

**LAPORAN PRAKTIKUM  
PEMROGRAMAN MOBILE  
MODUL 4**



**VIEWMODEL DAN DEBUGGING**

**Oleh:**

**Aiko Anatasha Wendiono**

**NIM. 2310817320013**

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

**LEMBAR PENGESAHAN**  
**LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE**  
**MODUL 4**

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

Nama Praktikan : Aiko Anatasha Wendiono  
NIM : 2310817320013

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

LEMBAR PENGESAHAN .....	2
DAFTAR ISI .....	3
DAFTAR GAMBAR.....	4
DAFTAR TABEL .....	5
SOAL 1 .....	6
A. Source Code.....	7
B. Output Program .....	20
C. Pembahasan .....	21
D. Tautan Git .....	28

## DAFTAR GAMBAR

Gambar 1. Contoh Penggunaan Debugger .....	7
Gambar 2. Screenshot Hasil Jawaban Soal 1 .....	20
Gambar 3. Screenshot Hasil Jawaban Soal 1 .....	21
Gambar 4. Screenshot Kode ViewModel .....	22
Gambar 5. Screenshot Kode ViewModelFactory .....	22
Gambar 6. Screenshot Kode StateFlow .....	23
Gambar 7. Screenshot Tampilan Logcat .....	23
Gambar 8. Screenshot Kode yang menggunakan Log .....	23
Gambar 9. Screenshot Kode yang menggunakan Log .....	24
Gambar 10. Screenshot Kode yang menggunakan Log .....	24
Gambar 11. Screenshot pada saat Debugging .....	25
Gambar 12. Screenshot Fitur Step Into .....	26
Gambar 13. Screenshot Fitur Step Into .....	26
Gambar 14. Screenshot Fitur Step Over .....	26
Gambar 15. Screenshot Fitur Step Into .....	27

## DAFTAR TABEL

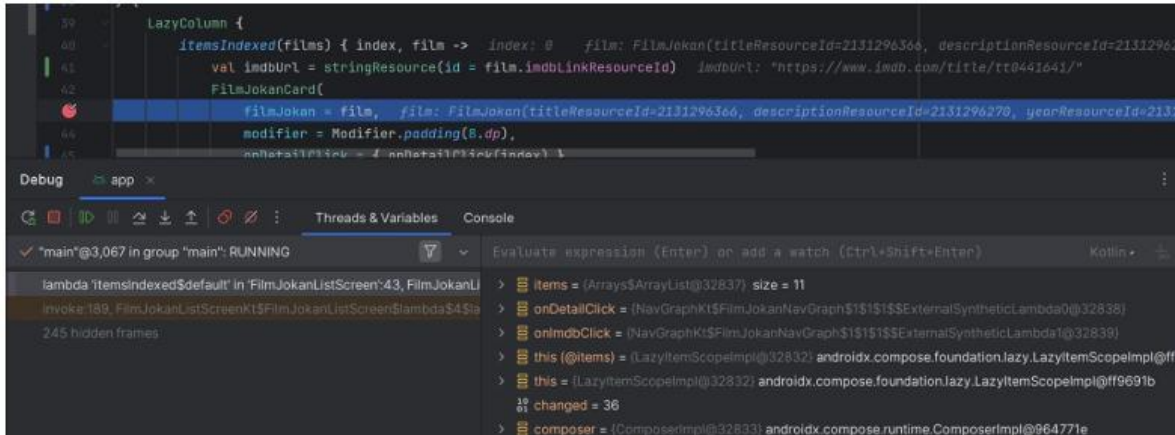
Tabel 1. Source Code Jawaban Soal 1.....	8
Tabel 2. Source Code Jawaban Soal 1.....	8
Tabel 3. Source Code Jawaban Soal 1.....	8
Tabel 4. Source Code Jawaban Soal 1.....	12
Tabel 5. Source Code Jawaban Soal 1.....	15
Tabel 6. Source Code Jawaban Soal 1.....	18
Tabel 7. Source Code Jawaban Soal 1.....	18
Tabel 8. Source Code Jawaban Soal 1.....	19

## SOAL 1

### Soal Praktikum:

1. Lanjutkan aplikasi Android berbasis XML atau 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 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

```

1 package com.example.mlcharacters
2
3 import android.os.Bundle
4 import androidx.activity.ComponentActivity
5 import androidx.activity.compose.setContent
6 import androidx.compose.material3.MaterialTheme
7 import androidx.compose.material3.Surface
8 import androidx.lifecycle.viewmodel.compose.viewModel
9 import
10 androidx.navigation.compose.rememberNavController
11 import com.example.mlcharacters.ui.CharacterViewModel
12 import
13 com.example.mlcharacters.ui.CharacterViewModelFactory
14 import
15 com.example.mlcharacters.ui.theme.MLCharactersTheme
16
17 class MainActivity : ComponentActivity() {
18     override fun onCreate(savedInstanceState: Bundle?)
19     {
20         super.onCreate(savedInstanceState)
21         setContent {
22             MLCharactersTheme {
23                 Surface(color
24 MaterialTheme.colorScheme.background) {
25                     val
26 rememberNavController()
27

```

28	val viewModel: CharacterViewModel	
29	= viewModel(	
30	factory	=
31	CharacterViewModelFactory()	
32	)	
33		
34	NavGraph(navController	=
35	navController, viewModel = viewModel)	
36	}	
37	}	
38	}	
39	}	
40	}	

Tabel 1. Source Code Jawaban Soal 1

## 2. Character

1	package com.example.mlcharacters
2	
3	data class Character(
4	val name: String,
5	val alias: String,
6	val imageRes: Int,
7	val description: String,
8	val wikiUrl: String
9	)

Tabel 2. Source Code Jawaban Soal 1

## 3. MLCharacterApplication.kt

1	package com.example.mlcharacters
2	
3	import android.app.Application
4	
5	class MLCharactersApplication : Application() {
6	override fun onCreate() {
7	super.onCreate()
8	}
9	}
10	}

Tabel 3. Source Code Jawaban Soal 1

## 4. CharacterListScreen.kt

1	package com.example.mlcharacters
2	
3	import android.content.Intent
4	import android.net.Uri
5	import androidx.compose.foundation.Image
6	import androidx.compose.foundation.layout.*



```

7 import androidx.compose.foundation.lazy.LazyColumn
8 import androidx.compose.foundation.lazy.items
9 import
10 androidx.compose.foundation.shape.RoundedCornerShape
11 import androidx.compose.material3.*
12 import androidx.compose.runtime.Composable
13 import androidx.compose.runtime.collectAsState
14 import androidx.compose.runtime.getValue
15 import androidx.compose.ui.Alignment
16 import androidx.compose.ui.Modifier
17 import androidx.compose.ui.graphics.Color
18 import androidx.compose.ui.layout.ContentScale
19 import androidx.compose.ui.platform.LocalContext
20 import androidx.compose.ui.res.painterResource
21 import androidx.compose.ui.text.font.FontWeight
22 import androidx.compose.ui.unit.dp
23 import androidx.navigation.NavController
24 import
25 com.example.mlcharacters.ui.CharacterViewModel
26
27 @OptIn(ExperimentalMaterial3Api::class)
28 @Composable
29 fun CharacterListScreen(
30     navController: NavController,
31     viewModel: CharacterViewModel
32 ) {
33     val characters by
34     viewModel.characterList.collectAsState()
35     val context = LocalContext.current
36
37     LazyColumn(
38         modifier = Modifier
39             .fillMaxSize()
40             .padding(8.dp),
41         contentPadding = PaddingValues(top = 60.dp),
42         verticalArrangement =
43     Arrangement.spacedBy(12.dp)
44     ) {
45         items(characters) { character ->
46             Card(
47                 shape = RoundedCornerShape(12.dp),
48                 colors =
49     CardDefaults.cardColors(containerColor =
50     Color(0xFF2B2B2B)),
51                 elevation =
52     CardDefaults.cardElevation(4.dp),
53                 modifier = Modifier.fillMaxWidth()

```

```

54         ) {
55             Row(modifier =
56 Modifier.padding(12.dp)) {
57                 Image(
58                     painter = painterResource(id
59 = character.imageRes),
60                     contentDescription =
61 character.name,
62                     contentScale =
63 ContentScale.Crop,
64                     modifier = Modifier
65                         .width(90.dp)
66                         .height(140.dp)
67                         .padding(end = 12.dp)
68                 )
69
70                 Column(modifier =
71 Modifier.weight(1f)) {
72                     Row(
73                         modifier =
74 Modifier.fillMaxWidth(),
75                         horizontalArrangement =
76 Arrangement.SpaceBetween
77                     ) {
78                         Text(
79                             text =
80 character.name,
81                             style =
82 MaterialTheme.typography.titleMedium,
83                             color = Color.White,
84                             fontWeight =
85 FontWeight.Bold
86                         )
87                         Text(
88                             text = "2016", //
89 Masih dummy, bisa pakai data jika tersedia
90                             style =
91 MaterialTheme.typography.labelMedium,
92                             color =
93 Color.LightGray
94                         )
95                     }
96
97                     Spacer(modifier =
98 Modifier.height(6.dp))
99
100                    Row(

```

```

101 modifier =
102 Modifier.fillMaxWidth(),
103 verticalAlignment =
104 Alignment.Top
105 ) {
106     Text(
107         text = "Deskripsi: ",
108         color = Color.White,
109         fontWeight =
110         FontWeight.Bold,
111         style =
112         MaterialTheme.typography.bodyMedium
113     )
114
115     Spacer(modifier =
116     Modifier.width(8.dp))
117
118     Text(
119         text =
120         character.description,
121         color = Color.White,
122         style =
123         MaterialTheme.typography.bodySmall,
124         modifier =
125         Modifier.weight(1f)
126     )
127 }
128
129     Spacer(modifier =
130     Modifier.height(10.dp))
131
132     Row(horizontalArrangement =
133     Arrangement.spacedBy(8.dp)) {
134         Button(
135             onClick = {
136
137             viewModel.logItemClick(character, "Detail Hero")
138                 val intent =
139                 Intent(Intent.ACTION_VIEW,
140                 Uri.parse(character.wikiUrl))
141
142             context.startActivity(intent)
143             },
144             colors =
145             ButtonDefaults.buttonColors(containerColor =
146             Color(0xFF9BB1EB)),
147

```

```

148                                     modifier =
149 Modifier.weight(1f)
150                                 ) {
151                                     Text("Detail Hero",
152 color = Color.White)
153                                 }
154
155                                 Button(
156                                     onClick = {
157
158 viewModel.logItemClick(character, "Deskripsi")
159 navController.navigate("detail/${character.name}")
160                                     },
161                                     colors =
162 ButtonDefaults.buttonColors(containerColor =
163 Color(0xFF9BB1EB)),
164                                     modifier =
165 Modifier.weight(1f)
166                                 ) {
167                                     Text("Deskripsi",
168 color = Color.White)
169                                 }
170                             }
171                         }
172                     }
173                 }
174             }
        }
    }
}

```

---

Tabel 4. Source Code Jawaban Soal 1

## 5. NavGraph.kt

```
1 package com.example.mlcharacters
2
3 import androidx.compose.runtime.Composable
4 import androidx.navigation.NavHostController
5 import androidx.navigation.compose.NavHost
6 import androidx.navigation.compose.composable
7 import com.example.mlcharacters.ui.CharacterViewModel
8 import
9 com.example.mlcharacters.ui.theme.CharacterDetailScreen
10
11
12 @Composable
13 fun NavGraph(navController: NavHostController,
14 viewModel: CharacterViewModel) {
```

```

15     val characters = listOf(
16         Character(
17             "Crocell", "Mage, Order: Light Equipment",
18             R.drawable.crocell,
19             "Role: Mage\nType: Order - Light
20 Equipment\n\nSaya menyukai hero ini karena tampilannya
21 yang sangat cantik dan positif dan voicenya yang ceria,
22 menarik serta lucu. Saya paling menyukai voice line
23 ketika menang, 'Horay, we won!'. Ultimate Crocell juga
24 berguna untuk team fight dan efek ultimatanya terang dan
25 bagus, seperti bintang yang jatuh di langit dengan
26 sukacita.",
27             "https://mla.fandom.com/wiki/Crocell"
28         ),
29         Character(
30             "Astraia Sipra", "Support, Hybrid, Light -
31 Tech: Light Equipment", R.drawable.sipra,
32             "Role: Support\nType: Hybrid (Light - Tech)
33 - Light Equipment\n\nDikenal dengan panggilan Sipra.
34 Hero ini juga termasuk dalam favorit saya karena
35 ultimatanya yang sangat berguna untuk war. Bahkan Sipra
36 B4 bisa mengimbangi war hero yang sudah awakened.
37 Ultimatanya adalah meng-summon hero secara acak.",
38             "https://mla.fandom.com/wiki/Astraia_Sipra"
39         ),
40         Character(
41             "Nana", "Spirit, Martial: Light Equipment",
42             R.drawable.nana,
43             "Role: Spirit\nType: Martial - Light
44 Equipment\n\nNana di Mobile Legends Bang Bang dengan
45 Mobile Legends Adventure sangat berbeda. Jujur ketika
46 saya pertama kali memakai hero ini, skill nya sangat
47 berbeda dengan yang versi moba nya. Skill 2 disini adalah
48 memberi heal pada tim dan ultinya membangkitkan hero
49 yang sudah mati.",
50             "https://mla.fandom.com/wiki/Nana"
51         ),
52         Character(
53             "Mecha Layla", "Marksman, Hybrid (Light -
54 Tech): Light Equipment", R.drawable.mechalayla,
55             "Role: Marksman\nType: Hybrid (Light - Tech)
56 - Light Equipment\n\nLayla di MLA ada 2 versi. Mecha
57 Layla memberi heal dan damage secara terus-menerus. Tapi
58 jika semua hero tim mati, Mecha Layla juga ikut lenyap.",
59             "https://mla.fandom.com/wiki/Mecha_Layla"
60         ),
61     )

```

```

62         Character(
63             "Natsu", "Fighter, Hybrid (Dark - Martial):
64 Medium Equipment", R.drawable.natsu,
65             "Role: Fighter\nType:Hybrid (Dark -
66 Martial) - Medium Equipment\n\nNatsu hero kolaborasi
67 Fairy Tail. Punya skill dorong musuh ke kanan. Menurut
68 saya sangat berguna saat team fight, apalagi jika
69 dikombo dengan AoE seperti Alice.",
70             "https://mla.fandom.com/wiki/Natsu"
71         ),
72         Character(
73             "Naiad Rafaela", "Hybrid (Light -
74 Elemental): Light Equipment", R.drawable.rafaela,
75             "Role:Support\nType: Hybrid (Light -
76 Elemental) - Light Equipment\n\nDesain sangat anggun dan
77 menyatu dengan alam. Hero ini saya sukai karena
78 estetikanya meskipun belum terlalu dipakai untuk war.",
79             "https://mla.fandom.com/wiki/Naiad_Rafaela"
80         ),
81         Character(
82             "Shah Torre", "Mage, Chaos: Light
83 Equipment", R.drawable.torre,
84             "Role: Mage\nType: Chaos - Light
85 Equipment\n\nShah Torre menyerap bar energi musuh,
86 sehingga bisa menunda ultimate lawan. Sangat menyebalkan
87 jika menjadi musuh.",
88             "https://mla.fandom.com/wiki/Shah_Torre"
89         ),
90         Character(
91             "Forseti", "Marksman, Order: Light
92 Equipment", R.drawable.forseti,
93             "Role: Marksman\nType: Order - Light
94 Equipment\n\nUltimatenya membuat Forseti langsung mati.
95 Butuh support respawn seperti Nana atau Lunox. Saya baru
96 sadar potensi hero ini setelah lihat performanya di
97 Tower of Babel.",
98             "https://mla.fandom.com/wiki/Forseti"
99         ),
100     ),
101 )
102
103 viewModel.setCharacterList(characters)
104
105 val charMap = characters.associateBy { it.name }
106
107 NavHost(navController = navController,
108 startDestination = "list") {

```

109	composable("list") {
110	CharacterListScreen(
111	navController = navController,
112	viewModel = viewModel
113	)
114	}
115	composable("detail/{name}") { backStackEntry ->
116	val name =
117	backStackEntry.arguments?.getString("name")
118	val character = charMap[name]
119	character?.let {
120	viewModel.logNavigateDetail(it)
121	CharacterDetailScreen(character = it,
122	navController = navController)
123	}
124	}
125	}
126	}
127	
128	

Tabel 5. Source Code Jawaban Soal 1

## 6. CharacterDetailScreen.kt – ui.theme

1	package com.example.mlcharacters.ui.theme
2	
3	import androidx.compose.foundation.Image
4	import androidx.compose.foundation.layout.*
5	import androidx.compose.foundation.lazy.LazyColumn
6	import androidx.compose.material.icons.Icons
7	import
8	androidx.compose.material.icons.filled.ArrowBack
9	import androidx.compose.material3.*
10	import androidx.compose.runtime.Composable
11	import androidx.compose.ui.Modifier
12	import androidx.compose.ui.graphics.Color
13	import androidx.compose.ui.layout.ContentScale
14	import androidx.compose.ui.res.painterResource
15	import androidx.compose.ui.text.font.FontWeight
16	import androidx.compose.ui.unit.dp
17	import androidx.compose.ui.unit.sp
18	import androidx.navigation.NavController
19	import com.example.mlcharacters.Character
20	
21	@OptIn(ExperimentalMaterial3Api::class)
22	@Composable
23	fun CharacterDetailScreen(character: Character,
24	navController: NavController) {

```

25 Scaffold(
26     containerColor = Color(0xFF1C1C1E),
27     topBar = {
28         CenterAlignedTopAppBar(
29             title = {},
30             navigationIcon = {
31                 IconButton(onClick = {
32 navController.popBackStack() }) {
33                     Icon(
34                         imageVector =
35 Icons.Default.ArrowBack,
36                         contentDescription =
37 "Back",
38                         tint = Color.White
39                     )
40                 }
41             },
42             colors =
43 TopAppBarDefaults.centerAlignedTopAppBarColors(
44                 containerColor =
45 Color.Transparent
46             )
47         )
48     }
49 ) { innerPadding ->
50     LazyColumn(
51         modifier = Modifier
52             .padding(innerPadding)
53             .fillMaxSize()
54     ) {
55         item {
56             Image(
57                 painter = painterResource(id =
58 character.imageRes),
59                 contentDescription =
60 character.name,
61                 contentScale = ContentScale.Crop,
62                 modifier = Modifier
63                     .fillMaxWidth()
64                     .height(500.dp)
65             )
66         }
67
68         item {
69             Column(modifier =
70 Modifier.padding(16.dp)) {
71                 Row(

```



```

72         modifier =
73 Modifier.fillMaxWidth(),
74         horizontalArrangement =
75 Arrangement.SpaceBetween
76     ) {
77         Text(
78             text = character.name,
79             color = Color.White,
80             style =
81 MaterialTheme.typography.titleLarge,
82             fontWeight =
83 FontWeight.Bold
84         )
85         Text(
86             text = "2016",
87             color = Color.LightGray,
88             style =
89 MaterialTheme.typography.bodyMedium
90         )
91     }
92
93     Spacer(modifier =
94 Modifier.height(8.dp))
95
96     Text(
97         text = "Deskripsi:",
98         color = Color.White,
99         fontWeight = FontWeight.Bold,
100        style =
101 MaterialTheme.typography.bodyMedium
102    )
103
104    Spacer(modifier =
105 Modifier.height(4.dp))
106
107    Text(
108        text = character.description,
109        color = Color(0xFFCCCCCC),
110        style =
111 MaterialTheme.typography.bodySmall,
112        lineHeight = 20.sp
113    )
114    }
115 }
116 }
117 }
118 }

```

Tabel 6. Source Code Jawaban Soal 1

**7. CharacterViewModel.kt – ui.theme**

1	package com.example.mlacharacters.ui
2	
3	import android.util.Log
4	import androidx.lifecycle.ViewModel
5	import androidx.lifecycle.viewModelScope
6	import com.example.mlacharacters.Character
7	import kotlinx.coroutines.flow.MutableStateFlow
8	import kotlinx.coroutines.flow.StateFlow
9	import kotlinx.coroutines.launch
10	
11	class CharacterViewModel : ViewModel() {
12	
13	private val _characterList =
14	MutableStateFlow<List<Character>>(emptyList())
15	val characterList: StateFlow<List<Character>> =
16	_characterList
17	
18	fun setCharacterList(characters: List<Character>)
19	{
20	viewModelScope.launch {
21	Log.d("CharacterViewModel", "Data item
22	masuk ke dalam list")
23	_characterList.value = characters
24	}
25	}
26	
27	fun logItemClick(character: Character, action:
28	String) {
29	Log.d("CharacterViewModel", "Tombol \$action
30	ditekan untuk \${character.name}")
31	}
32	
33	fun logNavigateDetail(character: Character) {
34	Log.d("CharacterViewModel", "Navigasi ke
35	detail karakter: \${character.name}")
36	}
37	}

Tabel 7. Source Code Jawaban Soal 1

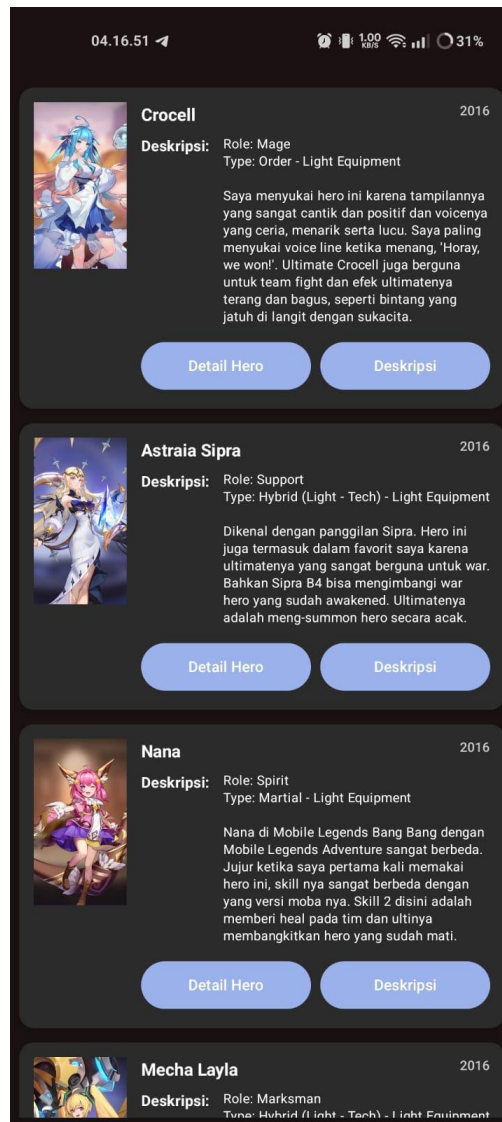
**8. CharacterViewModelFactory – ui.model**

1	package com.example.mlacharacters.ui
2	
3	import androidx.lifecycle.ViewModel
4	import androidx.lifecycle.ViewModelProvider

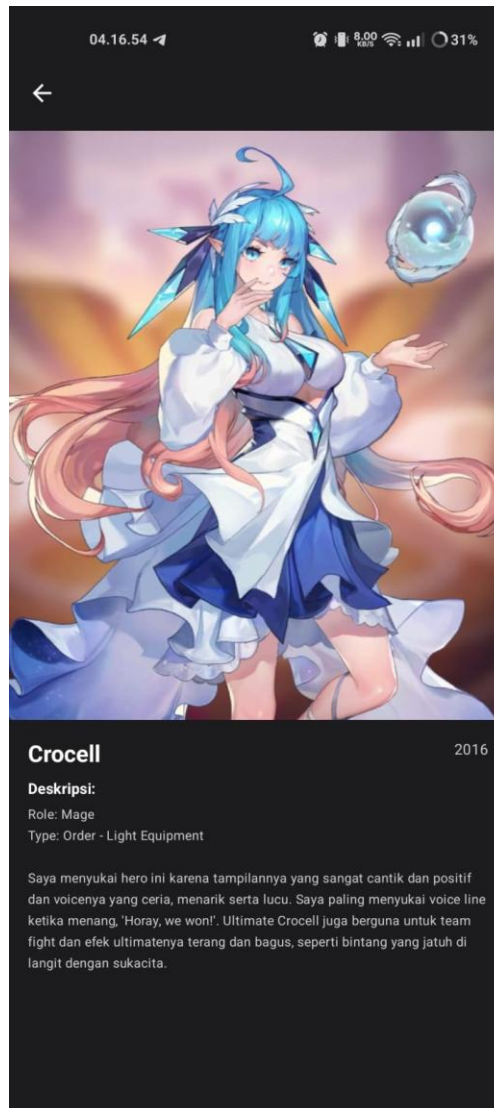
5	
6	class CharacterViewModelFactory :
7	ViewModelProvider.Factory {
8	override fun <T : ViewModel> create(modelClass:
9	Class<T>): T {
10	if
11	(modelClass.isAssignableFrom(CharacterViewModel::class
12	.java)) {
13	return CharacterViewModel() as T
14	}
15	throw IllegalArgumentException("Unknown
16	ViewModel class")
17	}
18	}

Tabel 8. Source Code Jawaban Soal 1

## B. Output Program



Gambar 2. Screenshot Hasil Jawaban Soal 1



*Gambar 3. Screenshot Hasil Jawaban Soal 1*

## **C. Pembahasan**

### **1. Melanjutkan aplikasi Android berbasis Jetpack Compose tanpa mengubah fitur-fitur sebelumnya.**

#### **a. Membuat sebuah ViewModel untuk menyimpan dan mengelola data dari list item.**

ViewModel digunakan untuk menyimpan dan mengelola data UI secara efisien dan aman terhadap perubahan siklus aplikasi. ViewModel memiliki peran utama sebagai penyimpanan logika bisnis dan data aplikasi yang digunakan oleh antarmuka pengguna. dengan menggunakan

ini, maka data seperti daftar karakter tidak akan hilang ketika aktivitas diubah, misalnya seperti orientasi layar. Contoh implementasi ViewModel dalam kode saya pada bagian ini:

#### **CharacterViewModel.kt**

```
11 class CharacterViewModel : ViewModel() {
12
13     private val _characterList = MutableStateFlow<List<Character>>(emptyList())
14     val characterList: StateFlow<List<Character>> = _characterList
```

*Gambar 4. Screenshot Kode ViewModel*

#### **b. Menggunakan ViewModelFactory dalam pembuatan ViewModel**

File ini menyediakan cara khusus untuk membuat CharacterViewModel, terutama jika ingin memberi argument ke ViewModel nanti, semisal repository atau parameter lainnya. File ini diperlukan karena viewModel() di Compose memerlukan Factory data ViewModel memiliki non-default. Pembuatan file ViewModelFactory ini agar kode lebih mudah dikembangkan dan dikelola ketika aplikasi akan bertambah besar dan semakin kompleks. Contoh implementasi kode ViewModelFactory saya adalah sebagai berikut:

#### **CharacterViewModelFactory.kt**

```
6 class CharacterViewModelFactory : ViewModelProvider.Factory {
7     override fun <T : ViewModel> create(modelClass: Class<T>): T {
8         if (modelClass.isAssignableFrom(CharacterViewModel::class.java)) {
9             return CharacterViewModel() as T
10        }
11        throw IllegalArgumentException("Unknown ViewModel class")
12    }
13 }
```

*Gambar 5. Screenshot Kode ViewModelFactory*

#### **c. Menggunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment**

StateFlow digunakan untuk menyimpan state secara reaktif. Jadi, ketika data berubah, maka UI akan otomatis terupdate. Contoh implementasi StateFlow di kode saya adalah sebagai berikut:

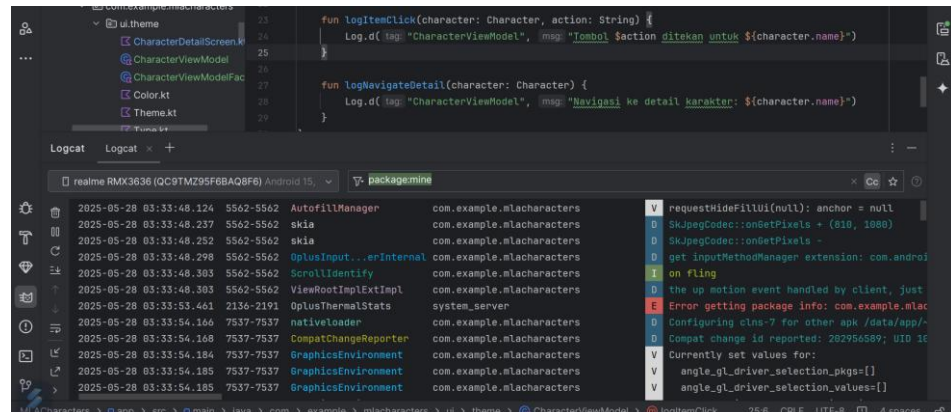
### CharacterViewModel.kt

```
13     private val _characterList = MutableStateFlow<List<Character>>(emptyList())
14     val characterList: StateFlow<List<Character>> = _characterList
```

Gambar 6. Screenshot Kode StateFlow

#### d. Menggunakan logging untuk event:

Tampilan pada Logcat:



Gambar 7. Screenshot Tampilan Logcat

#### a. Log saat data item masuk ke dalam list

Log ini akan tercetak saat setCharacterList() dipanggil—biasanya setelah kamu memuat data karakter. Tujuannya untuk memastikan bahwa list karakter berhasil dimasukkan ke dalam StateFlow.

### CharacterViewModel.kt

```
18     Log.d(tag: "CharacterViewModel", msg: "Data item masuk ke dalam list")
```

Gambar 8. Screenshot Kode yang menggunakan Log

#### b. Log saat tombol Detail dan tombol Explicit Intent ditekan

Fungsi ini dipanggil saat user menekan tombol Detail atau tombol Explicit Intent. Parameternya adalah nama karakter dan jenis tombol yang ditekan.

#### **CharacterViewModel.kt**

```
23 fun logItemClick(character: Character, action: String) {  
24     Log.d(tag: "CharacterViewModel", msg: "Tombol $action ditekan untuk ${character.name}")  
25 }
```

*Gambar 9. Screenshot Kode yang menggunakan Log*

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

Fungsi ini dicetak saat user memilih karakter untuk melihat halaman detail. Ini penting karena untuk tahu karakter mana yang dikirim saat navigasi terjadi.

#### **CharacterViewModel.kt**

```
27 fun logNavigateDetail(character: Character) {  
28     Log.d(tag: "CharacterViewModel", msg: "Navigasi ke detail karakter: ${character.name}")  
29 }
```

*Gambar 10. Screenshot Kode yang menggunakan Log*

- e. Menggunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi dengan menggunakan breakpoint dan menggunakan fitur Step Over, Step Over, dan Step Out

##### **a) Debugging dengan beberapa breakpoint**

Fungsi debugger adalah alat untuk membantu developer menemukan dan memperbaiki kesalahan (bug) dalam kode program dengan cara menjalankan program secara perlahan-lahan dan memeriksa apa yang terjadi di setiap langkahnya. Fungsi debugger adalah:

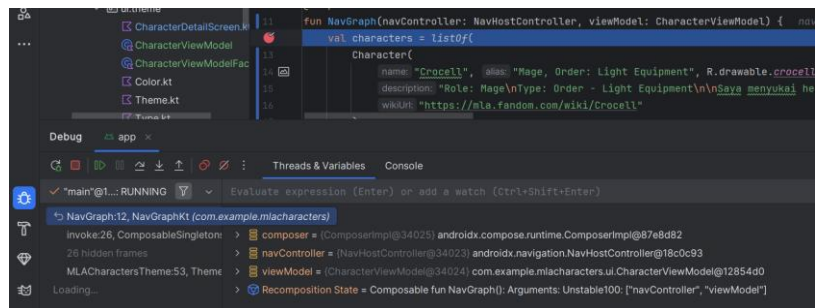
1. Menghentikan eksekusi program di titik tertentu menggunakan breakpoint.
2. Memeriksa nilai variabel saat program berhenti.
3. Menjalankan kode baris per baris agar bisa melihat alur program dan menemukan bug.



4. Memeriksa call stack, kondisi memori, dan status thread.
5. Mengubah nilai variabel saat debugging untuk menguji hasilnya.

Cara menggunakan debugger adalah sebagai berikut:

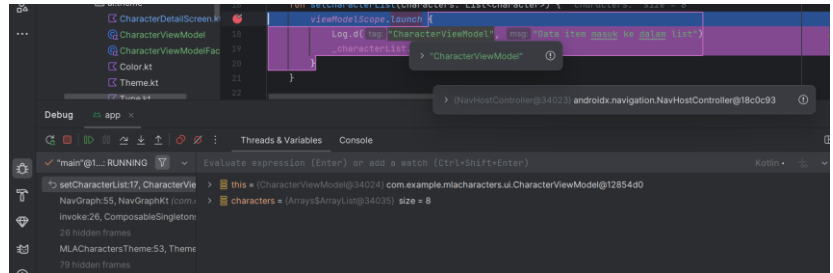
1. Pasang breakpoint (terletak di sebelah kiri kode, di angka baris nomor) pada baris yang ingin diperiksa.
2. Jalankan aplikasi dengan icon kumbang (bug), jangan run biasa.
3. Jalankan aplikasi di emulator atau device. Saya memakai device agar lebih cepat.
4. Memeriksa variabel dan status.
5. Melanjutkan eksekusi setelah berhenti di breakpoint.



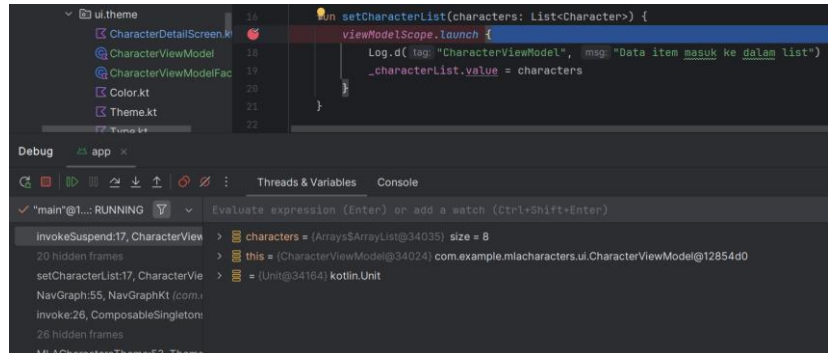
*Gambar 11. Screenshot pada saat Debugging*

#### **b) Menggunakan fitur Step Into (fn + F7)**

Digunakan untuk melangkah masuk ke dalam fungsi (method) yang sedang dipanggil di baris tersebut agar developer dapat melihat bagaimana isi fungsi itu bekerja secara detail.



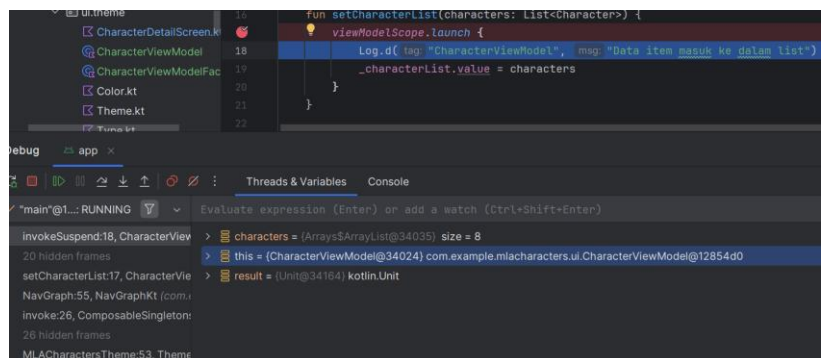
Gambar 12. Screenshot Fitur Step Into



Gambar 13. Screenshot Fitur Step Into

### c) Menggunakan fitur Step Over (fn + F8)

Digunakan untuk melangkah ke baris berikutnya di fungsi yang sama tanpa masuk ke fungsi lain, gunanya agar developer dapat melompat ke perintah berikutnya tanpa melihat isi fungsi yang dipanggil.

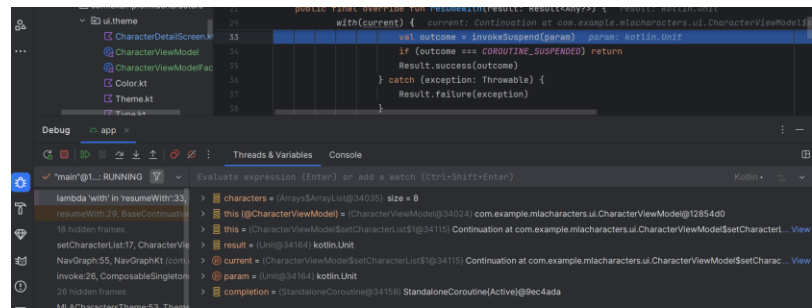


Gambar 14. Screenshot Fitur Step Over

### d) Menggunakan fitur Step Out (fn + Shift + F8)

Digunakan untuk melanjutkan eksekusi sampai keluar dari fungsi yang sedang developer jalani sekarang,

gunanya agar jika developer sudah masuk terlalu dalam ke fungsi dan ingin langsung keluar ke fungsi pemanggil.



Gambar 15. Screenshot Fitur Step Into

## 2. Fungsi Application class dalam arsitektur aplikasi Android

Dalam Application arsitektur Android, Application class adaah komponen inti yang mempresentasikan seluruh siklus hidup aplikasi. Class ini merupakan turunan dari android.app.Application dan bertindak sebagai entry point global untuk menjaga status dan konfigurasi aplikasi yang bersifat global selama aplikasi berjalan. Fungsi-fungsinya adalah sebagai berikut:

### a. Tempat menyimpan pengaturan global

Jika memiliki data atau pengaturan yang dibutuhkan di banyak bagian aplikasi, misalnya status login, tema, atau bahasa), developer bisa menyimpan di class ini agar tidak perlu mengatur ulang di setiap layar.

### b. Inisialisasi satu kali di awal aplikasi

Jika developer memakai library atau tools tambahan (misalnya database, Firebase, atau logger), bisa meletakkan kodenya di class ini agar langsung aktif saat aplikasi dibuka.

### c. Memiliki context global

Class ini bisa memberikan konteks yang bisa diakses dari mana saja di aplikasi karena terkadang developer membutuhkan konteks yang tidak bergantung pada layar tertentu

### d. Melacak aktivitas aplikasi

Bisa memantau aplikasi seperti kapan layar dibuka atau ditutup. Cocok untuk aplikasi besar yang perlu log atau analytics.

#### **D. Tautan Git**

Berikut adalah tautan untuk source code yang telah dibuat.