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

DAFTAR GAMBAR

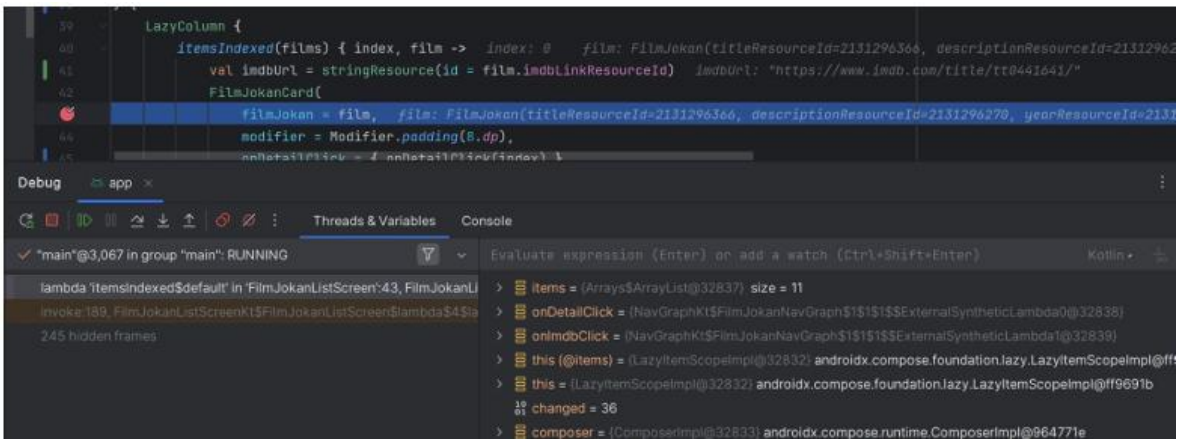
DAFTAR TABEL

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 pembuatam 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 lisy 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.mlacharacters
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.mlacharacters.ui.CharacterViewModel
12  import
13  com.example.mlacharacters.ui.CharacterViewModelFactory
14  import
15  com.example.mlacharacters.ui.theme.MLACharactersTheme
16
17  class MainActivity : ComponentActivity() {
18      override fun onCreate(savedInstanceState: Bundle?)
19  {
20          super.onCreate(savedInstanceState)
21          setContent {
22              MLACharactersTheme {
23                  Surface(color                    =
24  MaterialTheme.colorScheme.background) {
25                      val         navController        =
26  rememberNavController()
27
```

```
                    val viewModel: CharacterViewModel
= viewModel(
                        factory                         =
CharacterViewModelFactory()
                    )

                    NavGraph(navController          =
navController, viewModel = viewModel)
                }
            }
        }
    }
}
```

Tabel 1. Source Code Jawaban Soal 1

2. Character

```
package com.example.mlacharacters

data class Character(
    val name: String,
    val alias: String,
    val imageRes: Int,
    val description: String,
    val wikiUrl: String
)
```

Tabel 2. Source Code Jawaban Soal 1

3. MLACharacterApplication.kt

```
package com.example.mlacharacters

import android.app.Application

class MLACharactersApplication : Application() {
    override fun onCreate() {
        super.onCreate()
    }
}
```

Tabel 3. Source Code Jawaban Soal 1

4. CharacterListScreen.kt

```
package com.example.mlacharacters

import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.Image
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.mlacharacters.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()
```

```
            ) {
                Row(modifier =
Modifier.padding(12.dp)) {
                    Image(
                        painter = painterResource(id
= character.imageRes),
                        contentDescription =
character.name,
                        contentScale =
ContentScale.Crop,
                        modifier = Modifier
                            .width(90.dp)
                            .height(140.dp)
                            .padding(end = 12.dp)
                    )

                    Column(modifier =
Modifier.weight(1f)) {
                        Row(
                            modifier =
Modifier.fillMaxWidth(),
                            horizontalArrangement =
Arrangement.SpaceBetween
                        ) {
                            Text(
                                text =
character.name,
                                style =
MaterialTheme.typography.titleMedium,
                                color = Color.White,
                                fontWeight =
FontWeight.Bold
                            )
                            Text(
                                text = "2016", //
Masih dummy, bisa pakai data jika tersedia
                                style =
MaterialTheme.typography.labelMedium,
                                color =
Color.LightGray
                            )
                        }

                        Spacer(modifier =
Modifier.height(6.dp))

                        Row(
```

```
                                modifier =
Modifier.fillMaxWidth(),
                                verticalAlignment =
Alignment.Top
                            ) {
                                Text(
                                    text = "Deskripsi: ",
                                    color = Color.White,
                                    fontWeight =
FontWeight.Bold,
                                    style =
MaterialTheme.typography.bodyMedium
                                )

                                Spacer(modifier =
Modifier.width(8.dp))

                                Text(
                                    text =
character.description,
                                    color = Color.White,
                                    style =
MaterialTheme.typography.bodySmall,
                                    modifier =
Modifier.weight(1f)
                                )
                            }

                            Spacer(modifier =
Modifier.height(10.dp))

                            Row(horizontalArrangement =
Arrangement.spacedBy(8.dp)) {
                                Button(
                                    onClick = {

viewModel.logItemClick(character, "Detail Hero")
                                        val intent =
Intent(Intent.ACTION_VIEW,
Uri.parse(character.wikiUrl))

context.startActivity(intent)
                                    },
                                    colors =
ButtonDefaults.buttonColors(containerColor =
Color(0xFF9BB1EB)),

```

```
                                    modifier =
Modifier.weight(1f)
                                ) {
                                    Text("Detail Hero",
color = Color.White)
                                }

                                Button(
                                    onClick = {

viewModel.logItemClick(character, "Deskripsi")

navController.navigate("detail/${character.name}")
                                    },
                                    colors =
ButtonDefaults.buttonColors(containerColor =
Color(0xFF9BB1EB)),
                                    modifier =
Modifier.weight(1f)
                                ) {
                                    Text("Deskripsi",
color = Color.White)
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Tabel 4. Source Code Jawaban Soal 1

5. NavGraph.kt

```
package com.example.mlacharacters

import androidx.compose.runtime.Composable
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import com.example.mlacharacters.ui.CharacterViewModel
import
com.example.mlacharacters.ui.theme.CharacterDetailScre
en

@Composable
fun     NavGraph(navController:     NavHostController,
viewModel: CharacterViewModel) {
```

```
    val characters = listOf(
        Character(
            "Crocell", "Mage, Order: Light Equipment",
R.drawable.crocell,
            "Role:  Mage\nType:  Order  -  Light
Equipment\n\nSaya menyukai hero ini karena tampilannya
yang sangat cantik dan positif dan voicenya yang ceria,
menarik serta lucu. Saya paling menyukai voice line
ketika menang, 'Horay, we won!'. Ultimate Crocell juga
berguna untuk team fight dan efek ultimatenya terang dan
bagus, seperti bintang yang jatuh di langit dengan
sukacita.",
            "https://mla.fandom.com/wiki/Crocell"
        ),
        Character(
            "Astraia Sipra", "Support, Hybrid, Light -
Tech: Light Equipment", R.drawable.sipra,
            "Role: Support\nType: Hybrid (Light - Tech)
- Light Equipment\n\nDikenal dengan panggilan Sipra.
Hero ini juga termasuk dalam favorit saya karena
ultimatenya yang sangat berguna untuk war. Bahkan Sipra
B4 bisa mengimbangi war hero yang sudah awakened.
Ultimatenya adalah meng-summon hero secara acak.",

"https://mla.fandom.com/wiki/Astraia_Sipra"
        ),
        Character(
            "Nana", "Spirit, Martial: Light Equipment",
R.drawable.nana,
            "Role:  Spirit\nType:  Martial  -  Light
Equipment\n\nNana di Mobile Legends Bang Bang dengan
Mobile Legends Adventure sangat berbeda. Jujur ketika
saya pertama kali memakai hero ini, skill nya sangat
berbeda dengan yang versi moba nya. Skill 2 disini adalah
memberi heal pada tim dan ultinya membangkitkan hero
yang sudah mati.",
            "https://mla.fandom.com/wiki/Nana"
        ),
        Character(
            "Mecha Layla", "Marksman, Hybrid (Light -
Tech): Light Equipment", R.drawable.mechalayla,
            "Role: Marksman\nType: Hybrid (Light - Tech)
- Light Equipment\n\nLayla di MLA ada 2 versi. Mecha
Layla memberi heal dan damage secara terus-menerus. Tapi
jika semua hero tim mati, Mecha Layla juga ikut lenyap.",
            "https://mla.fandom.com/wiki/Mecha_Layla"
        ),
```

```
        Character(
            "Natsu", "Fighter, Hybrid (Dark - Martial): Medium Equipment", R.drawable.natsu,
            "Role:  Fighter\nType:Hybrid (Dark - Martial) - Medium Equipment\n\nNatsu hero kolaborasi Fairy Tail. Punya skill dorong musuh ke kanan. Menurut saya sangat berguna saat team fight, apalagi jika dikombo dengan AoE seperti Alice.",
            "https://mla.fandom.com/wiki/Natsu"
        ),
        Character(
            "Naiad Rafaela", "Hybrid (Light - Elemental): Light Equipment", R.drawable.rafaela,
            "Role:Support\nType: Hybrid (Light - Elemental) - Light Equipment\n\nDesain sangat anggun dan menyatu dengan alam. Hero ini saya sukai karena estetikanya meskipun belum terlalu dipakai untuk war.",

"https://mla.fandom.com/wiki/Naiad_Rafaela"
        ),
        Character(
            "Shah Torre", "Mage, Chaos: Light Equipment", R.drawable.torre,
            "Role:  Mage\nType:  Chaos - Light Equipment\n\nShah Torre menyerap bar energi musuh, sehingga bisa menunda ultimate lawan. Sangat menyebalkan jika menjadi musuh.",
            "https://mla.fandom.com/wiki/Shah_Torre"
        ),
        Character(
            "Forseti", "Marksman, Order: Light Equipment", R.drawable.forseti,
            "Role: Marksman\nType: Order - Light Equipment\n\nUltimatenya membuat Forseti langsung mati. Butuh support respawn seperti Nana atau Lunox. Saya baru sadar potensi hero ini setelah lihat performanya di Tower of Babel.",
            "https://mla.fandom.com/wiki/Forseti"
        ),
    )

    viewModel.setCharacterList(characters)

    val charMap = characters.associateBy { it.name }

    NavHost(navController = navController, 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.mlacharacters.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.mlacharacters.Character
20
21  @OptIn(ExperimentalMaterial3Api::class)
22  @Composable
23  fun CharacterDetailScreen(character: Character,
24  navController: NavController) {
```

```
    Scaffold(
        containerColor = Color(0xFF1C1C1E),
        topBar = {
            CenterAlignedTopAppBar(
                title = {},
                navigationIcon = {
                    IconButton(onClick = {
navController.popBackStack() }) {
                        Icon(
                            imageVector =
Icons.Default.ArrowBack,
                            contentDescription =
"Back",
                            tint = Color.White
                        )
                    }
                },
                colors =
TopAppBarDefaults.centerAlignedTopAppBarColors(
                    containerColor =
Color.Transparent
                )
            )
        }
    ) { innerPadding ->
        LazyColumn(
            modifier = Modifier
                .padding(innerPadding)
                .fillMaxSize()
        ) {
            item {
                Image(
                    painter = painterResource(id =
character.imageRes),
                    contentDescription =
character.name,
                    contentScale = ContentScale.Crop,
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(500.dp)
                )
            }

            item {
                Column(modifier =
Modifier.padding(16.dp)) {
                    Row(
```

```
                        modifier =
Modifier.fillMaxWidth(),
                        horizontalArrangement =
Arrangement.SpaceBetween
                    ) {
                        Text(
                            text = character.name,
                            color = Color.White,
                            style =
MaterialTheme.typography.titleLarge,
                            fontWeight =
FontWeight.Bold
                        )
                        Text(
                            text = "2016",
                            color = Color.LightGray,
                            style =
MaterialTheme.typography.bodyMedium
                        )
                    }

                    Spacer(modifier =
Modifier.height(8.dp))

                    Text(
                        text = "Deskripsi:",
                        color = Color.White,
                        fontWeight = FontWeight.Bold,
                        style =
MaterialTheme.typography.bodyMedium
                    )

                    Spacer(modifier =
Modifier.height(4.dp))

                    Text(
                        text = character.description,
                        color = Color(0xFFCCCCCC),
                        style =
MaterialTheme.typography.bodySmall,
                        lineHeight = 20.sp
                    )
                }
            }
        }
    }
}
```

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

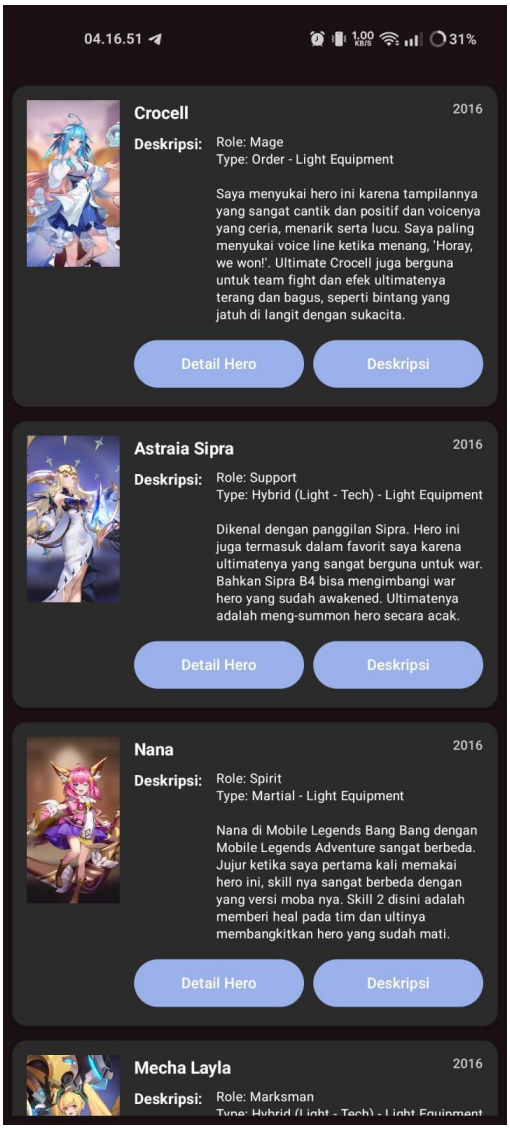
8. CharacterViewModelFactory – ui.model

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

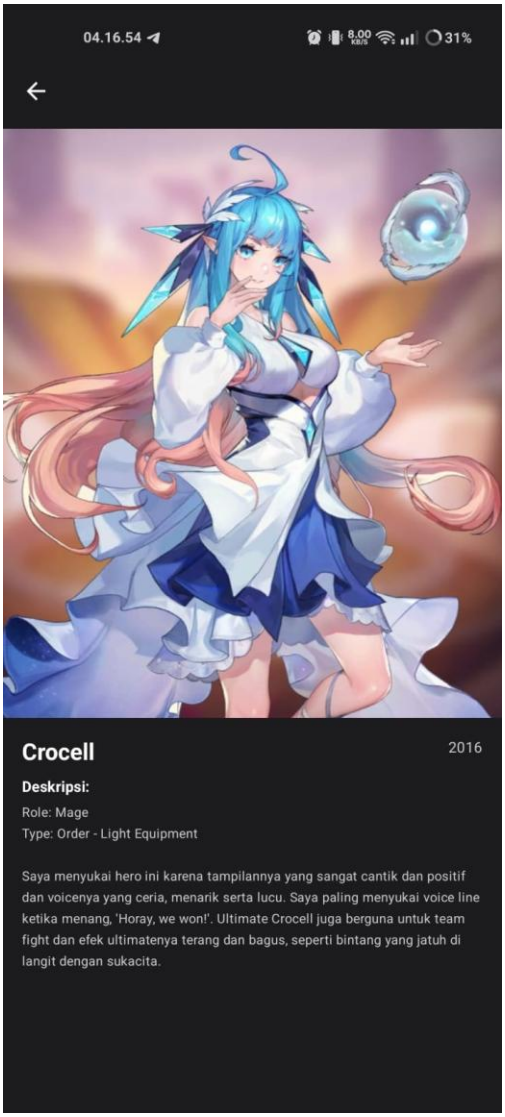
```
5
6   class CharacterViewModelFactory :
7   ViewModelProvider.Factory {
8       override fun <T : ViewModel> create(modelClass:
9   Class<T>): T {
10          if
11  (modelClass.isAssignableFrom(CharacterViewModel::class
12  .java)) {
13              return CharacterViewModel() as T
14          }
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

```
class CharacterViewModel : ViewModel() {

    private val _characterList = MutableStateFlow<List<Character>>(emptyList())
    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 memebri 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

```
class CharacterViewModelFactory : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(CharacterViewModel::class.java)) {
            return CharacterViewModel() as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

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:

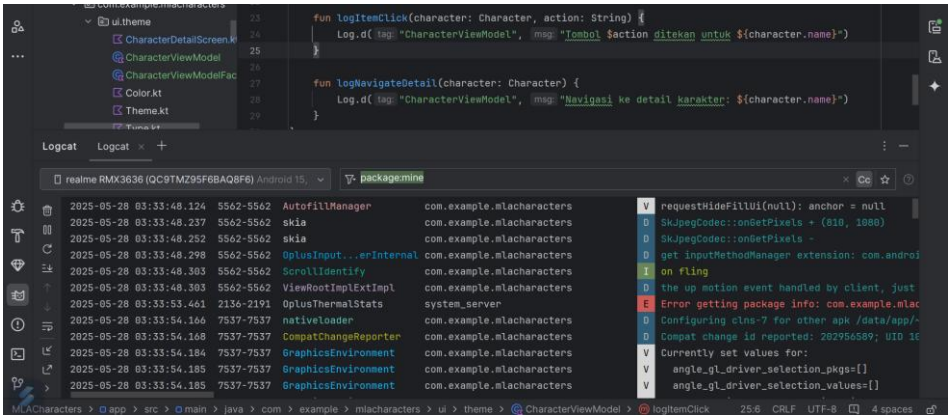
CharacterViewModel.kt

```
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

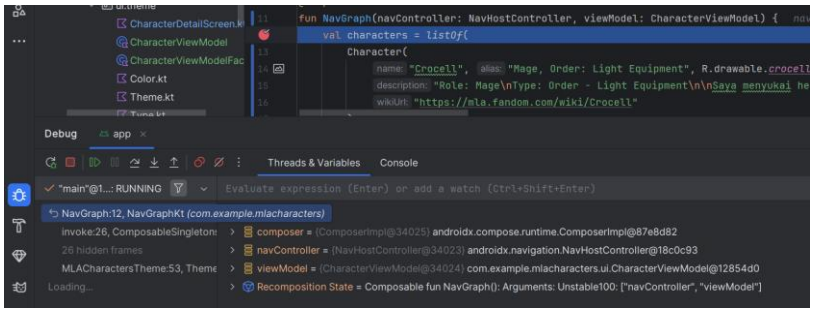
Fungsi debugger adalah alat untuk membantu developer menemukan dan memperbaiki kesalahan (bug) dlam kode program dengan cara menjalankan program secara perlahan-lahan dan memeriksa apa yang terjadi di setiap langkahnya. Fungsi debugger adalah:

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 uhntuk menguji hasilnya.

Cara menggunakan debugger adalah sebagai berikut:

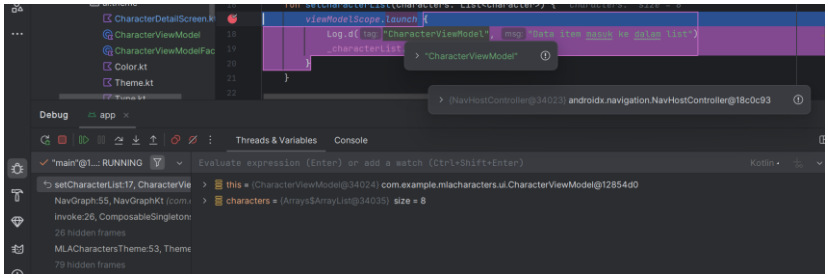
1. Pasang breakpoint (terletak di sebelah kiri kode, di angka baris nomor) pada baris yang ingin diperiksa.
2. Jalankan aplikasi dengan icon kumbang (bug), jangan run biasa.
3. Jalankan aplikasi di emulator atau device. Saya memakai device agar lebih cepat.
4. Memeriksa variabel dan status.
5. Melanjutkan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



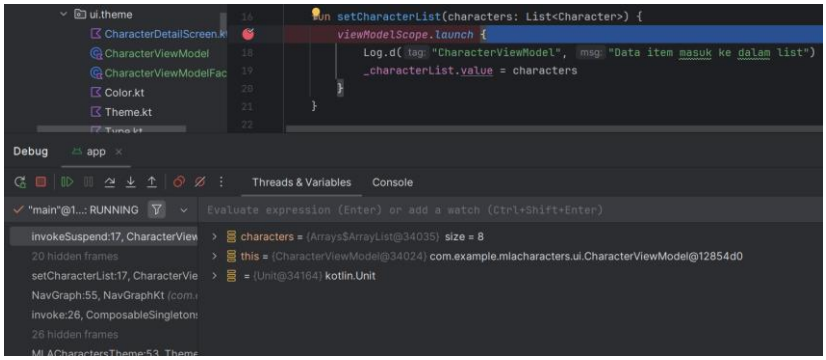
Gambar 11. Screenshot pada saat Debugging

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.



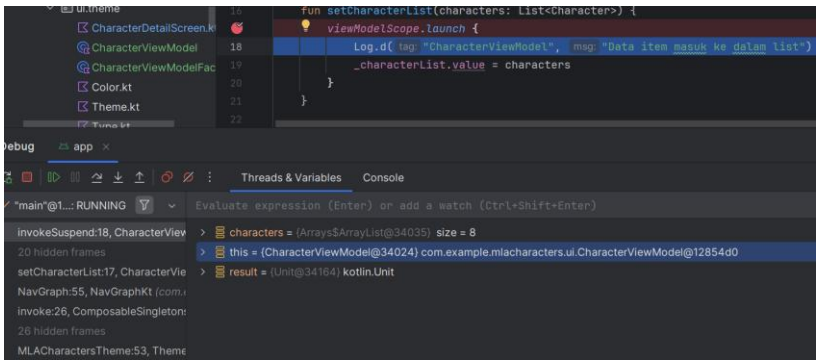
Gambar 12. Screenshot Fitur Step Into



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.

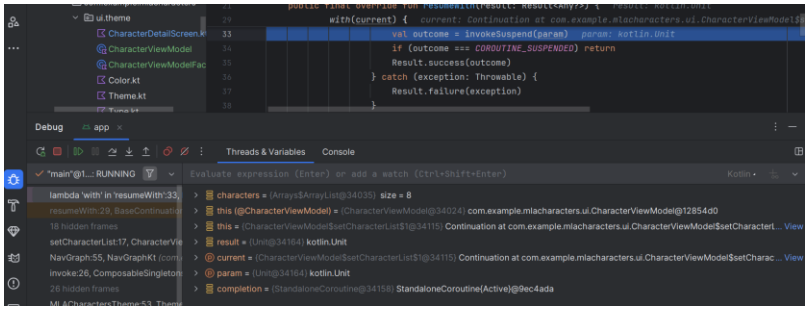


Gambar 14. Screenshot Fitur Step Over

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.

Laporan-Praktikum-Pemrograman-Mobile/MODUL 4 at main · aikoanatashawendiono/Laporan-Praktikum-Pemrograman-Mobile