

**LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4**



ViewModel and Debugging

Oleh:

Muhammad Ibnu Sina

NIM. 2310817210009

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

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 4

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

Nama Praktikan : Muhammad Ibnu Sina
NIM : 2310817210009

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Muhammad Ibnu Sina
NIM. 2310817210009

Muti`a Maulida S.Kom M.T.I
NIP. 19881027 201903 20 13

DAFTAR ISI

LEMBAR PENGESAHAN	2
DAFTAR ISI	3
DAFTAR GAMBAR	4
DAFTAR TABEL	5
SOAL 1	6
A. Source Code	6
B. Output Program	23
C. Pembahasan	26
D. Tautan Git	33

DAFTAR GAMBAR

Gambar 1 Contoh Penggunaan Debugger	6
Gambar 2 Hasil Tampilan UI List Soal 1	23
Gambar 3 Hasil Tampilan UI Detail Soal 1	24
Gambar 4 Screenshot E-commerce setelah Intent Soal 1	25
Gambar 5 Screenshot Debungging	25

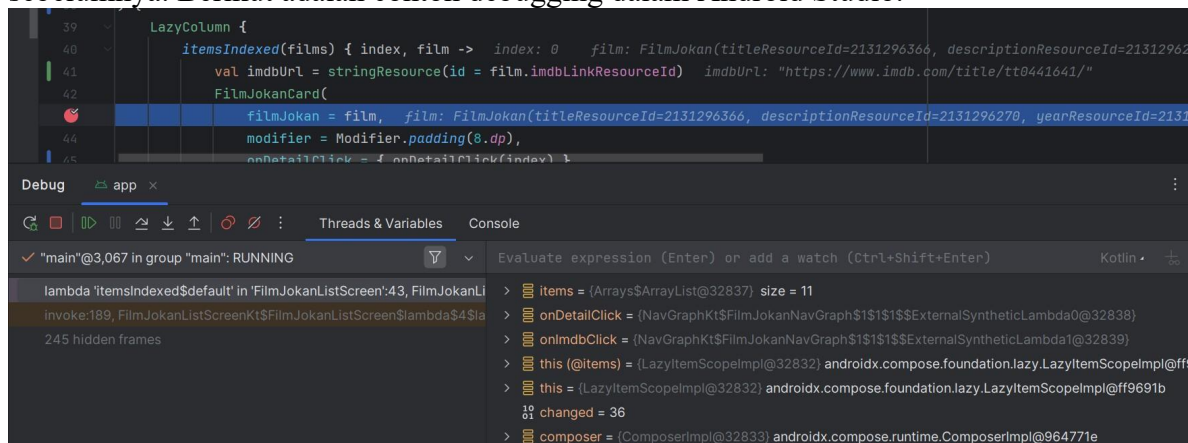
DAFTAR TABEL

Tabel 1 Source Code Jawaban Soal 1	6
Tabel 2 Source Code Jawaban Soal 1	7
Tabel 3 Source Code Jawaban Soal 1	8
Tabel 4 Source Code Jawaban Soal 1	10
Tabel 5 Source Code Jawaban Soal 1	11
Tabel 6 Source Code Jawaban Soal 1	15
Tabel 7 Source Code Jawaban Soal 1	16
Tabel 8 Source Code Jawaban Soal 1	17
Tabel 9 Source Code Jawaban Soal 1	17
Tabel 10 Source Code Jawaban Soal 1	19
Tabel 11 Source Code Jawaban Soal 1	20
Tabel 12 Source Code Jawaban Soal 1	22

SOAL 1

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
2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

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



Gambar 1 Contoh Penggunaan Debugger

A. Source Code

1. MainActivity.kt

Tabel 1 Source Code Jawaban Soal 1

```
1 package com.example.ootdlist
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5
6 class MainActivity : AppCompatActivity() {
```

7	override fun onCreate(savedInstanceState: Bundle?) {
8	super.onCreate(savedInstanceState)
9	setContentView(R.layout.activity_main)
10	}
11	}

2. DetailFragment.kt

Tabel 2 Source Code Jawaban Soal 1

1	package com.example.ootdlist.fragments
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 com.example.ootdlist.data.Ootd
11	import com.example.ootdlist.databinding.FragmentDetailBinding
12	import java.net.URLEncoder
13	
14	class DetailFragment : Fragment() {
15	private var _binding: FragmentDetailBinding? = null
16	private val binding get() = _binding!!
17	
18	override fun onCreateView(
19	inflater: LayoutInflater,
20	container: ViewGroup?,
21	savedInstanceState: Bundle?
22): View {
23	_binding = FragmentDetailBinding.inflate(inflater,
24	container, false)
25	return binding.root
26	}
27	override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
28	super.onViewCreated(view, savedInstanceState)
29	
30	val ootdId = requireArguments().getInt("ootdId")
31	val ootd = Ootd.getById(ootdId)
32	
33	if (ootd != null) {
34	binding.ootdTitle.text = ootd.title
35	binding.ootdGender.text = ootd.gender
36	binding.ootdStyle.text = ootd.style
37	binding.ootdDescription.text = ootd.description
38	binding.ootdImage.setImageResource(ootd.imageResId)
39	
40	binding.btnShop.setOnClickListener {

41	val searchQuery = "\${ootd.title} \${ootd.style}
42	val encodedQuery =
	URLEncoder.encode(searchQuery, "UTF-8")
43	val url =
	"https://shopee.co.id/search?keyword=\$encodedQuery"
44	val intent = Intent(Intent.ACTION_VIEW,
	Uri.parse(url))
45	startActivity(intent)
46	}
47	}
48	}
49	
50	override fun onDestroyView() {
51	super.onDestroyView()
52	_binding = null
53	}
54	}
55	

3. ListFragment.kt

Tabel 3 Source Code Jawaban Soal 1

1	package com.example.ootdlist.fragments
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.navigation.fragment.findNavController
11	import androidx.recyclerview.widget.LinearLayoutManager
12	import com.example.ootdlist.R
13	import com.example.ootdlist.adapter.OotdAdapter
14	import com.example.ootdlist.data.Ootd
15	import com.example.ootdlist.databinding.FragmentListBinding
16	import java.net.URLEncoder
17	
18	class ListFragment : Fragment() {
19	private var _binding: FragmentListBinding? = null
20	private val binding get() = _binding!!
21	
22	override fun onCreateView(
23	inflater: LayoutInflater,
24	container: ViewGroup?,
25	savedInstanceState: Bundle?
26): View {
27	_binding = FragmentListBinding.inflate(inflater, container,


```

28         false)
29         return binding.root
30     }
31     override fun onViewCreated(view: View, savedInstanceState: Bundle?)
32     {
33         super.onViewCreated(view, savedInstanceState)
34
35         val ootdList = Ootd.ootdList
36         val adapter = OotdAdapter(ootdList, object :
OotdAdapter.OnItemClickListener {
37             override fun onDetailClick(ootd: Ootd) {
38                 val bundle = Bundle().apply {
39                     putInt("ootdId", ootd.id)
40                 }
41
42                 findNavController().navigate(R.id.action_listFragment_to_detailFragment,
bundle)
43
44                 override fun onShopClick(ootd: Ootd) {
45                     val searchQuery = "${ootd.title} ${ootd.style}
46                     ${ootd.gender} fashion"
47                     val encodedQuery = URLEncoder.encode(searchQuery, "UTF-
8")
48                     val url =
"https://shopee.co.id/search?keyword=$encodedQuery"
49                     val intent = Intent(Intent.ACTION_VIEW, Uri.parse(url))
50                     startActivity(intent)
51                 }
52             })
53         binding.recyclerView.layoutManager =
54         LinearLayoutManager(context)
55         binding.recyclerView.adapter = adapter
56
57         if (ootdList.isEmpty()) {
58             binding.recyclerView.visibility = View.GONE
59             binding.emptyView.visibility = View.VISIBLE
60         } else {
61             binding.recyclerView.visibility = View.VISIBLE
62             binding.emptyView.visibility = View.GONE
63         }
64     }
65     override fun onDestroyView() {
66         super.onDestroyView()
67         _binding = null
68     }
69 }

```

4. OotdAdapter.kt

Tabel 4 Source Code Jawaban Soal 1

```
1 package com.example.ootdlist.adapter
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
5 import androidx.recyclerview.widget.RecyclerView
6 import com.example.ootdlist.R
7 import com.example.ootdlist.data.Ootd
8 import com.example.ootdlist.databinding.ItemOotdBinding
9
10 class OotdAdapter(private val ootdList: List<Ootd>, private val
listener: OnItemClickListener) :
11     RecyclerView.Adapter<OotdAdapter.OotdViewHolder>() {
12
13     interface OnItemClickListener {
14         fun onDetailClick(ootd: Ootd)
15         fun onShopClick(ootd: Ootd)
16     }
17
18     inner class OotdViewHolder(private val binding:
ItemOotdBinding) : RecyclerView.ViewHolder(binding.root) {
19         fun bind(ootd: Ootd) {
20             binding.ootdTitle.text = ootd.title
21             binding.ootdGender.text = ootd.gender
22             binding.ootdStyle.text = ootd.style
23             binding.ootdDescription.text = ootd.description
24             binding.ootdImage.setImageResource(ootd.imageResId)
25
26             binding.btnImdb.setOnClickListener {
27                 listener.onShopClick(ootd)
28             }
29
30             binding.btnDetail.setOnClickListener {
31                 listener.onDetailClick(ootd)
32             }
33
34             val imageContentDesc = itemView.context.getString(
35                 R.string.ootd_image_content_description,
36                 ootd.title,
37                 ootd.gender
38             )
39             binding.ootdImage.contentDescription =
40             imageContentDesc
41         }
42
43         override fun onCreateViewHolder(parent: ViewGroup, viewType:

```

44	Int): OotdViewHolder {
	val binding =
45	ItemOotdBinding.inflate(LayoutInflater.from(parent.context),
	parent, false)
	return OotdViewHolder(binding)
46	}
47	
48	override fun onBindViewHolder(holder: OotdViewHolder,
49	position: Int) {
	holder.bind(ootdList[position])
50	}
51	
52	override fun getItemCount(): Int = ootdList.size
53	}

5. OotdData.kt

Tabel 5 Source Code Jawaban Soal 1

1	package com.example.ootdlist.data
2	
3	import com.example.ootdlist.R
4	
5	data class Ootd(
6	val id: Int,
7	val title: String,
8	val style: String,
9	val description: String,
10	val imageResId: Int,
11	val gender: String
12) {
13	companion object {
14	val ootdList = listOf(
15	Ootd(1, "Block Core", "Streetwear", "gaya streetwear
	modern yang lahir dari semangat bereksperimen dan mengaburkan
	batas antara dunia olahraga dan fashion jalanan. " +
16	"Keunikannya terletak pada penggunaan jersey
	bola sebagai item utama, menjadikannya lebih dari sekadar simbol
	fandom-melainkan pernyataan gaya yang berani. " +
17	"Gaya ini merepresentasikan generasi muda
	yang dinamis, penuh karakter, dan tidak takut memadukan elemen
	yang tampak kontras. Block Core bukan hanya soal pakaian," +
18	" tapi juga tentang cara mengekspresikan
	diri dengan bebas dan orisinal." +
19	"\n\n Atasan: Jersey klub bola bergaya retro
	dengan motif kotak merah-putih dan logo "PACHA Ibiza" yang
	menonjol." +
20	"\n\n Bawahan: Celana wide leg bahan wol
	berwarna abu-abu tua, menciptakan kontras menarik antara sporty

21	dan formal.", R.drawable.block_core_man, "Men"),
22	Ootd(2, "Block Core", "Streetwear", " gaya
23	streetwear modern yang lahir dari semangat bereksperimen dan
24	mengaburkan batas antara dunia olahraga dan fashion jalanan. " +
25	"Keunikannya terletak pada penggunaan jersey
26	bola sebagai item utama, menjadikannya lebih dari sekadar simbol
27	fandom—melainkan pernyataan gaya yang berani. " +
28	"Gaya ini merepresentasikan generasi muda
29	yang dinamis, penuh karakter, dan tidak takut memadukan elemen
30	yang tampak kontras. Block Core bukan hanya soal pakaian, " +
31	"tapi juga tentang cara mengekspresikan diri
32	dengan bebas dan orisinal." +
33	"\n\n Atasan: Jersey FC Barcelona dengan
34	potongan crop top, menampilkan logo Nike dan sponsor Spotify
35	yang mencolok." +
36	"\n\n Bawahan: Celana cargo hitam longgar,
37	menciptakan siluet santai namun tetap bold.",
38	R.drawable.block_core_girl, "Women"),
39	Ootd(3, "Cottage Core", "Romantic", "gaya fashion
40	yang terinspirasi dari kehidupan pedesaan yang tenang, romantis,
41	dan penuh nostalgia. " +
42	"Gaya ini mengangkat elemen alami dan
43	vintage—mulai dari motif bunga kecil, bahan katun atau linen,
44	warna-warna hangat dan lembut," +
45	" hingga siluet yang mengalir dan nyaman.
46	Cottage Core bukan hanya tentang penampilan, tetapi juga
47	menggambarkan gaya hidup yang menghargai " +
48	" keindahan sederhana, kedekatan dengan
49	alam, dan suasana yang damai." +
50	"\n\n Atasan: Sweater rajut motif folk
51	berwarna dasar hitam dengan aksen pola merah-putih yang memberi
52	kesan hangat." +
53	"\n\n Bawahan: Celana panjang corduroy
54	coklat muda berpotongan longgar, menambah kesan earthy dan
55	santai.", R.drawable.cottage_core_man, "Men"),
56	Ootd(4, "Cottage Core", "Romantic", "gaya fashion
57	yang terinspirasi dari kehidupan pedesaan yang tenang, romantis,
58	dan penuh nostalgia." +
59	" Gaya ini mengangkat elemen alami dan
60	vintage—mulai dari motif bunga kecil, bahan katun atau linen,
61	warna-warna hangat dan lembut," +
62	" hingga siluet yang mengalir dan nyaman.
63	Cottage Core bukan hanya tentang penampilan, tetapi juga
64	menggambarkan gaya hidup yang menghargai " +
65	"keindahan sederhana, kedekatan dengan alam,
66	dan suasana yang damai." +
67	"\n\n Atasan: Kardigan rajut tipis berwarna
68	merah dipadukan dengan dress floral putih bermotif bunga kecil
69	dan detail renda." +

41	"\n\n Bawahan: Dress panjang mengalir dengan potongan klasik dan aksan pita, menciptakan siluet lembut yang feminin.", R.drawable.cottage_core_girl, "Women"),
42	
43	Ootd(5, "Grunge", "Alternative", "Outfit bergaya grunge identik dengan kesan santai, berantakan yang terkesan effortless namun tetap keren. " +
44	"Ciri khasnya adalah dominasi warna gelap, layering, bahan denim, serta kaos band sebagai elemen penting. " +
45	"Gaya ini memadukan unsur musik alternatif dengan sikap anti-mainstream khas era 90-an." +
46	"\n\n Atasan: Kaos grafis berwarna hitam dilapisi kemeja flanel motif kotak merah-hijau yang terbuka, menciptakan tampilan layering khas grunge klasik." +
47	"\n\n Bawahan: Celana longgar berbahan corduroy hijau tua, memberi tekstur earthy yang sejalan dengan elemen grunge vintage.", R.drawable.grunge_man, "Men"),
48	
49	Ootd(6, "Grunge", "Alternative", "Outfit bergaya grunge identik dengan kesan santai, berantakan yang terkesan effortless namun tetap keren. " +
50	"Ciri khasnya adalah dominasi warna gelap, layering, bahan denim, serta kaos band sebagai elemen penting." +
51	" Gaya ini memadukan unsur musik alternatif dengan sikap anti-mainstream khas era 90-an." +
52	"\n\n Atasan: Kaos band The Beatles berwarna hitam kusam dengan print besar, memberikan nuansa retro dan rebellious." +
53	"\n\n Bawahan: Celana jeans baggy berpotongan longgar berwarna biru pudar, menambah kesan santai dan nonchalant.", R.drawable.grunge_girl, "Women"),
54	
55	Ootd(7, "Japanese Americana", "Fusion", "Outfit bergaya Japanese Americana memadukan gaya kasual klasik khas Amerika seperti denim dan workwear dengan sentuhan rapi, minimalis, dan estetika layering ala Jepang. " +
56	"Ciri khasnya meliputi jaket denim, celana berpotongan longgar, warna earthy seperti olive dan navy, serta perhatian terhadap siluet dan tekstur." +
57	"\n\n Atasan: Kemeja putih berkerah dengan dasi cokelat yang dilapisi jaket denim fitted berwarna indigo, menciptakan tampilan rapi dengan lapisan Americana yang kuat." +
58	"\n\n Bawahan: Celana workwear hijau zaitun berpotongan lurus yang terinspirasi dari gaya militer, menyeimbangkan kesan maskulin dan vintage.", R.drawable.japanese_americana_man, "Men"),
59	
60	Ootd(8, "Japanese Americana", "Fusion", "Outfit bergaya Japanese Americana memadukan gaya kasual klasik khas Amerika seperti denim dan workwear dengan sentuhan rapi,

61	minimalis, dan estetika layering ala Jepang. " + "Ciri khasnya meliputi jaket denim, celana berpotongan longgar, warna earthy seperti olive dan navy, serta perhatian terhadap siluet dan tekstur." +
62	"\n\n Atasan: Kaos bergaris hitam-putih bergaya Breton yang dimasukkan ke dalam celana, memberikan kesan clean namun santai. Sebuah jaket denim biru dikenakan secara kasual di bahu sebagai outer, menambah sentuhan Americana klasik." +
63	"\n\n Bawahan: Celana cargo hijau army berpotongan lebar yang memberi nuansa utilitarian khas workwear, sekaligus menciptakan siluet longgar yang modis dan nyaman.", R.drawable.japanese_american_girl, "Women"),
64	
65	Ootd(9, "Old Money", "Classic", "Outfit bergaya Old Money mencerminkan kemewahan yang tidak mencolok, elegan, dan timeless. Gaya ini lekat dengan nuansa klasik, potongan yang rapi," +
66	" bahan berkualitas tinggi, serta warna- warna netral yang terkesan mahal. Tidak berfokus pada tren, tetapi pada keanggunan dan prestise." +
67	"\n\n Atasan: Kemeja putih klasik dengan kerah terbuka, memberikan kesan effortless dan bersih namun tetap terjaga rapi. Diselipkan ke dalam celana untuk tampilan yang lebih polished." +
68	"\n\n Bawahan: Celana wool cokelat berpotongan longgar dan pleated, menambah kesan sophisticated khas gentleman lama. ", R.drawable.old_money_man, "Men"),
69	
70	Ootd(10, "Old Money", "Classic", "Outfit bergaya Old Money mencerminkan kemewahan yang tidak mencolok, elegan, dan timeless. Gaya ini lekat dengan nuansa klasik, potongan yang rapi," +
71	" bahan berkualitas tinggi, serta warna- warna netral yang terkesan mahal. Tidak berfokus pada tren, tetapi pada keanggunan dan prestise." +
72	"\n\n Atasan: Kemeja satin berwarna ivory dengan detail kerah klasik, memberikan tampilan sophisticated yang mewah namun understated. Dipadukan dengan blazer oversized dalam nuansa beige yang memberikan siluet structured dan elegan." +
73	"\n\n Bawahan: Celana tailored high-waist warna senada yang jatuh longgar hingga menyentuh lantai, menegaskan citra refined dan berkelas.", R.drawable.old_money_girl, "Women"),
74	
75	Ootd(11, "Y2K", "Retro Futurism", "Outfit bergaya Y2K (Year 2000) identik dengan nuansa futuristik, playful, dan eksperimental yang merefleksikan semangat awal milenium. " +
76	"Gaya ini sering menggabungkan elemen-elemen seperti low-rise jeans, atasan ketat, bahan denim dengan efek faded, " +

77	"serta aksesoris mencolok yang terinspirasi dari dunia pop culture, teknologi, dan fashion selebriti tahun 2000-an." +
78	"\n\n Atasan: Kemeja putih sebagai dasar dipadukan dengan sweater hitam dan jaket kulit hitam sebagai outer, menghasilkan tampilan layering maskulin dan berani yang populer di era awal 2000-an." +
79	"\n\n Bawahan: Celana jeans longgar berwarna hitam pudar, menciptakan kesan laid-back namun tetap tajam, selaras dengan gaya street style Y2K.", R.drawable.y2k_man, "Men"),
80	
81	Ootd(12, "Y2K", "Retro Futurism", "Outfit bergaya Y2K (Year 2000) identik dengan nuansa futuristik, playful, dan eksperimental yang merefleksikan semangat awal milenium. " +
82	"Gaya ini sering menggabungkan elemen-elemen seperti low-rise jeans, atasan ketat, bahan denim dengan efek faded, " +
83	"serta aksesoris mencolok yang terinspirasi dari dunia pop culture, teknologi, dan fashion selebriti tahun 2000-an." +
84	"\n\n Atasan: Cropped cardigan abu-abu tua yang ketat dipadukan dengan tank top abu-abu muda, menciptakan tampilan layering yang menggoda dan edgy." +
85	"\n\n Bawahan: Celana jeans flare berpotongan low-rise dengan efek washed-out, memperkuat siluet tubuh dan nuansa retro futuristik khas Y2K.", R.drawable.y2k_girl, "Women")
86)
87	
88	fun getById(id: Int): Ootd? = ootdList.find { it.id ==
89	id }
90	}

6. OotdView Model.kt

Tabel 6 Source Code Jawaban Soal 1

1	package com.example.modul4
2	
3	import android.util.Log
4	import androidx.lifecycle.ViewModel
5	import kotlinx.coroutines.flow.MutableStateFlow
6	import kotlinx.coroutines.flow.StateFlow
7	
8	class OotdViewModel : ViewModel() {
9	
10	private val _ootdList =
	MutableStateFlow<List<Ootd>>(emptyList())

11	val ootdList: StateFlow<List<Ootd>> get() = _ootdList
12	
13	private val _selectedOotd =
14	MutableStateFlow<Ootd?>(null)
15	val selectedOotd: StateFlow<Ootd?> get() =
16	_selectedOotd
17	init {
18	_ootdList.value = Ootd.ootdList
19	Log.d("OotdViewModel", "Data dimuat ke dalam list:
20	\${_ootdList.value.size} item")
21	}
22	fun selectOotd(ootd: Ootd) {
23	_selectedOotd.value = ootd
24	Log.d("OotdViewModel", "Item dipilih:
25	\${ootd.title} (\${ootd.gender})")
26	}
	}

7. OotdView ModelFactory.kt

Tabel 7 Source Code Jawaban Soal 1

1	package com.example.modul4
2	
3	import androidx.lifecycle.ViewModel
4	import androidx.lifecycle.ViewModelProvider
5	
6	class OotdViewModelFactory : ViewModelProvider.Factory {
7	override fun <T : ViewModel> create(modelClass:
8	Class<T>): T {
9	if
10	(modelClass.isAssignableFrom(OotdViewModel::class.java)) {
11	return OotdViewModel() as T
12	}
13	throw IllegalArgumentException("Unknown ViewModel
14	class")
	}
	}

8. activity_main.xml

Tabel 8 Source Code Jawaban Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<FrameLayout
	xmlns:android="http://schemas.android.com/apk/res/android"
3	xmlns:app="http://schemas.android.com/apk/res-auto"
4	android:layout_width="match_parent"
5	android:layout_height="match_parent"
6	android:background="@color/background">
7	
8	
9	<androidx.fragment.app.FragmentContainerView
10	android:id="@+id/nav_host_fragment"
11	android:name="androidx.navigation.fragment.NavHostFragment"
12	android:layout_width="match_parent"
13	android:layout_height="match_parent"
14	app:defaultNavHost="true"
15	app:navGraph="@navigation/nav_graph" />
	</FrameLayout>

9. Fragment_detail.xml

Tabel 9 Source Code Jawaban Soal 1

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:layout_width="match_parent"
5	android:layout_height="match_parent"
6	android:background="@color/background"
7	android:padding="16dp">
8	
9	<androidx.constraintlayout.widget.ConstraintLayout
10	android:layout_width="match_parent"
11	android:layout_height="wrap_content">
12	
13	<ImageView
14	android:id="@+id/ootdImage"
15	android:layout_width="match_parent"
16	android:layout_height="500dp"
17	android:background="@color/gray_300"
18	android:scaleType="centerCrop"
19	app:layout_constraintTop_toTopOf="parent" />
20	
21	<TextView
22	android:id="@+id/ootdTitle"

```

23         android:layout_width="wrap_content"
24         android:layout_height="wrap_content"
25         android:layout_marginTop="16dp"
26         android:textColor="@color/text_primary"
27
28         android:textAppearance="@style/TextAppearance.AppCompat.Headline"
29         app:layout_constraintStart_toStartOf="parent"
30         app:layout_constraintTop_toBottomOf="@id/oofdImage"
31     />
32
33     <TextView
34         android:id="@+id/oofdGender"
35         android:layout_width="wrap_content"
36         android:layout_height="wrap_content"
37         android:layout_marginStart="8dp"
38         android:textColor="@color/text_secondary"
39
40         android:textAppearance="@style/TextAppearance.AppCompat.Subhead"
41         app:layout_constraintBaseline_toBaselineOf="@id/oofdTitle"
42         app:layout_constraintStart_toEndOf="@id/oofdTitle" />
43
44     <TextView
45         android:id="@+id/oofdStyle"
46         android:layout_width="wrap_content"
47         android:layout_height="wrap_content"
48         android:layout_marginTop="8dp"
49         android:textColor="@color/text_secondary"
50
51         android:textAppearance="@style/TextAppearance.AppCompat.Subhead"
52         app:layout_constraintStart_toStartOf="@id/oofdTitle"
53         app:layout_constraintTop_toBottomOf="@id/oofdTitle"
54     />
55
56     <TextView
57         android:id="@+id/oofdDescription"
58         android:layout_width="match_parent"
59         android:layout_height="wrap_content"
60         android:layout_marginTop="16dp"
61         android:textColor="@color/text_secondary"
62
63         android:textAppearance="@style/TextAppearance.AppCompat.Body1"
64         app:layout_constraintTop_toBottomOf="@id/oofdStyle"
65     />
66
67     <Button
68         android:id="@+id/btnShop"
69         android:layout_width="match_parent"
70         android:layout_height="wrap_content"
71         android:layout_marginTop="24dp"
72         android:text="@string/shop_style_button"
73         style="@style/Widget.OOTDList.Button"

```

66	
67	app:layout_constraintTop_toBottomOf="@id/ootdDescription" />
68	</androidx.constraintlayout.widget.ConstraintLayout>
69	</ScrollView>

10. Fragment_list.xml

Tabel 10 Source Code Jawaban Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	xmlns:app="http://schemas.android.com/apk/res-auto"
5	android:layout_width="match_parent"
6	android:layout_height="match_parent"
7	android:background="@color/background">
8	
9	<androidx.recyclerview.widget.RecyclerView
10	android:id="@+id/recyclerView"
11	android:layout_width="match_parent"
12	android:layout_height="match_parent"
13	android:clipToPadding="false"
14	android:padding="8dp"
15	app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
16	android:contentDescription="@string/recycler_content_description"
17	app:layout_constraintBottom_toBottomOf="parent"
18	app:layout_constraintEnd_toEndOf="parent"
19	app:layout_constraintStart_toStartOf="parent"
20	app:layout_constraintTop_toTopOf="parent" />
21	
22	<TextView
23	android:id="@+id/emptyView"
24	android:layout_width="match_parent"
25	android:layout_height="500dp"
26	android:gravity="center"
27	android:text="@string/empty_list_message"
28	android:visibility="gone"
29	app:layout_constraintBottom_toBottomOf="parent"
30	app:layout_constraintEnd_toEndOf="parent"
31	app:layout_constraintStart_toStartOf="parent"
32	app:layout_constraintTop_toTopOf="parent" />
33	</androidx.constraintlayout.widget.ConstraintLayout>

11. item_ootd.xml

Tabel 11 Source Code Jawaban Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.cardview.widget.CardView
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	xmlns:app="http://schemas.android.com/apk/res-auto"
5	android:layout_width="match_parent"
6	android:layout_height="wrap_content"
7	android:layout_margin="8dp"
8	style="@style/CardView.OOTD">
9	
10	<androidx.constraintlayout.widget.ConstraintLayout
11	android:layout_width="match_parent"
12	android:layout_height="wrap_content"
13	android:padding="8dp">
14	
15	<ImageView
16	android:id="@+id/ootdImage"
17	android:layout_width="match_parent"
18	android:layout_height="150dp"
19	android:scaleType="centerCrop"
20	android:background="@color/gray_300"
21	android:contentDescription="@string/ootd_image_content_description"
22	app:layout_constraintTop_toTopOf="parent" />
23	<TextView
24	android:id="@+id/ootdTitle"
25	android:layout_width="wrap_content"
26	android:layout_height="wrap_content"
27	android:layout_marginTop="8dp"
28	android:textColor="@color/text_primary"
29	android:textAppearance="@style/TextAppearance.AppCompat.Subhead"
30	app:layout_constraintStart_toStartOf="parent"
31	app:layout_constraintTop_toBottomOf="@id/ootdImage" />
32	<TextView
33	android:id="@+id/ootdGender"
34	android:layout_width="wrap_content"
35	android:layout_height="wrap_content"
36	android:layout_marginStart="8dp"
37	android:textColor="@color/text_secondary"
38	android:textAppearance="@style/TextAppearance.AppCompat.Caption"
39	app:layout_constraintBaseline_toBaselineOf="@id/ootdTitle"
40	app:layout_constraintStart_toEndOf="@id/ootdTitle" />
	<TextView

```

41         android:id="@+id/ootdStyle"
42         android:layout_width="wrap_content"
43         android:layout_height="wrap_content"
44         android:layout_marginTop="4dp"
45         android:textColor="@color/text_secondary"
46         android:textAppearance="@style/TextAppearance.AppCompat.Body1"
47         app:layout_constraintStart_toStartOf="@id/ootdTitle"
48         app:layout_constraintTop_toBottomOf="@id/ootdTitle" />
49
50     <TextView
51         android:id="@+id/ootdDescription"
52         android:layout_width="match_parent"
53         android:layout_height="wrap_content"
54         android:layout_marginTop="4dp"
55         android:textColor="@color/text_secondary"
56         android:maxLines="2"
57         android:ellipsize="end"
58         android:textAppearance="@style/TextAppearance.AppCompat.Body2"
59         app:layout_constraintTop_toBottomOf="@id/ootdStyle" />
60
61     <LinearLayout
62         android:layout_width="match_parent"
63         android:layout_height="wrap_content"
64         android:layout_marginTop="8dp"
65         android:orientation="horizontal"
66         app:layout_constraintTop_toBottomOf="@id/ootdDescription">
67
68         <Button
69             android:id="@+id/btnImdb"
70             android:layout_width="0dp"
71             android:layout_height="wrap_content"
72             android:layout_weight="1"
73             android:text="@string/shop_button"
74             android:layout_marginEnd="4dp"
75             style="@style/Widget.OOTDList.Button" />
76
77         <Button
78             android:id="@+id/btnDetail"
79             android:layout_width="0dp"
80             android:layout_height="wrap_content"
81             android:layout_weight="1"
82             android:text="@string/detail_button"
83             android:layout_marginStart="4dp"
84             style="@style/Widget.OOTDList.Button" />
85     </LinearLayout>
86 </androidx.constraintlayout.widget.ConstraintLayout>
87 </androidx.cardview.widget.CardView>

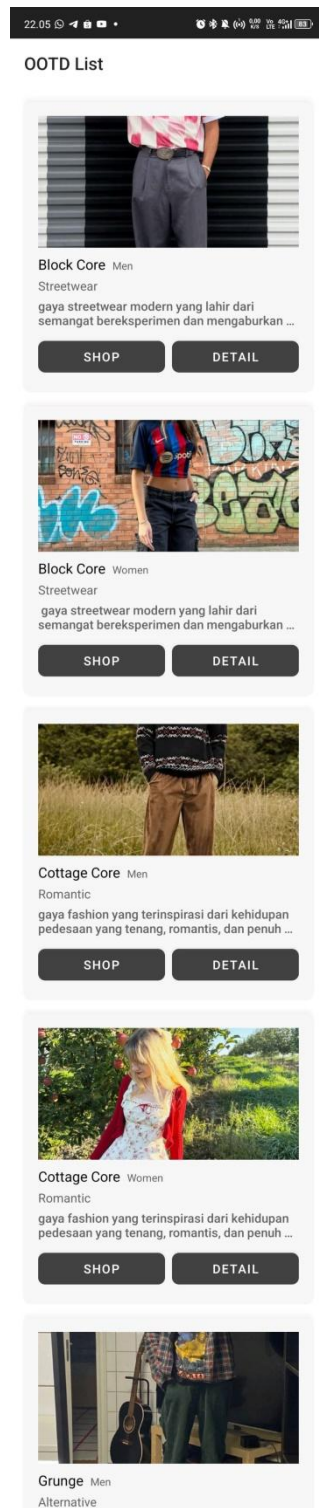
```

12. nav_graph.xml

Tabel 12 Source Code Jawaban Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<navigation
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	xmlns:app="http://schemas.android.com/apk/res-auto"
5	xmlns:tools="http://schemas.android.com/tools"
6	android:id="@+id/nav_graph"
7	app:startDestination="@id/listFragment">
8	<fragment
9	android:id="@+id/listFragment"
10	android:name="com.example.ootdlist.fragments.ListFragment"
	android:label="OOTD List"
	tools:layout="@layout/fragment_list">
11	<action
12	
13	android:id="@+id/action_listFragment_to_detailFragment"
14	app:destination="@id/detailFragment" />
15	</fragment>
16	<fragment
17	android:id="@+id/detailFragment"
18	
19	android:name="com.example.ootdlist.fragments.DetailFragment"
20	tools:layout="@layout/fragment_detail">
21	<argument
22	android:name="ootdId"
23	app:argType="integer" />
24	</argument>
25	</fragment>
26	</navigation>
27	

B. Output Program



Gambar 2 Hasil Tampilan UI List Soal 1



Block Core Men

Streetwear

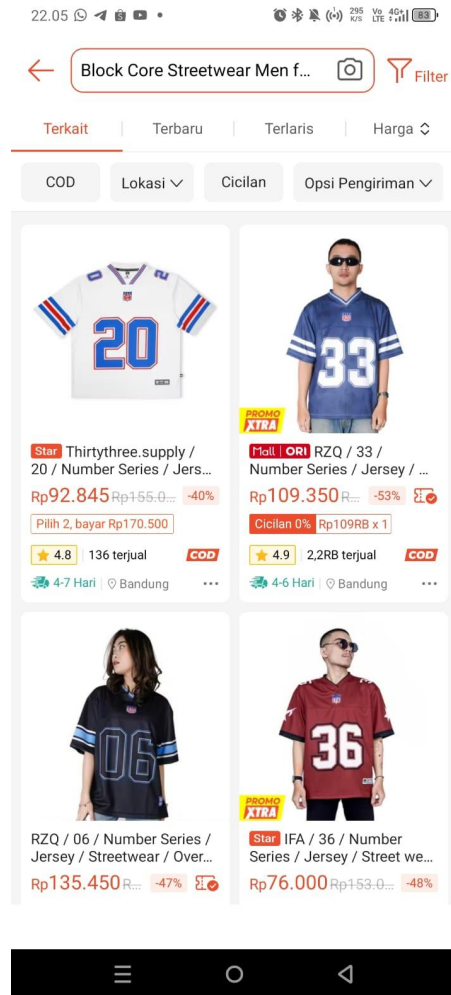
gaya streetwear modern yang lahir dari semangat bereksperimen dan mengaburkan batas antara dunia olahraga dan fashion jalanan. Keunikannya terletak pada penggunaan jersey bola sebagai item utama, menjadikannya lebih dari sekadar simbol fandom —melainkan pernyataan gaya yang berani. Gaya ini merepresentasikan generasi muda yang dinamis, penuh karakter, dan tidak takut memadukan elemen yang tampak kontras. Block Core bukan hanya soal pakaian, tapi juga tentang cara mengekspresikan diri dengan bebas dan orisinal.

Atasan: Jersey klub bola bergaya retro dengan motif kotak merah-putih dan logo "PACHA Ibiza" yang menonjol.

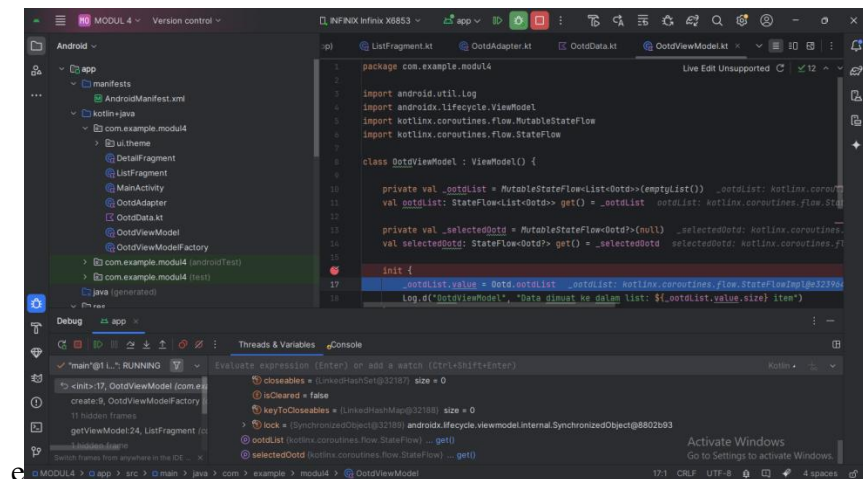
Bawahan: Celana wide leg bahan wol berwarna abu-abu tua, menciptakan kontras menarik antara sporty dan formal.

SHOP THIS STYLE

Gambar 3 Hasil Tampilan UI Detail Soal 1



Gambar 4 Screenshot E-commerce setelah Intent Soal 1



Gambar 5 Screenshot Debugging

C. Pembahasan

1. MainActivity.kt

File ini merupakan titik masuk utama aplikasi Android. Di dalamnya, kelas MainActivity mewarisi AppCompatActivity, yang merupakan salah satu komponen utama dalam Android untuk mendukung kompatibilitas UI lintas versi. Di dalam metode onCreate(), dipanggil setContentView(R.layout.activity_main), yang berarti aplikasi akan menampilkan tampilan (UI) dari activity_main.xml sebagai antarmuka awal. File ini secara struktur sangat sederhana, namun sangat penting karena menginisialisasi layout utama dan menjadi wadah fragment yang akan dimuat, seperti ListFragment dan DetailFragment. Biasanya MainActivity juga menjadi pengatur navigasi fragment menggunakan NavHostFragment jika menggunakan Jetpack Navigation, meskipun tidak eksplisit terlihat di file ini.

2. DetailFragment.kt

File ini adalah fragment yang bertanggung jawab menampilkan detail dari satu item OOTD (Outfit of The Day). DetailFragment menggunakan *View Binding* melalui FragmentDetailBinding untuk mengakses elemen UI secara langsung tanpa perlu findViewById, yang meningkatkan keamanan dan efisiensi kode.

Di dalam metode onCreateView(), fragment menerima argument ootdId yang dikirim dari fragment sebelumnya (kemungkinan ListFragment). ID ini digunakan untuk mengambil objek Ootd tertentu melalui metode Ootd.getId(ootdId). Setelah data OOTD diperoleh, informasi seperti judul, jenis kelamin, gaya, deskripsi, dan gambar akan ditampilkan melalui komponen UI yang telah di-bind.

Menariknya, terdapat fitur interaktif berupa tombol btnShop, yang ketika ditekan akan membuat URL pencarian berdasarkan properti OOTD (judul, gaya, dan gender) ke situs Shopee. Proses ini dilakukan dengan menyusun searchQuery, kemudian mengenkripsi query-nya menggunakan URLEncoder, lalu membuat Intent untuk membuka browser ke URL pencarian tersebut. Ini adalah contoh integrasi sederhana antara aplikasi dan layanan eksternal.

Metode onDestroyView() digunakan untuk menghapus referensi binding agar tidak terjadi memory leak, mengikuti praktik standar pengelolaan memori dalam penggunaan View Binding pada fragment.

3. ListFragment.kt:

ListFragment adalah fragment yang menampilkan daftar semua OOTD yang tersedia dalam bentuk RecyclerView. Sama seperti DetailFragment, fragment ini menggunakan FragmentListBinding untuk efisiensi akses ke komponen UI.

Pada onCreateView, diinisialisasi sebuah OotdAdapter yang menerima data dari Ootd.ootdList, yaitu daftar statis atau dinamis dari semua outfit. Adapter ini menggunakan pola listener interface (OnItemClickListener) yang memiliki dua aksi utama: onDetailClick() dan onShopClick(). Pada onDetailClick, fragment akan melakukan navigasi ke DetailFragment dengan membawa ootdId sebagai argumen menggunakan Jetpack Navigation. Sedangkan pada onShopClick, konsepnya sama seperti pada DetailFragment, yaitu membuat query pencarian berdasarkan detail OOTD dan membuka Shopee melalui Intent ke browser.

RecyclerView dikonfigurasi dengan LinearLayoutManager untuk menampilkan daftar secara vertikal. Terdapat juga logika untuk menangani keadaan jika daftar kosong (ootdList.isEmpty()), yang akan menyembunyikan RecyclerView dan menampilkan emptyView, semacam teks atau ikon placeholder yang menunjukkan bahwa tidak ada data tersedia.

Seperti sebelumnya, metode onDestroyView() digunakan untuk mencegah memory leak dengan menghapus binding saat fragment dihancurkan.

4. OotdAdapter.kt

File OotdAdapter.kt merupakan salah satu komponen utama dalam arsitektur RecyclerView Adapter yang bertugas menghubungkan data outfit (OOTD) dengan tampilan daftar item di UI. Kelas OotdAdapter menerima dua parameter utama: ootdList, yaitu daftar objek Ootd, dan listener, yang merupakan antarmuka.OnItemClickListener untuk menangani aksi klik dari pengguna. Adapter ini mewarisi RecyclerView.Adapter dan di dalamnya didefinisikan kelas OotdViewHolder yang bertugas untuk mem-bind setiap elemen data ke tampilan item yang sudah ditentukan dalam item_ootd.xml.

Pada metode bind(), setiap properti dari objek Ootd seperti title, gender, style, description, dan imageResId ditampilkan ke dalam elemen visual seperti TextView dan ImageView. Tombol btnDetail dan btnImdb (yang seharusnya secara konteks bisa diubah menjadi btnShop) masing-masing memiliki click listener yang memanggil fungsi dari OnItemClickListener. Ini menunjukkan bahwa adapter ini juga menangani interaksi pengguna, seperti ketika ingin melihat detail atau menuju toko. Selain itu, adapter juga menerapkan konten deskriptif gambar (contentDescription) demi mendukung aksesibilitas.

Metode onCreateViewHolder() berfungsi untuk meng-inflate layout item dari file XML dan mengembalikan OotdViewHolder. Sedangkan

onBindViewHolder() memanggil fungsi bind() untuk memasukkan data OOTD berdasarkan posisi. Metode getItemCount() cukup sederhana, mengembalikan jumlah total data yang akan ditampilkan.

5. OotdData.kt

File OotdData.kt berisi deklarasi data model Ootd dan data source statis yang mewakili berbagai gaya fashion. Kelas Ootd menggunakan anotasi data class yang secara otomatis menyediakan fungsi seperti toString(), equals(), dan copy(). Setiap objek Ootd memiliki properti: id, title, style, description, imageResId, dan gender. Informasi ini mewakili detail outfit termasuk judul tampilan, kategori gaya (misalnya Streetwear, Romantic, Alternative), deskripsi naratif yang mendalam, serta referensi ke resource gambar.

Objek-objek Ootd dikelompokkan ke dalam daftar statis ootdList yang berfungsi sebagai data dummy atau konten statis dalam aplikasi. Masing-masing entri menggambarkan gaya fashion spesifik, lengkap dengan narasi inspiratif serta penjelasan atasan dan bawahan, yang sangat berguna untuk menciptakan pengalaman pengguna yang imersif dan edukatif. Ada berbagai gaya seperti Block Core, Cottage Core, Grunge, Japanese Americana, Old Money, dan Y2K, yang semuanya menjelaskan elemen fashion yang populer pada segmennya.

Fungsi getById() disediakan untuk mencari OOTD berdasarkan ID-nya, sangat berguna ketika aplikasi perlu menampilkan detail item tertentu saat pengguna memilih dari daftar.

6. OotdViewModel.kt

File OotdViewModel.kt ini berisi class OotdViewModel yang merupakan turunan dari ViewModel. ViewModel ini bertugas sebagai jembatan antara UI dan data yang ditampilkan, supaya data tetap terjaga meskipun terjadi perubahan konfigurasi seperti rotasi layar. Di dalamnya terdapat dua properti MutableStateFlow yaitu _ootdList dan _selectedOotd. _ootdList menyimpan list data outfit (OOTD) yang nanti akan ditampilkan di UI, sedangkan _selectedOotd menyimpan data OOTD yang sedang dipilih. Untuk menjaga encapsulation, kedua variabel tersebut dibungkus dengan StateFlow agar hanya bisa diakses dari luar tanpa bisa diubah secara langsung. Pada bagian init, list OOTD langsung diisi dari Ootd.ootdList dan dicetak ke log jumlah datanya. Ada juga fungsi selectOotd() yang dipanggil saat user memilih salah satu OOTD, lalu log akan menampilkan item mana yang dipilih berdasarkan judul dan gender-nya. File ini menunjukkan implementasi reactive programming menggunakan Kotlin Flow dan ViewModel.

7. OotdViewModelFactory.kt

File `OotdViewModelFactory.kt` berisi class `OotdViewModelFactory` yang mengimplementasikan `ViewModelProvider.Factory`. Fungsi utamanya adalah untuk membuat instance dari `OotdViewModel`. Factory ini digunakan saat kita ingin membuat `ViewModel` secara manual, terutama kalau `ViewModel`-nya butuh parameter khusus atau untuk menjaga fleksibilitas pembuatan `ViewModel`. Pada override method `create`, terdapat pengecekan apakah `modelClass` yang diminta sesuai dengan class `OotdViewModel`, jika iya maka akan dikembalikan instance-nya. Jika tidak cocok, maka akan dilempar exception `IllegalArgumentException`. Penggunaan `ViewModelFactory` ini penting untuk memisahkan logika pembuatan `ViewModel` dari UI, sehingga kode lebih rapi dan mudah diuji.

8. activity_main.xml

File `activity_main.xml` merupakan file tata letak utama (main layout) dari aplikasi yang mendefinisikan struktur dasar UI dengan menggunakan `FrameLayout`. Di dalamnya, terdapat komponen `FragmentContainerView` yang berfungsi sebagai host dari navigasi berbasis fragment. View ini menggunakan `NavHostFragment` dari Jetpack Navigation Component untuk mengelola transisi antar fragment berdasarkan navigation graph (`nav_graph.xml`).

Atribut `app:defaultNavHost="true"` memastikan bahwa fragment ini akan menjadi host utama untuk navigasi (misalnya saat menekan tombol back). Atribut `app:navGraph="@navigation/nav_graph"` menunjukkan file navigasi XML yang berisi arah dan struktur perpindahan fragment. Tata letak ini juga memiliki latar belakang yang ditentukan melalui warna `@color/background`, memberikan kesan visual yang konsisten di seluruh aplikasi.

9. fragment_detail.xml

File `fragment_detail.xml` adalah layout tampilan detail untuk setiap OOTD yang dipilih. Layout dibungkus dalam `ScrollView`, memungkinkan pengguna menggulir jika kontennya panjang. Di dalamnya terdapat `ConstraintLayout` yang mengatur elemen-elemen UI secara fleksibel dan responsif. Elemen-elemen yang ditampilkan mencakup `ImageView` (`ootdImage`) untuk gambar outfit, serta `TextView` untuk `ootdTitle`, `ootdGender`, `ootdStyle`, dan `ootdDescription`. Semua teks menggunakan `TextAppearance` untuk menjaga konsistensi desain dan tema aplikasi.

Di bagian akhir terdapat tombol `btnShop`, yang memungkinkan pengguna melakukan aksi lanjut seperti membuka halaman belanja atau eksternal link ke produk serupa. Tata letak ini memberikan pengalaman detail yang informatif dan estetis bagi pengguna untuk memahami outfit secara menyeluruh. Gaya desainnya bersih, rapi, dan memprioritaskan keterbacaan serta hirarki informasi.

10. fragment_list.xml

File `fragment_list.xml` merupakan layout XML untuk tampilan utama dari `ListFragment`, yaitu sebuah fragment yang menampilkan daftar item OOTD (Outfit of The Day) dalam aplikasi. Layout ini menggunakan `ConstraintLayout` sebagai root layout, yang memungkinkan setiap elemen UI diposisikan dengan fleksibel dan efisien berdasarkan constraint antar elemen. Di dalamnya terdapat dua elemen utama: `RecyclerView` dan `TextView`.

`RecyclerView` dengan ID `@+id/recyclerView` berfungsi untuk menampilkan daftar OOTD secara vertikal menggunakan `LinearLayoutManager`. Atribut `clipToPadding="false"` dan padding sebesar 8dp memastikan tampilan daftar lebih lega dan tidak terpotong. `RecyclerView` ini akan diisi oleh adapter yang mengatur item-item OOTD. Sebagai fallback, `TextView` dengan ID `@+id/emptyView` disediakan dan akan ditampilkan ketika daftar kosong. `TextView` ini menampilkan pesan dari string resource `@string/empty_list_message`, dan secara default disembunyikan (`android:visibility="gone"`), hanya muncul saat diperlukan. Layout ini sangat mendukung prinsip *responsive UI*, karena semua elemen menggunakan constraint terhadap parent, memastikan kompatibilitas pada berbagai ukuran layar.

11. item_ootd.xml

File ini adalah layout dari setiap item OOTD yang akan ditampilkan dalam `RecyclerView`. Root layout-nya menggunakan `CardView`, yang memberikan efek bayangan dan sudut membulat, menciptakan tampilan yang lebih menarik dan modern. Selanjutnya, `ConstraintLayout` digunakan untuk mengatur tata letak elemen-elemen di dalam card tersebut.

Elemen utama dari layout ini adalah `ImageView` (ID: `ootdImage`) yang menampilkan gambar OOTD dengan tinggi tetap 150dp dan properti `centerCrop` agar gambar proporsional memenuhi area tampilan. Kemudian terdapat beberapa `TextView`, masing-masing untuk menampilkan judul outfit (`ootdTitle`), jenis kelamin target outfit (`ootdGender`), gaya berpakaian (`ootdStyle`), dan deskripsi singkat (`ootdDescription`). Deskripsi dibatasi maksimal dua baris dengan `ellipsize="end"` untuk menjaga tampilan tetap rapi.

Pada bagian bawah terdapat `LinearLayout` horizontal yang memuat dua tombol (`Button`). Tombol pertama (`btnImdb`) biasanya digunakan untuk mengarahkan ke halaman toko atau e-commerce, dan tombol kedua (`btnDetail`) digunakan untuk membuka detail lengkap dari OOTD yang dipilih. Masing-masing tombol memiliki gaya yang konsisten sesuai style `Widget.OOTDList.Button`.

Dengan struktur ini, tiap item OOTD tidak hanya informatif secara visual, tetapi juga interaktif, memungkinkan pengguna mengambil tindakan langsung.

12. nav_graph.xml

File `nav_graph.xml` adalah bagian dari Android Jetpack Navigation Component yang mendefinisikan struktur navigasi antar fragment dalam aplikasi. Di dalam file ini terdapat dua fragment utama: `listFragment` dan `detailFragment`. `listFragment` berfungsi sebagai *startDestination*, artinya ketika aplikasi dijalankan, pengguna akan langsung diarahkan ke fragment ini yang menampilkan daftar OOTD.

Dari `listFragment`, terdapat satu action menuju `detailFragment`, yaitu `action_listFragment_to_detailFragment`. Action ini memungkinkan pengguna berpindah ke tampilan detail dari item yang dipilih. Sementara itu, `detailFragment` didefinisikan dengan satu argument bernama `ootdId` bertipe integer, yang akan digunakan untuk mengambil dan menampilkan data detail berdasarkan ID yang dikirim dari fragment sebelumnya. Konfigurasi ini memungkinkan navigasi antar UI dilakukan dengan mudah dan aman, karena argument telah diketik secara eksplisit (type-safe).

13. jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out

Debugger itu kayak detektif buat ngubugin kode. Fungsinya buat ngecek alur program, ngelihat nilai variabel pas runtime, dan nyari di mana kesalahan terjadi. Jadi, kita bisa nge-pause kode di titik tertentu (breakpoint) terus ngelakuin investigasi.

Cara pake debugger di Android Studio:

1. **Set breakpoint:** Klik area sebelah kiri nomor baris (nanti muncul titik merah).
2. **Jalankan debug mode:** Klik tombol debug (yang ada icon kumbang).
3. **Pantau eksekusi:** Pas kode berhenti di breakpoint, kita bisa liat nilai variabel di panel *Variables* atau *Watches*.

Fitur penting debugger:

- **Step Into (F7):** Masuk ke dalam fungsi/method buat ngecek isinya. Misal, kalo ada `calculateTotal()`, kita bisa masuk ke dalam fungsi itu.
- **Step Over (F8):** Lanjut ke baris selanjutnya *tanpa* masuk ke fungsi. Berguna kalo kita gak perlu ngecek dalam fungsi itu.

- **Step Out (Shift + F8):** Keluar dari fungsi sekarang dan balik ke pemanggilnya. Cocok kalo udah selesai ngecek suatu fungsi.

2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

Application class di Android adalah kelas inti yang mewakili seluruh aplikasi dan hidup selama aplikasi berjalan. Fungsinya sebagai tempat inisialisasi komponen global (seperti database, library networking), menyimpan data yang perlu diakses di banyak tempat (misalnya token autentikasi), dan menangani callback siklus hidup aplikasi. Cara pakainya dengan membuat subclass dari `android.app.Application` lalu mendaftarkannya di `AndroidManifest.xml`. Contohnya, ketika menggunakan Room Database

D. Tautan Git

[illegible]