LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

ViewModel and Debugging

Oleh:

Muhammad Rizki Ramadhan NIM. 2310817310008

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 4

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

Nama Praktikan : Muhammad Rizki Ramadhan
NIM : 2310817310008

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Raka Azwar NIM. 2210817210012	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 19930703 20190301011

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 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 untuk membuat parameter dengan tipe data String di dalam ViewModel
c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
d. Install dan gunakan library Timber untuk logging 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
    d) Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi.
e. 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

XML:

MainActivity.kt

Table 1. MainActivity.kt

```
1  package com.example.scrollablexml
2
3  import android.content.Intent
4  import android.net.Uri
5  import android.os.Bundle
6  import androidx.activity.viewModels
7  import androidx.appcompat.app.AppCompatActivity
8  import androidx.lifecycle.lifecycleScope
9  import androidx.recyclerview.widget.LinearLayoutManager
10 import androidx.recyclerview.widget.RecyclerView
11 import com.example.scrollablexml.ui.ChocolateAdapter
12 import com.example.scrollablexml.ui.DetailActivity
13 import com.example.scrollablexml.viewmodel.ChocolateViewModel
14 import
   com.example.scrollablexml.viewmodel.ChocolateViewModelFactory
15 import kotlinx.coroutines.launch
16 import timber.log.Timber
17
18 class MainActivity : AppCompatActivity() {
19     private val viewModel: ChocolateViewModel by viewModels {
20         ChocolateViewModelFactory("MainActivity")
21     }
22
23     override fun onCreate(savedInstanceState: Bundle?) {
24         super.onCreate(savedInstanceState)
25         Timber.plant(Timber.DebugTree())
26         setContentView(R.layout.activity_main)
27
28         val        recyclerView:        RecyclerView        =
   findViewById(R.id.recyclerView)
```

```
29          recyclerView.layoutManager                                  =
    LinearLayoutManager(this)
30
31          lifecycleScope.launch {
32              viewModel.chocolateList.collect { chocolate ->
33                  recyclerView.adapter                                =
    ChocolateAdapter(this@MainActivity, chocolate,
34                      onDetailClick = {
35                          Timber.d("Detail clicked: $it")
36                          viewModel.onChocolateSelected(it)
37                          val             intent             =
    Intent(this@MainActivity, DetailActivity::class.java)
38                          intent.putExtra("id", it.id)
39                          startActivity(intent)
40                      },
41                      onLinkClick = {
42                          Timber.d("IMDB               clicked:
    ${it.btnLink}")
43                          val             intent             =
    Intent(Intent.ACTION_VIEW, Uri.parse(it.btnLink))
44                          startActivity(intent)
45                      }
46                  )
47              }
48          }
49      }
50  }
```

Chocolate.kt

Table 2. Chocolate.kt

```
1  package com.example.scrollablexml.data
2
3  data class Chocolate(
4      val id: Int,
5      val name: String,
6      val description: String,
7      val chocolateImageId: Int,
8      val btnLink: String
9  )
```

ChocolateProvider.kt

Table 3. ChocolateProvider.kt

```
1   package com.example.scrollablexml.data
2
3   import com.example.scrollablexml.R
4
5   object ChocolateProvider {
6       val chocolateList = listOf(
            Chocolate(1, "Dark Chocolate", "Dark chocolate is
    a form of chocolate made from cocoa solids, cocoa butter
    and sugar. " +
7                   "It has a higher cocoa percentage than
    white   chocolate,   milk   chocolate,   and   semisweet
    chocolate.",                  R.drawable.darkchocolate,
    "https://en.wikipedia.org/wiki/Dark_chocolate"),
8           Chocolate(2, "Milk Chocolate", "Milk chocolate is
    a form of solid chocolate containing cocoa, sugar and
    milk. " +
9                   "Milk chocolate contains smaller amounts
    of cocoa solids than dark chocolates do, and (as with
    white     chocolate)     contains     milk     solids.",
    R.drawable.milkchocolate,
    "https://en.wikipedia.org/wiki/Milk_chocolate"),
10          Chocolate(3, "White Chocolate", "White chocolate
    is a chocolate made from cocoa butter, sugar and milk
    solids. " +
11                  "It is ivory in color and lacks the dark
    appearance of most other types of chocolate as it does
    not contain the non-fat components of cocoa (cocoa
    solids).",                    R.drawable.whitechocolate,
    "https://en.wikipedia.org/wiki/White_chocolate"),
12          Chocolate(4,  "Hazelnut  Chocolate",  "Hazelnut
    chocolate adds a new dimension to the flavor by combining
    the flavor of chocolate with the distinctive hazelnut
    nut. " +
13                  "The hazelnut    provides a savory,
    crunchy, and slightly sweet touch that complements the
```

```
    richness of the chocolate.",
    R.drawable.huzelnutchocolate,
    "https://en.wikipedia.org/wiki/Chocolate#Flavors"),
14          Chocolate(5, "Mint Chocolate", "The hazelnut
    provides a savory, crunchy, and slightly sweet touch that
    complements the richness of the chocolate. " +
15                  "Mint chocolate can be found in a wide
    variety of confectionery items, such as candy, mints,
    cookies, mint chocolate chip ice cream, hot chocolate,
    and others. ", R.drawable.mintchocolate,
    "https://en.wikipedia.org/wiki/Chocolate#Flavors")
        )
16  }
17
```

ChocolateAdapter.kt

Table 4. ChocolateAdapter.kt

```
1   package com.example.scrollablexml.ui
2
3   import android.content.Context
4   import android.view.LayoutInflater
5   import android.view.View
6   import android.view.ViewGroup
7   import android.widget.Button
8   import android.widget.ImageView
9   import android.widget.TextView
10  import androidx.recyclerview.widget.RecyclerView
11  import com.example.scrollablexml.R
12  import com.example.scrollablexml.data.Chocolate
13
14  class ChocolateAdapter(
15      private val context: Context,
16      private val chocolateList: List<Chocolate>,
17      private val onDetailClick: (Chocolate) -> Unit,
18      private val onLinkClick: (Chocolate) -> Unit
19  ) : RecyclerView.Adapter<ChocolateAdapter.ViewHolder>() {

20      inner class ViewHolder(view: View) :
21  RecyclerView.ViewHolder(view) {
```

```
22          val image: ImageView =
    view.findViewById(R.id.imageChocolate)
23          val name: TextView = view.findViewById(R.id.textName)
            val description: TextView =
24  view.findViewById(R.id.textDescription)
            val btnImdb: Button =
25  view.findViewById(R.id.buttonImdb)
            val btnDetail: Button =
26  view.findViewById(R.id.buttonDetail)
        }
27
28      override fun onCreateViewHolder(parent: ViewGroup,
    viewType: Int): ViewHolder {
29          val view =
    LayoutInflater.from(context).inflate(R.layout.row_chocolate,
    parent, false)
30          return ViewHolder(view)
31      }
32
33      override fun getItemCount(): Int = chocolateList.size
34
35      override fun onBindViewHolder(holder: ViewHolder,
    position: Int) {
36          val chocolate = chocolateList[position]
37
38  holder.image.setImageResource(chocolate.chocolateImageId)
39          holder.name.text = chocolate.name
40          holder.description.text = chocolate.description
41
42          holder.btnImdb.setOnClickListener {
43              onLinkClick(chocolate)
44          }
45
46          holder.btnDetail.setOnClickListener {
47              onDetailClick(chocolate)
48          }
49      }
50  }
```

DetailActivity.kt

Table 5. DetailActivity.kt

```
1  package com.example.scrollablexml.ui
2
3  import android.os.Bundle
4  import android.widget.ImageView
5  import android.widget.TextView
6  import androidx.appcompat.app.AppCompatActivity
7  import com.example.scrollablexml.R
8  import com.example.scrollablexml.data.ChocolateProvider
9  import timber.log.Timber
10
11 class DetailActivity : AppCompatActivity() {
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         Timber.d("DetailActivity opened")
15
16         setContentView(R.layout.activity_detail)
17         val id = intent.getIntExtra("id", -1)
18         val                  chocolate                  =
   ChocolateProvider.chocolateList.find { it.id == id }
19
20         chocolate?.let {
21             Timber.d("Displaying chocolate detail: $it")
22
23 findViewById<ImageView>(R.id.imageChocolateDetail).setI
   mageResource(it.chocolateImageId)
24
25 findViewById<TextView>(R.id.textTitleDetail).text      =
   it.name
26
27 findViewById<TextView>(R.id.textDescriptionDetail).text
   = it.description
28         }
29     }
30 }
```

ChocolateViewModel.kt

Table 6. ChocolateViewModel.kt

```
1  package com.example.scrollablexml.viewmodel
2
3  import androidx.lifecycle.ViewModel
4  import com.example.scrollablexml.data.Chocolate
5  import com.example.scrollablexml.data.ChocolateProvider
6  import kotlinx.coroutines.flow.MutableStateFlow
7  import kotlinx.coroutines.flow.StateFlow
8  import timber.log.Timber
9
10 class ChocolateViewModel(private val source: String) :
11 ViewModel() {
12
13     private val _chocolateList =
   MutableStateFlow<List<Chocolate>>(emptyList())
14     val chocolateList: StateFlow<List<Chocolate>> get()
   = _chocolateList
15
       private val _selectedChocolate =
16 MutableStateFlow<Chocolate?>(null)

17     init {
           Timber.d("ViewModel created with source:
18 $source")
           _chocolateList.value =
19 ChocolateProvider.chocolateList
           Timber.d("Chocolate list loaded:
20 ${_chocolateList.value}")
21     }
22
       fun onChocolateSelected(chocolate: Chocolate) {
23         _selectedChocolate.value = chocolate
           Timber.d("Selected chocolate: $chocolate")
24     }
25 }
```

ChocolateViewModelFactory.kt

Table 7. ChocolateViewModelFactory.kt

```
1  package com.example.scrollablexml.viewmodel
2
3  import androidx.lifecycle.ViewModel
4  import androidx.lifecycle.ViewModelProvider
5
6  class ChocolateViewModelFactory(private val source:
7  String) : ViewModelProvider.Factory {
8      override fun <T : ViewModel> create(modelClass:
   Class<T>): T {
9          return ChocolateViewModel(source) as T
10     }
11 }
```

activity_main.xml

Table 8. activity_main.xml

```
1  <LinearLayout
2  xmlns:android="http://schemas.android.com/apk/res/android"

3  android:orientation="vertical"
4  android:layout_width="match_parent"
5      android:layout_height="match_parent">
6
7      <androidx.recyclerview.widget.RecyclerView
8          android:id="@+id/recyclerView"
9          android:layout_width="match_parent"
10         android:layout_height="match_parent"/>
11 </LinearLayout>
```

activity_detail.xml

Table 9. activity_detail.xml

```
1  <ScrollView
2  xmlns:android=http://schemas.android.com/apk/res/android
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      xmlns:app="http://schemas.android.com/apk/res-auto"
6      android:padding="16dp">
7
8      <androidx.constraintlayout.widget.ConstraintLayout
9          android:layout_width="match_parent"
10         android:layout_height="wrap_content">
11
12         <ImageView
13             android:id="@+id/imageChocolateDetail"
14             android:layout_width="0dp"
15             android:layout_height="240dp"
16             android:scaleType="centerCrop"
17             app:layout_constraintTop_toTopOf="parent"
18             app:layout_constraintStart_toStartOf="parent"
19             app:layout_constraintEnd_toEndOf="parent" />
20
21         <TextView
22             android:id="@+id/textTitleDetail"
23             android:layout_marginTop="16dp"
24             android:textSize="24sp"
25             android:textStyle="bold"
26             android:layout_width="0dp"
27             android:layout_height="wrap_content"
28
29 app:layout_constraintTop_toBottomOf="@id/imageChocolateDetail"
30             app:layout_constraintStart_toStartOf="parent"
31             app:layout_constraintEnd_toEndOf="parent" />
32
33         <TextView
34             android:id="@+id/textLabelPlot"
35             android:textStyle="bold"
```

```
            android:layout_marginTop="8dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

app:layout_constraintTop_toBottomOf="@id/textTitleDetail"
            app:layout_constraintStart_toStartOf="parent" />

        <TextView
            android:id="@+id/textDescriptionDetail"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"

app:layout_constraintTop_toBottomOf="@id/textLabelPlot"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintBottom_toBottomOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>
```

row_chocolate.xml

Table 10. row_chocolate.xml

```
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:padding="16dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/imageChocolateDetail"
            android:layout_width="0dp"
            android:layout_height="240dp"
            android:scaleType="centerCrop"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintStart_toStartOf="parent"
```

```
            app:layout_constraintEnd_toEndOf="parent" />

        <TextView
            android:id="@+id/textTitleDetail"
            android:layout_marginTop="16dp"
            android:textSize="24sp"
            android:textStyle="bold"
            android:layout_width="0dp"
            android:layout_height="wrap_content"

app:layout_constraintTop_toBottomOf="@id/imageChocolateDetail"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent" />

        <TextView
            android:id="@+id/textLabelPlot"
            android:textStyle="bold"
            android:layout_marginTop="8dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

app:layout_constraintTop_toBottomOf="@id/textTitleDetail"
            app:layout_constraintStart_toStartOf="parent" />

        <TextView
            android:id="@+id/textDescriptionDetail"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"

app:layout_constraintTop_toBottomOf="@id/textLabelPlot"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintBottom_toBottomOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>
```

Jetpack Compose:

MainActivity.kt

Table 11. MainActivity.kt

```
1  package com.example.scrollablecompose
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
6  import androidx.activity.enableEdgeToEdge
7  import androidx.compose.foundation.layout.fillMaxSize
8  import androidx.compose.foundation.layout.padding
9  import androidx.compose.foundation.lazy.LazyColumn
10 import androidx.compose.foundation.lazy.items
11 import androidx.compose.material3.Scaffold
12 import androidx.compose.runtime.collectAsState
13 import androidx.compose.runtime.getValue
14 import androidx.compose.ui.Modifier
15 import androidx.lifecycle.viewmodel.compose.viewModel
16 import androidx.navigation.compose.NavHost
17 import androidx.navigation.compose.composable
18 import androidx.navigation.compose.rememberNavController
19 import com.example.scrollablecompose.data.ChocolateProvider
20
21 import com.example.scrollablecompose.uiScreen.DetailScreen
22 import
23 com.example.scrollablecompose.ui.theme.ScrollableComposeTheme
24 import com.example.scrollablecompose.uiComponent.ChocolateItem
25 import com.example.scrollablecompose.uiComponent.ListScreen
26 import com.example.scrollablecompose.viewModel.ChocolateViewModel
27 import
   com.example.scrollablecompose.viewModel.ChocolateViewModelFactory
   import timber.log.Timber
28
29 class MainActivity : ComponentActivity() {
30     override fun onCreate(savedInstanceState: Bundle?) {
31         super.onCreate(savedInstanceState)
32         Timber.plant(Timber.DebugTree())
```

```
33          enableEdgeToEdge()
34          setContent {
35              val navController = rememberNavController()
36              val viewModel: ChocolateViewModel = viewModel(
37                  factory  =  ChocolateViewModelFactory("parameter-
mu")
38              )
39
40              ScrollableComposeTheme {
41                  Scaffold(modifier  =  Modifier.fillMaxSize())  {
innerPadding ->
42                      NavHost(
43                          navController = navController,
44                          startDestination = "list",
45                          modifier = Modifier.padding(innerPadding)
46                      ) {
47                          composable("list") {
48                              val          chocolateList          by
viewModel.chocolateList.collectAsState()
49                              ListScreen(
50                                  chocolates = chocolateList,
52                                  onDetailClicked = {
53                                      Timber.d("Navigating       to
Detail of: ${it.title}")
54
55  navController.navigate("detail/${it.no}")
56                                  }
57                              )
58                          }
59
60                          composable("detail/{id}")               {
backStackEntry ->
61                              val                id                =
backStackEntry.arguments?.getString("id")?.toIntOrNull()
62                              DetailScreen(id  =  id,  viewModel  =
viewModel)
63                          }
64                      }
65                  }
66              }
67
```

```
        }
    }
}
```

Chocolate.kt

Table 12. Chocolate.kt

```
package com.example.scrollablecompose.data

data class Chocolate(
    val no: Int,
    val title: String,
    val description: String,
    val chocolateImageId: Int,
    val btnLink: String
)
```

ChocolateProvider.kt

Table 13. ChocolateProvider.kt

```
package com.example.scrollablecompose.data

import com.example.scrollablecompose.R

object ChocolateProvider {
        val chocolateList = listOf(
            Chocolate(
                no = 1,
                title = "Dark Chocolate",
                description = "Dark chocolate is a form of chocolate made from cocoa solids, cocoa butter and sugar. " +
                        "It has a higher cocoa percentage than white chocolate, milk chocolate, and semisweet chocolate.",
                chocolateImageId = R.drawable.darkchocolate,
                btnLink = "https://en.wikipedia.org/wiki/Dark_chocolate"
            ),
```

```
14              Chocolate(
15                  no = 2,
16
17                  title = "Milk Chocolate",
                    description = "Milk chocolate is a form
18  of solid chocolate containing cocoa, sugar and milk. " +
                            "Milk chocolate contains smaller
    amounts of cocoa solids than dark chocolates do, and (as
19  with white chocolate) contains milk solids.",
                    chocolateImageId                        =
20  R.drawable.milkchocolate,
                    btnLink                                 =
21  "https://en.wikipedia.org/wiki/Milk_chocolate"
22              ),
23              Chocolate(
24                  no = 3,
25                  title = "White Chocolate",
                    description  =  "White  chocolate  is  a
    chocolate made from cocoa butter, sugar and milk solids.
26  " +
                            "It is ivory in color and lacks
    the dark appearance of most other types of chocolate as
    it does not contain the non-fat components of cocoa (cocoa
27  solids).",
                    chocolateImageId                        =
28  R.drawable.whitechocolate,
                    btnLink                                 =
29  "https://en.wikipedia.org/wiki/White_chocolate"
30              ),
31              Chocolate(
32                  no = 4,
33                  title = "Hazelnut Chocolate",
                    description = "Hazelnut  chocolate adds
    a new dimension to the flavor by combining the flavor of
34  chocolate with the distinctive hazelnut  nut. " +
                            "The hazelnut  provides a savory,
    crunchy, and slightly sweet touch that complements the
    richness of the chocolate.",
35                  chocolateImageId                        =
    R.drawable.huzelnutchocolate,
36
```

```
                btnLink                                          =
    "https://kioskcokelat.com/blogs/news/perbedaan-coklat-
37  dan-coklat-hazelnut"
38          ),
39          Chocolate(
40              no = 5,
41              title = "Mint Chocolate",
                description  =  "Mint  chocolate  (or
    chocolate mint) is a popular flavor of chocolate, made
    by  adding  a  mint  flavoring,  such  as  peppermint,
42  spearmint, or crème de menthe, to chocolate. " +
                        "Mint chocolate can be found in
    a  wide  variety  of  confectionery  items,  such  as  candy,
    mints,  cookies,  mint  chocolate  chip  ice  cream,  hot
43  chocolate, and others.",
                chocolateImageId                                 =
44  R.drawable.mintchocolate,
45              btnLink                                          =
46  "https://en.wikipedia.org/wiki/Chocolate#Flavors"
47          )
        )
    }
```

ChocolateItem.kt

Table 14. ChocolateItem.kt

```
1   package          com.example.scrollablecompose.uiComponent
2
3   import                                android.content.Intent
4   import                                       android.net.Uri
5   import                    androidx.compose.foundation.Image
6   import             androidx.compose.foundation.layout.Column
7   import                androidx.compose.foundation.layout.Row
8   import             androidx.compose.foundation.layout.Spacer
9   import      androidx.compose.foundation.layout.fillMaxWidth
10  import             androidx.compose.foundation.layout.height
11  import           androidx.compose.foundation.layout.padding
12  import              androidx.compose.foundation.layout.width
13  import androidx.compose.foundation.shape.RoundedCornerShape
14  import                    androidx.compose.material3.Button
15  import                      androidx.compose.material3.Card
```

```
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
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.res.painterResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import com.example.scrollablecompose.R
import com.example.scrollablecompose.data.Chocolate
import timber.log.Timber

@Composable
fun ChocolateItem(
    chocolate: Chocolate,
    onDetailClicked: (Chocolate) -> Unit
) {
    val context = LocalContext.current

    Card(
        modifier = Modifier
            .padding(8.dp)
            .fillMaxWidth(),
        shape = RoundedCornerShape(16.dp)
    ) {
        Row(
            modifier = Modifier.padding(8.dp),
            verticalAlignment = Alignment.CenterVertically
        ) {

            Image(
                painter = painterResource(id = chocolate.chocolateImageId),
                contentDescription = null,
                contentScale = ContentScale.Crop,
                modifier = Modifier
                    .width(100.dp)
                    .height(140.dp)
```

```
56                      .clip(RoundedCornerShape(12.dp))
57              )
58
59              Spacer(modifier     =     Modifier.width(12.dp))
60
61              Column(
62                  modifier              =              Modifier
63                      .padding(16.dp)
64                      .weight(1f)
65              )                                                 {
66                  Text(text         =         chocolate.title)
67                  Text(text    =    "No:    ${chocolate.no}")
68                  Text(
69                      text      =      chocolate.description,
70                      style                                     =
    MaterialTheme.typography.bodySmall,
71                      modifier = Modifier.padding(top = 4.dp)
72                  )
73
74                  Row(modifier = Modifier.padding(top = 8.dp))
75  {
76                      Button(onClick           =           {
77                          Timber.d("Link   button   clicked:
    ${chocolate.btnLink}")
78                          val             intent             =
    Intent(Intent.ACTION_VIEW,    Uri.parse(chocolate.btnLink))
79                          context.startActivity(intent)
80                      })                                        {
81                          Text("IMDB")
82                      }
83
84                      Spacer(modifier                           =
    Modifier.padding(horizontal             =             4.dp))
85
86                      Button(onClick           =           {
87                          Timber.d("Detail   button   clicked:
    ${chocolate.title}")
88                          onDetailClicked(chocolate)
89
90                      })                                        {
91                          Text("Detail")
```

```
                    }
                }
            }
        }
    }
}


@Preview
@Composable
private fun ChocolateItemPreview(){
    val chocolate = Chocolate(
        no = 1,
        title = "Dark Chocolate",
        description = "Dark chocolate is a form of chocolate made from cocoa solids, cocoa butter and sugar. " +
                "It has a higher cocoa percentage than white chocolate, milk chocolate, and semisweet chocolate.",
        chocolateImageId = R.drawable.darkchocolate,
        btnLink = "https://en.wikipedia.org/wiki/Dark_chocolate"
    )
    ChocolateItem(
        chocolate= chocolate,
        onDetailClicked= { }

    )
}
```

ChocolateListScreen.kt

Table 15. ChocolateListScreen.kt

```
1   package com.example.scrollablecompose.uiComponent
2
3   import androidx.compose.foundation.lazy.LazyColumn
4   import androidx.compose.foundation.lazy.items
5   import androidx.compose.runtime.Composable
6   import com.example.scrollablecompose.data.Chocolate
7
8   @Composable
9   fun ListScreen(
10      chocolates: List<Chocolate>,
11      onDetailClicked: (Chocolate) -> Unit
12  ) {
13      LazyColumn {
14          items(chocolates) { chocolate ->
15              ChocolateItem(chocolate     =     chocolate,
    onDetailClicked = onDetailClicked)
16          }
17      }
18  }
```

DetailScreen.kt

Table 16. DetailScreen.kt

```
1   package com.example.scrollablecompose.uiScreen
2
3   import androidx.compose.foundation.Image
4   import androidx.compose.foundation.layout.Column
5   import androidx.compose.foundation.layout.Row
6   import androidx.compose.foundation.layout.Spacer
7   import androidx.compose.foundation.layout.fillMaxWidth
8   import androidx.compose.foundation.layout.height
9   import androidx.compose.foundation.layout.padding
10  import
11  androidx.compose.foundation.shape.RoundedCornerShape
12  import androidx.compose.material3.MaterialTheme
```

```
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import
com.example.scrollablecompose.viewModel.ChocolateViewMo
del
import timber.log.Timber

@Composable
fun DetailScreen(
    id: Int?,
    viewModel: ChocolateViewModel
) {
    val           chocolate           =           id?.let
{viewModel.getChocolateById(it)}

    if (chocolate != null) {
        Timber.d("Navigating     to     Detail     of:
${chocolate.title}")

        Column(
            modifier = Modifier
                .padding(16.dp)
                .fillMaxWidth()
        ) {
            Image(
                painter    =    painterResource(id    =
chocolate.chocolateImageId),
                contentDescription = null,
                contentScale = ContentScale.Crop,
                modifier = Modifier
                    .fillMaxWidth()
                    .height(240.dp)
                    .clip(RoundedCornerShape(12.dp))
            )

```

```
51              Spacer(modifier = Modifier.height(16.dp))
52
53              Row(
54                  modifier = Modifier.fillMaxWidth(),
55                  verticalAlignment                       =
    Alignment.CenterVertically
56              ) {
57                  Text(
58                      text = chocolate.title,
59                      style                               =
    MaterialTheme.typography.titleLarge,
60                      modifier = Modifier.weight(1f)
61                  )
62              }
63
64              Spacer(modifier = Modifier.height(8.dp))
65
66              Text(
67                  text = chocolate.description,
68                  style                                   =
    MaterialTheme.typography.bodyMedium
69              )
70          }
71      } else{
72          Timber.e("Chocolate with ID $id not found")
73          Text("Data    tidak    ada",    modifier    =
    Modifier.padding(16.dp))
74      }
75  }
```

ChocolateViewModel.kt

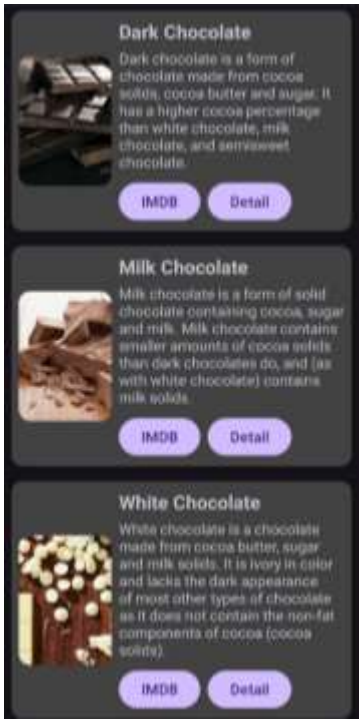
Table 17. ChocolateViewModel.kt

```
1   package com.example.scrollablecompose.viewModel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5   import com.example.scrollablecompose.data.Chocolate
6
```

```
import
7  com.example.scrollablecompose.data.ChocolateProvider
8  import kotlinx.coroutines.flow.MutableStateFlow
9  import kotlinx.coroutines.flow.StateFlow
10 import timber.log.Timber
11
   class ChocolateViewModelFactory(private val param:
12 String) : ViewModelProvider.Factory {
       override fun <Choco : ViewModel> create(modelClass:
13 Class<Choco>): Choco {
           if
       (modelClass.isAssignableFrom(ChocolateViewModel::class.
14     java)) {
15             return ChocolateViewModel(param) as Choco
16         }
           throw          IllegalArgumentException("Unknown
17     ViewModel class")
18     }
19 }
20
   class ChocolateViewModel(private val param: String) :
21 ViewModel() {
22
       private         val         _chocolateList         =
23 MutableStateFlow<List<Chocolate>>(emptyList())
       val   chocolateList:   StateFlow<List<Chocolate>>   =
24 _chocolateList
25
26     init {
27         loadChocolates()
28     }
29
30     private fun loadChocolates() {
           _chocolateList.value                            =
31 ChocolateProvider.chocolateList
           Timber.d("Chocolate         list         loaded:
32 ${_chocolateList.value.size} items")
33
34     }
35
       fun getChocolateById(id: Int): Chocolate? {
```

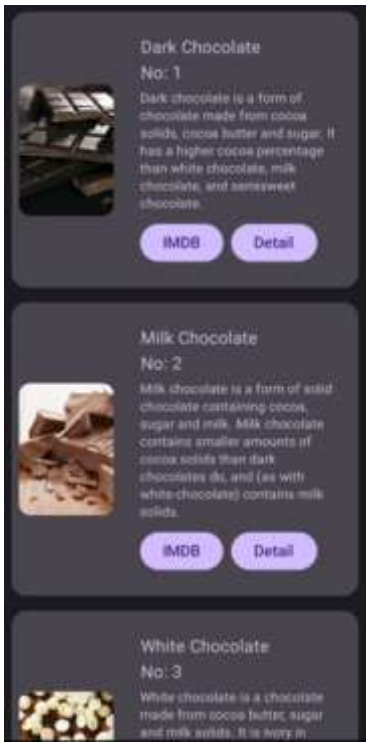
```
36          return _chocolateList.value.find { it.no == id }
37      }
38  }
```

B. Output Program



Gambar 1. UI List (XML)

Gambar 2. UI Detail (XML)



Gambar 3. UI List (Jetpack Compose)

Gambar 4. UI Detail (Jetpack Compose)

C. Pembahasan

MainActivity.kt (XML):

1. Syntax package `package com.example. scrollablexml` merupakan deklarasi nama package file Kotlin.
2. Syntax `class MainActivity : AppCompatActivity() {}` merupakan deklarasi kelas bernama MainActivity yang extend dari AppCompatActivity.
3. Syntax `private val viewModel: ChocolateViewModel by viewModels {ChocolateViewModelFactory("MainActivity")}` membuat instance ChocolateViewModel dengan by viewModels. Menggunakan ChocolateViewModelFactory untuk mengirim parameter "MainActivity" ke ViewModel.
4. Syntax `override fun onCreate(savedInstanceState: Bundle?){}` merupakan lifecycle method pertama yang dipanggil saat Activity dibuat.
5. Syntax `super.onCreate(savedInstanceState)` merupakan program untuk memanggil method onCreate() dari superclass untuk menjalankan proses inisialisasi default dan wajib dipanggil agar activity dapat berjalan dengan benar.
6. Syntax `Timber.plant(Timber.DebugTree())` program untuk mengatur Timber untuk mencetak log debug selama development.

7. Syntax `setContentView(R.layout.activity_main)` program untuk menetapkan layout XML yang digunakan sebagai UI untuk activity ini.
8. Syntax `val recyclerView: RecyclerView = findViewById(R.id.recyclerView)` merupakan fungsi untuk mendeklarasikan dan menginisialisasi variabel recyclerView dari layout XML, findViewById mencari view berdasarkan IDE @+id/recyclerView yang telah didefinisikan di XML.
9. Syntax `recyclerView.layoutManager = LinearLayoutManager(this)` berfungsi untuk mengatur adapter untuk RecyclerView, adapter ChocolateAdapter bertanggung jawab untuk menghubungkan data (chocolateList) dengan tampilan list, this mengacu pada konteks activity, dan DataProvider.chocolateList adalah sumber data.
10. Syntax `lifecycleScope.launch {` berfungsi untuk meluncurkan coroutine yang mengikuti lifecycle activity (otomatis berhenti saat activity dihancurkan).
11. Syntax `viewModel.chocolateList.collect { chocolate ->` berfungsi untuk mengumpulkan aliran data chocolateList dari ViewModel (dengan StateFlow atau Flow).
12. Syntax `recyclerView.adapter = ChocolateAdapter(this@MainActivity, chocolate,` berfungsi untuk menetapkan adapter RecyclerView dengan data chocolate.
13. Syntax `onDetailClick = {Timber.d("Detail clicked: $it")viewModel.onChocolateSelected(it) val intent = Intent(this@MainActivity, DetailActivity::class.java) intent.putExtra("id", it.id) startActivity(intent) },` berfungsi untuk mengatur item diklik untuk detail log data, memanggil fungsi di ViewModel untuk menyimpan yang dipilih, dan mavigasi ke DetailActivity, sambil mengirim ID item.
14. Syntax `onLinkClick = {Timber.d("IMDB clicked:${it.btnLink}")val intent = Intent(Intent.ACTION_VIEW, Uri.parse(it.btnLink)) startActivity(intent) }` berfungsi untuk mengatur tombol link ditekan logging link, membuat Intent untuk membuka link menggunakan browser atau app lain, dan menjalankan intent tersebut.

Chocolate.kt (XML):

1. Syntax package `package com.example.scrollablelistxml` merupakan deklarasi package tempat file ini berada, yaitu package data, com.example.scrollablexml.

2. Syntax `data class Chocolate(` merupakan definisi data class bernama Chocolate, data class secara otomatis menyediakan fungsi-fungsi penting, class ini biasanya dipakai untuk menyimpan data sederhana, seperti model atau entitas.
3. Syntax `val id: Int,val name: String,val description: String,val chocolateImageId: Int,val btnLink: String` mendeklarasikan variabel dengan tipe datanya.

ChocolateProvider.kt (XML):

1. Syntax package `package com.example.scrollablexml.data` merupakan deklarasi package tempat file ini berada, yaitu di package data, com.example.scrollablexml.
2. Syntax `object ChocolateProvider {` digunakan untuk membuat singleton, yaitu satu instance global dan menyimpan data list cokelat yang bisa diakses dari mana saja.
3. Syntax `val chocolateList = listOf(` mendeklarasikan properti immutable (val) yang berisi daftar objek Chocolate dan menggunakan listOf() untuk membuat list statis.

ChocolateAdapter.kt (XML):

1. Syntax `class ChocolateAdapter(...) : RecyclerView.Adapter<ChocolateAdapter.ViewHolder>()` membuat class ChocolateAdapter, turunan dari RecyclerView.Adapter. Adapter ini digunakan untuk menjembatani data Chocolate dengan tampilan di RecyclerView.
2. Syntax `class ViewHolder(view: View) : RecyclerView.ViewHolder(view)` merupakan wadah (wrapper) untuk setiap item yang akan ditampilkan di RecyclerView.
3. Syntax `val image: ImageView = view.findViewById(R.id.imageChocolate)` menghubungkan view dari XML (R.id.imageChocolate) ke variabel image.
4. Syntax `val name: TextView = view.findViewById(R.id.textName)` merupakan program untuk menghubungkan TextView dengan cokelat.
5. Syntax `val description: TextView = view.findViewById(R.id.textDescription)` merupakan program untuk untuk menyambungkan TextView deskripsi cokelat.

6. Syntax `val btnImdb: Button = view.findViewById(R.id.buttonImdb)` program untuk menghubungkan tombol IMDb atau link ke varianel btnImdb.
7. Syntax `val btnDetail: Button = view.findViewById(R.id.buttonDetail)` merupakan fungsi untuk menghubungkan tombol Detail ke variabel btnDetail.
8. Syntax `override fun onCreateViewHolder(...)` berfungsi untuk mengubah row_chocolate.xml menjadi objek view dan membuat ViewHolder dari tampilan tersebut.
9. Syntax `override fun getItemCount(): Int = chocolateList.size` merupakan program untuk mengembalikan jumlah item dalam chocolateList dan untuk menentukan berapa banyak item yang ditampilkan.
10. Syntax `override fun onBindViewHolder(holder: ViewHolder, position: Int) {` mengambil data cokelat pada posisi position.
11. Syntax `holder.name.text = chocolate.name` menampilkan nama cokelat.
12. Syntax `holder.btnImdb.setOnClickListener { ... }` merupakan program untuk menetapkan aksi saat tombol IMDb diklik dan membuat intent dengan ACTION_VIEW untuk membuka chocolate.btnLink di browser.
13. Syntax `holder.btnDetail.setOnClickListener { ... }` merupakan aksi untuk tombol Detail, membuat Intent untuk membuka DetailActivity, menyisipkan id dari cokelat sebagai data ekstra (putExtra("id", chocolate.id)).

DetailActivity.kt (XML):

1. Syntax package `package com.example. scrollablexml` merupakan deklarasi nama package file Kotlin.
2. Syntax `class DetailActivity : AppCompatActivity() {}` merupakan deklarasi kelas bernama DetailActivity yang extend dari AppCompatActivity.
3. Syntax `Timber.d("DetailActivity opened")` merupakan log debug untuk menandai bahwa DetailActivity sudah dibuka.
4. Syntax `override fun onCreate(savedInstanceState: Bundle?) {}` mendeklarasikan fungsi onCreate dipanggil saat activity dibuat pertama kali, parameter menyimpan state sebelumnya jika ada.
5. Syntax `super.onCreate(savedInstanceState)` merupakan fungsi untuk memanggil implementasi onCreate dari superclass (AppCompatActivity), wajib dilakukan agar lifecycle Android tetap berjalan normal.

6. Syntax `setContentView(R.layout.activity_detail)` menetapkan layout XML activity_detail.xml sebagai tampilan utama activity ini.
7. Syntax `val id = intent.getIntExtra("id", -1)` mengambil data id yang dikirim dari ChocolateAdapter via Intent, jika id tidak ditemukan, default-nya adalah -1.
8. Syntax `val chocolate = DataProvider.chocolateList.find { it.id == id }` merupakan Mencari objek Chocolate dalam DataProvider.chocolateList yang ID-nya cocok dengan data id dari intent, menggunakan fungsi find { ... }, yang akan mengembalikan elemen pertama yang cocok atau null jika tidak ditemukan.
9. Syntax `chocolate?.let { ... }` berfungsi untuk mengecek apakah chocolate tidak null, jika ya, maka blok let akan dijalankan untuk menampilkan data.
10. Syntax `findViewById<ImageView>(R.id.imageChocolateDetail).setImageResource(it.chocolateImageId)` berfungsi untuk mengatur gambar pada ImageView di layout detail berdasarkan gambar dari objek Chocolate.
11. Syntax `findViewById<TextView>(R.id.textTitleDetail).text = it.name` berfungsi untuk menampilkan nama cokelat ke TextView dengan ID textTitleDetail.
12. Syntax `findViewById<TextView>(R.id.textDescriptionDetail).text = it.description` berfungsi untuk menampilkan deskripsi cokelat ke TextView dengan ID textDescriptionDetail.

ChocolateViewModel.kt (XML):

1. Syntax package `package com.example. scrollablexml. viewmodel` merupakan deklarasi nama package viewmodel.
2. Syntax `class ChocolateViewModel(private val source: String) : ViewModel() {` merupakan ViewModel yang memegang data cokelat. Menerima parameter source untuk menandai asal atau konteks pembuatan ViewModel.
3. Syntax `private val _chocolateList = MutableStateFlow<List<Chocolate>>(emptyList())`
4. `val chocolateList: StateFlow<List<Chocolate>> get() = _chocolateList` merupakan fungsi untuk menyimpan list cokelat secara mutable.

5. Syntax `private val _selectedChocolate = MutableStateFlow<Chocolate?>(null)` menyimpan cokelat yang sedang dipilih, dengan nilai awal null.
6. Syntax `init {Timber.d("ViewModel created with source: $source")_chocolateList.value = ChocolateProvider.chocolateListTimber.d("Chocolate list loaded: ${_chocolateList.value}")}` merupakan Blok inisialisasi yang dipanggil saat ViewModel dibuat, menampilkan sumber ViewModel, mengisi _chocolateList dengan data dari ChocolateProvider, dan log pesan debug daftar cokelat yang sudah dimuat.

ChocolateViewModelFactory.kt (XML):

1. Syntax `package com.example.scrollablexml.viewmodel` merupakan deklarasi nama package viewmodel.
2. Syntax `class ChocolateViewModelFactory(private val source: String) : ViewModelProvider.Factory {` merupakan kelas factory ChocolateViewModelFactory yang mengimplementasikan interface ViewModelProvider.Factory. Factory ini menerima sebuah parameter source (tipe String) yang akan diteruskan ke ViewModel.
3. Syntax `override fun <T : ViewModel> create(modelClass: Class<T>): T {` membuat instance ViewModel sesuai kelas yang diminta.
4. Syntax `return ChocolateViewModel(source) as T}` mengembalikan instance ViewModel yang sudah dibuat.

activity_detail.xml

1. Syntax package `ScrollView` merupakan komponen yang memungkinkan isi layout dapat digulir secara vertikal jika kontennya lebih tinggi dari layar.
2. Syntax `ConstraintLayout` merupakan layout fleksibel dan powerful yang memungkinkan kita mengatur posisi elemen UI berdasarkan hubungan (constraint) antar elemen atau dengan parent.
3. Syntax `ImageView` untuk menampilkan gambar.
4. Syntax `TextView` merupakan program untuk menampilkan nama cokelat, deskripsi.

activity_main.xml

1. Syntax package `<LinearLayout>` merupakan layout linear yang menyusun elemen-elemen UI secara vertikal (satu di atas yang lain) karena orientation="vertical".
2. Syntax `RecyclerView` merupakan komponen yang digunakan untuk menampilkan daftar item (seperti daftar cokelat) secara efisien dan bisa discroll.Syntax `ImageView` untuk menampilkan gambar.

row_chocolate.xml

1. Syntax `<androidx.cardview.widget.CardView` membuat sebuah card dengan sudut membulat.
2. Syntax `<androidx.constraintlayout.widget.ConstraintLayout` merupakan komponen yang digunakan sebagai container utama di dalam card.
3. Syntax `android:id="@+id/imageChocolate"` merupakan gambar untuk menampilkan foto cokelat.
4. Syntax `<LinearLayoutandroid:id="@+id/infoLayout"android:orientation="vertical"` merupakan ontainer vertikal untuk teks dan tombol di sebelah kanan gambar.
5. Syntax `<TextView android:id="@+id/textName"` merupakan komponen yang digunakan sebagai enampilkan nama cokelat..
6. Syntax `<TextView android:id="@+id/textDescription"` merupakan komponen yang digunakan untuk menampilkan deskripsi singkat cokelat.
7. Syntax `<Button android:id="@+id/buttonImdb"` `<Button android:id="@+id/buttonDetail"` merupakan tombol untuk membuka link dan detail coklat.

MainActivity.kt (Jetpack Compose):

1. Syntax package `package com.example.scrollablecompose` merupakan deklarasi nama package file Kotlin.
2. Syntax `class MainActivity : ComponenttActivity() {}` merupakan activity untuk Jetpack Compose yang berisi program UI penuh di layer, mengatur UI dengan compose, dan program untuk tema dan memanggil composable utama.

3. Syntax `override fun onCreate(savedInstanceState: Bundle?) {super.onCreate(savedInstanceState)` merupakan override fungsi onCreate untuk menginisialisasi UI saat activity dibuat.
4. Syntax `Timber.plant(Timber.DebugTree())` merupakan program untuk enginisialisasi Timber, library logging, agar mencatat log selama pengembangan/debugging.
5. Syntax `enableEdgeToEdge()` untuk mengaktifkan mode Edge-to-Edge UI agar konten aplikasi bisa tampil full screen sampai ke tepi layar.
6. Syntax `setContent {` merupakan fungsi memulai blok Compose UI. Semua UI dibuat di dalam blok ini.
7. Syntax `val navController = rememberNavController()` merupakan program untuk membuat dan mengingat (mem-remember) instance NavController untuk navigasi antar layar Compose.
8. Syntax `val viewModel: ChocolateViewModel = viewModel(factory = ChocolateViewModelFactory("parameter-mu"))` merupakan program untuk mengambil instance ViewModel ChocolateViewModel dengan bantuan ChocolateViewModelFactory.
9. Syntax `ScrollableComposeTheme {` merupakan program membungkus seluruh UI dengan tema aplikasi khusus ScrollableComposeTheme yang mengatur warna, typography, dan lain-lain.
10. Syntax `Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->` merupakan layout dasar yang mendukung struktur UI standar seperti app bar, floating button, dan konten utama. InnerPadding adalah padding internal yang Scaffold berikan agar konten tidak tertutup oleh elemen UI seperti app bar.
11. Syntax `NavHost(){` merupakan container navigasi yang mengelola switching antar layar berdasarkan rute.
12. Syntax `composable("list") {` merupakan program untuk mendefinisikan route "list" untuk layar daftar cokelat.
13. Syntax `ListScreen(){` merupakan composable ListScreen untuk menampilkan daftar cokelat.
14. Syntax `DetailScreen(id = id, viewModel = viewModel)` merupakan program untuk memanggil composable DetailScreen untuk menampilkan detail cokelat berdasarkan id.

Chocolate.kt (Jetpack Compose):

1. Syntax `package com.example.scrollablelistxml` merupakan deklarasi package tempat file ini berada, yaitu package data, com.example.scrollablexml.

2. Syntax `data class Chocolate(` merupakan definisi data class bernama Chocolate, data class secara otomatis menyediakan fungsi-fungsi penting, class ini biasanya dipakai untuk menyimpan data sederhana, seperti model atau entitas.
3. Syntax `val id: Int,val name: String,val description: String,val chocolateImageId: Int,val btnLink: String` mendeklarasikan variabel dengan tipe datanya.

ChocolateProvider.kt (Jetpack Compose):

1. Syntax `package com.example.scrollablexml.data` merupakan deklarasi package tempat file ini berada, yaitu di package data, com.example.scrollablexml.
2. Syntax `object ChocolateProvider {` digunakan untuk membuat singleton, yaitu satu instance global dan menyimpan data list cokelat yang bisa diakses dari mana saja.
3. Syntax `val chocolateList = listOf(` mendeklarasikan properti immutable (val) yang berisi daftar objek Chocolate dan menggunakan listOf() untuk membuat list statis.

ChocolateListItem.kt (Jetpack Compose):

1. Syntax package `package com.example.scrollablecompose` merupakan deklarasi nama package file Kotlin.
2. Syntax `@Composable fun ChocolateListItem(chocolate: Chocolate, onDetailClick: (Chocolate) -> Unit) {` merupakan program untuk menandai fungsi sebagai composable.
3. Syntax `Card (){` merupakan program untuk latar belakang.
4. Syntax `Row(){` merupakan row utama untuk mengatur konsen dengan berbagai ketentuan.
5. Syntax `Image()` merupakan fungsi untuk menampilkan gambar dengan resource ID.

6. Syntax `Spacer(modifier = Modifier.width(12.dp))` merupakan fungsi untuk memberi jarak horizontal 12dp antara gambar dan konten teks.
7. Syntax `Column(){}` merupakan program untuk mengatur teks dan tombol pada aplikasi.
8. Syntax `Text(text = chocolate.title), Text(text = "No: ${chocolate.no}"), Text(text = chocolate.description, style = MaterialTheme.typography.bodySmall, modifier = Modifier.padding(top = 4.dp))` merupakan program untuk menampilkan judul, nomor, dan deskripsi item.
9. Syntax `Button(onClick = {Timber.d("Link button clicked: ${chocolate.btnLink}")val intent =Intent(Intent.ACTION_VIEW, Uri.parse(chocolate.btnLink)) context.startActivity(intent)}) {Text("IMDB")}` merupakan program untuk mencatat log dan membuka browser ke URL yang ada di chocolate.btnLink menggunakan Intent.ACTION_VIEW.
10. Syntax `Button(onClick = {Timber.d("Detail button clicked: ${chocolate.title}")onDetailClicked(chocolate)}) {Text("Detail")}` merupakan program untuk mencatat mencatat log dan memanggil callback onDetailClicked dengan objek cokelat.

ChocolateListScreen.kt (Jetpack Compose):

1. Syntax `package com.example.scrollablecompose.uiComponent` merupakan deklarasi nama package UIComponent.
2. Syntax `@Composable fun ListScreen( chocolates: List<Chocolate>, onDetailClicked: (Chocolate) -> Unit) {` merupakan program untuk mendeklarasikan fungsi composable ListScreen.
3. Syntax `LazyColumn {` merupakan program untuk membuat list vertikal yang bisa di-scroll dengan performa optimal. Berbeda dengan Column, LazyColumn hanya membuat dan menampilkan item yang terlihat saja.
4. Syntax `items(chocolates) { chocolate ->` merupakan program untuk membuat elemen list untuk setiap item di dalam daftar chocolates.
5. Syntax `ChocolateItem(chocolate = chocolate, onDetailClicked = onDetailClicked)` merupakan program untuk emanggil composable ChocolateItem untuk setiap item di list.

DetailScreen.kt (Jetpack Compose):

1. Syntax `package com.example.scrollablecompose.uiScreen` merupakan deklarasi nama package uiScreen.
2. Syntax `@Composable fun DetailScreen(id: Int?, viewModel: ChocolateViewModel) {` merupakan program untuk mendefinisikan fungsi composable DetailScreen.
3. Syntax `val chocolate = id?.let { viewModel.getChocolateById(it) }` merupakan program untuk mengambil data Chocolate berdasarkan ID menggunakan ViewModel.
4. Syntax `if (chocolate != null) {` merupakan program untuk mengecek apakah data chocolate ditemukan.
5. Syntax `Timber.d("Navigating to Detail of: ${chocolate.title}")` merupakan program untuk menampilkan log debug jika data ditemukan.
6. Syntax `Column( modifier = Modifier.padding(16.dp).fillMaxWidth()) {` merupakan program untuk mengatur layout vertikal (atas ke bawah) dengan padding dan lebar penuh.
7. Syntax `Image()` merupakan program untuk enampilkan gambar dari resource berdasarkan ID dari objek chocolate.
8. Syntax `Text(text = chocolate.title, style = MaterialTheme.typography.titleLarge, modifier = Modifier.weight(1f))` merupakan program untuk menampilkan judul chocolate dalam satu baris (Row).
9. Syntax `Text(text = chocolate.description, style = MaterialTheme.typography.bodyMedium)` merupakan program untuk menampilkan deskripsi chocolate menggunakan gaya teks dari MaterialTheme.
10. Syntax `} else {Timber.e("Chocolate with ID $id not found")Text("Data tidak ada", modifier =Modifier.padding(16.dp))}` merupakan program Jika chocolate null (tidak ditemukan): Log error menggunakan Timber dan menampilkan teks “Data tidak ada”.

DetailScreen.kt (Jetpack Compose):

1. Syntax `package com.example.scrollablecompose.uiScreen` merupakan deklarasi nama package uiScreen.
2. Syntax `class ChocolateViewModelFactory(private val param: String) : ViewModelProvider.Factory {` merupakan program untuk membuat ViewModelFactory kustom.

3. Syntax `override fun <Choco : ViewModel> create(modelClass: Class<Choco>): Choco {` merupakan program Override fungsi create() untuk menghasilkan instance ViewModel.
4. Syntax `if (modelClass.isAssignableFrom(ChocolateViewModel::class.java)) {return ChocolateViewModel(param) as Choco}` merupakan program untuk mengecek apakah modelClass adalah ChocolateViewModel, lalu membuat instance-nya.
5. Syntax `class ChocolateViewModel(private val param: String) : ViewModel() {` merupakan program untuk menyimpan dan mengelola data list cokelat.
6. Syntax `private val _chocolateList = MutableStateFlow<List<Chocolate>>(emptyList())val chocolateList: StateFlow<List<Chocolate>> = _chocolateList` merupakan program mutable (bisa diubah) tapi hanya bisa diakses internal.
7. Syntax `Image()` merupakan program untuk enampilkan gambar dari resource berdasarkan ID dari objek chocolate.
8. Syntax `private fun loadChocolates() {_chocolateList.value = ChocolateProvider.chocolateListTimber.d("Chocolate list loaded: ${_chocolateList.value.size} items")}` merupakan program untuk mengambil data dari ChocolateProvider dan simpan ke _chocolateList.
9. Syntax `fun getChocolateById(id: Int): Chocolate? {return _chocolateList.value.find { it.no == id }}` merupakan program untuk mendapatkan 1 objek chocolate berdasarkan nomor. Berguna untuk menampilkan detail berdasarkan ID.

SOAL 2

Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

D. Pembahasan

Karena RecyclerView masih kompatibilitas saat ini, terlebih keterbatasan LazzyColumn untuk kasus-kasus spesifik, seperti pagination yang kompleks, data binding tingkat lanjut, multi-view types, dan item click listeners. Kemudian RecyclerView juga dapat memberikan lebih banyak kontrol terkait optimasi memori dengan fitur seperti ViewPool, yang memungkinkan daftar item tampilan yang sudah ada untuk meningkatkan kinerja. Selain itu, perpindahan dari XML ke Jetpact Compose juga memerlukan waktu dan perubahan besar dalam struktur aplikasi apalagi aplikasi yang dari awal dikembangkan dengan XML yang membutuhkan pemahaman dan penguasaan baru untuk bisa berpindah menggunakan Jetpack Compose.

E. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/rizkiiirr/PraktikumMobile/tree/main/PRAKTIKUM%20MOBILE/Modul4