LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

ViewModel and Debugging

Oleh:

Ghani Mudzakir **NIM. 2310817110011**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE I
MODUL 5

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

Nama Praktikan : Ghani Mudzakir
NIM : 2310817110011

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 19
C. Pembahasan 21
D. Tautan Git 25

DAFTAR GAMBAR

Gambar 1 Contoh Penggunaan Debugger 7
Gambar 2 Tampilan Awal UI Aplikasi 19
Gambar 3 Tampilan Halaman Detail 20
Gambar 4 Tampilan Saat Menuju Website 20
Gambar 5 Log saat data item masuk ke dalam list 21
Gambar 6 Log saat tombol Detail dan tombol Explicit Intent ditekan 21
Gambar 7 Log data dari list yang dipilih ketika berpindah ke halaman Detail 21

DAFTAR TABEL

Tabel 1 Source Code Pembukaan.kt 7
Tabel 2 Source Code PembukaanRepository.kt 7
Tabel 3 Source Code viewmodel/PembukaanViewModel.kt 12
Tabel 4 Source Code viewmodel/PembukaanviewModelFactory.kt 13
Tabel 5 Source Code MainActivity.kt 13

SOAL 1

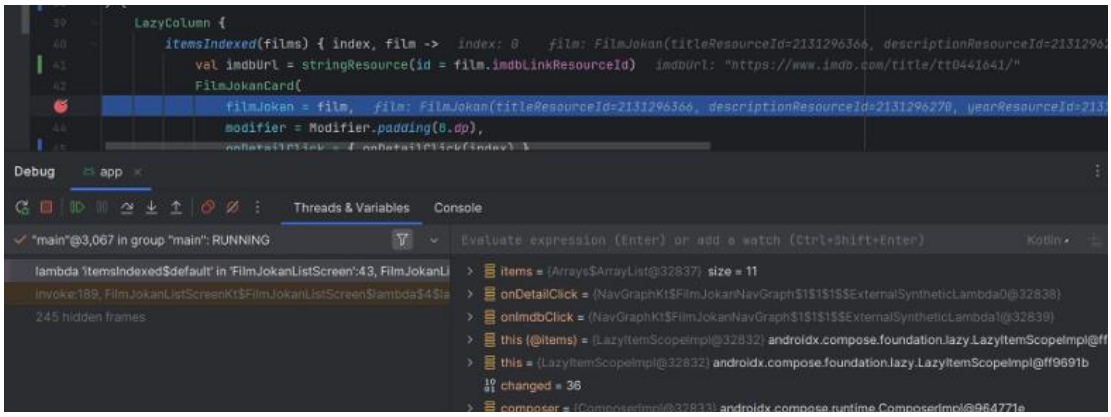
Soal Praktikum:

1. Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:

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

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. Pembukaan.kt

Tabel 1 Source Code Pembukaan.kt

```
1  package com.example.pembukaan_catur
2
3  import androidx.annotation.DrawableRes
4
5  data class TampilanPembukaan(
6      val nama_pembukaan: String,
7      val penjelasan_singkat: String,
8      val penjelasan_dua_paragraf: String,
9      @DrawableRes val imageResId: Int,
10     val link: String
11 )
```

2. PembukaanRepository.kt

Tabel 2 Source Code PembukaanRepository.kt

```
1  package com.example.pembukaan_catur
2
3  class PembukaanRepository {
4      fun ambilSemuaPembukaan(): List<TampilanPembukaan>{
5          return listOf(
6              TampilanPembukaan(
7                  nama_pembukaan = "Sicilian Defense",
8                  penjelasan_singkat = "Sebuah pembukaan
9  agresif yang dimulai dengan e4 c5, sering digunakan untuk
10 menciptakan permainan tidak seimbang.",
11                 penjelasan_dua_paragraf = """
12         Sicilian Defense adalah salah satu pembukaan
13 paling populer dan tajam dalam catur modern. Dengan
```

```
membalas e4 dengan c5, hitam menghindari simetri dan berusaha mengambil inisiatif dengan permainan sayap.

        Pembukaan ini terkenal menghasilkan posisi yang kompleks dan penuh taktik. Banyak juara dunia, seperti Kasparov dan Fischer, mengandalkan Sicilian untuk menghadapi pemain e4.
    """.trimIndent(),
                imageResId = R.drawable.p1,
                link                                        = "https://www.chess.com/openings/Sicilian-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "French Defense",
                penjelasan_singkat   =   "Strategi   yang dimulai dengan e4 e6, di mana hitam memblokir pion putih dan merencanakan serangan melalui posisi yang lebih tertutup.",
                penjelasan_dua_paragraf = """
        French Defense memberikan struktur pion yang kokoh dan solid untuk hitam. Ia menawarkan banyak kemungkinan strategis, terutama dalam permainan tengah.

        Salah satu ciri khasnya adalah konflik antara pion e5 putih dan struktur pertahanan hitam di d5. Taktik dan strategi memainkan peran penting dalam membuka posisi ini.
    """.trimIndent(),
                imageResId = R.drawable.p2,
                link                                        = "https://www.chess.com/openings/French-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Ruy López Opening",
                penjelasan_singkat   =   "Pembukaan   klasik yang dimulai dengan e4 e5 2.Nf3 Nc6 3.Bb5, bertujuan untuk mengontrol pusat dan menekan pertahanan hitam.",
                penjelasan_dua_paragraf = """
        Ruy López adalah salah satu pembukaan tertua yang masih dimainkan di level tinggi. Dengan menekan kuda di c6, putih mencoba mengganggu kontrol hitam terhadap pusat.

        Posisi yang timbul sering bersifat strategis, dengan ruang untuk manuver jangka panjang dan potensi serangan raja di tahap akhir pembukaan.
    """.trimIndent(),
                imageResId = R.drawable.p3,

```

```
                link                                                =
"https://www.chess.com/openings/Ruy-Lopez-Opening"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Caro-Kann Defense",
                penjelasan_singkat = "Dimulai dengan e4
c6, hitam berusaha untuk memperkuat pusat dan membangun
pertahanan yang solid.",
                penjelasan_dua_paragraf = """
        Caro-Kann terkenal karena stabilitas dan keamanan
bagi raja hitam. Ini adalah pembukaan pilihan bagi pemain
yang menyukai posisi bertahan namun aktif.

        Hitam membentuk d5 segera setelah c6, berusaha
menetralisir pusat putih tanpa menciptakan kelemahan
signifikan.
    """.trimIndent(),
                imageResId = R.drawable.p4,
                link                                                =
"https://www.chess.com/openings/Caro-Kann-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Italian Game",
                penjelasan_singkat = "Sebuah pembukaan
yang sering mengarah pada permainan terbuka, dimulai
dengan e4 e5 2.Nf3 Nc6 3.Bc4, dengan tujuan menyerang pusat
lawan.",
                penjelasan_dua_paragraf = """
        Italian Game memberikan peluang pengembangan cepat
bagi kedua pihak. Putih langsung mengincar titik lemah f7,
titik lemah paling rentan bagi hitam di awal permainan.

        Pembukaan ini cocok untuk pemain pemula hingga
master karena keseimbangan antara taktik dan strategi.
    """.trimIndent(),
                imageResId = R.drawable.p5,
                link                                                =
"https://www.chess.com/openings/Italian-Game"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Queen's Gambit",
                penjelasan_singkat = "Pembukaan populer
yang dimulai dengan d4 d5 2.c4, di mana putih menawarkan
pion untuk mengontrol pusat papan.",
                penjelasan_dua_paragraf = """
```

```
        Queen's Gambit adalah salah satu pembukaan tertua dan paling dihormati dalam catur. Meski disebut 'gambit', pion yang dikorbankan biasanya dapat direbut kembali.

        Tujuan utama putih adalah mengalihkan pion d5 hitam dan menciptakan dominasi penuh di pusat papan. Banyak juara dunia telah menggunakan pembukaan ini dengan sukses besar.
    """.trimIndent(),
                imageResId = R.drawable.p6,
                link                                    =
"https://www.chess.com/openings/Queens-Gambit"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Slav Defense",
                penjelasan_singkat = "Dimulai dengan d4 d5
2.c4 c6, hitam bertujuan untuk menjaga pusat dan bersiap untuk melawan serangan putih.",
                penjelasan_dua_paragraf = """
        Slav Defense adalah respon solid terhadap Queen’s Gambit. Dengan memainkan c6, hitam memperkuat kontrol atas d5 tanpa membuka terlalu banyak ruang.

        Ini menghasilkan posisi yang seimbang namun fleksibel, memungkinkan transisi ke berbagai rencana strategis tergantung perkembangan permainan.
    """.trimIndent(),
                imageResId = R.drawable.p7,
                link                                    =
"https://www.chess.com/openings/Slav-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "King's Indian Defense",
                penjelasan_singkat   =   "Strategi   yang
dimulai dengan 1.d4 Nf6 2.c4 g6, hitam berencana menyerang pusat dengan pion dan pasukan yang dikembangkan setelahnya.",
                penjelasan_dua_paragraf = """
        King's Indian Defense dikenal dengan pendekatannya yang agresif dan asimetris. Hitam mengizinkan putih membangun pusat besar, lalu menyerangnya.

        Strategi khasnya adalah serangan raja oleh hitam, bahkan ketika putih mengembangkan keunggulan ruang. Ini adalah pilihan ideal bagi pecatur taktis.
    """.trimIndent(),
                imageResId = R.drawable.p8,
```

```
                link                                                =
"https://www.chess.com/openings/Kings-Indian-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Nimzo-Indian Defense",
                penjelasan_singkat = "Dimulai dengan d4
Nf6 2.c4 e6 3.Nc3 Bb4, hitam bertujuan untuk mengendalikan
pusat sambil mengembangkan tekanan terhadap pion putih.",
                penjelasan_dua_paragraf = """
        Nimzo-Indian Defense adalah pembukaan strategis
yang memanfaatkan ancaman terhadap struktur pion putih di
awal. Dengan Bb4, hitam menekan Nc3 dan menciptakan
potensi kerusakan struktur.

        Pembukaan ini menggabungkan kontrol pusat dengan
tekanan posisi dan cocok bagi pemain yang menyukai
fleksibilitas.
    """.trimIndent(),
                imageResId = R.drawable.p9,
                link                                                =
"https://www.chess.com/openings/Nimzo-Indian-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan    =    "Queen's    Indian
Defense",
                penjelasan_singkat = "Dimulai dengan 1.d4
Nf6  2.c4  e6  3.Nf3  b6,  hitam  berusaha  mengembangkan
bidaknya  dengan  cara  yang  fleksibel  dan  mengontrol
pusat.",
                penjelasan_dua_paragraf = """
        Queen’s Indian Defense fokus pada perkembangan
bidak yang fleksibel dan pengendalian diagonal panjang
dengan gajah di b7. Ini adalah pembukaan yang solid dan
penuh manuver.

        Pembukaan ini cocok untuk pemain yang ingin
menghindari konflik langsung di awal, namun siap melawan
balik saat permainan berkembang.
    """.trimIndent(),
                imageResId = R.drawable.p10,
                link                                                =
"https://www.chess.com/openings/Queens-Indian-Defense"
            )

        )
    }
}
```

3. PembukaanViewModel.kt

Tabel 3 Source Code viewmodel/PembukaanViewModel.kt

```
1  package com.example.pembukaan_catur.viewmodel
2
3  import androidx.lifecycle.ViewModel
4  import
5  com.example.pembukaan_catur.model.TampilanPembukaan
6  import
7  com.example.pembukaan_catur.model.PembukaanRepository
8  import kotlinx.coroutines.flow.MutableStateFlow
9  import kotlinx.coroutines.flow.StateFlow
10 import android.util.Log
11
   class PembukaanViewModel(private val repository:
   PembukaanRepository) : ViewModel() {

       private val _pembukaanList =
   MutableStateFlow<List<TampilanPembukaan>>(emptyList())
       val pembukaanList:
   StateFlow<List<TampilanPembukaan>> = _pembukaanList

       private val _selectedPembukaan =
   MutableStateFlow<TampilanPembukaan?>(null)
       val selectedPembukaan: StateFlow<TampilanPembukaan?>
   = _selectedPembukaan

       init {
           val data = repository.ambilSemuaPembukaan()
           _pembukaanList.value = data

           Log.d("PembukaanViewModel", "Data item dimuat ke
   dalam list: ${data.size} items")
       }

       fun pilihPembukaan(pembukaan: TampilanPembukaan) {
           _selectedPembukaan.value = pembukaan
           Log.d("PembukaanViewModel", "Item dipilih untuk
   detail: ${pembukaan.nama_pembukaan}")
       }

       fun logTombolDetail(item: TampilanPembukaan) {
           Log.d("PembukaanViewModel", "Tombol Detail
   ditekan: ${item.nama_pembukaan}")
       }

       fun logTombolExplicitIntent(item: TampilanPembukaan)
   {
           Log.d("PembukaanViewModel", "Tombol Website
```

```
    ditekan:                          ${item.nama_pembukaan}")
        }
    }
```

4. PembukaanviewModelFactory.kt

Tabel 4 Source Code viewmodel/PembukaanviewModelFactory.kt

```
1  package com.example.pembukaan_catur.viewmodel
2
3  import androidx.lifecycle.ViewModel
4  import androidx.lifecycle.ViewModelProvider
5  import
6  com.example.pembukaan_catur.model.PembukaanRepository
7
8  class PembukaanViewModelFactory(
9      private val repository: PembukaanRepository
10 ) : ViewModelProvider.Factory {
11
       @Suppress("UNCHECKED_CAST")
       override fun <T : ViewModel> create(modelClass:
       Class<T>): T {
           if
       (modelClass.isAssignableFrom(PembukaanViewModel::class.j
       ava)) {
               return PembukaanViewModel(repository) as T
           }
           throw IllegalArgumentException("Unknown
       ViewModel class")
       }
   }
```

5. MainActivity.kt

Tabel 5 Source Code MainActivity.kt

```
1  package com.example.pembukaan_catur
2
3  import android.content.Intent
4  import android.net.Uri
5  import android.os.Bundle
6  import android.util.Log
7  import androidx.activity.ComponentActivity
8  import androidx.activity.compose.setContent
9  import androidx.activity.enableEdgeToEdge
10 import androidx.compose.foundation.Image
```

```
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.lifecycle.ViewModelProvider
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavController
import androidx.navigation.NavHostController
import androidx.navigation.NavType
import androidx.navigation.compose.*
import androidx.navigation.navArgument
import com.example.pembukaan_catur.model.PembukaanRepository
import com.example.pembukaan_catur.ui.theme.Pembukaan_caturTheme
import com.example.pembukaan_catur.viewmodel.PembukaanViewModel
import
com.example.pembukaan_catur.viewmodel.PembukaanViewModelFactory
import kotlinx.coroutines.flow.collectLatest

class MainActivity : ComponentActivity() {

    private lateinit var pembukaanViewModelFactory:
PembukaanViewModelFactory
    private lateinit var pembukaanViewModel: PembukaanViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()

        pembukaanViewModelFactory =
PembukaanViewModelFactory(PembukaanRepository())
        pembukaanViewModel = ViewModelProvider(this,
pembukaanViewModelFactory)[PembukaanViewModel::class.java]


        setContent {
            Pembukaan_caturTheme {
                Surface(modifier = Modifier.fillMaxSize()) {
                    val navController = rememberNavController()
```

```
                    NavHost(
                        navController = navController,
                        startDestination = "listPembukaan"
                    ) {
                        composable("listPembukaan") {
                            // Ambil state list dari ViewModel dengan collectAsState()
                            val pembukaanList by pembukaanViewModel.pembukaanList.collectAsState()
                            val context = LocalContext.current

                            DaftarPembukaan(
                                pembukaanItems = pembukaanList,
                                navController = navController,
                                onDetailClick = { item ->

pembukaanViewModel.logTombolDetail(item)  // Panggil tanpa argumen
                                    val encodedDesc = Uri.encode(item.penjelasan_dua_paragraf)

navController.navigate("penjelasan/$encodedDesc/${item.imageResId}")
                                },

                                onWebsiteClick = { item ->

pembukaanViewModel.logTombolExplicitIntent(item)
                                    val intent = Intent(Intent.ACTION_VIEW, Uri.parse(item.link))
                                    context.startActivity(intent)
                                }
                            )

                        }
                        composable(
                            route = "penjelasan/{desc}/{img}",
                            arguments = listOf(
                                navArgument("desc") { type = NavType.StringType },
                                navArgument("img") { type = NavType.IntType }
                            )
                        ) { backStackEntry ->
                            val desc = backStackEntry.arguments?.getString("desc") ?: ""
                            val img = backStackEntry.arguments?.getInt("img") ?: 0

                            // Log saat berpindah ke halaman Detail
                            Log.d("PembukaanViewModel", "Navigasi ke
```

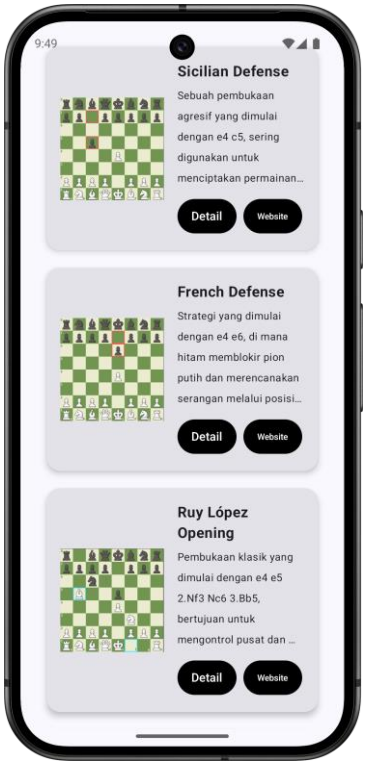
```
detail dengan deskripsi: $desc")

                    tampilanDetail(img = img, nama = "Detail
Pembukaan", penjelasan = desc)
                }
            }
        }
    }
    }
}

// Composable daftar pembukaan dengan callback klik
@Composable
fun DaftarPembukaan(
    pembukaanItems:
List<com.example.pembukaan_catur.model.TampilanPembukaan>,
    navController: NavHostController,
    onDetailClick:
(com.example.pembukaan_catur.model.TampilanPembukaan) -> Unit,
    onWebsiteClick:
(com.example.pembukaan_catur.model.TampilanPembukaan) -> Unit
) {
    LazyColumn(
        modifier = Modifier
            .fillMaxWidth()
            .padding(20.dp)
    ) {
        items(items = pembukaanItems) { item ->
            TampilanEntitasPembukaan(
                name = item.nama_pembukaan,
                image = item.imageResId,
                url = item.link,
                description = item.penjelasan_singkat,
                penjelasan = item.penjelasan_dua_paragraf,
                navController = navController,
                onDetailClick = { onDetailClick(item) },
                onWebsiteClick = { onWebsiteClick(item) }
            )
        }
    }
}

// Perbaikan fungsi TampilanEntitasPembukaan dengan callback tombol
@Composable
fun TampilanEntitasPembukaan(
    name: String,
    image: Int,
    url: String,
```

```
    description:                                                String,
    penjelasan:                                                 String,
    navController:                                          NavController,
    onDetailClick:                ()                ->              Unit,
    onWebsiteClick:                ()                ->              Unit
)                                                                       {
    val           context           =           LocalContext.current
    Card(
        modifier                          =                     Modifier
            .fillMaxWidth()
            .padding(10.dp),
        shape                =               RoundedCornerShape(16.dp),
        elevation  =  CardDefaults.cardElevation(defaultElevation  =
4.dp)
    )                                                                   {
        Row(
            modifier                      =                     Modifier
                .padding(16.dp)
                .fillMaxWidth(),
            verticalAlignment      =      Alignment.CenterVertically
        )                                                               {
            Image(
                painter    =    painterResource(id    =    image),
                contentDescription                =               null,
                modifier                    =                   Modifier
                    .size(width   =   120.dp,   height   =   120.dp)
            )
            Spacer(modifier         =         Modifier.width(16.dp))
            Column(
                modifier                    =                   Modifier
                    .weight(1f)
            )                                                           {
                Text(text  =  name,  fontWeight  =  FontWeight.Bold,
fontSize                              =                              17.sp)
                Spacer(modifier       =       Modifier.height(4.dp))
                Text(
                    text                =               description,
                    fontSize                  =                 12.sp,
                    maxLines                    =                   5,
                    overflow         =         TextOverflow.Ellipsis
                )
                Spacer(modifier       =       Modifier.height(8.dp))
                Row(
                    horizontalArrangement                             =
Arrangement.SpaceBetween,
                    modifier                  =                 Modifier
                        .fillMaxWidth()
                        .wrapContentWidth(Alignment.Start),
                )                                                       {
```

```
                    Button(
                        onClick = onDetailClick,
                        contentPadding = PaddingValues(horizontal =
16.dp,  vertical = 8.dp),
                        shape = RoundedCornerShape(50),
                        modifier = Modifier.defaultMinSize(minWidth
=  1.dp),
                        colors = ButtonDefaults.buttonColors(
                            containerColor = Color.Black,
                            contentColor = Color.White
                        )
                    ) {
                        Text("Detail", fontSize = 14.sp)
                    }

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

                    Button(
                        onClick = onWebsiteClick,
                        contentPadding = PaddingValues(horizontal =
16.dp,  vertical = 8.dp),
                        shape = RoundedCornerShape(50),
                        modifier = Modifier.defaultMinSize(minWidth
=  1.dp),
                        colors = ButtonDefaults.buttonColors(
                            containerColor = Color.Black,
                            contentColor = Color.White
                        )
                    ) {
                        Text("Website", fontSize = 10.sp)
                    }
                }
            }
        }
    }
}

@Composable
fun tampilanDetail(img: Int, nama: String, penjelasan: String) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
            .padding(WindowInsets.statusBars.asPaddingValues()),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Image(
            painter = painterResource(id = img),
            contentDescription = "Gambar Detail dari $nama",
```

```
            modifier                    =                    Modifier
                .fillMaxWidth()
                .height(380.dp)
        )
        Spacer(modifier          =          Modifier.height(16.dp))
        Text(
            text                        =                        nama,
            fontSize                      =                    30.sp,
            fontWeight                  =                FontWeight.W800
        )
        Spacer(modifier          =          Modifier.height(16.dp))
        Text(
            text                      =                    penjelasan,
            fontSize                        =                  20.sp,
        )
    }
}
```

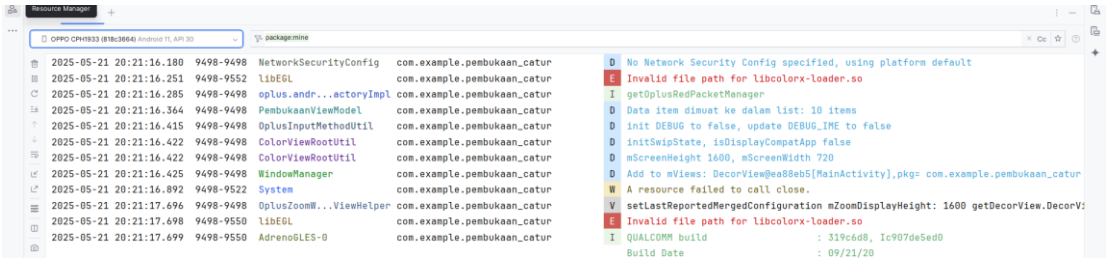
B. Output Program



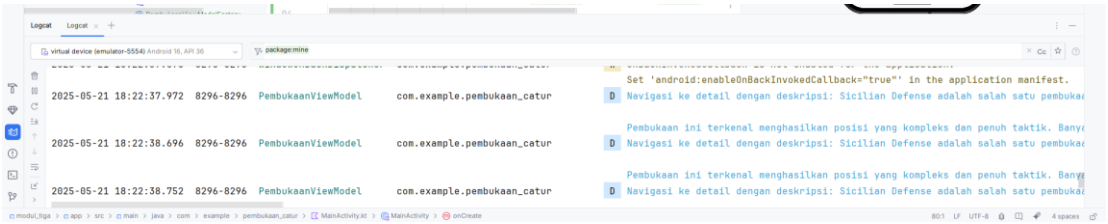
Gambar 2 Tampilan Awal UI Aplikasi

Gambar 3 Tampilan Halaman Detail

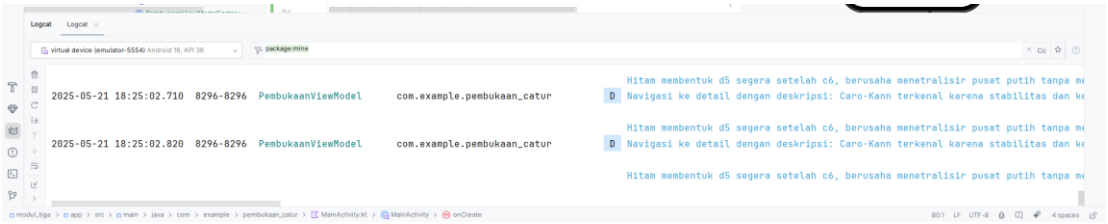
Gambar 4 Tampilan Saat Menuju Website



Gambar 5 Log saat data item masuk ke dalam list



Gambar 6 Log saat tombol Detail dan tombol Explicit Intent ditekan



Gambar 7 Log data dari list yang dipilih ketika berpindah ke halaman Detail

C. Pembahasan

1. Pembukaan.kt

Pada file ini kita sebenarnya hanya membuat sebuah class dengan nama TampilanPembukaan dengan beberapa val parameter penting seperti nama_pembukaan dengan tipe data string, penjelasan_singkat dengan tipe data string, penjelasan_dua_paragraf dengan tipe data string, val dengan tipe data Drawable yang memiliki default value Int, dan yang terakhir adalah link dengan tipe data String. File ini akan digunakan data awal yang akan ditampilkan di bagian looping sesuai dengan MainActivity.kt nantinya.

2. PembukaanRepository.kt

Pada file ini kita membuat class dengan nama ambilSemuaPembukaan yang mengambil fungsi TampilanPembukaan dari file Pembukaan.kt dan akan mengembalikan fungsi itu dengan data-data yang dideklarasikan di file ini di dalam MainAcivity.kt

3. PembukaanViewModel.kt

Pada file ini, kita membuat sebuah kelas bernama PembukaanViewModel yang mewarisi dari ViewModel. File ini bertanggung jawab untuk mengatur dan mengelola data yang akan ditampilkan di tampilan UI, khususnya data yang berhubungan dengan pembukaan catur. Di dalam kelas ini, terdapat dua buah StateFlow, yaitu _pembukaanList yang berisi daftar seluruh pembukaan catur dan _selectedPembukaan yang menyimpan data dari satu item pembukaan yang dipilih oleh pengguna. Data pembukaan ini sendiri diambil dari PembukaanRepository, yang sebelumnya sudah di-inject lewat constructor.

Pada bagian init, semua data pembukaan langsung dimuat dari repository dan dimasukkan ke dalam _pembukaanList, sehingga nantinya bisa diamati oleh tampilan antarmuka. Selain itu, terdapat beberapa fungsi penting seperti pilihPembukaan() yang digunakan untuk menetapkan pembukaan yang sedang dipilih, serta dua fungsi logging tambahan yaitu logTombolDetail() dan logTombolExplicitIntent() yang hanya bertugas mencatat ke logcat saat tombol-tombol tertentu ditekan.

4. PembukaanViewModelFactory.kt

Pada file ini, kita membuat sebuah class dengan nama PembukaanViewModelFactory yang berfungsi sebagai factory atau pabrik pembuat objek PembukaanViewModel. Kelas ini mengimplementasikan interface ViewModelProvider.Factory, yang memang biasa digunakan ketika kita ingin mengirim parameter ke dalam sebuah ViewModel. Dalam kasus ini, parameter yang dikirim adalah repository, yaitu instance dari PembukaanRepository yang akan digunakan oleh ViewModel nantinya.

Di dalam fungsi create(), kita melakukan pengecekan terlebih dahulu apakah modelClass yang diminta merupakan turunan dari PembukaanViewModel. Jika ya,

maka kita akan mengembalikan instance dari PembukaanViewModel dengan repository sebagai parameternya. Namun jika tidak cocok, maka akan dilemparkan sebuah exception dengan pesan “Unknown ViewModel class”.

5. **MainActivity.kt**

File MainActivity.kt merupakan titik masuk utama dari aplikasi, yang berfungsi untuk mengatur tampilan dan navigasi antar layar menggunakan Jetpack Compose. Di dalam onCreate, kita menginisialisasi PembukaanViewModelFactory dengan menyisipkan PembukaanRepository sebagai dependensinya, lalu menggunakan factory ini untuk mendapatkan instance dari PembukaanViewModel melalui ViewModelProvider. Ini memungkinkan ViewModel digunakan secara terpusat dan dapat mengakses data dari repository.

Selanjutnya, fungsi setContent digunakan untuk menyusun UI menggunakan Composable dalam tema Pembukaan_caturTheme. Di dalamnya terdapat struktur navigasi NavHost dengan dua rute utama, yaitu "listPembukaan" dan "penjelasan/{desc}/{img}". Rute listPembukaan menampilkan daftar pembukaan catur yang diambil dari pembukaanList, yaitu StateFlow yang dikoleksi menggunakan collectAsState. Masing-masing entri daftar ditampilkan melalui komponen TampilanEntitasPembukaan, lengkap dengan tombol untuk melihat detail atau membuka link website eksternal, dengan intent eksplisit.

Jika pengguna menekan tombol “Detail”, maka aplikasi akan melakukan navigasi ke halaman penjelasan menggunakan rute "penjelasan/{desc}/{img}". Parameter desc dan img akan diterima pada halaman tujuan sebagai argumen. Halaman penjelasan tersebut ditampilkan melalui fungsi tampilanDetail, yang menunjukkan gambar pembukaan catur, nama, dan penjelasan lengkap. Seluruh struktur ini menunjukkan penggunaan prinsip arsitektur MVVM dan Compose Navigation secara efektif dan modular, serta memanfaatkan pendekatan reactive UI dari Jetpack Compose.

2. RecyclerView masih digunakan karena memberikan kontrol penuh atas perilaku dan performa daftar seperti pengelolaan tampilan yang banyak serta efisiensi memori yang baik, hal ini menjadi sangat penting untuk aplikasi kita apabila aplikasi kita sudah berskala besar atau dengan data dinamis. Meskipun kode RecyclerView cenderung boiler-plate, yang berarti fleksibilitasnya lebih tinggi dibandingkan *LazyColumn* di Jetpack Compose, hal ini menyebabkan bahwa RecyclerView lebih cocok untuk antarmuka deklaratif dan kebutuhan yang lebih sederhana atau modern.

D. Tautan Git

https://github.com/KunyitAlami/Praktikum-Pemrograman-Mobile-I-Ghani-Mudzakir.git