LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

ViewModel and Debugging

Oleh:

Avantio Fierza Patria NIM. 2310817310001

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 Prakitkum ini dikerjakan oleh:

Nama Praktikan : Avantio Fierza Patria
NIM : 2310817310001

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Raka Azwar NIM. 2210817210012	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 19930703 201903 01 011

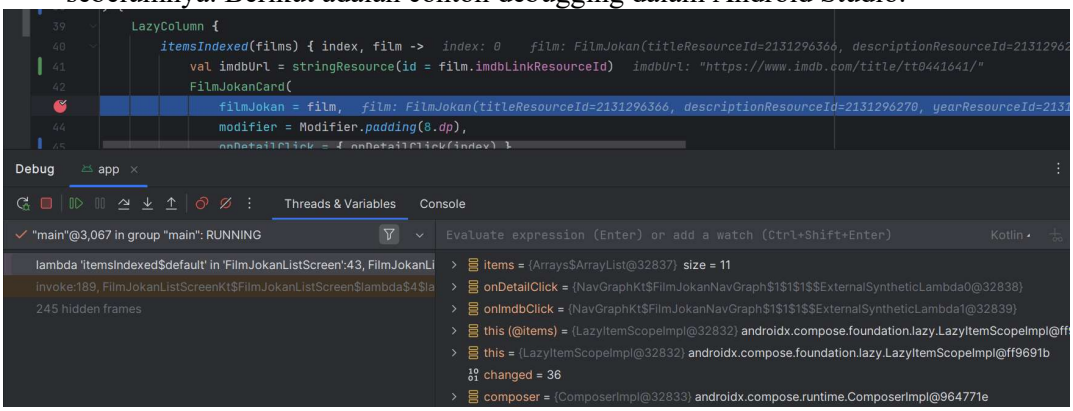
DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Soal Praktikum:

1. Buatlah sebuah aplikasi Android menggunakan XML dan Jetpack Compose yang dapat menampilkan list dengan 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
    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. 1 Contoh Penggunaan Debugger

A. Source Code menggunakan Jetpack Compose

MainActivity.kt

```
1   package com.example.scrollablelist
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import com.example.scrollablelist.navigation.NavGraph
7   import com.example.scrollablelist.ui.theme.ScrollableListTheme
8
9   class MainActivity : ComponentActivity() {
10      override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        setContent {
            ScrollableListTheme {
                NavGraph()
            }
        }
    }
}
```

Tabel 1. 1 Source Code MainActivity.kt Jetpack Compose

Book.kt

```
package com.example.scrollablelist.model

data class Book(
    val id: Int,
    val title: String,
    val imageUrl: String,
    val webUrl: String,
    val shortDescription: String,
    val fullDescription: String
)
```

Tabel 1. 2 Source Code Book.kt Jetpack Compose

BookViewModel.kt

```
package com.example.scrollablelist.viewmodel

import android.util.Log
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import androidx.lifecycle.viewModelScope
import com.example.scrollablelist.model.Book
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch
import timber.log.Timber

class BookViewModel(private val query: String) : ViewModel() {

    private             val             _bookList             =
MutableStateFlow<List<Book>>(emptyList())
    val bookList: StateFlow<List<Book>> = _bookList

    private val _event = MutableStateFlow<Event?>(null)
    val event: StateFlow<Event?> = _event

    init {
        Timber.d("ViewModel initialized with query: $query")
        loadBooks(query)
    }

    private fun loadBooks(query: String) {
        viewModelScope.launch {

```

```
            // Simulasi mengambil data dari sumber data
(misalnya database, API, dll)
            val books = listOf(
                Book(
                    1,
                    "Atomic Habits",
                    "https://m.media-
amazon.com/images/I/91bYsX41DVL.jpg",

"https://www.goodreads.com/book/show/40121378-atomic-habits",
                    "Build good habits and break bad ones.",
                    "Atomic Habits by James Clear offers a
proven framework for improving everyday life. Learn how tiny
changes add up to remarkable results."
                ),
                Book(
                    2,
                    "Deep Work",
                    "https://m.media-
amazon.com/images/I/91nujEwIpYL._SL1500_.jpg",

"https://www.goodreads.com/book/show/25744928-deep-work",
                    "Focused success in a distracted world.",
                    "Deep Work is a book about the benefits of
intense focus and how to systematically train your mind and
habits to achieve it."
                ),
                Book(
                    3,
                    "The Subtle Art of Not Giving a F*ck",
                    "https://m.media-
amazon.com/images/I/71QKQ9mwV7L.jpg",

"https://www.goodreads.com/book/show/28257707-the-subtle-art-
of-not-giving-a-f-ck",
                    "Counterintuitive approach to living a good
life.",
                    "Mark Manson's book shows that life's
struggles give it meaning, and teaches how to stop trying to be
positive all the time."
                ),
                Book(
                    4,
                    "Thinking, Fast and Slow",
                    "https://m.media-
amazon.com/images/I/61fdrEuPJwL._SL1500_.jpg",

"https://www.goodreads.com/book/show/11468377-thinking-fast-
and-slow",
                    "Explores how we think, make decisions, and
act.",
                    "Nobel laureate Daniel Kahneman explores
two modes of thought: 'fast', intuitive thinking, and 'slow',
deliberate thinking."
```

```
                ),
                Book(
                    5,
                    "Start With Why",
                    "https://m.media-
amazon.com/images/I/71NBZIExBCL._SY466_.jpg",

"https://www.goodreads.com/book/show/7108725-start-with-why",
                    "Find your why and inspire others.",
                    "Simon Sinek explains how leaders can
inspire cooperation, trust and change by focusing on the WHY
behind their mission."
                ),
                Book(
                    6,
                    "The Power of Now",
                    "https://m.media-
amazon.com/images/I/61Ij8nLooNL._SL1500_.jpg",
                    "https://www.amazon.com/Power-Now-Guide-
Spiritual-Enlightenment/dp/1577314808",
                    "A guide to spiritual enlightenment.",
                    "Eckhart Tolle’s guide helps readers
discover the importance of living in the present moment and
letting go of the past and future."
                ),
                Book(
                    7,
                    "Can't Hurt Me",
                    "https://m.media-
amazon.com/images/I/81gTRv2HXrL._SL1500_.jpg",

"https://www.goodreads.com/book/show/41721428-can-t-hurt-me",
                    "Master your mind and defy the odds.",
                    "David Goggins shares his incredible life
story and teaches readers how to overcome pain, fear, and self-
doubt to reach their full potential."
                ),
                Book(
                    8,
                    "Educated",
                    "https://m.media-
amazon.com/images/I/81WojUxbbFL.jpg",

"https://www.goodreads.com/book/show/35133922-educated",
                    "A memoir of transformation through
education.",
                    "Tara Westover tells her story of growing
up in a strict and abusive household in rural Idaho and how she
escaped through learning and education."
                )
            )

            // Log saat data masuk ke dalam list
            Timber.d("Books loaded: $books")
```

```
139             _bookList.value = books
140         }
141     }
142
143     fun onItemClicked(book: Book) {
144         // Log saat detail tombol ditekan
145         Timber.d("Detail    button    clicked    for    book:
146 ${book.title}")
147         _event.value = Event.NavigateToDetail(book)
148     }
149
150     fun onWebButtonClicked(book: Book) {
151         // Log saat tombol Explicit Intent ditekan
152         Timber.d("Web button clicked for book: ${book.title}")
153         _event.value = Event.OpenWebUrl(book.webUrl)
154     }
155
156     sealed class Event {
157         data class NavigateToDetail(val book: Book) : Event()
158         data class OpenWebUrl(val url: String) : Event()
159     }
160 }
161
162 class  BookViewModelFactory(private  val  query:  String)  :
163 ViewModelProvider.Factory {
164     override fun <T : ViewModel> create(modelClass: Class<T>):
165 T {
166         if
167 (modelClass.isAssignableFrom(BookViewModel::class.java)) {
168             return BookViewModel(query) as T
169         }
170         throw    IllegalArgumentException("Unknown    ViewModel
171 class")
172     }
    }
```

Tabel 1. 3 Source Code BookViewModel.kt Jetpack Compose

DetailScreen.kt

```
1   package com.example.scrollablelist.screens
2
3   import androidx.compose.foundation.Image
4   import androidx.compose.foundation.layout.*
5   import androidx.compose.foundation.shape.RoundedCornerShape
6   import androidx.compose.material3.*
7   import androidx.compose.runtime.Composable
8   import androidx.compose.ui.Modifier
9   import androidx.compose.ui.draw.clip
10  import androidx.compose.ui.graphics.Color
11  import androidx.compose.ui.text.font.FontWeight
12  import androidx.compose.ui.unit.dp
13  import androidx.compose.ui.unit.sp
14  import coil.compose.rememberAsyncImagePainter
15
16  @Composable
```

```
fun DetailScreen(title: String, imageUrl: String,
fullDescription: String) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
    ) {
        Image(
            painter = rememberAsyncImagePainter(imageUrl),
            contentDescription = title,
            modifier = Modifier
                .fillMaxWidth()
                .height(250.dp)
                .clip(RoundedCornerShape(16.dp))
        )

        Spacer(modifier = Modifier.height(16.dp))

        Text(
            text = title,
            style = MaterialTheme.typography.headlineSmall,
            fontWeight = FontWeight.Bold,
            fontSize = 24.sp,
            color = Color.Black
        )

        Spacer(modifier = Modifier.height(12.dp))

        Text(
            text = fullDescription,
            style = MaterialTheme.typography.bodyLarge,
            fontSize = 18.sp,
            color = Color.DarkGray
        )
    }
}
```

Tabel 1. 4 Source Code DetailScreen.kt Jetpack Compose

HomeScreen.kt

```
package com.example.scrollablelist.screens

import android.content.Intent
import android.net.Uri
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.draw.clip
import androidx.compose.ui.graphics.Color
```

```
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavController
import com.example.scrollablelist.model.Book
import com.example.scrollablelist.viewmodel.BookViewModel
import
com.example.scrollablelist.viewmodel.BookViewModelFactory
import coil.compose.rememberAsyncImagePainter
import timber.log.Timber

@Composable
fun HomeScreen(navController: NavController) {
    val  viewModel:  BookViewModel  =  viewModel(factory  =
BookViewModelFactory("some query"))
    val books by viewModel.bookList.collectAsState()
    val event by viewModel.event.collectAsState()

    LaunchedEffect(event) {
        when (event) {
            is BookViewModel.Event.NavigateToDetail -> {
                val       book       =       (event       as
BookViewModel.Event.NavigateToDetail).book

navController.navigate("detail/${Uri.encode(book.title)}/${Ur
i.encode(book.imageUrl)}/${Uri.encode(book.fullDescription)}"
)
            }
            is BookViewModel.Event.OpenWebUrl -> {
                val       url       =       (event       as
BookViewModel.Event.OpenWebUrl).url
                val   intent   =   Intent(Intent.ACTION_VIEW,
Uri.parse(url))
                navController.context.startActivity(intent)
            }
            null -> Unit
        }
    }

    LazyColumn(
        contentPadding = PaddingValues(16.dp),
        verticalArrangement = Arrangement.spacedBy(16.dp)
    ) {
        items(books) { book ->
            Card(
                shape = RoundedCornerShape(16.dp),
                colors                                          =
CardDefaults.cardColors(containerColor = Color(0xFF2C2C2C)),
                modifier = Modifier.fillMaxWidth()
            ) {
                Row(
                    modifier = Modifier
                        .padding(16.dp)
```

```
                            .fillMaxWidth()
                    ) {
                        Image(
                            painter                                 =
rememberAsyncImagePainter(book.imageUrl),
                            contentDescription = book.title,
                            modifier = Modifier
                                .size(100.dp)
                                .clip(RoundedCornerShape(16.dp))
                        )

                        Spacer(modifier = Modifier.width(16.dp))

                        Column(
                            modifier = Modifier.weight(1f)
                        ) {
                            Text(
                                text = book.title,
                                style                               =
MaterialTheme.typography.titleMedium,
                                color = Color.White,
                                fontWeight = FontWeight.Bold,
                                fontSize = 18.sp
                            )
                            Spacer(modifier                         =
Modifier.height(4.dp))
                            Text(
                                text = book.shortDescription,
                                style                               =
MaterialTheme.typography.bodyMedium,
                                color = Color.Gray,
                                maxLines = 2
                            )

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

                            Row {
                                Button(
                                    onClick = {

viewModel.onWebButtonClicked(book)
                                    },
                                    colors                          =
ButtonDefaults.buttonColors(
                                        containerColor              =
Color(0xFFB3C7F9),
                                        contentColor = Color.Black
                                    ),
                                    shape                           =
RoundedCornerShape(50),
                                    modifier                        =
Modifier.height(40.dp)
                                ) {
```

```
                                    Text(text = "Website")
                                }

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

                                Button(
                                    onClick = {

viewModel.onItemClicked(book)
                                    },
                                    colors                      =
ButtonDefaults.buttonColors(
                                        containerColor          =
Color(0xFFB3C7F9),
                                        contentColor = Color.Black
                                    ),
                                    shape                       =
RoundedCornerShape(50),
                                    modifier                    =
Modifier.height(40.dp)
                                ) {
                                    Text(text = "Detail")
                                }
                            }
                        }
                    }
                }
            }
        }
```

Tabel 1. 5 Source Code HomeScreen.kt Jetpack Compose

MyApplication.kt

```
package com.example.scrollablelist

import android.app.Application
import timber.log.Timber

class MyApplication : Application() {
    override fun onCreate() {
        super.onCreate()
            Timber.plant(Timber.DebugTree())
    }
}
```

Tabel 1. 6 Source Code MyApplication.kt Jetpack Compose

Navigation.kt

```
package com.example.scrollablelist.navigation

import androidx.compose.runtime.Composable
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
```

```
6  import androidx.navigation.compose.composable
7  import androidx.navigation.compose.rememberNavController
8  import com.example.scrollablelist.screens.DetailScreen
9  import com.example.scrollablelist.screens.HomeScreen
10
11 @Composable
12 fun      NavGraph(navController:      NavHostController      =
13 rememberNavController()) {
14     NavHost(
15         navController = navController,
16         startDestination = "home"
17     ) {
18         composable("home") {
19             HomeScreen(navController)
20         }
21
22 composable("detail/{title}/{imageUrl}/{fullDescription}")     {
23 backStackEntry ->
24             val                      title                     =
25 backStackEntry.arguments?.getString("title") ?: ""
26             val                    imageUrl                    =
27 backStackEntry.arguments?.getString("imageUrl") ?: ""
28             val                fullDescription                 =
29 backStackEntry.arguments?.getString("fullDescription") ?: ""
30             DetailScreen(title, imageUrl, fullDescription)
31         }
32     }
33 }
```

Tabel 1. 7 Source Code Navigation.kt Jetpack Compose

B. Source Code menggunakan XML

MainActivity.kt

```
1  package com.example.scrollablelist2
2
3  import android.os.Bundle
4  import androidx.appcompat.app.AppCompatActivity
5  import
6  com.example.scrollablelist2.databinding.ActivityMainBinding
7
8  class MainActivity : AppCompatActivity() {
9      private lateinit var binding: ActivityMainBinding
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         binding = ActivityMainBinding.inflate(layoutInflater)
14         setContentView(binding.root)
15     }
16 }
```

Tabel 1. 8 Source Code MainActivity.kt XML

Book.kt

```
1  package com.example.scrollablelist2.data
2
```

```
3  data class Book(
4      val id: Int,
5      val title: String,
6      val imageUrl: String,
7      val webUrl: String,
8      val shortDescription: String,
9      val fullDescription: String
10 )
```

Tabel 1. 9 Source Code Book.kt XML

BookAdapter.kt

```
1  package com.example.scrollablelist2.ui
2
3  import android.view.LayoutInflater
4  import android.view.ViewGroup
5  import androidx.recyclerview.widget.RecyclerView
6  import com.bumptech.glide.Glide
7  import com.example.scrollablelist2.data.Book
8  import com.example.scrollablelist2.databinding.ItemBookBinding
9
10 class BookAdapter(private var bookList: List<Book>, private val
11 onBookClicked: (Book) -> Unit) :
12     RecyclerView.Adapter<BookAdapter.BookViewHolder>() {
13
14     inner class BookViewHolder(val binding: ItemBookBinding) :
15 RecyclerView.ViewHolder(binding.root)
16
17     override fun onCreateViewHolder(parent: ViewGroup, viewType:
18 Int): BookViewHolder {
19         val                          binding                          =
20 ItemBookBinding.inflate(LayoutInflater.from(parent.context),
21 parent, false)
22         return BookViewHolder(binding)
23     }
24
25     override    fun    onBindViewHolder(holder:    BookViewHolder,
26 position: Int) {
27         val book = bookList[position]
28         holder.binding.apply {
29             titleTextView.text = book.title
30             descTextView.text = book.shortDescription
31
32 Glide.with(imageView.context).load(book.imageUrl).into(imageView)
33
34             root.setOnClickListener {
35                 onBookClicked(book)
36             }
37         }
38     }
39
40     override fun getItemCount(): Int = bookList.size
41
42     fun updateBooks(newBooks: List<Book>) {
43         bookList = newBooks
```

```
44          notifyDataSetChanged()
45      }
46  }
```

Tabel 1. 10 Source Code BookAdapter.kt XML

BookDetailFragment.kt

```
1   package com.example.scrollablelist2.ui
2
3   import android.os.Bundle
4   import android.view.LayoutInflater
5   import android.view.View
6   import android.view.ViewGroup
7   import androidx.fragment.app.Fragment
8   import com.bumptech.glide.Glide
9   import
10  com.example.scrollablelist2.databinding.FragmentBookDetailBinding
11
12  class BookDetailFragment : Fragment() {
13
14      private lateinit var binding: FragmentBookDetailBinding
15
16      override fun onCreateView(
17          inflater:    LayoutInflater,    container:    ViewGroup?,
18  savedInstanceState: Bundle?
19      ): View? {
20          binding   =   FragmentBookDetailBinding.inflate(inflater,
21  container, false)
22
23          val                          args                          =
24  BookDetailFragmentArgs.fromBundle(requireArguments())
25          binding.detailTitle.text = args.title
26          binding.detailDescription.text = args.description
27
28          Glide.with(this)
29              .load(args.imageUrl)
30              .into(binding.detailImageView)
31
32          return binding.root
33      }
34  }
```

Tabel 1. 11 Source Code BookDetailFragment.kt XML

BookListFragment.kt

```
1   package com.example.scrollablelist2.ui
2
3   import android.os.Bundle
4   import android.view.LayoutInflater
5   import android.view.View
6   import android.view.ViewGroup
7   import androidx.fragment.app.Fragment
8   import androidx.fragment.app.activityViewModels
9   import androidx.lifecycle.lifecycleScope
10
```

```
11 import
12 com.example.scrollablelist2.databinding.FragmentBookListBinding
13 import kotlinx.coroutines.flow.collect
14 import timber.log.Timber
15
16 class BookListFragment : Fragment() {
17
18     private lateinit var binding: FragmentBookListBinding
19     private val viewModel: BookViewModel by activityViewModels
20 { BookViewModelFactory() }
21
22     override    fun    onCreateView(inflater:    LayoutInflater,
23 container: ViewGroup?, savedInstanceState: Bundle?): View? {
24         binding   =   FragmentBookListBinding.inflate(inflater,
25 container, false)
26
27         val adapter = BookAdapter(emptyList()) { book ->
28             viewModel.onBookSelected(book)
29         }
30         binding.bookRecyclerView.adapter = adapter
31
32         // Mengamati perubahan daftar buku
33         lifecycleScope.launchWhenStarted {
34             viewModel.books.collect { books ->
35                 Timber.d("Books data collected: ${books.size}")
36                 adapter.updateBooks(books)
37             }
38         }
39
40         return binding.root
41     }
   }
```

Tabel 1. 12 Source Code BookListFragment.kt XML

BookViewModel.kt

```
1  package com.example.scrollablelist2.ui
2
3  import androidx.lifecycle.ViewModel
4  import androidx.lifecycle.viewModelScope
5  import kotlinx.coroutines.flow.MutableStateFlow
6  import kotlinx.coroutines.flow.StateFlow
7  import kotlinx.coroutines.launch
8  import com.example.scrollablelist2.data.Book
9  import timber.log.Timber
10
11 class BookViewModel : ViewModel() {
12
13     private              val              _books              =
14 MutableStateFlow<List<Book>>(emptyList())
15     val books: StateFlow<List<Book>> = _books
16
17     init {
18         fetchBooks()
19     }
```

```
20
21      private fun fetchBooks() {
22          // Log ketika data buku di-load
23          Timber.d("Loading books data")
24
25          // Simulasi mengambil data buku
26          val bookList = listOf(
27              Book(1, "Atomic Habits",
28                  "https://m.media-
29  amazon.com/images/I/91bYsX41DVL.jpg",
30                  "https://www.goodreads.com/book/show/40121378-
31  atomic-habits",
32                  "Build good habits and break bad ones.",
33                  "Atomic Habits by James Clear offers a proven
34  framework for improving everyday life. Learn how tiny changes
35  add up to remarkable results."),
36
37              Book(2, "Deep Work",
38                  "https://m.media-
39  amazon.com/images/I/91nujEwIpYL._SL1500_.jpg",
40                  "https://www.goodreads.com/book/show/25744928-
41  deep-work",
42                  "Focused success in a distracted world.",
43                  "Deep Work is a book about the benefits of
44  intense focus and how to systematically train your mind and
45  habits to achieve it."),
46
47              Book(3, "The Subtle Art of Not Giving a F*ck",
48                  "https://m.media-
49  amazon.com/images/I/71QKQ9mwV7L.jpg",
50                  "https://www.goodreads.com/book/show/28257707-
51  the-subtle-art-of-not-giving-a-f-ck",
52                  "Counterintuitive approach to living a good
53  life.",
54                  "Mark Manson's book shows that life's struggles
55  give it meaning, and teaches how to stop trying to be positive
56  all the time."),
57
58              Book(4, "Thinking, Fast and Slow",
59                  "https://m.media-
60  amazon.com/images/I/61fdrEuPJwL._SL1500_.jpg",
61                  "https://www.goodreads.com/book/show/11468377-
62  thinking-fast-and-slow",
63                  "Explores how we think, make decisions, and
64  act.",
65                  "Nobel laureate Daniel Kahneman explores two
66  modes of thought: 'fast', intuitive thinking, and 'slow',
67  deliberate thinking."),
68
69              Book(5, "Start With Why",
70                  "https://m.media-
71  amazon.com/images/I/71NBZIExBCL._SY466_.jpg",
72                  "https://www.goodreads.com/book/show/7108725-
73  start-with-why",
```

```
                "Find your why and inspire others.",
                "Simon Sinek explains how leaders can inspire
cooperation, trust and change by focusing on the WHY behind
their mission."),

            Book(6, "The Power of Now",
                "https://m.media-
amazon.com/images/I/61Ij8nLooNL._SL1500_.jpg",
                "https://www.amazon.com/Power-Now-Guide-
Spiritual-Enlightenment/dp/1577314808",
                "A guide to spiritual enlightenment.",
                "Eckhart Tolle's guide helps readers discover
the importance of living in the present moment and letting go
of the past and future."),

            Book(7, "Can't Hurt Me",
                "https://m.media-
amazon.com/images/I/81gTRv2HXrL._SL1500_.jpg",
                "https://www.goodreads.com/book/show/41721428-
can-t-hurt-me",
                "Master your mind and defy the odds.",
                "David Goggins shares his incredible life story
and teaches readers how to overcome pain, fear, and self-doubt
to reach their full potential."),

            Book(8, "Educated",
                "https://m.media-
amazon.com/images/I/81WojUxbbFL.jpg",
                "https://www.goodreads.com/book/show/35133922-
educated",
                "A memoir of transformation through
education.",
                "Tara Westover tells her story of growing up in
a strict and abusive household in rural Idaho and how she
escaped through learning and education."),

            Book(9, "Sapiens: A Brief History of Humankind",
                "https://m.media-
amazon.com/images/I/91MJzUOZ3CL.jpg",
                "https://www.goodreads.com/book/show/23692271-
sapiens",
                "A journey through human history.",
                "Yuval Noah Harari explores the history of
humankind from the Stone Age to the modern age, examining how
biology and history have defined human societies."),

            Book(10, "Becoming",
                "https://m.media-
amazon.com/images/I/71kW6hbcfPL.jpg",
                "https://www.goodreads.com/book/show/38746485-
becoming",
                "The memoir of Michelle Obama.",


```

```
                "Becoming by Michelle Obama is a powerful memoir where she reflects on her personal journey from the South Side of Chicago to the White House, and beyond."),

            Book(11, "Educated",
                "https://m.media-amazon.com/images/I/71N9Sx5GBxL.jpg",
                "https://www.goodreads.com/book/show/25733515-educated",
                "A memoir about a woman who sought education.",
                "Educated is a memoir by Tara Westover, detailing her upbringing in a strict and abusive household and her eventual escape through education. A story of resilience and perseverance.")
        )

        // Mengupdate StateFlow dengan data buku
        viewModelScope.launch {
            _books.emit(bookList)
            Timber.d("Books loaded: ${bookList.size} books")
        }
    }

    // Event untuk tombol di klik
    fun onBookSelected(book: Book) {
        Timber.d("Book selected: ${book.title}")
    }
}
```

Tabel 1. 13 Source Code BookViewModel.kt XML

BookViewFactory.kt

```
package com.example.scrollablelist2.ui

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import androidx.lifecycle.viewmodel.CreationExtras

class BookViewModelFactory : ViewModelProvider.Factory {

    override fun <T : ViewModel> create(modelClass: Class<T>, extras: CreationExtras): T {
        if (modelClass.isAssignableFrom(BookViewModel::class.java)) {
            // Return instance dari BookViewModel
            return BookViewModel() as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

Tabel 1. 14 Source Code BookViewModelFactory.kt XML

activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:id="@+id/mainLayout"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      tools:context=".MainActivity">
10
11     <!-- Tempat untuk menampilkan fragment -->
12     <androidx.fragment.app.FragmentContainerView
13         android:id="@+id/nav_host_fragment"
14
15 android:name="androidx.navigation.fragment.NavHostFragment"
16         android:layout_width="0dp"
17         android:layout_height="0dp"
18         app:defaultNavHost="true"
19         app:navGraph="@navigation/nav_graph"
20         app:layout_constraintTop_toTopOf="parent"
21         app:layout_constraintBottom_toBottomOf="parent"
22         app:layout_constraintStart_toStartOf="parent"
23         app:layout_constraintEnd_toEndOf="parent" />
24 </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 1. 15 Source Code activity_maint .XML

fragment_book_detail.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ScrollView
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:padding="16dp">
7
8      <LinearLayout
9          android:orientation="vertical"
10         android:layout_width="match_parent"
11         android:layout_height="wrap_content">
12
13         <ImageView
14             android:id="@+id/detailImageView"
15             android:layout_width="match_parent"
16             android:layout_height="250dp"
17             android:scaleType="centerCrop"
18             android:layout_marginBottom="16dp"
19             android:background="@drawable/rounded_image" />
20
21         <TextView
22             android:id="@+id/detailTitle"
23             android:layout_width="match_parent"
24             android:layout_height="wrap_content"
25             android:textSize="20sp"
```

```
            android:textStyle="bold" />

        <TextView
            android:id="@+id/detailDescription"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:textSize="16sp" />

    </LinearLayout>
</ScrollView>
```

Tabel 1. 16 Source Code fragment_book_detail .XML

fragment_book_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/bookRecyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:padding="8dp"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 1. 17 Source Code fragment_book_list .XML

item_book.xml

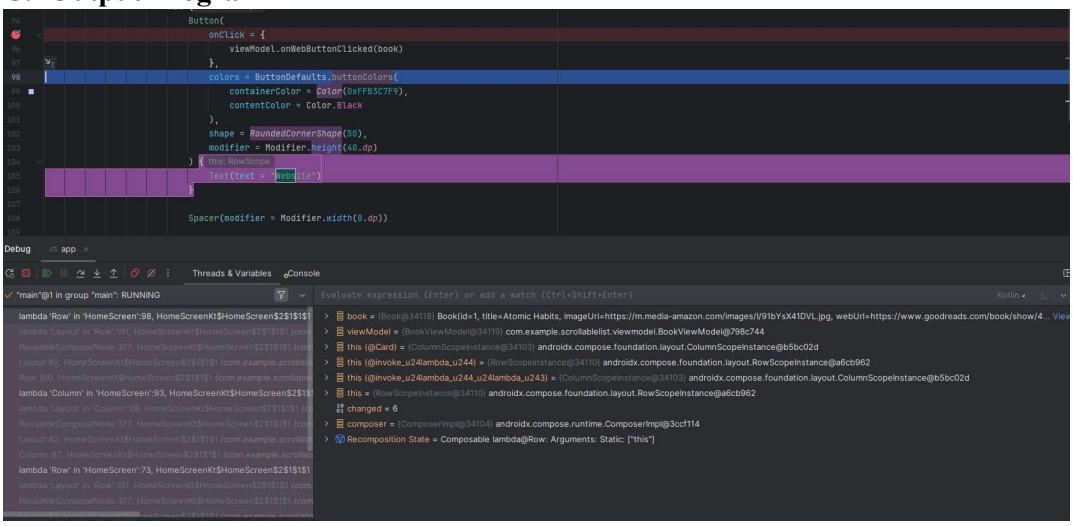
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    card_view:cardCornerRadius="16dp"
    card_view:cardElevation="6dp"
    card_view:cardBackgroundColor="#2C2C2C">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="16dp">

```

```
18          <ImageView
19              android:id="@+id/imageView"
20              android:layout_width="100dp"
21              android:layout_height="150dp"
22              android:scaleType="centerCrop"
23              android:contentDescription="@string/book_cover"
24              android:layout_marginEnd="16dp"
25              android:background="@drawable/image_background"
26              android:clipToOutline="true" />
27
28          <LinearLayout
29              android:layout_width="0dp"
30              android:layout_height="match_parent"
31              android:layout_weight="1"
32              android:orientation="vertical">
33
34              <TextView
35                  android:id="@+id/titleTextView"
36                  android:layout_width="wrap_content"
37                  android:layout_height="wrap_content"
38                  android:text="Book Title"
39                  android:textColor="@android:color/white"
40                  android:textSize="18sp"
41                  android:textStyle="bold" />
42
43              <TextView
44                  android:id="@+id/yearTextView"
45                  android:layout_width="wrap_content"
46                  android:layout_height="wrap_content"
47                  android:text="2025"
48                  android:textColor="@android:color/darker_gray"
49                  android:textSize="14sp"
50                  android:layout_marginBottom="8dp" />
51
52              <TextView
53                  android:id="@+id/descTextView"
54                  android:layout_width="wrap_content"
55                  android:layout_height="wrap_content"
56                  android:text="Short description goes here"
57                  android:textColor="@android:color/white"
58                  android:textSize="14sp"
59                  android:layout_marginBottom="8dp" />
60
61              <LinearLayout
62                  android:layout_width="wrap_content"
63                  android:layout_height="wrap_content"
64                  android:orientation="horizontal"
65                  android:layout_marginTop="8dp">
66
67                  <Button
68                      android:id="@+id/buttonOpenLink"
69                      android:layout_width="wrap_content"
70                      android:layout_height="36dp"
71                      android:text="Goodreads"
```

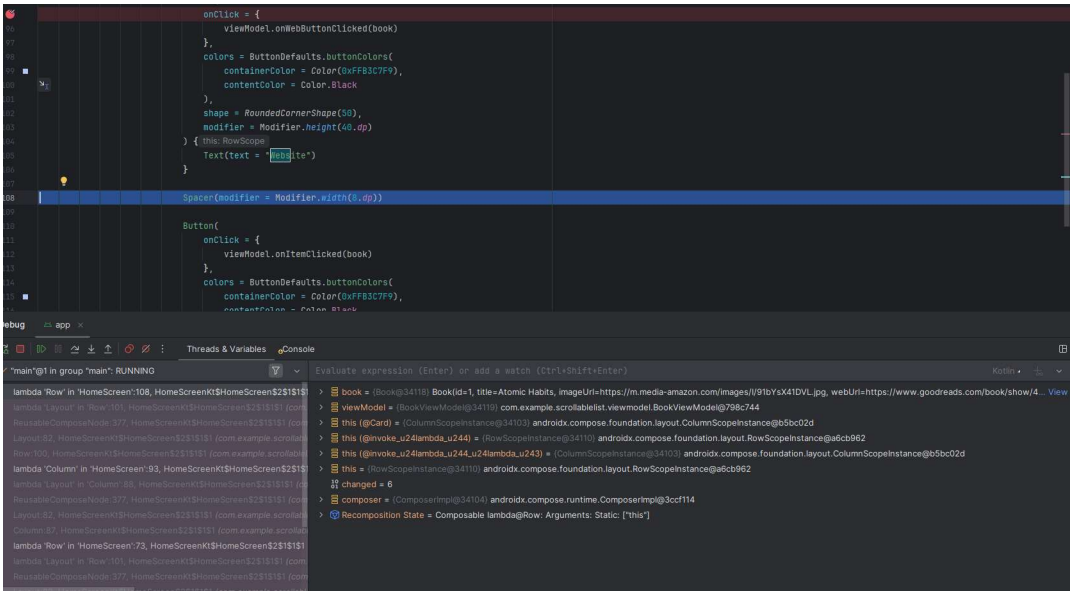
```
                        android:textColor="#000000"
                        android:backgroundTint="#B3C7F9"
                        android:textSize="14sp"
                        android:layout_marginEnd="8dp" />

                    <Button
                        android:id="@+id/buttonDetails"
                        android:layout_width="wrap_content"
                        android:layout_height="36dp"
                        android:text="Detail"
                        android:textColor="#000000"
                        android:backgroundTint="#B3C7F9"
                        android:textSize="14sp" />
                </LinearLayout>
            </LinearLayout>
        </LinearLayout>
</androidx.cardview.widget.CardView>
```

Tabel 1. 18 Source Code item_book_ .XML

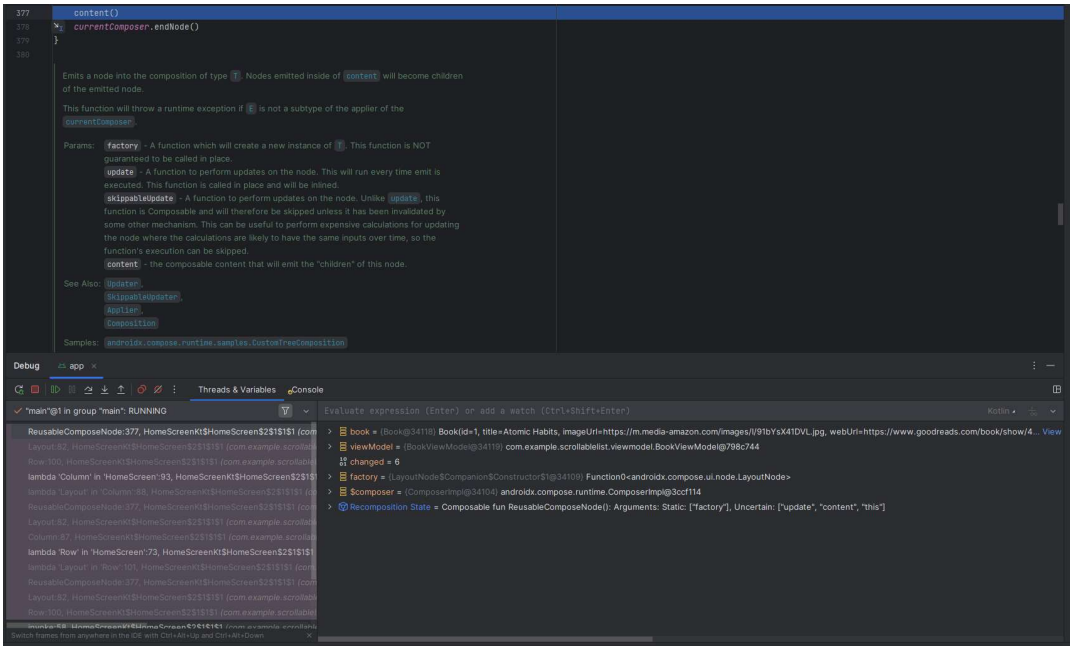
C. Output Program



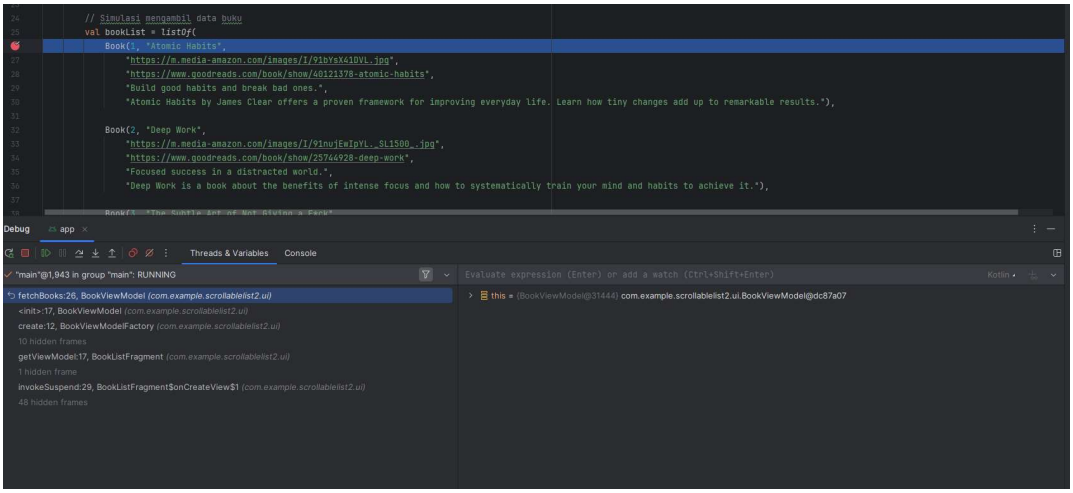
Gambar 1. 2 Screenshot Debugging Step Into Jetpack Compose



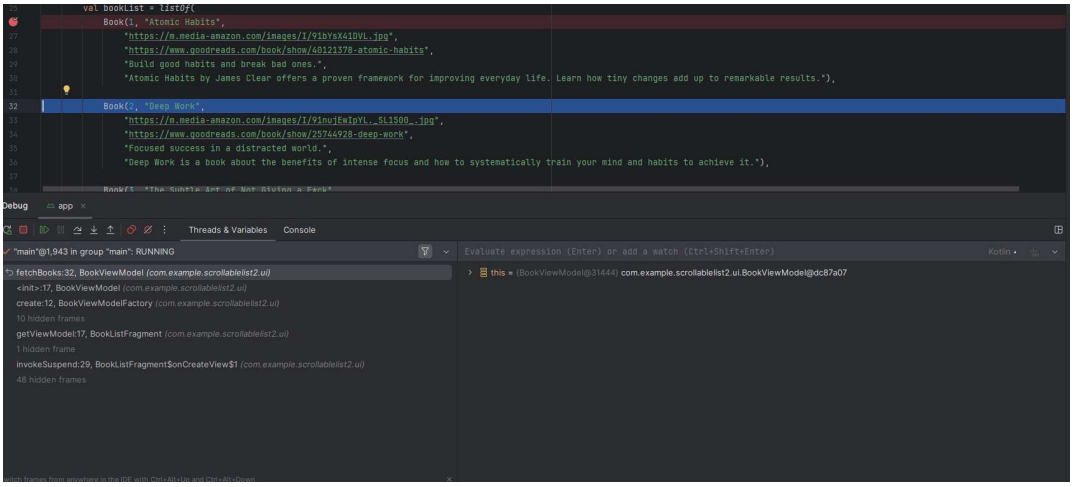
Gambar 1. 3 Screenshot Debugging Step Over Jetpack Compose



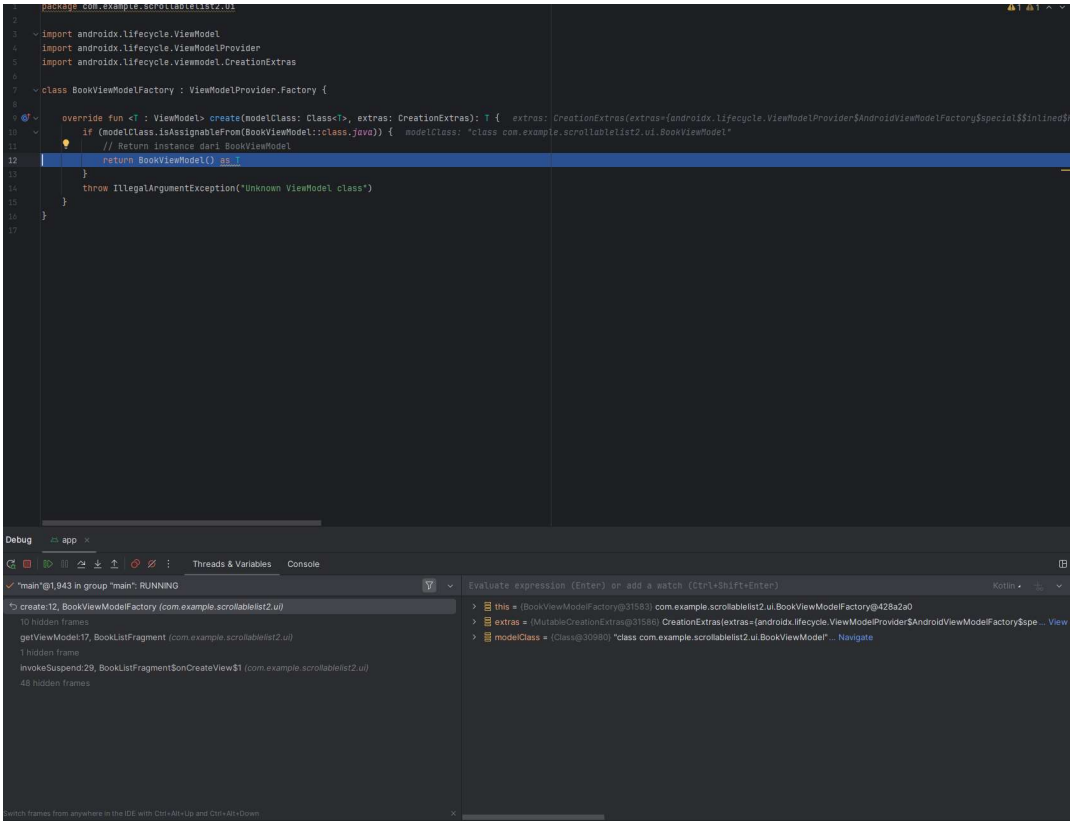
Gambar 1. 4 Screenshot Debugging Step Out Jetpack Compose



Gambar 1. 5 Screenshot Debugging Step Into XML



Gambar 1. 6 Screenshot Debugging Step Over XML



Gambar 1. 7 Screenshot Debugging Step Out XML

D. Pembahasan

MainActivity.kt Jetpack Compose:

- Entry point aplikasi Android Compose.
- NavGraph() dipanggil untuk menangani semua navigasi antar layar.
- Menggunakan tema ScrollableListTheme.

MyApplication.kt Jetpack Compose :

- Class `Application` yang digunakan untuk menginisialisasi Timber.
- Logger ini akan mencatat aktivitas penting, seperti klik tombol dan data yang dikirim ke layar detail.

NavGraph.kt Jetpack Compose :

- Menyediakan sistem navigasi antar layar menggunakan Jetpack Navigation Compose.
- Mendefinisikan dua rute: home dan detail/{...}.

Book.kt Jetpack Compose :

- Model data Book berisi informasi buku yang akan ditampilkan di daftar dan detail.

BookViewModel.kt Jetpack Compose :

- ViewModel menyimpan data Book dan status selectedBook.
- Menggunakan StateFlow agar UI bisa mengobservasi perubahan data secara reaktif.
- Melakukan logging menggunakan Timber.

BookViewModelFactory.kt

- Custom ViewModelFactory untuk memberikan parameter (source) ke ViewModel.

HomeScreen.kt

- Menampilkan daftar Book dalam bentuk LazyColumn.
- Ketika tombol Detail diklik, memanggil onBookClick di ViewModel dan navigasi ke halaman detail.

DetailScreen.kt

- Menampilkan informasi lengkap tentang buku yang dipilih.
- Logging judul buku yang diterima ke Logcat.

MainActivity.kt XML:

- AppCompatActivity digunakan karena UI berbasis pada XML dan ViewBinding atau findViewById.
- setContentView(...) memuat layout dari file XML (activity_main.xml).
- Navigasi dan fragment biasanya dikelola melalui NavHostFragment, FragmentManager, dan NavController.

Book.kt

- Model data buku yang digunakan di seluruh aplikasi.

BookViewModel.kt

- Menyediakan data buku secara reaktif (melalui StateFlow) untuk digunakan oleh UI.

BookViewModelFactory.kt

- Alas Digunakan jika ingin membuat BookViewModel secara manual (misal inject dependency).
- Menyesuaikan dengan API baru ViewModelProvider.Factory.

BookAdapter.kt

- Menampilkan setiap buku di dalam RecyclerView.
- Menangani klik tombol untuk membuka link atau navigasi ke detail.

BookListFragment.kt

- Mengamati StateFlow dari BookViewModel.
- Menampilkan daftar buku di RecyclerView.

BookDetailFragment.kt

- Menampilkan detail lengkap buku berdasarkan argumen navigasi dari BookListFragment.

activity_main.xml

- FragmentContainerView: tempat menampilkan fragment.
- app:navGraph: menunjuk ke file navigasi (res/navigation/nav_graph.xml).
- defaultNavHost: agar tombol back sistem bisa bekerja otomatis.

fragment_book_list.xml

- RecyclerView: menampilkan daftar buku.
- Digunakan oleh BookListFragment untuk mengisi dengan BookAdapter.

item_book.xml

- ImageView: menampilkan gambar buku.
- TextView: judul dan deskripsi singkat.
- Dua tombol: buka link dan lihat detail.
- Layout ini digunakan oleh BookAdapter.

fragment_book_detail.xml

- ScrollView: agar konten bisa digulir.
- Menampilkan judul buku, gambar, dan deskripsi lengkap.
- Data diisi dari BookDetailFragmentArgs.

Apa itu Application class?
Application adalah kelas dasar di Android yang digunakan untuk menyimpan dan mengelola status global aplikasi. Android membuat instance dari class ini sebelum aktivitas, service, atau receiver pertama kali dijalankan.

Fungsi dan Kegunaan :

1. **Inisialisasi Global** : Misalnya Timber, Hilt, Firebase, atau Analytics hanya perlu diinisialisasi sekali di seluruh aplikasi.
2. **Menyediakan Dependency Global** : Bisa digunakan sebagai semacam service locator sederhana untuk menyediakan object atau config global.
3. **Melacak Lifecycle Global** : User bisa menambahkan ActivityLifecycleCallbacks untuk mengetahui aktivitas apa yang aktif, masuk background, dll.
4. **Mocking/Test Support** : Di testing, user bisa mengganti Application class untuk menyuntikkan dependency yang berbeda.

E. Tautan GIT

https://github.com/gr1ff0m/Pemrograman-Mobile-Praktikum