samping Gedung
      Parlemen di tepi Sungai Thames. Meski banyak mengira
      namanya merujuk pada menaranya, Big Ben sebenarnya
      adalah lonceng besar seberat lebih dari 13 ton di
```

```
        dalam Elizabeth Tower adalah nama resmi menara
        tersebut, yang diberikan untuk menghormati Ratu
        Elizabeth II.\n" +
49                          "\n" + "Dentang Big Ben punya
        makna emosional yang dalam, sering terdengar dalam
        momen penting seperti pergantian tahun atau peringatan
        nasional, dan bahkan disiarkan BBC sejak 1920-an.
        Arsitekturnya yang anggun menjadi latar favorit para
        wisatawan, baik saat disinari mentari pagi maupun
        diterangi lampu malam hari. Walau akses ke dalam
        menara terbatas, cukup berdiri di dekatnya sudah
        membuat pengunjung merasa terhubung dengan sejarah dan
        semangat kota London yang tak lekang waktu",
50                  descriptionsingkat = "Jam besar
        London",
51                  year = 1859,
52                  image = R.drawable.bigben,
53                  link =
        "https://en.wikipedia.org/wiki/Big_Ben"
54              ),
55              MyData(
56                  nama = "Buckingham Palace",
57                  description = "Buckingham Palace
        adalah simbol monarki Inggris yang berdiri megah di
        pusat London, berfungsi sebagai kediaman resmi Raja
        dan pusat berbagai acara kenegaraan. Dengan lebih dari
        700 ruangan, istana ini mencerminkan kemewahan dan
        sejarah yang hidup, bahkan dari luar pagar hitamnya
        yang ikonik.\n" +
58                          "\n" + "Salah satu atraksi
        utama adalah Upacara Pergantian Penjaga, prosesi
        tradisional dengan seragam merah dan musik marching
        band yang menarik ribuan wisatawan setiap harinya.
        Jika bendera kerajaan berkibar di atas istana, itu
        menandakan Raja sedang berada di dalam momen sederhana
        yang membuat banyak orang merasa lebih dekat dengan
        sejarah kerajaan.\n" +
59                          "\n" + "Pada musim panas,
        beberapa ruang kenegaraan dibuka untuk umum,
        menampilkan interior menawan lengkap dengan kristal,
        lukisan klasik, dan kemegahan khas kerajaan. Meski
        banyak pengunjung hanya melihat dari luar, pesona dan
        wibawa istana ini menjadikannya salah satu destinasi
        paling ikonik di London.",
60                  descriptionsingkat = "Istana resmi
        Kerajaan Inggris",
61                  year = 1703,
62                  image = R.drawable.buckinghampalace,
```

```
63                  link =
    "https://en.wikipedia.org/wiki/Buckingham_Palace"
64              ),
65              MyData(
66                  nama = "Tower of London",
67                  description = "Tower of London adalah
    benteng bersejarah di tepi Sungai Thames yang
    menyimpan kisah dramatis tentang kekuasaan,
    pengkhianatan, dan warisan kerajaan Inggris. Dulu
    berfungsi sebagai penjara bagi bangsawan, termasuk
    Anne Boleyn yang dieksekusi di sana, tempat ini
    memancarkan nuansa mencekam sekaligus megah, terutama
    di lokasi-lokasi penting seperti halaman eksekusi.\n"
    +
68                          "\n" + "Namun, sisi gelap itu
    berpadu dengan kemewahan karena di sinilah Permata
    Mahkota Inggris disimpan, termasuk mahkota dan tongkat
    kerajaan yang berkilau menakjubkan. Kontras antara
    sejarah kelam dan simbol kejayaan membuat kunjungan ke
    Tower of London terasa seperti menyusuri lorong waktu,
    menghadirkan pengalaman mendalam yang tak
    terlupakan.",
69                  descriptionsingkat = "Benteng
    mempunyai banyak sejarah",
70                  year = 1066,
71                  image = R.drawable.toweroflondon,
72                  link =
    "https://en.wikipedia.org/wiki/Tower_of_London"
73              ),
74              MyData(
75                  nama = "Warner Bros. Studio Tour
    London",
76                  description = "Warner Bros. Studio
    Tour London adalah destinasi impian bagi penggemar
    Harry Potter, menawarkan pengalaman imersif ke dunia
    sihir yang sebelumnya hanya bisa dilihat di layar.
    Begitu masuk, kamu akan dibawa ke set asli seperti
    Great Hall, Diagon Alley, dan Privet Drive dimana
    semuanya penuh detail yang membuatmu merasa benar-
    benar berada di dunia Hogwarts.\n" +
77                          "\n" + "Selain menjelajahi
    lokasi ikonik, pengunjung juga bisa melihat properti
    film seperti Horcrux, kostum rumah-rumah Hogwarts,
    hingga proses pembuatan efek visual yang menghidupkan
    sihir di layar. Setiap sudut studio menyuguhkan
    keajaiban yang membuat kamu makin menghargai imajinasi
    dan kerja keras di balik film. Dan tentu saja,
    mencicipi Butterbeer jadi penutup manis dari kunjungan
```

```
        yang terasa seperti pulang ke dunia masa kecil yang
        penuh keajaiban.",
78                  descriptionsingkat = "Studio Harry
        Potter London",
79                  year = 2012,
80                  image = R.drawable.harrypotterstudio,
81                  link =
        "https://www.wbstudiotour.co.uk/"
82              ),
83              MyData(
84                  nama = "Hyde Park",
85                  description = "Hyde Park adalah oase
        hijau di tengah London yang menawarkan ketenangan dan
        ruang bebas bagi siapa saja, dari warga lokal hingga
        turis. Dengan luas lebih dari 140 hektar, taman ini
        menjadi tempat ideal untuk jogging, bersepeda, membaca
        buku, atau sekadar duduk santai di tepi danau
        Serpentine.\n" +
86                              "\n" + "Suasananya santai dan
        cocok buat piknik, bermain, atau menikmati kopi di
        bawah rindangnya pepohonan. Salah satu sudut paling
        unik adalah Speaker’s Corner, simbol kebebasan
        berpendapat di mana siapa pun bisa berbicara di depan
        umum. Selain sebagai tempat pelarian dari hiruk pikuk
        kota, Hyde Park juga kerap menjadi lokasi konser besar
        dan festival, menjadikannya ruang publik yang dinamis
        dan menyatu dengan jiwa kota London.",
87                  descriptionsingkat = "Taman pusat
        kota",
88                  year = 1637,
89                  image = R.drawable.hydepark,
90                  link =
        "https://en.wikipedia.org/wiki/Hyde_Park,_London"
91              ),
92              MyData(
93                  nama = "The Sherlock Holmes Museum",
94                  description = "The Sherlock Holmes
        Museum di 221B Baker Street adalah surga bagi
        penggemar detektif legendaris ini, membawa pengunjung
        langsung ke dunia fiksi era Victoria yang terasa
        hidup. Interiornya merekonstruksi ruang kerja dan
        rumah Holmes secara detail—lengkap dengan perapian,
        kaca pembesar, dan barang-barang pribadi khas karakter
        dalam cerita.\n" +
95                              "\n" + "Setiap sudut museum
        dirancang agar kamu seolah benar-benar berada di
        tengah kisah misteri bersama Holmes dan Watson. Lebih
        dari sekadar pameran, museum ini menawarkan pengalaman
```

```
        yang memuaskan rasa ingin tahu para penggemar dan
        pencinta cerita klasik.",
96                  descriptionsingkat = "Museum seorang
        detektif yang ikonik",
97                  year = 1990,
98                  image =
        R.drawable.thesherlockholmesmuseum,
99                  link = "https://www.sherlock-
        holmes.co.uk/"
100             ),
101             MyData(
102                 nama = "St Paul's Cathedral",
103                 description = "St Paul’s Cathedral
        adalah salah satu ikon arsitektur paling menakjubkan
        di London, dikenal dengan kubah raksasanya yang
        mendominasi cakrawala kota. Dari luar terlihat megah,
        tapi keindahan sejatinya baru benar-benar terasa saat
        kamu melangkah masuk—ruang dalamnya hening, agung, dan
        sarat nuansa spiritual. Langit-langit tinggi, kaca
        patri indah, dan cahaya alami menciptakan atmosfer
        yang membuat siapa pun terdiam dalam kekaguman.\n" +
104                             "\n" + "Salah satu pengalaman
        paling tak terlupakan di sini adalah menaiki ratusan
        anak tangga menuju puncak kubah. Dari atas, kamu bisa
        menikmati panorama kota London yang luas dan penuh
        sejarah, dari Sungai Thames hingga gedung-gedung
        modern yang berdiri berdampingan dengan bangunan
        bersejarah. Tak heran, St Paul’s sering menjadi lokasi
        berbagai momen penting nasional, karena tempat ini
        bukan hanya gereja, tapi simbol kekuatan dan keindahan
        yang hidup modern di kota.",
105                 descriptionsingkat = "Katedral
        bersejarah di London",
106                 year = 1710,
107                 image = R.drawable.stpaulcathedral,
108                 link =
        "https://en.wikipedia.org/wiki/St_Paul%27s_Cathedral"
109             ),
110             MyData(
111                 nama = "Camden Market",
112                 description = "Camden Market adalah
        salah satu tempat paling unik dan penuh karakter di
        London, ideal bagi kamu yang suka berburu barang-
        barang tak biasa dan merasakan suasana kota yang
        dinamis. Pasar ini dipenuhi toko-toko kecil yang
        menjual pakaian vintage, aksesori handmade, dan
        berbagai barang nyentrik yang sering kali hanya ada
        satu di dunia. Suasananya ramai tapi seru, dengan
```

```
        musik jalanan, aroma makanan internasional, dan gaya
        busana pengunjung yang beragam.\n" +
113                             "\n" + "Selain jadi pusat
        belanja, Camden juga merupakan ruang ekspresi
        subkultur alternatif seperti punk, goth, dan hippie—
        tempat di mana semua orang bisa tampil sesuai dirinya
        sendiri. Kalau lapar, pilihan kulinernya luar biasa
        banyak dan menggoda, dari makanan Asia, Latin, Timur
        Tengah, sampai fusion kreatif. Setiap kunjungan ke
        Camden terasa seperti eksplorasi baru bukan sekadar
        belanja, tapi pengalaman hidup kota London yang bebas,
        kreatif, dan penuh kejutan.",
114                     descriptionsingkat = "Pasar terbesar
        di London",
115                     year = 1974,
116                     image = R.drawable.camdenmarket,
117                     link =
        "https://en.wikipedia.org/wiki/Camden_Market"
118                 )
119
120             )
121             _destinationList.value = data
122             Log.d("HomeViewModel", "List data berhasil
        dimuat sebanyak ${data.size} item")
123         }
124
125         fun onItemClicked(item: MyData) {
126             _selectedItem.value = item
127         }
128
129         fun resetSelectedItem() {
130             _selectedItem.value = null
131         }
132 }
```

Tabel 1. 4 Source Code HomeViewModel

5. HomeViewModelFactory.kt

```
1   package com.example.londondestination
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5
6   class HomeViewModelFactory : ViewModelProvider.Factory
    {
7       override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
```

```
8           if
    (modelClass.isAssignableFrom(HomeViewModel::class.java)
    ) {
9               return HomeViewModel() as T
10          }
11          throw IllegalArgumentException("Unknown
    ViewModel class")
12      }
13  }
```

Tabel 1. 5 Source Code HomeViewModelFactory

6. MyAdapter.kt

```
1   package com.example.londondestination
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.util.Log
6   import android.view.LayoutInflater
7   import android.view.ViewGroup
8   import androidx.recyclerview.widget.RecyclerView
9   import
    com.example.londondestination.databinding.ItemListBindi
    ng
10
11  class MyAdapter(
12      private val destinations: List<MyData>,
13      private val onDetailClick: (MyData) -> Unit
14  ) : RecyclerView.Adapter<MyAdapter.ViewHolder>() {
15
16      inner class ViewHolder(val binding:
    ItemListBinding) :
17          RecyclerView.ViewHolder(binding.root)
18
19      override fun onCreateViewHolder(parent: ViewGroup,
    viewType: Int): ViewHolder {
20          val binding = ItemListBinding.inflate(
21              LayoutInflater.from(parent.context),
22              parent,
23              false
24          )
25          return ViewHolder(binding)
26      }
27
28      override fun onBindViewHolder(holder: ViewHolder,
    position: Int) {
29          val destination = destinations[position]
```

```
30          with(holder.binding) {
31              textViewName.text = destination.nama
32              textViewYear.text =
    destination.year.toString()
33              textViewDesc.text =
    destination.descriptionsingkat
34  imageView.setImageResource(destination.image)
35
36              buttonLink.setOnClickListener {
37                  val context = it.context
38                  val intent = Intent(Intent.ACTION_VIEW,
    Uri.parse(destination.link))
39                  context.startActivity(intent)
40                  Log.d("MyAdapter", "Tombol Link ditekan
    untuk: ${destination.nama}")
41              }
42
43              buttonDetail.setOnClickListener {
44                  onDetailClick(destination)
45                  Log.d("MyAdapter", "Tombol Detail
    ditekan untuk: ${destination.nama}")
46              }
47          }
48      }
49
50      override fun getItemCount(): Int =
    destinations.size
51  }
```

Tabel 1. 6 Source Code MyAdapter

7. MyData.kt

```
1   package com.example.londondestination
2
3   import android.os.Parcelable
4   import kotlinx.parcelize.Parcelize
5
6   @Parcelize
7   data class MyData(
8       val nama: String,
9       val description: String,
10      val descriptionsingkat: String,
11      val year: Int,
12      val image:Int,
13      val link: String
14  ): Parcelable
```

Tabel 1. 7 Source Code MyData

8. Detail_fragment.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ScrollView
   xmlns:android="http://schemas.android.com/apk/res/androi
   d"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      android:id="@+id/detailScrollView"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:padding="16dp">
8
9      <androidx.cardview.widget.CardView
10         android:layout_width="match_parent"
11         android:layout_height="wrap_content"
12         app:cardCornerRadius="16dp"
13         app:cardElevation="8dp">
14
15         <LinearLayout
16             android:layout_width="match_parent"
17             android:layout_height="wrap_content"
18             android:orientation="vertical">
19
20             <ImageView
21                 android:id="@+id/detailImage"
22                 android:layout_width="match_parent"
23                 android:layout_height="200dp"
24                 android:scaleType="centerCrop"/>
25
26             <TextView
27                 android:id="@+id/detailTitle"
28                 android:layout_width="match_parent"
29                 android:layout_height="wrap_content"
30                 android:text="Nama Tempat"
31                 android:textStyle="bold"
32                 android:textSize="20sp"
33                 android:padding="16dp" />
34
35             <TextView
36                 android:id="@+id/detailDescription"
37                 android:layout_width="match_parent"
38                 android:layout_height="wrap_content"
39                 android:text="Deskripsi lengkap tempat
   destinasi."
40                 android:paddingStart="16dp"
41                 android:paddingEnd="16dp"
42                 android:paddingBottom="16dp"
43                 android:textSize="16sp" />
```

```
44
45          </LinearLayout>
46      </androidx.cardview.widget.CardView>
47  </ScrollView>
```

Tabel 1. 8 Source Code detail_fragment

9. activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:id="@+id/main_layout"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       tools:context=".MainActivity">
9
10
11      <androidx.fragment.app.FragmentContainerView
12          android:id="@+id/fragment_container_view"
13  android:name="androidx.navigation.fragment.NavHostFragme
    nt"
14          android:layout_width="match_parent"
15          android:layout_height="match_parent"
16          app:navGraph="@navigation/nav_graph"
17          app:defaultNavHost="true" />
18  </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 1. 9 Source Code activity_main

10. home_fragment.xml

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

```
14          app:layout_constraintEnd_toEndOf="parent"
15          app:layout_constraintStart_toStartOf="parent"
16          app:layout_constraintTop_toTopOf="parent" />
17  </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 1. 10 Source Code home_fragment

11. item_list.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="wrap_content"
7       android:layout_margin="12dp"
8       app:cardCornerRadius="16dp"
9       app:cardElevation="6dp">
10
11      <androidx.constraintlayout.widget.ConstraintLayout
12          android:layout_width="match_parent"
13          android:layout_height="wrap_content"
14          android:padding="12dp">
15
16          <ImageView
17              android:id="@+id/imageView"
18              android:layout_width="100dp"
19              android:layout_height="150dp"
20              android:scaleType="centerCrop"
21              app:layout_constraintTop_toTopOf="parent"
22             app:layout_constraintStart_toStartOf="parent"
    />
23
24          <TextView
25              android:id="@+id/textViewName"
26              android:layout_width="0dp"
27              android:layout_height="wrap_content"
28              android:text="Main Title"
29              android:textStyle="bold"
30              android:textSize="18sp"
31              android:textColor="#000000"
32              android:layout_marginStart="12dp"
33              android:layout_marginTop="8dp"
34        app:layout_constraintStart_toEndOf="@id/imageView"
35              app:layout_constraintEnd_toEndOf="parent"
36              tools:ignore="MissingConstraints" />
```

```
37
38          <TextView
39              android:id="@+id/textViewYear"
40              android:layout_width="0dp"
41              android:layout_height="wrap_content"
42              android:text="1990"
43              android:textSize="16sp"
44              android:textColor="#555555"
45              android:layout_marginStart="12dp"
46              android:layout_marginTop="4dp"
47        app:layout_constraintStart_toEndOf="@id/imageView"
48  app:layout_constraintTop_toBottomOf="@id/textViewName"
49              app:layout_constraintEnd_toEndOf="parent" />
50
51          <TextView
52              android:id="@+id/textViewDesc"
53              android:layout_width="0dp"
54              android:layout_height="wrap_content"
55              android:text="Secondary description text
    that can span multiple lines."
56              android:textSize="15sp"
57              android:textColor="#555555"
58              android:layout_marginStart="12dp"
59              android:layout_marginTop="8dp"
60  app:layout_constraintStart_toEndOf="@id/imageView"
61  app:layout_constraintTop_toBottomOf="@id/textViewYear"
62              app:layout_constraintEnd_toEndOf="parent" />
63
64          <LinearLayout
65              android:id="@+id/buttonRow"
66              android:layout_width="wrap_content"
67              android:layout_height="wrap_content"
68              android:orientation="horizontal"
69              android:gravity="center"
70              android:layout_marginTop="16dp"
71  app:layout_constraintTop_toBottomOf="@id/imageView"
72           app:layout_constraintBottom_toBottomOf="parent"
73             app:layout_constraintStart_toStartOf="parent"
74              app:layout_constraintEnd_toEndOf="parent">
75
76              <Button
77                  android:id="@+id/buttonLink"
78                  android:layout_width="wrap_content"
79                  android:layout_height="wrap_content"
80                  android:layout_marginEnd="8dp"
81                android:backgroundTint="@color/purple_200"
82                  android:text="Detail"
83                  android:textAllCaps="false" />
```

```
84
85              <Button
86                  android:id="@+id/buttonDetail"
87                  android:layout_width="wrap_content"
88                  android:layout_height="wrap_content"
89                  android:layout_marginStart="8dp"
90                android:backgroundTint="@color/purple_200"
91                  android:text="Info"
92                  android:textAllCaps="false" />
93          </LinearLayout>
94
95      </androidx.constraintlayout.widget.ConstraintLayout>
96  </androidx.cardview.widget.CardView>
```

Tabel 1. 11 Source Code item_list

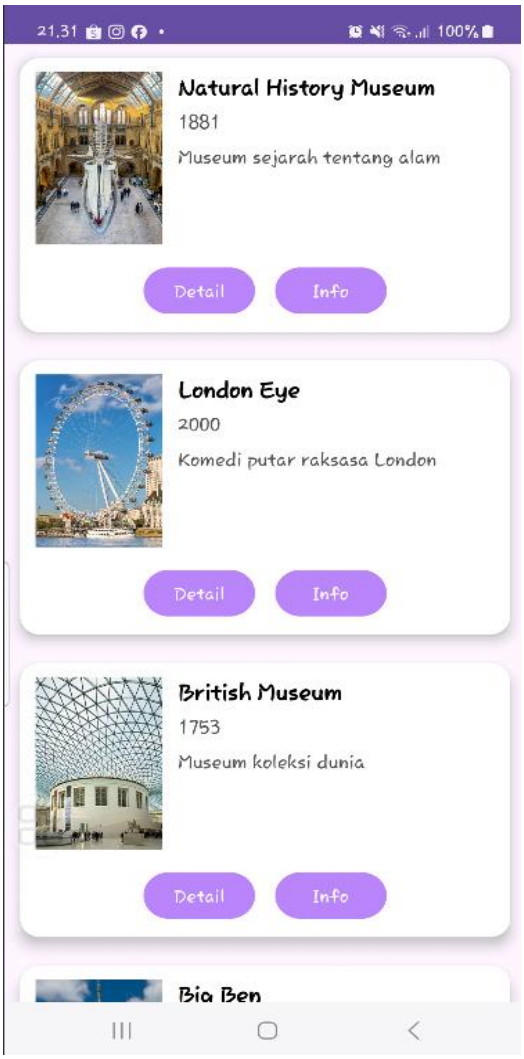
12. nav_graph.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <navigation
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       android:id="@+id/nav_graph"
5       app:startDestination="@id/HomeFragment">
6
7       <fragment
8           android:id="@+id/HomeFragment"
9   android:name="com.example.londondestination.HomeFragment
    "
10          android:label="HomeFragment" >
11          <action
12  android:id="@+id/action_HomeFragment_to_detailFragment"
13              app:destination="@id/detailFragment" />
14      </fragment>
15
16      <fragment
17          android:id="@+id/detailFragment"
18  android:name="com.example.londondestination.FragmentGuwe
    h"
19          android:label="DetailFragment" >
20          <argument
21              android:name="imageResId"
22              app:argType="integer" />
23          <argument
24              android:name="nama"
25              app:argType="string" />
26          <argument
```

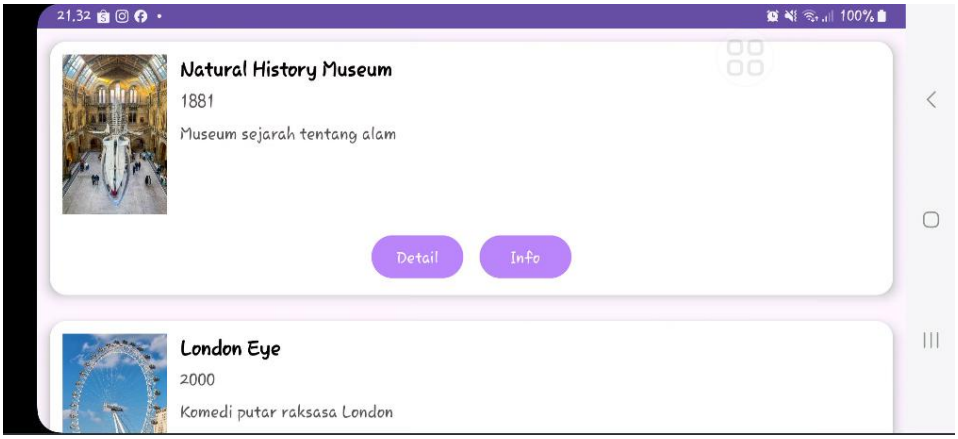
```
27              android:name="deskripsi"
28              app:argType="string" />
29      </fragment>
30
31  </navigation>
```

Tabel 1. 12 Source Code nav_graph

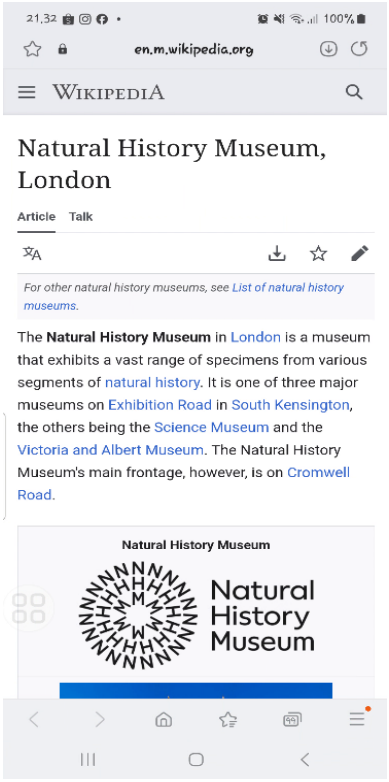
B. Output Program



Gambar 1 Screenshot Hasil Jawaban Soal 1



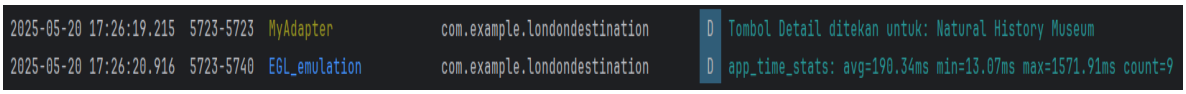
Gambar 2 Screenshot Hasil Jawaban Soal 1



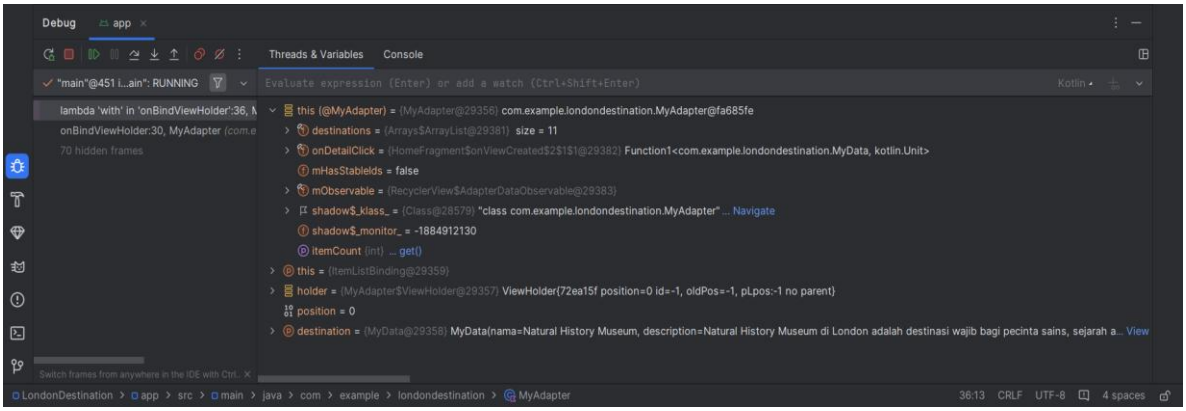
Gambar 3 Screenshot tombol Detail

Gambar 4 Screenshot tombol Info

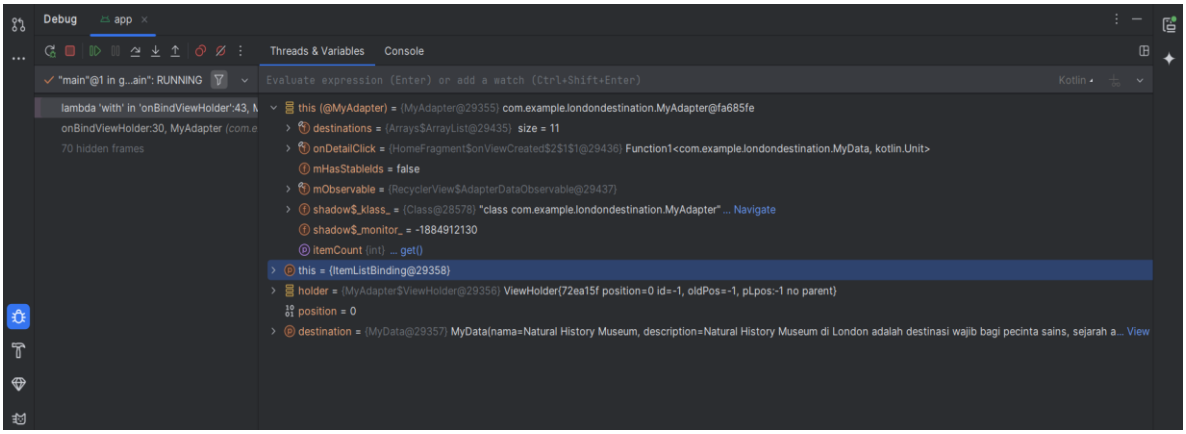
Gambar 5 Screenshot list data



Gambar 6 Screenshot Tombol Detail



Gambar 7 Screenshot Debugger



Gambar 8 Screenshot Debugger Tombol Detail

C. Pembahasan

1. MainActivity.kt

Pada file `MainActivity` ini merupakan activity utama dalam aplikasi Android yang menggunakan view binding melalui class `ActivityMainBinding` agar bisa mengakses tampilan yang sudah didefinisikan di `activity_main.xml`. Dalam metode `onCreate` disini activity akan diinisialisasi dengan cara memanggil `super.onCreate(savedInstanceState)`, kemudian binding diatur menggunakan `ActivityMainBinding.inflate(layoutInflater)` dan tampilan activity diatur dengan `setContentView(binding.root)`.

Kita lihat dari segi implementasi logging, meskipun file `MainActivity` yang ditampilkan tidak secara eksplisit menampilkan proses terkait list data, tombol, atau navigasi ke halaman detail, tetapi `MainActivity` berperan sebagai kontainer utama dari fragment atau komponen lain yang memuat logika tersebut. Mari kita lihat soal d bagian `Log` saat data item masuk ke dalam list (a), log saat tombol `Detail` dan tombol `Explicit Intent` ditekan (b), dan `log` data dari list yang dipilih ketika berpindah ke halaman `Detail` (c) kemungkinan besar diatur oleh fragment atau komponen lain yang berada di dalam `MainActivity`, seperti `HomeFragment`. Oleh karena itu, meskipun `MainActivity` tidak secara langsung menangani logging untuk ketiga event atau permintaan soal tersebut, semua proses tersebut terjadi dalam `lifecycle` dari `MainActivity`. Maka, jika

diperlukan `logging` untuk aktivitas aplikasi dapat pula dilakukan di sini, misalnya dengan menambahkan `log` di dalam `onCreate` agar bisa mencatat bahwa aktivitas utama dimulai atau bahwa aplikasi dalam keadaan aktif dan siap menampilkan konten yang akan dimasukkan.

2. FragmentGuweh.kt

Pada class `FragmentGuweh` merupakan implementasi dari sebuah fragment yang berguna untuk menampilkan halaman detail dari suatu item destinasi di aplikasi. Dimana fragment ini menggunakan view binding melalui `DetailFragmentBinding` agar bisa mengakses komponen UI yang didefinisikan dalam layout `detail_fragment.xml`. Adanya metode `onCreateView` layout di-inflate dan binding disiapkan. Selanjutnya, di bagian `onViewCreated` fragment mengambil data dari argument bundle yang dikirim saat navigasi yaitu `imageResId`, `nama`, dan `deskripsi`, dengan nilai default jika tidak ditemukan. Nilai-nilai ini kemudian digunakan untuk mengatur konten tampilan seperti, gambar, judul, dan deskripsi.

Dilihat terkait implementasi logging untuk event yang disebutkan seperti, log saat data item masuk ke dalam list seperti soal d bagian (a) tidak terjadi di dalam `FragmentGuweh` karena fragment ini hanya bertugas menampilkan `detail` satu item bukannya menangani seluruh list. Namun, `log` saat tombol `Detail` dan tombol `Explicit Intent` ditekan seperti soal d bagian (b) dapat dihubungkan menuju fragment ini karena fragment ini dipanggil akibat salah satu dari tombol tersebut ditekan. Maka, saat metode `onViewCreated` dijalankan dan data ditampilkan dapat dipastikan bahwa salah satu tombol telah ditekan, serta saat itu bisa dilakukan logging. Kita lihat lagi di `Log` data dari list yang dipilih ketika berpindah ke halaman `Detail` seperti soal d bagian (c) karena fragment ini menerima data dari item yang dipilih saja dengan demikian, di dalam `onViewCreated` dapat ditambahkan logging yang mencatat data `imageResId`, `nama`, dan `deskripsi` sebagai bentuk pencatatan informasi dari item dipilih saat berpindah menuju halaman `detail`. Jadi, meskipun `log` tidak dituliskan secara eksplisit dalam kode yang ditampilkan,

FragmentGuweh merupakan tempat yang tepat agar bisa mencatat event log terkait pemilihan dan penampilan data detail.

3. HomeFragment.kt

Pada class HomeFragment berfungsi sebagai UI utama yang menampilkan daftar destinasi wisata menggunakan RecyclerView. Di sini fragment ini dengan arsitektur MVVM yang mana data destinasi diperoleh dari HomeViewModel melalui objek destinationList merupakan Flow. Ketika data dikirim oleh ViewModel dan fragment ini mengumpulkannya menggunakan collectLatest, lalu menginisialisasi kembali MyAdapter dengan data tersebut dan menyetelnya menuju RecyclerView. Nah, ini bagian dari alur masuknya data ke dalam list.

Kita lihat di bagian logging untuk event ada saat tombol Detail dan tombol Explicit Intent karena pencatatan dilakukan di dalam class MyAdapter yang diinisialisasi dalam HomeFragment. Ketika kita (user) menekan tombol Detail atau Link pada setiap item daftar, adapter nantinya mencetak log melalui Log.d merekam aktivitas kita (user) terhadap item yang ditekan.

Selanjutnya, saat sebuah item dipilih dengan data tersebut diteruskan melalui selectedItem di ViewModel. Di dalam fragment, data ini dikumpulkan menggunakan collectLatest dan jika item tidak null, maka nantinya dicetak log berisi informasi lengkap mengenai item yang dipilih seperti, nama, tahun, dan deskripsi. Setelah log tercetak, fragment melakukan navigasi menuju DetailFragment menggunakan findNavController().navigate() dengan membawa data yang diperlukan dalam bentuk Bundle.

Jadi, kode HomeFragment udah mencakup logging untuk tiga poin penting seperti, saat data masuk ke daftar melalui proses pengumpulan dari Flow, saat tombol ditekan melalui adapter, dan saat item dipilih dan digunakan untuk navigasi ke halaman detail.

4. HomeViewModel.kt

Pada class `HomeViewModel` ini berperan sebagai bagian dari arsitektur `MVVM` yang berguna dalam menyimpan dan mengelola data dari destinasi wisata. Dimana data tersebut disimpan dalam `_destinationList` sebuah objek `MutableStateFlow bertipe List<MyData>` dan kemudian diekspose ke luar sebagai `destinationList` bersifat `StateFlow`. Nah, saat kita inisialisasi `ViewModel` memuat daftar destinasi menuju `_destinationList` dengan membuat list berisi objek-objek `MyData`, lalu menetapkan data ini menjadi nilai `StateFlow`. Di momen data dimuat ke dalam list tersebut, maka dilakukan logging menggunakan `Log.d` dengan `"HomeViewModel"` mencatat jumlah item yang berhasil dimuat ke dalam daftar `(Log.d("HomeViewModel", "List data berhasil dimuat sebanyak ${data.size} item"))`. Nah, logging ini menjawab kebutuhan untuk kode ini karena mencatat event saat data item masuk ke dalam list termasuk ke poin soal d bagian (a).

Kemudian, kode juga menyediakan fungsi `onItemClicked` berguna untuk menetapkan sebuah item yang dipilih dari list sebagai nilai `selectedItem`. Ketika fungsi ini dipanggil oleh `HomeFragment` melalui interaksi kita atau pengguna (seperti menekan tombol `Detail` atau `Explicit Intent` di dalam `MyAdapter`), item yang ditekan nantinya disimpan di dalam `_selectedItem`. Dari tombol-tombol tersebut ditangani langsung di adapter (`MyAdapter`) yang sudah disiapkan di `HomeFragment` dan logging untuk event saat tombol Detail dan tombol Explicit Intent ditekan seperti poin soal d bagian (b) dicatat menggunakan `Log.d` dari dalam adapter saat fungsi-fungsi tersebut dipanggil.

Selain itu, jika item yang dipilih diambil dari `selectedItem` oleh `HomeFragment` dan digunakan agar bisa berpindah menuju halaman detail dimana data dari item tersebut akan dicetak melalui `log`. Nah, logging ini mencakup informasi seperti nama tempat, tahun, dan deskripsi singkat dari item yang dipilih. Hal ini menjawab kebutuhan `event log` data dari list yang dipilih ketika berpindah ke halaman Detail sesuai pion soal d bagian (c). Dengan begitu, proses pencatatan

aktivitas penting dalam alur aplikasi telah diimplementasikan sesuai kebutuhan menggunakan `Log.d` dari `ViewModel`, `Adapter`, hingga `Fragment`.

5. HomeViewModelFactory.kt

Pada class `HomeViewModelFactory` merupakan implementasi dari `ViewModelProvider.Factory` berguna agar bisa menghasilkan instance dari `HomeViewModel`. Dengan metode create yang dilakukan sebagai pemeriksaan apakah `modelClass` turunan dari `HomeViewModel`. Jika benar, maka `HomeViewModel` nantinya dibuat dan dikembalikan sebagai instance dari `T`. Jika tidak sesuai, maka akan dilemparkan `exception IllegalArgumentException` dengan pesan `"Unknown ViewModel class"`.

Meskipun kode `HomeViewModelFactory` ini tidak secara langsung memuat data ataupun menangani aksi kita (sebagai pengguna saat mengklik tombol), namun perannya sangat penting dalam memastikan bahwa instance `HomeViewModel` dapat dibuat dan dipergunakan oleh komponen seperti `Fragment` atau `Activity`. Saat proses logging yang berkaitan dengan log saat data item masuk ke dalam list di bagian poin soal d bagian (a) dilakukan di dalam `HomeViewModel` diciptakan melalui factory ini. Jadi, ketika `HomeViewModelFactory` memproduksi `HomeViewModel`, maka secara tidak langsung proses logging dari pemuatan data akan terjadi karena sudah tertulis di blok init dalam `HomeViewModel`.

Kesimpulan pada file ini adalah meskipun `HomeViewModelFactory` tidak berisi kode logging secara langsung, tetapi peran utamanya itu sebagai penyedia `ViewModel` yang menjalankan log pada bagian-bagian yang relevan dalam alur aplikasi atau alur data aplikasinya.

6. MyAdapter.kt

Pada class `MyAdapter` merupakan class untuk `RecyclerView` dalam aplikasi Android yang menampilkan daftar destinasi wisata di London. Dimana

adapter ini menerima dua parameter utama sepeti daftar data destinations bertipe `List<MyData>` dan `onDetailClick` nantinya dipanggil saat kita klik tombol Detail. Nah, `Logging event` disini diimplementasikan menggunakan `Log.d()` agar bisa mencatat aktivitas kita selama run aplikasi atau mengklik salah satu tombol dan alur datanya yang sangat berguna saat kita melakukan debugging.

Pertama di poin soal d bagian (a), kode `MyAdapter` adalah adapter untuk `RecyclerView` berguna saat menampilkan daftar objek `MyData` ke dalam tampilan daftar (list). Di dalam `onBindViewHolder` itu setiap item dari daftar destinations diproses dan diikat menuju tampilan `item_list` menggunakan view binding dari `ItemListBinding`. Saat proses pengikatan ini terjadi berarti data item akan masuk menuju list dan tampil di layar, sehingga event ini bisa dianggap sebagai pencatatan saat data sudah dimasukkan ke dalam daftar.

Kedua di poin soal d bagian (b), logging event sudah diimplementasikan saat kita (user) menekan dua tombol yaitu, tombol `Link` dan tombol `Detail`. Dimana bagian `buttonLink.setOnClickListener` saat kita (user) mengklik tombol membuka URL menggunakan `Intent.ACTION_VIEW` dan mencatat aksi tersebut dengan `Log.d("MyAdapter", "Tombol Link ditekan untuk: ${destination.nama}")`. Begitu juga dengan tombol `Detail` saat kita klik fungsi `onDetailClick(destination)` dijalankan dan aplikasi mencatat interaksi pengguna dengan `Log.d("MyAdapter", "Tombol Detail ditekan untuk: ${destination.nama}")`. Nah, ini sepenuhnya memenuhi permintaan agar bisa mencatat log pada tombol-tombol tersebut ketika di klik.

Ketiga di poin soal d bagian (c), data dari list yang dipilih ketika berpindah menuju halaman detail dicatat melalui pemanggilan fungsi `onDetailClick(destination)` juga sudah dilengkapi dengan log. Dari fungsi ini berguna untuk membuka halaman baru (misalnya `DetailActivity`) dan membawa data item dipilih. Dengan mencatat `Log.d("MyAdapter", "Tombol Detail ditekan untuk: ${destination.nama}")`, maka data yang dipilih saat pindah halaman pun sudah ter-log dengan jelas.

Jadi, kode ini sudah mengimplementasikan logging untuk ketiga kebutuhan yaitu, pencatatan item yang tampil dalam list, interaksi kita saat klik tombol, dan data yang dipilih untuk detail.

7. MyData.kt

Pada data class bernama `MyData` yang berfungsi sebagai model data utama untuk aplikasi destinasi wisata London. Dimana class ini menggunakan anotasi `@Parcelize` yang memberi kemungkinan objek `MyData` bisa dikirim antar komponen Android seperti antar fragment atau activity dengan lebih mudah melalui Bundle. Dengan adanya class ini menyimpan enam properti penting untuk tiap destinasi seperti, nama (nama tempat wisata), description (deskripsi lengkap), descriptionsingkat (deskripsi singkat), year (tahun pendirian atau peresmian), image (ID dari gambar sumber daya), dan link (tautan ke halaman informasi lebih lanjut). Pentingnya struktur ini agar aplikasi bisa menampilkan informasi yang lengkap dan interaktif mengenai setiap lokasi wisata dalam bentuk daftar, serta meneruskan data dengan rapi ke tampilan detail saat kita memilih salah satu destinasi. Jadi, file ini menjadi pondasi data yang akan digunakan oleh adapter dan fragment lainnya di aplikasi ini.

8. Detail_fragment.xml

Pada bagian ini digunakan untuk menampilkan tampilan detail dari sebuah tempat wisata dalam aplikasi. Dimana seluruh konten dibungkus di dalam sebuah elemen `ScrollView` yang memungkinkan kita bisa menggulir layar ke bawah jika isi kontennya lebih panjang dari ukuran layar. Hal ini penting agar seluruh informasi tetap bisa diakses meskipun banyak atau Panjang isinya.

Di dalam `ScrollView` terdapat sebuah `CardView` yang berfungsi memberikan tampilan lebih menarik karena memiliki sudut melengkung (dengan `cardCornerRadius="16dp"`) dan (`cardElevation="8dp"`), sehingga konten tampak seperti kartu sedikit terangkat dari latar belakang membuat tampilan lebih rapi dan enak dilihat.

Isi dari CardView diletakkan di dalam LinearLayout yang diatur secara vertikal. Di dalam layout ini terdapat tiga komponen utama yaitu, pertama ada ImageView menggunakan ID detailImage berguna untuk menampilkan gambar tempat wisata yang biasanya gambar ini nantinya ditampilkan dari file drawable atau dari sumber lain. Dengan gambar ini disetel menggunakan scaleType="centerCrop" agar mengisi seluruh ruang yang disediakan dengan proporsional.

Selanjutnya, dua TextView yang pertama ada detailTitle digunakan untuk menampilkan nama tempat dengan ukuran teks yang cukup besar (20sp) dan gaya teks tebal (bold), serta diberi padding agar teks tidak menempel langsung ke sisi layar. Yang kedua, detailDescription berfungsi untuk menampilkan deskripsi lengkap dari tempat tersebut dengan deskripsi ini menggunakan ukuran teks sedikit lebih kecil (16sp) dan diberi padding sisi kiri, kanan, dan bawah agar memudahkan saat membaca. Jadi, secara keseluruhan, layout ini dibuat untuk memberikan tampilan detail tempat wisata secara bersih, informatif, dan nyaman dilihat.

9. activity_main.xml

Pada bagian ini berguna untuk mengelola navigasi antar-fragment. Dimana seluruh tampilan dibungkus di dalam ConstraintLayout merupakan salah satu jenis layout fleksibel yang memungkinkan pengaturan posisi antar elemen secara dinamis dan efisien. Namun dalam kasus ini, hanya ada satu elemen di dalamnya, jadi ConstraintLayout tidak benar-benar dimanfaatkan secara penuh. Dengan elemen utamanya itu FragmentContainerView berfungsi sebagai wadah (container) agar bisa menampilkan fragment. Adanya FragmentContainerView ini memiliki ID fragment_container_view dan diatur untuk mengisi seluruh lebar dan tinggi layar (match_parent). Terdapat property berisi android:name yang menunjuk ke bagian androidx.navigation.fragment.NavHostFragment berarti view ini berperan sebagai host fragment nantinya saat mengelola navigasi.

Adanya atribut `app:navGraph` menunjuk ke file `nav_graph` di direktori `res/navigation` berisi struktur navigasi fragment–seperti daftar tujuan (destination) dan hubungan antar fragment (misalnya aksi berpindah antar fragment). Sementara `app:defaultNavHost="true"` digunakan dalam menyatakan bahwa fragment ini adalah host default agar bisa menangani navigasi sistem, seperti tombol `"Back"` di Android. Jadi, keseluruhan bagian ini XML ini menyusun fondasi navigasi fragment berbasis Jetpack Navigation dengan melakukan penempatan satu `NavHostFragment` untuk mengatur transisi antar-fragment di dalam aplikasi tanpa perlu berpindah antar activity yang menjadikan navigasi lebih efisien dan lebih terstruktur.

10. home_fragment.xml

Pada bagian ini adalah layout untuk sebuah Fragment lebih tepatnya `HomeFragment` yang menggunakan `ConstraintLayout` sebagai layout utama. Dimana layout ini hanya memiliki satu elemen di dalamnya, yaitu sebuah `RecyclerView` dengan `ID rv_character` dengan fungsi utama dari `RecyclerView` ini untuk menampilkan daftar item secara efisien dan dapat discroll tergantung pada pengaturan adapter dan layout manager-nya.

Pada pengaturannya `RecyclerView` ini dibuat agar memenuhi seluruh ruang layar karena semua constraint-nya dihubungkan ke tepi-tepi parent layout. Hal ini terlihat dari penggunaan `0dp` untuk `layout_width` dan `layout_height` berarti ukuran akan disesuaikan berdasarkan constraint yang diberikan. Dilihat bagian Constraint-nya menghubungkan atas (Top), bawah (Bottom), kiri (Start), dan kanan (End) ke parent, sehingga `RecyclerView` menutupi seluruh tampilan yang tersedia dalam fragment.

Sementara itu, kita lihat di atribut `tools:context=".HomeFragment"` berguna hanya untuk keperluan preview di Android Studio agar tampilan ini bisa dikenali sedang digunakan `HomeFragment` saat mendesain antarmuka. Jadi, secara keseluruhan layout bagian ini sangat sederhana tapi efektif demi bisa menampilkan

daftar yang dinamis, seperti daftar tempat wisata, karakter, atau data lain nantinya diisi melalui adapter dalam kode program aplikasi ini.

11. item_list.xml

Pada layout file untuk komponen tampilan dalam aplikasi Android yang mana layout ini menggunakan `CardView` sebagai wadah utama yang memberikan efek tampilan seperti kartu, lengkap dengan sudut melengkung dan bayangan. Di dalam `CardView` terdapat `ConstraintLayout` digunakan sebagai layout utama agar bisa menyusun elemen-elemen UI dengan fleksibilitas tinggi sesuai posisi satu sama lain. Dimana bagian pertama itu `ImageView` berguna untuk menampilkan gambar, dengan ukurannya ditetapkan 100dp x 150dp dan letaknya di pojok kiri atas tampilan. Gambar diatur agar "crop" ke tengah (centerCrop) agar mengisi seluruh ruang.

Selanjutnya, ada tiga `TextView` yang masing-masing menampilkan nama (`textViewName`), tahun atau angkatan (`textViewYear`), dan deskripsi tambahan (`textViewDesc`). Dari semua `TextView` ini diletakkan di sebelah kanan `ImageView` yang disusun secara vertikal satu per satu dari atas ke bawah. Adanya `textViewName` yang menggunakan teks tebal dan ukuran huruf lebih besar cocok untuk judul atau nama utama. Sedangkan `textViewYear` dan `textViewDesc` memiliki ukuran teks yang lebih kecil dan warna abu-abu gelap (`#555555`) untuk memberikan perbedaan teks.

Di bagian paling bawah layout terdapat `LinearLayout` yang berisi dua tombol (`Button`), yaitu tombol "`Detail`" dan "`Info`". Dengan layout tombol ini diposisikan di bawah `ImageView` dan diberi margin atas agar tidak menempel langsung. Dimana warna latar tombol diambil dari `@color/purple_200` yanh memberi warna ungu dari resources aplikasi. Kedua tombol ini disiapkan untuk tindakan lebih lanjut, misalnya "`Detail`" untuk membuka tampilan biodata lengkap, dan "`Info`" untuk menampilkan informasi tambahan.

Jadi, secara keseluruhan, XML ini digunakan untuk item dalam RecyclerView karena desainnya ringkas, informatif, dan responsif. Dengan struktur layout sudah rapi sesuai pengaturan ConstraintLayout yang fleksibel dan CardView mempercantik tampilan menjadi tipe layout biasanya digunakan agar bisa menampilkan daftar data dalam bentuk kartu.

12. nav_graph.xml

Pada file navigation graph fungsinya untuk mendefinisikan alur navigasi antar tampilan atau fragment dalam sebuah aplikasi. Di bagian dalamnya akan dideklarasikan dua buah fragment yaitu, HomeFragment dan detailFragment. Dimana layout navigasi ini untuk aplikasi dapat berpindah dari satu fragment menuju fragment lainnya secara terstruktur dan mudah dikelola.

Pertama, elemen <navigation> menyatakan bahwa ini bagian root dari navigation graph yang terdapat atribut ini app:startDestination="@id/HomeFragment" yang menandakan bahwa saat aplikasi dijalankan, maka tampilan awal (halaman pertama) nantinya ditampilkan itu HomeFragment. Dengan fragment ini diberi ID @+id/HomeFragment dan terhubung menuju bagian dari class com.example.londondestination.HomeFragment.

Di dalam HomeFragment terdapat elemen <action> dengan ID action_HomeFragment_to_detailFragment merupakan aksi navigasi yang berarti jika dipicu (misalnya kita klik tombol), maka aplikasi akan berpindah dari HomeFragment menuju detailFragment. Nama action ini bisa dipanggil di kode Kotlin atau Java untuk mentrigger perpindahan halaman ke selanjutnya.

Adanya fragment tujuan yaitu detailFragment dikaitkan dengan class Java atau Kotlin com.example.londondestination.FragmentGuweh. Dimana fragment ini nantinya menerima data melalui tiga buah argument seperti, imageResId (tipe integer, biasanya ID gambar dari resource drawable), nama (tipe string), dan deskripsi (juga string) dari ketiga argumen ini memberi kemungkinan

fragment tujuan bisa menerima data yang dikirim dari `HomeFragment`, misalnya saat kita klik tombol "`Detail`". Jadi, file ini memastikan bahwa aplikasi punya alur navigasi yang rapi dari halaman utama ke halaman detail, dan sudah siap menerima data agar bisa ditampilkan.

13. Debugger

Pada debugger saya menggunakan file `MyAdapter` karena breakpoint dipasang di dua lokasi penting, yaitu `buttonLink.setOnClickListener` dan `buttonDetail.setOnClickListener`. Dimana breakpoint berguna untuk memastikan bahwa kedua tombol berfungsi sebagaimana mestinya dan bahwa data yang ditampilkan atau dikirim sesuai yang diharapkan. Sebagai contoh, ketika saya klik tombol "`Link`", maka debugger dapat memperlihatkan nilai `destination.link` dan memastikan intent menuju browser terbentuk dengan benar. Begitu pula saat tombol "`Detail`" ditekan, maka debugger dapat menunjukkan bahwa fungsi `onDetailClick(destination)` menerima data yang benar seperti nama dan deskripsi destinasi.

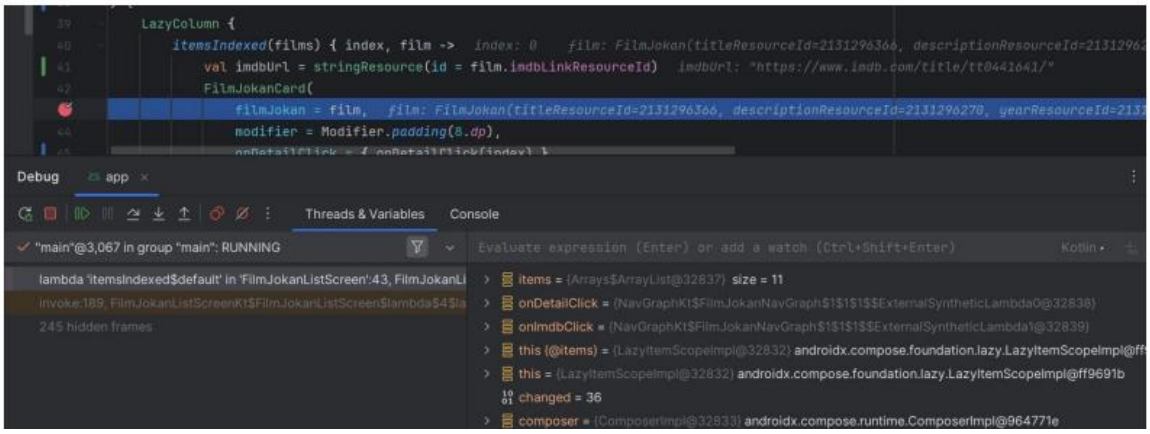
Selama proses debugging, Android Studio menyediakan tiga fitur utama yaitu, `Step Into`, `Step Over`, dan `Step Out`. Dari fitur `Step Into` digunakan untuk masuk ke dalam detail implementasi sebuah metode atau fungsi yang sedang dipanggil, misalnya masuk ke dalam `onDetailClick(destination)`. Hal ini sangat berguna saat kita ingin mengetahui lebih dalam bagaimana suatu fungsi bekerja. Sementara itu, `Step Over` digunakan untuk melompati baris kode saat ini tanpa perlu masuk ke dalam fungsi, cocok digunakan jika kita sudah yakin bahwa fungsi tersebut bekerja dengan baik dan tidak perlu ditelusuri lagi. Lalu, ada `Step Out` digunakan untuk keluar dari fungsi yang sedang ditelusuri dan kembali ke pemanggilnya, sangat membantu ketika kita sudah masuk terlalu dalam menuju struktur kode dan ingin kembali ke bagian level sebelumnya. Dengan cara memanfaatkan breakpoint dan fitur langkah demi langkah tersebut, proses penelusuran bug menjadi jauh lebih mudah dan efisien. Kemudian, dilihat lagi *gambar output program 7 dan 8* yang memperlihatkan bahwa proses

debugging berhasil menghentikan eksekusi program di dalam `onBindViewHolder` dan nilai objek destination telah ditampilkan dengan benar di jendela debugger, menandakan bahwa data yang digunakan saat itu sesuai dengan yang diharapkan.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

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



Gambar 9 Contoh Penggunaan Debugger

Jawab:

Pada `Application` berfungsi sebagai titik awal dan pusat konfigurasi aplikasi Android sebelum `Activity` atau `Service` pertama dipanggil. Untuk menggunakannya, pengembang cukup membuat class yang mewarisi dari `android.app.Application` dan mendaftarkannya di file `AndroidManifest.xml`.

Nah, kita masuk ke bagian proses debugging pada kode `MyAdapter` dimana saya sudah menempatkan breakpoint di dua bagian penting, yaitu `buttonLink.setOnClickListener` dan `buttonDetail.setOnClickListener`. Dari kedua bagian kode ini menangani interaksi kita (user) saat tombol ditekan pada tiap item `RecyclerView`. Ketika debugging mencapai kode tombol "`Link`", maka debugger nantinya memperlihatkan bahwa nilai

`destination.link` sudah benar dan menghasilkan intent dengan `ACTION_VIEW` agar bisa membuka tautan di browser. Hal ini memastikan bahwa aplikasi akan mengarahkan kita ke situs yang sesuai. Sedangkan pada tombol "`Detail`" ini debugger menunjukkan bahwa fungsi `onDetailClick(destination)` berhasil dipanggil dengan objek destination yang benar. Misalnya `nama`, `descriptionsingkat`, dan properti lainnya sesuai data dikirimkan. Informasi inilah yang sangat penting untuk memastikan bahwa interaksi kita (user) diproses sesuai yang diharapkan.

Penggunaan debugger seperti ini memungkinkan kita (user) agar bisa menelusuri alur program lebih teliti. Saat breakpoint aktif, saya mulai menggunakan fitur `Step Into` untuk masuk ke dalam implementasi fungsi (misalnya `onDetailClick`), `Step Over` untuk melewati eksekusi fungsi tanpa masuk ke dalamnya, dan `Step Out` untuk keluar dari fungsi saat ini dan kembali ke pemanggilnya. Dari seluruh fitur ini (`Step Into`, `Step Over`, dan `Step Out`) sangat berguna dalam memahami secara mendalam tentang bagaimana cara aplikasi bekerja dan memastikan bahwa tidak ada kesalahan logika atau data yang salah selama proses interaksi kita (saat kita mencoba mengklik salah satu tombol bagik itu detail maupun info) dengan tampilan aplikasi yang dimuculkan di layar.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/alysaarmelia/AlysaArmelia_2310817120009_Pemrograman_Mobile/tree/d0ebe1f8ea4e6c5485df1b46f5b68e7be9252c3c/PRAK_MODUL4