LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

ViewModel and Debugging

Oleh:

Muhammad Rizky NIM. 2310817310011

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

Nama Praktikan : Muhammad Rizky
NIM : 2310817310011

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

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

A. Source Code

1. MainActivity.kt

Table 1. Source Code MainActivity.kt

```
1   package com.example.modul4
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.navigation.compose.rememberNavController
9   import com.example.modul4.navigation.AppNavGraph
10
11  class MainActivity : ComponentActivity() {
12      override fun onCreate(savedInstanceState: Bundle?) {
13          super.onCreate(savedInstanceState)
14          setContent {
15
```

```
            Surface(color =
MaterialTheme.colorScheme.background) {
                val navController =
rememberNavController()
                AppNavGraph(navController =
navController)
            }
        }
    }
}
```

2. DataTournament.kt

Table 2. Source Code DataTournament.kt

```
package com.example.modul4.data
import com.example.modul4.model.Item

val sampleItems = listOf(
Item("Indonesia Open", "Indonesia Open adalah
sebuah kejuaraan bulu tangkis internasional di Indonesia.
Kejuaraan ini diselenggarakan oleh Persatuan Bulu Tangkis
Seluruh Indonesia dengan edisi terkini berlangsung di Istora
Gelora
Bung Karno, Jakarta.","https://img.bwfbadminton.com/image/u
pload/v1746 607228/assets/tournaments/logo/8E66BAA2-CF0F
-4FFA-8798-FF700D1CDBEC.png", "https://bwfworldtour.bwfb
adminton.com/tournament/5236/indonesia-open-2025/results/2
025-06-03"),Item("Yonex All England Open", "Turnamen
bulutangkis tertua di dunia yang diadakan di Inggris sejak
1899. Saat ini menjadi bagian dari BWF World Tour Super 1000
dan sangat bergengsi karena Sejarah dan
prestisenya.","https://img.bwfbadminton.com/image/upload
/v1734076435/assets/tournaments/logo/F6304F7C-D298-4E3D
AA62A6B933CBB8C7.png","https://bwfworldtour.bwfbadminton.com/
tournament/5238/yonex-all-england-open-badminton-champions
hips-2025/results/podium/"),
Item("Victor China Open", "Turnamen bulutangkisintern
asional bergengsi yang diadakan setiap tahun di Tiongkok.
Merupakan bagian dari BWF World Tour Super 1000 (bersama
All England dan Indonesia Open). Menarik banyak pemain top
dunia karena hadiah besar dan poin peringkat
tinggi.","https://img.bwfbadminton.com/image/upload/v17
46149516/assets/tournaments/logo/5A59A1E7-1366-49BA-863D
-33FB844D51C0.png", "https://bwfworldtour.bwfbadmin
ton.com/tournament/5281/victor-china-open-2025/results
/2025-07-22"),
Item("Daihatsu Indonesia Master", "Indonesia Mast
```

```
er adalah sebuah kejuaraan bulu tangkis internasional di
Indonesia. Kejuaraan ini diselenggarakan oleh Persatuan Bulu
Tangkis Seluruh Indonesia dengan edisi terkini berlangsung di
Istora Gelora Bung Karno,
Jakarta.","https://img.bwfbadminton.com/image/upload/
v1731292348/assets/tournaments/logo/C166BF11-2128-4F8C-B1ED-
F30565DE9089.png",
"https://bwfworldtour.bwfbadminton.com/tournament/5234/
daihatsu-indonesia-masters-2025/results/podium/"),
Item("HSBC BWF World Tour Final", "Turnamen penutup
musim bulutangkis yang hanya mempertemukan 8 pemain atau
pasangan terbaik dunia dari setiap sektor berdasarkan
perolehan poin sepanjang musim BWF World Tour. Turnamen ini
sangat  bergengsi dan berhadiah besar, setara dengan kejuaraan
elit lainnya.","https://img.bwfbadminton.com/image/
upload/v1734514049/assets/tournaments/logo/F34EBDDB-AA5D-
4F1E-A14F-
A94473DD1295.png","https://bwfworldtourfinals.bwfbadm
inton.com/results/5259/hsbc-bwf-world-tour-finals-
2025/draw/"),
Item("TotalEnergies BWF Sudirman Cup Finals", "Sudirman Cup
adalah kejuaraan bulu tangkis internasional untuk nomor beregu
campuran, mempertandingkan nomor tunggal putra, tunggal putri,
ganda putra, ganda putri, dan ganda campuran.
Kejuaraan ini digelar setiap dua tahun
sekali.","https://img.bwfbadminton.com/image/upload/v
1734511970/assets/tournaments/logo/C9F72B3B-2C6A-4D63-
A782-78968933BDE9.png", "https://bwfsudirmancup.bwfb
adminton.com/results/5260/totalenergies-bwf-sudirman-cu
p-finals-2025/podium"),
ListItem("TotalEnergies BWF Thomas & Uber Cup Finals",
 "Thomas Cup dan Uber Cup adalah kejuaraan bulutangkis
beregu putra dan putri antarnegara yang diselenggarakan
dua tahun sekali oleh BWF. Setiap tim terdiri dari pemain
tunggal dan ganda, dan bertanding untuk menjadi tim
 nasional putra dan putri terbaik di dunia",
"https://bwfthomasubercups.bwfbadminton.com/wp-conten
t/plugins/bwf-menu
system/images/TUCTournamentLogoLandscape.png",
"https://bwfthomasubercups.bwfbadminton.com/results/5
147/totalenergies-bwf-thomas-uber-cup-finals-2024/podium"),
)
```

3. Item.kt

Table 3. Source Code Item.kt

```
1  package com.example.modul4.model
```

```
2
3   data class Item(
4       val id: Int,
5       val title: String,
6       val desc: String,
7       val imageUrl: String,
8       val linkUrl: String)
```

4. NavGraph.kt

Table 4. Source Code NavGraph.kt

```
1   package com.example.modul4.navigation
2
3   import androidx.compose.runtime.Composable
4   import androidx.compose.ui.Modifier
5   import androidx.lifecycle.viewmodel.compose.viewModel
6   import androidx.navigation.NavType
7   import androidx.navigation.compose.NavHost
8   import androidx.navigation.compose.composable
9   import androidx.navigation.navArgument
10  import androidx.navigation.NavHostController
11  import com.example.modul4.screen.DetailScreen
12  import com.example.modul4.screen.ListScreen
13  import com.example.modul4.viewModel.ItemViewModel
14  import com.example.modul4.viewModel.ItemViewModelFactory
15
16  @Composable
17  fun AppNavGraph(
18      navController: NavHostController,
19      modifier: Modifier = Modifier
20  ) {
21      val factory = ItemViewModelFactory()
22      val itemViewModel: ItemViewModel = viewModel(factory
23  = factory)
24
25      NavHost(navController        =        navController,
26  startDestination = "list", modifier = modifier) {
27          composable("list") {
28              ListScreen(navController   =   navController,
29  viewModel = itemViewModel)
30          }
31          composable(
32              "detail/{id}",
33              arguments = listOf(navArgument("id") {
34                  type = NavType.IntType
35              })
```

```
36          ) { backStackEntry ->
37              val                     id                      =
38  backStackEntry.arguments?.getInt("id") ?: 0
39              DetailScreen(id    =    id,    viewModel    =
40  itemViewModel)
41          }
41      }
42  }
```

5. DetailScreen.kt

Table 5. Source Code DetailScreen.kt

```
1   package com.example.modul4.screen
2
3   import androidx.compose.foundation.layout.*
4   import
5   androidx.compose.foundation.shape.RoundedCornerShape
6   import androidx.compose.material3.*
7   import androidx.compose.runtime.Composable
8   import androidx.compose.ui.Alignment
9   import androidx.compose.ui.Modifier
10  import androidx.compose.ui.draw.clip
11  import androidx.compose.ui.layout.ContentScale
12  import androidx.compose.ui.text.font.FontWeight
13  import androidx.compose.ui.unit.dp
14  import coil.compose.AsyncImage
15  import com.example.modul4.viewModel.ItemViewModel
16
17  @Composable
18  fun DetailScreen(viewModel: ItemViewModel, id: Int) {
19      val item = viewModel.getItemById(id)
20
21      if (item == null) {
22          // Item tidak ditemukan
23          Box(
24              modifier = Modifier.fillMaxSize(),
25              contentAlignment = Alignment.Center
26          ) {
27              Text(
28                  text = "Item tidak ditemukan",
29                  style                                       =
30  MaterialTheme.typography.bodyLarge.copy(
31                      fontWeight = FontWeight.Bold,
32                      color                                   =
33  MaterialTheme.colorScheme.error
34                  )
```

```
            )
        }
    } else {
        Column(
            horizontalAlignment =
Alignment.CenterHorizontally,
            modifier = Modifier.padding(16.dp)
        ) {
            Card(
                shape = RoundedCornerShape(16.dp),
                modifier = Modifier
                    .fillMaxWidth()
                    .height(250.dp)
                    .clip(RoundedCornerShape(16.dp))
            ) {
                AsyncImage(
                    model = item.imageUrl,
                    contentDescription = item.title,
                    modifier = Modifier
                        .fillMaxSize()

.clip(RoundedCornerShape(16.dp)),
                    contentScale = ContentScale.Crop
                )
            }

            Spacer(modifier = Modifier.height(24.dp))

            Text(
                text = item.title,
                style =
MaterialTheme.typography.titleMedium.copy(
                    fontWeight = FontWeight.Bold,
                    color =
MaterialTheme.colorScheme.primary
                ),
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(horizontal = 16.dp)
                    .align(Alignment.CenterHorizontally)
            )

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

            Text(
                text = "Description:",

```

```
                    style                                           =
MaterialTheme.typography.bodyMedium.copy(
                        fontWeight = FontWeight.SemiBold,
                        color                                       =
MaterialTheme.colorScheme.secondary
                    ),
                    modifier = Modifier.padding(horizontal =
16.dp)
                )

                Text(
                    text = item.desc,
                    style                                           =
MaterialTheme.typography.bodyLarge,
                    modifier = Modifier.padding(horizontal =
16.dp)
                )
            }
        }
}
```

6. ListScreen.kt

Table 6. Source Code ListScreen.kt

```
package com.example.modul4.screen

import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import
androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.clickable
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.core.content.ContextCompat
import androidx.navigation.NavController
import coil.compose.rememberAsyncImagePainter
```

```
import com.example.modul4.data.ListItem
import com.example.modul4.navigation.Screen

@Composable
fun HomeScreen(navController: NavController, items:
List<ListItem>) {
    LazyColumn(
        modifier = Modifier
            .fillMaxSize()
            .padding(8.dp)
    ) {
        items(items) { item ->
            Card(
                modifier = Modifier
                    .padding(8.dp)
                    .fillMaxWidth(),
                shape = RoundedCornerShape(16.dp),
                elevation =
CardDefaults.cardElevation(4.dp)
            ) {
                Column(
                    modifier = Modifier.padding(16.dp)
                ) {
                    Image(
                        painter =
rememberAsyncImagePainter(item.imageUrl),
                        contentDescription = null,
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(180.dp)

.clip(RoundedCornerShape(16.dp)),
                        contentScale = ContentScale.Crop
                    )
                    Spacer(modifier =
Modifier.height(8.dp))
                    Text(item.title,     fontWeight     =
FontWeight.Bold)

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

                    val context = LocalContext.current

                    Row(
                        modifier =
Modifier.fillMaxWidth(),
```

```
                        horizontalArrangement =
Arrangement.SpaceBetween
                    ) {
                        Button(onClick = {
                            val intent =
Intent(Intent.ACTION_VIEW, Uri.parse(item.url))

intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)

ContextCompat.startActivity(context, intent, null)
                        }) {
                            Text("Open Link")
                        }
                        Button(onClick = {

navController.navigate(Screen.Detail.createRoute(item.id))
                        }) {
                            Text("Detail")
                        }
                    }
                }
            }
        }
    }
}
```

7. ItemViewModel.kt

Table 7. Source Code ItemViewModel.kt

```
package com.example.modul4.viewModel

import androidx.lifecycle.ViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import com.example.modul4.model.Item
import com.example.modul4.data.sampleItems
import android.util.Log

class ItemViewModel : ViewModel() {
    private val _itemList =
MutableStateFlow<List<Item>>(emptyList())
    val itemList: StateFlow<List<Item>> = _itemList

    private val _selectedItem =
MutableStateFlow<Item?>(null)

```

```
    val selectedItem: StateFlow<Item?> get() = _selectedItem

    init {
        _itemList.value = sampleItems
        Log.d("ItemViewModel", "Item list initialized: ${sampleItems.map { it.title }}")
    }

    fun selectItem(item: Item) {
        _selectedItem.value = item
        Log.d("ItemViewModel", "Selected item: $item")
    }

    fun getItemById(id: Int): Item? {
        return _itemList.value.find { it.id == id }
    }
}
```

8. ItemViewModelFactory.kt

Table 8. Source Code ItemViewModelFactory.kt

```
package com.example.modul4.viewModel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

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

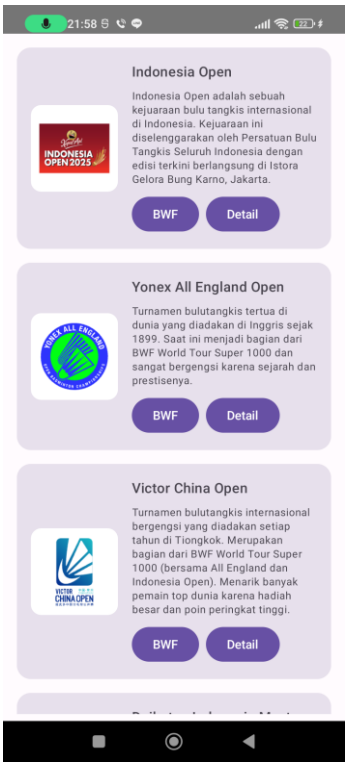
9. MainActivity.kt

Table 9. Source Code MainActivity.kt

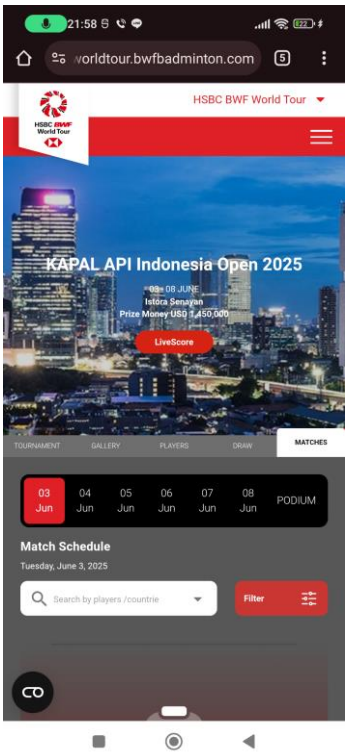
```
package com.example.modul4

```

```
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.navigation.compose.rememberNavController
9   import com.example.modul4.navigation.AppNavGraph
10
11  class MainActivity : ComponentActivity() {
12      override fun onCreate(savedInstanceState: Bundle?) {
13          super.onCreate(savedInstanceState)
14          setContent {
15              Surface(color                                    =
16  MaterialTheme.colorScheme.background) {
17                  val            navController              =
18  rememberNavController()
19                  AppNavGraph(navController                  =
20  navController)
21              }
22          }
23      }
24  }
```

B. Output Program



Gambar 1. Screenshot Halaman Awal



Gambar 2. Screenshot Ketika Tombol Ditekan Mengarah Ke Website

Gambar 3. Screenshot Ketika Tombol Detail Ditekan

C. Pembahasan

1. MainActivity.kt:

Pada line 1, dideklarasikan nama package file Kotlin, yaitu com.example.modul4.

Pada line 3 sampai 7, dilakukan import terhadap class dan fungsi yang dibutuhkan seperti ComponentActivity, setContent, komponen material 3, dan navigasi Compose.

Pada line 9, dideklarasikan class MainActivity yang merupakan subclass dari ComponentActivity. Class ini menjadi titik masuk utama dari aplikasi Android.

Pada line 10, dideklarasikan fungsi onCreate yang merupakan override dari fungsi lifecycle activity Android, dan menerima parameter savedInstanceState bertipe Bundle?.

Pada line 11, dipanggil super.onCreate(savedInstanceState) untuk menjalankan logika bawaan dari ComponentActivity.

Pada line 12, dipanggil setContent untuk menentukan UI dengan menggunakan Jetpack Compose.

Pada line 13, dibuat Surface dengan warna latar belakang sesuai dengan tema aplikasi (MaterialTheme.colorScheme.background).

Pada line 14, dideklarasikan navController menggunakan rememberNavController() yang digunakan untuk mengelola navigasi antar layar.

Pada line 15, dipanggil fungsi AppNavGraph dan diberikan parameter navController untuk mengatur jalur navigasi aplikasi.

Pada line 16, penutupan Surface.

Pada line 17, penutupan setContent.

Pada line 18, penutupan fungsi onCreate.

Pada line 19, penutupan class MainActivity.

2. DataTournament.kt

Pada line 1, dideklarasikan nama package file Kotlin, yaitu com.example.modul4.data.

Pada line 2, dilakukan import terhadap Item dari package com.example.modul4.model agar bisa digunakan dalam daftar data.

Pada line 4, dideklarasikan variabel sampleItems sebagai list yang berisi beberapa objek Item. List ini merupakan data contoh yang akan ditampilkan di aplikasi.

Pada line 5, dibuat objek Item pertama dengan id 1, judul "Indonesia Open", deskripsi mengenai turnamen, URL logo turnamen, dan URL resmi hasil turnamen.

Pada line 6, dibuat objek Item kedua dengan id 2, berjudul "Yonex All England Open", dengan deskripsi tentang sejarah turnamen tertua di dunia, URL logo, dan URL hasil turnamen.

Pada line 7, dibuat objek Item ketiga dengan id 3, berjudul "Victor China Open", beserta deskripsi turnamen, logo, dan tautan resmi hasil pertandingan.

Pada line 8, dibuat objek Item keempat dengan id 4, berjudul "Daihatsu Indonesia Master", dengan informasi deskripsi, logo turnamen, dan tautan hasil pertandingan.

Pada line 9, dibuat objek Item kelima dengan id 5, berjudul "HSBC BWF World Tour Final", dengan deskripsi tentang turnamen final musim, logo turnamen, dan URL hasil turnamen.

Pada line 10, dibuat objek Item keenam dengan id 6, berjudul "TotalEnergies BWF Sudirman Cup Finals", dengan deskripsi tentang kejuaraan beregu campuran, logo, dan tautan hasil.

Pada line 11, dibuat objek Item ketujuh dengan id 7, berjudul "TotalEnergies BWF Thomas & Uber Cup Finals", dengan deskripsi mengenai kejuaraan beregu putra dan putri, logo, dan URL hasil pertandingan.

Pada line 12, penutupan dari deklarasi list sampleItems.

3. Item.kt

Pada line 1, dideklarasikan nama package file Kotlin, yaitu com.example.modul4.model.

Pada line 3, didefinisikan sebuah data class dengan nama Item. Data class digunakan untuk merepresentasikan data dan secara otomatis menyediakan fungsi seperti toString(), equals(), dan hashCode().

Pada line 4, dideklarasikan properti id bertipe Int yang berfungsi sebagai identifikasi unik setiap item.

Pada line 5, dideklarasikan properti title bertipe String yang berfungsi sebagai judul atau nama dari item.

Pada line 6, dideklarasikan properti desc bertipe String yang berisi deskripsi atau penjelasan tentang item.

Pada line 7, dideklarasikan properti imageUrl bertipe String yang menyimpan URL dari gambar atau logo item.

Pada line 8, dideklarasikan properti linkUrl bertipe String yang berisi tautan menuju halaman detail atau hasil turnamen.

Pada line 9, dilakukan penutupan deklarasi class Item.

4. NavGraph.kt

Pada line 1, dideklarasikan nama package file Kotlin, yaitu com.example.modul4.navigation.

Pada line 3–9, dilakukan import terhadap berbagai komponen yang dibutuhkan untuk membangun navigasi dan tampilan menggunakan Jetpack Compose, seperti Composable, NavHost, composable, dan NavHostController.

Pada line 10–13, dilakukan import terhadap layar/tampilan DetailScreen dan ListScreen, serta ItemViewModel dan ItemViewModelFactory yang akan digunakan untuk manajemen data.

Pada line 15, didefinisikan fungsi AppNavGraph sebagai fungsi @Composable yang menerima parameter navController bertipe NavHostController dan modifier opsional bertipe Modifier.

Pada line 17, dibuat instance ItemViewModelFactory yang akan digunakan untuk menghasilkan ItemViewModel.

Pada line 18, dibuat instance ItemViewModel menggunakan fungsi viewModel() dengan factory yang telah dibuat.

Pada line 20, dideklarasikan NavHost yang menjadi container untuk semua destinasi navigasi. Start destination-nya adalah "list", dan modifier diteruskan dari parameter fungsi.

Pada line 21–23, didefinisikan destinasi "list" yang akan memanggil fungsi ListScreen dan mengoper navController serta itemViewModel.

Pada line 24–31, didefinisikan destinasi "detail/{id}" yang menerima argumen id bertipe Int. Nilai id diambil dari backStackEntry dan diteruskan ke DetailScreen bersama dengan itemViewModel.

Pada line 32, dilakukan penutupan dari blok NavHost.

Pada line 33, dilakukan penutupan dari fungsi AppNavGraph.

5. DetailScreen.kt

Pada line 1, dideklarasikan nama package file Kotlin, yaitu com.example.modul4.screen.

Pada line 3–13, dilakukan import terhadap berbagai komponen Jetpack Compose yang digunakan dalam pembuatan UI, seperti layout (Column, Box, Spacer), elemen visual (Text, Card), dan style (RoundedCornerShape, FontWeight, dll).

Pada line 14, dilakukan import terhadap AsyncImage dari library Coil untuk menampilkan gambar dari URL.

Pada line 15, dilakukan import terhadap ItemViewModel dari package com.example.modul4.viewModel yang digunakan untuk mendapatkan data item.

Pada line 17, didefinisikan fungsi DetailScreen sebagai fungsi @Composable dengan parameter viewModel dan id.

Pada line 18, data item diambil dari viewModel menggunakan fungsi getItemById(id).

Pada line 20–28, dilakukan pengecekan apakah item ditemukan atau tidak. Jika item == null, maka akan ditampilkan teks "Item tidak ditemukan" di tengah layar dengan gaya teks berwarna merah dan tebal.

Pada line 29, jika item ditemukan, UI akan ditampilkan dalam bentuk kolom (Column) dengan posisi horizontal di tengah dan padding sebesar 16dp.

Pada line 30–37, sebuah Card dibuat untuk menampilkan gambar dari item dengan bentuk rounded corner dan tinggi 250dp. Di dalamnya, AsyncImage digunakan untuk menampilkan gambar berdasarkan item.imageUrl. Gambar akan diisi penuh (fillMaxSize()) dan dipotong agar sesuai dengan bentuk rounded.

Pada line 39, Spacer digunakan untuk memberikan jarak vertikal sebesar 24dp.

Pada line 41–47, ditampilkan judul item (item.title) dengan gaya teks tebal dan warna utama aplikasi. Teks memenuhi lebar penuh dan ditempatkan di tengah.

Pada line 49–54, label "Description:" ditampilkan sebelum deskripsi item, menggunakan gaya teks semi-bold dan warna sekunder.

Pada line 56–58, deskripsi lengkap dari item (item.desc) ditampilkan dalam gaya teks body.

Pada line 59, penutupan dari blok fungsi DetailScreen.

6. ListScreen.kt

Pada line 1, dideklarasikan package com.example.modul4.screen.
Pada line 3–5, dilakukan import terhadap Intent, Uri, dan Log yang digunakan untuk eksplisit intent dan logging.
Pada line 6–31, berbagai komponen dari Jetpack Compose diimpor, seperti layout (Row, Column, Spacer), gambar (AsyncImage), tema (MaterialTheme), dan lain-lain.
Pada line 32, LocalContext diimpor untuk mendapatkan context saat ini.
Pada line 33, viewModel digunakan dari lifecycle library untuk menghubungkan ke ItemViewModel.
Pada line 34, NavController diimpor dari Navigation Compose.
Pada line 35, AsyncImage dari Coil digunakan untuk memuat gambar dari URL.
Pada line 36, ItemViewModel diimpor dari package viewModel.
Pada line 38, didefinisikan fungsi @Composable bernama ListScreen, dengan parameter navController dan viewModel.
Pada line 39, itemList diambil dari viewModel.itemList menggunakan collectAsState().
Pada line 40, context didapatkan dari LocalContext.current, digunakan untuk membuat intent eksplisit.
Pada line 42, digunakan LazyColumn untuk menampilkan daftar item secara efisien.
Pada line 45, digunakan items() untuk menampilkan setiap item dalam itemList.
Pada line 46–52, sebuah Card digunakan untuk menampilkan setiap item, dengan bentuk rounded dan padding. Komponen ini juga clickable, yang akan memanggil selectItem() di viewModel dan menavigasi ke screen detail berdasarkan item.id.
Pada line 53–60, digunakan Row untuk menyusun gambar dan teks secara horizontal.
Pada line 61–66, AsyncImage menampilkan gambar dengan ukuran 100dp dan rounded corner.
Pada line 68, Spacer digunakan untuk memberikan jarak horizontal.
Pada line 70–74, Column digunakan untuk menampilkan judul dan deskripsi item secara vertikal.
Pada line 75, Row digunakan untuk menampung dua Button: satu untuk membuka link, satu lagi untuk menavigasi ke detail.

Pada line 76–81, Button pertama menjalankan explicit intent (Intent.ACTION_VIEW) ke item.linkUrl, membuka browser atau aplikasi lain. Pada line 83–87, Button kedua menjalankan navigasi ke detail screen ("detail/${item.id}") secara langsung dari tombol "Detail".

Pada line 89, fungsi ListScreen ditutup.

7. ItemViewModel.kt

Pada line 1, dideklarasikan package com.example.modul4.viewModel.

Pada line 3 sampai 8, dilakukan import terhadap beberapa library dan class yang dibutuhkan, yaitu ViewModel dari Android Jetpack Lifecycle untuk manajemen siklus hidup data, MutableStateFlow dan StateFlow dari Kotlin Coroutines yang digunakan untuk mengelola data secara reaktif dan dapat diobservasi, class Item dari package model yang merepresentasikan data item, sampleItems sebagai data contoh dari package data, serta Log untuk pencatatan log debug.

Pada line 10, dideklarasikan class ItemViewModel yang merupakan turunan dari ViewModel.

Pada line 11, dibuat variabel privat _itemList bertipe MutableStateFlow yang menyimpan daftar item dengan nilai awal berupa list kosong.

Pada line 12, dideklarasikan variabel publik itemList bertipe StateFlow yang hanya bisa dibaca dan mengambil nilai dari _itemList, sehingga data item dapat dipantau dari luar class tanpa bisa diubah langsung.

Pada line 14, dibuat variabel privat _selectedItem bertipe MutableStateFlow yang menyimpan item yang sedang dipilih, dengan nilai awal null.

Pada line 15, variabel publik selectedItem bertipe StateFlow disediakan sebagai getter untuk mengakses nilai _selectedItem secara read-only.

Pada line 17 sampai 20, terdapat blok init yang merupakan inisialisasi ketika ItemViewModel dibuat. Pada blok ini, nilai _itemList diisi dengan data contoh sampleItems. Selain itu, sebuah log debug dicatat yang menampilkan daftar judul item yang sudah diinisialisasi.

Pada line 22 sampai 25, terdapat fungsi selectItem yang menerima parameter item bertipe Item. Fungsi ini mengubah nilai _selectedItem menjadi item yang dipilih dan mencatat log debug terkait item tersebut.

Pada line 27 sampai 29, terdapat fungsi getItemById yang menerima parameter id bertipe Int. Fungsi ini mencari dan mengembalikan objek Item dari _itemList yang memiliki id sesuai parameter. Jika tidak ditemukan, fungsi mengembalikan null.

8. ItemViewModelfactory.kt

Pada line 1, dideklarasikan package com.example.modul4.viewModel.

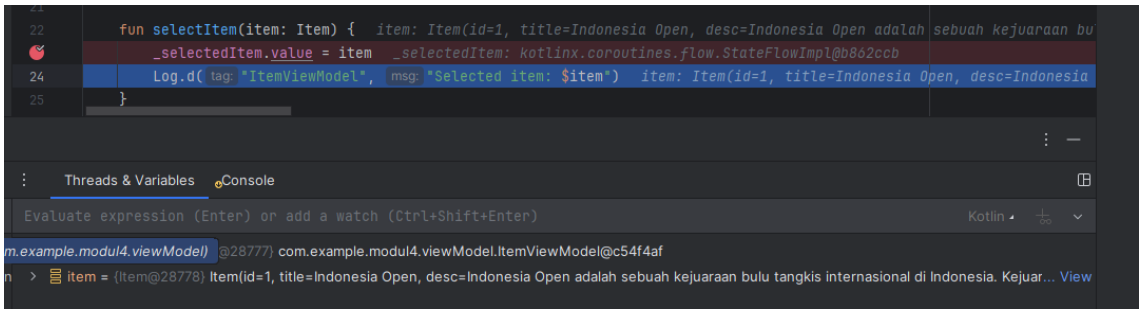
Pada line 3 dan 4, dilakukan import terhadap ViewModel dan ViewModelProvider dari Android Jetpack Lifecycle yang digunakan untuk membuat dan menyediakan instance ViewModel dengan siklus hidup yang benar.

Pada line 6 sampai 13, dideklarasikan class ItemViewModelFactory yang mengimplementasikan interface ViewModelProvider.Factory.

Pada line 7 sampai 11, terdapat override fungsi create yang bertugas membuat instance ViewModel berdasarkan kelas yang diminta. Fungsi ini menerima parameter modelClass bertipe Class<T>, kemudian memeriksa apakah kelas tersebut merupakan subclass dari ItemViewModel. Jika ya, maka dibuat dan dikembalikan instance ItemViewModel yang di-cast ke tipe T. Jika bukan, maka fungsi melemparkan IllegalArgumentException dengan pesan "Unknown ViewModel class" sebagai tanda bahwa kelas ViewModel yang diminta tidak dikenali atau tidak didukung oleh factory ini.

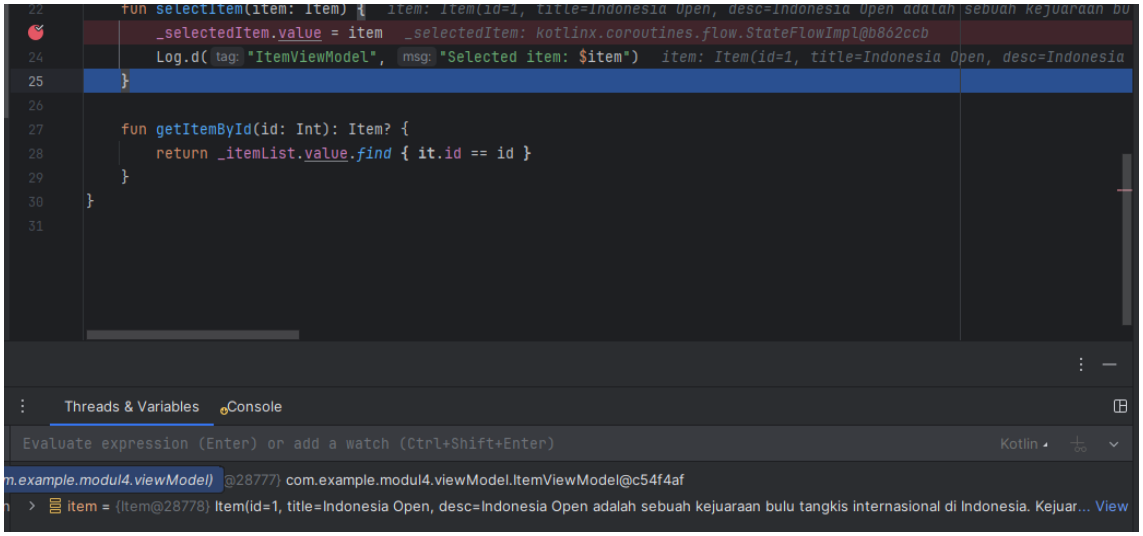
Debugging

1. Step Into



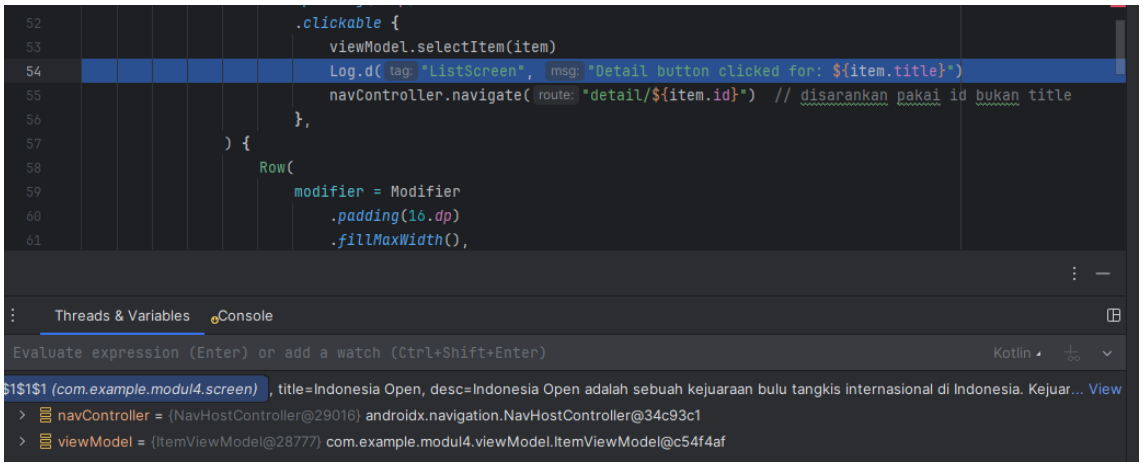
Ketika menekan Step Into di baris _selectedItem.value = item, debugger akan masuk ke implementasi properti value dari MutableStateFlow. Karena ini adalah bagian dari library Kotlin Coroutine, maka akan terlihat kode internal yang mengatur nilai baru disimpan dan bagaimana observer yang melihat perubahan nilai diberi tahu.

2. **Step Over**



Ketika melakukan Step Over, maka debugger akan mengeksekusi seluruh instruksi _selectedItem.value = item secara utuh tanpa masuk ke implementasi internalnya. Ini berguna jika tidak ingin melihat detail internal MutableStateFlow, dan langsung lanjut ke baris kode berikutnya.

3. **Step Out**



Jika sudah masuk ke dalam fungsi internal atau method dan ingin keluar kembali ke kode pemanggil, maka bisa menggunakan Step Out. Step Out akan mengeksekusi sisa

kode di dalam fungsi tersebut dan mengembalikan kontrol ke baris setelah pemanggilan fungsi yang kamu masuki sebelumnya.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/kkeshiian/Pemrograman-Mobilefix

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

A. Jawaban

Application class dalam arsitektur aplikasi Android adalah kelas dasar yang dijalankan terlebih dahulu saat aplikasi mulai berjalan. Kelas ini merupakan subclass dari android.app.Application dan berfungsi sebagai titik awal untuk menginisialisasi komponen atau data global yang diperlukan oleh seluruh aplikasi selama siklus hidupnya.

Fungsi utama Application class adalah menyediakan konteks aplikasi yang bersifat global dan bertahan selama aplikasi aktif. Dengan Application class, developer dapat melakukan inisialisasi satu kali seperti setup library pihak ketiga, konfigurasi database, atau penyimpanan data yang harus tersedia secara konsisten di seluruh bagian aplikasi. Selain itu, Application class membantu mengelola state aplikasi secara keseluruhan dan dapat digunakan untuk menyimpan data atau variabel yang perlu diakses oleh banyak komponen aplikasi tanpa perlu membuat ulang atau melewatkan data melalui Intent atau Bundle.