

**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 Praktikum 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

LEMBAR PENGESAHAN.....	1
DAFTAR ISI	2
DAFTAR GAMBAR.....	3
DAFTAR TABEL	4
Soal Praktikum:	5
A. Source Code menggunakan Jetpack Compose	5
B. Source Code menggunakan XML	Error! Bookmark not defined.
C. Output Program	22
D. Pembahasan	23
E. Tautan GIT	27

DAFTAR GAMBAR

Gambar 1. 1 Screenshot Tampilan List Menggunakan API.....	22
---	----

DAFTAR TABEL

Tabel 1. 1 Source Code MainActivity.kt Jetpack Compose.....	5
Tabel 1. 2 Source Code ApiService.kt Jetpack Compose	6
Tabel 1. 3 Source Code DetailScreen.kt Jetpack Compose.....	8
Tabel 1. 4 Source Code Film.kt Jetpack Compose.....	8
Tabel 1. 5 Source Code FilmRepository.kt Jetpack Compose	8
Tabel 1. 6 Source Code FilmViewModel.kt Jetpack Compose.....	9
Tabel 1. 7 Source Code FlmViewModelFactory.kt Jetpack Compose	10
Tabel 1. 8 Source Code HomeScreen.kt Jetpack Compose.....	13
Tabel 1. 9 Source Code MainScreen.kt Jetpack Compose	14
Tabel 1. 10 Source Code Mappers.kt Jetpack Compose	14
Tabel 1. 11 Source Code MovieResponse.kt Jetpack Compose.....	15
Tabel 1. 12 Source Code MyApplication.kt Jetpack Compose.....	16
Tabel 1. 13 Source Code Navigation.kt Jetpack Compose.....	17
Tabel 1. 14 Source Code NetworkFilm.kt Jetpack Compose.....	17
Tabel 1. 15 Source Code PreferenceManager.kt Jetpack Compose	18
Tabel 1. 16 Source Code Result.kt Jetpack Compose	18
Tabel 1. 17 Source Code RetrofitInstance.kt Jetpack Compose.....	19
Tabel 1. 18 Source Code SharedPreferenceViewModel.kt Jetpack Compose	19
Tabel 1. 19 Source Code SharedPreferenceViewModelFactory.kt Jetpack Compose..	20
Tabel 1. 20 Source Code TMDBApi.kt Jetpack Compose.....	20
Tabel 1. 21 Source Code SettingScreen.kt Jetpack Compose	21

Soal Praktikum:

1. Lanjutkan aplikasi Android yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi sesuai ketentuan berikut:
 - a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.
 - b. Gunakan KotlinX Serialization sebagai library JSON.
 - c. Gunakan library seperti Coil atau Glide untuk image loading.
 - d. API yang digunakan pada modul ini adalah The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API:
<https://developer.themoviedb.org/docs/getting-started>
 - e. Implementasikan konsep data persistence (aplikasi menyimpan data walau pengguna keluar dari aplikasi) dengan SharedPreferences untuk menyimpan data ringan (seperti pengaturan aplikasi) dan Room untuk data relasional.
 - f. Gunakan caching strategy pada Room. Dibebaskan untuk memilih caching strategy yang sesuai, dan sertakan penjelasan kenapa menggunakan caching strategy tersebut.
 - g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose.

Aplikasi harus mempertahankan fitur-fitur yang dibuat pada modul sebelumnya.

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
11     private val apiKey = "9fad0dc9a0338ecf00596875e4cf5645" //
12     Masukkan API Key di sini
13
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         setContent {
17             ScrollableListTheme {
18                 NavGraph()
19             }
20         }
21     }
22 }
```

Tabel 1. 1 Source Code MainActivity.kt Jetpack Compose

ApiService.kt

```
1 package com.example.scrollablelist.data.remote
```

2	
3	import com.example.scrollablelist.model.MovieResponse
4	import retrofit2.http.GET
5	import retrofit2.http.Query
6	
7	interface ApiService {
8	@GET("movie/popular")
9	suspend fun getPopularMovies(
10	@Query("api_key") apiKey: String
11): MovieResponse
12	}

Tabel 1. 2 Source Code ApiService.kt Jetpack Compose

DetailScreen.kt

1	package com.example.scrollablelist.screens
2	
3	import androidx.activity.compose.BackHandler
4	import androidx.compose.foundation.Image
5	import androidx.compose.foundation.layout.*
6	import androidx.compose.foundation.shape.RoundedCornerShape
7	import androidx.compose.material3.*
8	import androidx.compose.runtime.Composable
9	import androidx.compose.ui.Modifier
10	import androidx.compose.ui.draw.clip
11	import androidx.compose.ui.graphics.Color
12	import androidx.compose.ui.text.font.FontWeight
13	import androidx.compose.ui.unit.dp
14	import androidx.compose.ui.unit.sp
15	import coil.compose.rememberAsyncImagePainter
16	import androidx.compose.material.icons.filled.ArrowBack
17	
18	@Composable
19	fun DetailScreen(
20	title: String,
21	imageUrl: String,
22	fullDescription: String,
23	onBack: () -> Unit
24) {
25	val imageUrl = "https://image.tmdb.org/t/p/w500"
26	val fullImageUrl = imageUrl + imageUrl
27	
28	Scaffold(
29	topBar = {
30	SmallTopAppBar(
31	title = { Text(text = "Detail Film") },
32	onBack = onBack
33)
34	}
35) { padding ->
36	Column(
37	modifier = Modifier
38	.fillMaxSize()
39	.padding(padding)
40	.padding(16.dp)

```

41         ) {
42             Image(
43                 painter
44 rememberAsyncImagePainter(fullImageUrl),
45                 contentDescription = title,
46                 modifier = Modifier
47                     .fillMaxWidth()
48                     .height(300.dp)
49                     .clip(RoundedCornerShape(16.dp))
50             )
51
52             Spacer(modifier = Modifier.height(16.dp))
53
54             Text(
55                 text = title,
56                 style = MaterialTheme.typography.headlineSmall,
57                 fontWeight = FontWeight.Bold,
58                 fontSize = 24.sp,
59                 color = Color.Black
60             )
61
62             Spacer(modifier = Modifier.height(12.dp))
63
64             Text(
65                 text = fullDescription,
66                 style = MaterialTheme.typography.bodyLarge,
67                 fontSize = 18.sp,
68                 color = Color.DarkGray
69             )
70         }
71     }
72 }
73
74 @OptIn(ExperimentalMaterial3Api::class)
75 @Composable
76 fun SmallTopAppBar(
77     title: @Composable () -> Unit,
78     onBack: () -> Unit
79 ) {
80     TopAppBar(
81         title = title,
82         navigationIcon = {
83             IconButton(onClick = onBack) {
84                 Icon(
85                     imageVector
86 androidx.compose.material.icons.Icons.Default.ArrowBack,
87                     contentDescription = "Back"
88                 )
89             }
90         }
91     )
92 }
93
94

```


95	
96	

Tabel 1. 3 Source Code DetailScreen.kt Jetpack Compose

Film.kt

1	package com.example.scrollablelist.model
2	
3	data class Film(
4	val id: Int,
5	val title: String,
6	val overview: String,
7	val posterPath: String?,
8	val backdropPath: String?,
9	val releaseDate: String?,
10	val homepage: String? = null
11)

Tabel 1. 4 Source Code Film.kt Jetpack Compose

FilmRepository.kt

1	package com.example.scrollablelist.data
2	
3	import com.example.scrollablelist.model.Film
4	import com.example.scrollablelist.network.TmdbApi
5	
6	class FilmRepository(
7	private val api: TmdbApi,
8	private val apiKey: String
9) {
10	suspend fun getPopularFilms(): List<Film> {
11	val response = api.getPopularMovies(apiKey)
12	
13	return response.results.map { networkFilm ->
14	Film(
15	id = networkFilm.id,
16	title = networkFilm.title,
17	overview = networkFilm.overview,
18	posterPath = networkFilm.posterPath,
19	backdropPath = networkFilm.backdropPath,
20	releaseDate = networkFilm.releaseDate,
21	homepage = null
22)
23	}
24	}
25	}

Tabel 1. 5 Source Code FilmRepository.kt Jetpack Compose

FilmViewModel.kt

1	package com.example.scrollablelist.ui
2	
3	import androidx.lifecycle.ViewModel
4	import androidx.lifecycle.viewModelScope
5	import com.example.scrollablelist.data.FilmRepository
6	import com.example.scrollablelist.model.Film

```

7 import kotlinx.coroutines.flow.MutableStateFlow
8 import kotlinx.coroutines.flow.StateFlow
9 import kotlinx.coroutines.flow.asStateFlow
10 import kotlinx.coroutines.launch
11 import timber.log.Timber
26
27 class FilmViewModel(
28     private val repository: FilmRepository
29 ) : ViewModel() {
30
31     private val _filmList =
32     MutableStateFlow<List<Film>>(emptyList())
33     val filmList: StateFlow<List<Film>> =
34     _filmList.asStateFlow()
35
36     private val _event = MutableStateFlow<Event?>(null)
37     val event: StateFlow<Event?> = _event.asStateFlow()
38
39     init {
40         loadFilmsFromApi()
41     }
42
43     private fun loadFilmsFromApi() {
44         viewModelScope.launch {
45             try {
46                 val films = repository.getPopularFilms()
47                 _filmList.value = films as List<Film>
48             } catch (e: Exception) {
49                 Timber.e("Gagal memuat film dari API:
50 ${e.message}")
51             }
52         }
53     }
54
55     fun onItemClickClicked(film: Film) {
56         _event.value = Event.NavigateToDetail(film)
57     }
58
59     fun onWebButtonClicked(url: String) {
60         _event.value = Event.OpenWebUrl(url)
61     }
62
63     fun clearEvent() {
64         _event.value = null
65     }
66
67     sealed class Event {
68         data class NavigateToDetail(val film: Film) : Event()
69         data class OpenWebUrl(val url: String) : Event()
70     }
71 }

```

Tabel 1. 6 Source Code FilmViewModel.kt Jetpack Compose

FilmViewModelFactory.kt

```
1 package com.example.scrollablelist.ui
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5 import com.example.scrollablelist.data.FilmRepository
6
7 class FilmViewModelFactory(
8     private val repository: FilmRepository
9 ) : ViewModelProvider.Factory {
10
11     override fun <T : ViewModel> create(modelClass: Class<T>):
12 T {
13
14     if (modelClass.isAssignableFrom(FilmViewModel::class.java)) {
15         @Suppress("UNCHECKED_CAST")
16         return FilmViewModel(repository) as T
17     }
18     throw IllegalArgumentException("Unknown ViewModel
19 class")
20 }
21 }
```

Tabel 1. 7 Source Code FlmViewModelFactory.kt Jetpack Compose

HomeScreen.kt

```
1 package com.example.scrollablelist.screens
2
3 import android.content.Intent
4 import android.net.Uri
5 import androidx.compose.foundation.Image
6 import androidx.compose.foundation.layout.*
7 import androidx.compose.foundation.lazy.LazyColumn
8 import androidx.compose.foundation.lazy.items
9 import androidx.compose.foundation.shape.RoundedCornerShape
10 import androidx.compose.material3.*
11 import androidx.compose.runtime.*
12 import androidx.compose.ui.Modifier
13 import androidx.compose.ui.draw.clip
14 import androidx.compose.ui.graphics.Color
15 import androidx.compose.ui.platform.LocalContext
16 import androidx.compose.ui.text.font.FontWeight
17 import androidx.compose.ui.unit.dp
18 import androidx.compose.ui.unit.sp
19 import androidx.lifecycle.viewmodel.compose.viewModel
20 import androidx.navigation.NavController
21 import coil.compose.rememberAsyncImagePainter
22 import com.example.scrollablelist.data.FilmRepository
23 import com.example.scrollablelist.network.RetrofitInstance
24 import com.example.scrollablelist.ui.FilmViewModel
25 import com.example.scrollablelist.ui.FilmViewModelFactory
26
27 @Composable
28 fun HomeScreen(navController: NavController) {
```

```

29     val context = LocalContext.current
30
31     val repository = remember {
32         FilmRepository(
33             api = RetrofitInstance.api,
34             apiKey = "9fad0dc9a0338ecf00596875e4cf5645"
35         )
36     }
37     val viewModel: FilmViewModel = viewModel(factory =
38 FilmViewModelFactory(repository))
39
40     val films by viewModel.filmList.collectAsState()
41     val event by viewModel.event.collectAsState()
42
43     val imageUrl = "https://image.tmdb.org/t/p/w500"
44
45     LaunchedEffect(event) {
46         when (event) {
47             is FilmViewModel.Event.NavigateToDetail -> {
48                 val film = (event as
49 FilmViewModel.Event.NavigateToDetail).film
50                 navController.navigate(
51
52 "detail/${Uri.encode(film.title)}/${Uri.encode(film.posterPath
53 ? : "")}/${Uri.encode(film.overview)}"
54                 )
55                 viewModel.clearEvent()
56             }
57             is FilmViewModel.Event.OpenWebUrl -> {
58                 val url = (event as
59 FilmViewModel.Event.OpenWebUrl).url
60                 val intent = Intent(Intent.ACTION_VIEW,
61 Uri.parse(url))
62                 navController.context.startActivity(intent)
63                 viewModel.clearEvent()
64             }
65             null -> Unit
66         }
67     }
68
69     LazyColumn(
70         contentPadding = PaddingValues(16.dp),
71         verticalArrangement = Arrangement.spacedBy(16.dp)
72     ) {
73         items(films) { film ->
74             Card(
75                 shape = RoundedCornerShape(16.dp),
76                 colors
77 CardDefaults.cardColors(containerColor = Color(0xFF2C2C2C)),
78                 modifier = Modifier.fillMaxWidth()
79             ) {
80                 Row(
81                     modifier = Modifier
82                         .padding(16.dp)

```

```

83         .fillMaxWidth()
84     ) {
85         Image(
86             painter
87             rememberAsyncImagePainter(imageBaseUrl + (film.posterPath ?:
88             "")),
89             contentDescription = film.title,
90             modifier = Modifier
91                 .size(100.dp)
92                 .clip(RoundedCornerShape(16.dp))
93         )
94
95         Spacer(modifier = Modifier.width(16.dp))
96
97         Column(
98             modifier = Modifier.weight(1f)
99         ) {
100             Text(
101                 text = film.title,
102                 style
103                 MaterialTheme.typography.titleMedium,
104                 color = Color.White,
105                 fontWeight = FontWeight.Bold,
106                 fontSize = 18.sp
107             )
108             Spacer(modifier
109             Modifier.height(4.dp))
110             Text(
111                 text = film.overview,
112                 style
113                 MaterialTheme.typography.bodyMedium,
114                 color = Color.Gray,
115                 maxLines = 3,
116                 overflow
117                 androidx.compose.ui.text.style.TextOverflow.Ellipsis
118             )
119
120             Spacer(modifier
121             Modifier.height(8.dp))
122
123             Row {
124                 Button(
125                     onClick
126                     viewModel.onItemClicked(film) },
127                     colors
128                     ButtonDefaults.buttonColors(
129                         containerColor
130                         Color(0xFFB3C7F9),
131                         contentColor = Color.Black
132                     ),
133                     shape
134                     RoundedCornerShape(50),
135                     modifier
136                     Modifier.height(40.dp)

```

```

137         ) {
138             Text(text = "Detail")
139         }
140     }
141 }
142 }
143 }
144 }
145 }
146 }

```

Tabel 1. 8 Source Code HomeScreen.kt Jetpack Compose

MainScreen.kt

```

1 package com.example.scrollablelist.ui
2
3 import androidx.compose.foundation.Image
4 import androidx.compose.foundation.clickable
5 import androidx.compose.foundation.layout.*
6 import androidx.compose.foundation.lazy.LazyColumn
7 import androidx.compose.foundation.lazy.items
8 import androidx.compose.material3.*
9 import androidx.compose.runtime.Composable
10 import androidx.compose.runtime.collectAsState
11 import androidx.compose.ui.Modifier
12 import androidx.compose.ui.layout.ContentScale
13 import androidx.compose.ui.unit.dp
14 import coil.compose.rememberAsyncImagePainter
15 import com.example.scrollablelist.model.Film
16
17 @Composable
18 fun MainScreen(viewModel: FilmViewModel) {
19     // Ganti films ke filmList, dan pastikan collectAsState
20     // dengan tipe eksplisit
21     val films = viewModel.filmList.collectAsState(initial =
22     emptyList())
23
24     Scaffold(
25         topBar = {
26             SmallTopAppBar(title = { Text("Popular Movies") })
27         }
28     ) { padding ->
29         LazyColumn(
30             modifier = Modifier
31                 .padding(padding)
32                 .fillMaxSize()
33         ) {
34             items(films.value) { film ->
35                 FilmListItem(film)
36             }
37         }
38     }
39 }
40
41 @OptIn(ExperimentalMaterial3Api::class)

```

42	@Composable
43	fun SmallAppBar(title: @Composable () -> Unit) {
44	AppBar(title = title)
45	}
46	
47	@Composable
48	fun FilmListItem(film: Film) {
49	Row(
50	modifier = Modifier
51	.fillMaxWidth()
52	.clickable { /* Navigate to details if needed */ }
53	.padding(8.dp)
54) {
55	Image(
56	painter
57	rememberAsyncImagePainter("https://image.tmdb.org/t/p/w500\${film
58	m.posterPath}"),
59	contentDescription = film.title,
60	modifier = Modifier.size(80.dp),
61	contentScale = ContentScale.Crop
62)
63	Spacer(modifier = Modifier.width(16.dp))
64	Column(modifier = Modifier.fillMaxWidth()) {
65	Text(text = film.title, style =
66	MaterialTheme.typography.titleMedium)
67	Spacer(modifier = Modifier.height(4.dp))
68	Text(
69	text = film.overview,
70	maxLines = 3,
71	style = MaterialTheme.typography.bodyMedium
72)
73	}
74	}
75	}

Table 1. 9 Source Code mainScreen.kt Jetpack Compose

Mappers.kt

1	package com.example.scrollablelist.model
2	
3	fun NetworkFilm.toFilm(): Film {
4	return Film(
5	id = id,
6	title = title,
7	overview = overview,
8	posterPath = posterPath,
9	backdropPath = backdropPath,
10	releaseDate = releaseDate,
11	homepage = null
12)
13	}

Table 1. 10 Source Code Mappers.kt Jetpack Compose

MovieResponse.kt

```

1 package com.example.scrollablelist.model
2
3 fun NetworkFilm.toFilm(): Film {
4     return Film(
5         id = id,
6         title = title,
7         overview = overview,
8         posterPath = posterPath,
9         backdropPath = backdropPath,
10        releaseDate = releaseDate,
11        homepage = null
12    )
13 }

```

Tabel 1. 11 Source Code MovieResponse.kt Jetpack Compose

MyApplication.kt

```

1 package com.example.scrollablelist
2
3 import android.app.Application
4 import
5 com.example.scrollablelist.data.FilmRepository
6 import com.example.scrollablelist.network.TmdbApi
7 import
8 com.jakewharton.retrofit2.converter.kotlinx.serialization.asConverterFactory
9
10 import kotlinx.serialization.json.Json
11 import okhttp3.MediaType.Companion.toMediaType
12 import okhttp3.OkHttpClient
13 import retrofit2.Retrofit
14 import timber.log.Timber
15
16 class MyApplication : Application() {
17
18     lateinit var filmRepository: FilmRepository
19     private set
20
21     lateinit var preferenceManager:
22 PreferenceManager
23     private set
24
25     override fun onCreate() {
26         super.onCreate()
27
28         Timber.plant(Timber.DebugTree())
29
30         preferenceManager = PreferenceManager(this)
31
32         val contentType =
33 "application/json".toMediaType()
34         val json = Json { ignoreUnknownKeys = true
35     }
36
37         val okHttpClient =
38 OkHttpClient.Builder().build()

```



```

39         val retrofit = Retrofit.Builder()
40
41         .baseUrl("https://api.themoviedb.org/3/")
42         .client(okHttpClient)
43
44         .addConverterFactory(json.asConverterFactory(contentType))
45         .build()
46
47         val api = retrofit.create(TmdbApi::class.java)
48
49         val apiKey = "9fad0dc9a0338ecf00596875e4cf5645"
50
51         filmRepository = FilmRepository(api, apiKey)
52     }
53 }

```

Tabel 1. 12 Source Code MyApplication.kt Jetpack Compose

Navigation.kt

```

1 package com.example.scrollablelist.navigation
2
3 import androidx.compose.runtime.Composable
4 import androidx.navigation.NavHostController
5 import androidx.navigation.compose.NavHost
6 import androidx.navigation.compose.composable
7 import
8 androidx.navigation.compose.rememberNavController
9
10 import
11 com.example.scrollablelist.screens.DetailScreen
12 import
13 com.example.scrollablelist.screens.HomeScreen
14
15 @Composable
16 fun NavGraph(navController: NavHostController =
17 rememberNavController()) {
18     NavHost(
19         navController = navController,
20         startDestination = "home"
21     ) {
22         composable("home") {
23             HomeScreen(navController)
24         }
25
26         composable("detail/{title}/{imageUrl}/{fullDescription}") { backStackEntry ->
27             val title =
28             backStackEntry.arguments?.getString("title") ?:
29             ""
30
31

```

32	val	imageUrl	=
33	backStackEntry.arguments?.getString("imageUrl")		
34	?: ""		
35	val	fullDescription	=
36	backStackEntry.arguments?.getString("fullDescription")		
37	?: ""		
38			
39	DetailScreen(
40	title = title,		
41	imageUrl = imageUrl,		
42	fullDescription	=	
43	fullDescription,		
44	onBack	=	{
45	navController.popBackStack() }		
46)		
47	}		
48	}		
49	}		
50			

Table 1. 13 Source Code Navigation.kt Jetpack Compose

NetworkFilm.kt

1	package com.example.scrollablelist.navigation
2	package com.example.scrollablelist.model
3	
4	import kotlinx.serialization.SerialName
5	import kotlinx.serialization.Serializable
6	import
7	kotlinx.serialization.InternalSerializationApi
8	import kotlin.annotation.AnnotationTarget.CLASS
9	
10	@OptIn(InternalSerializationApi::class)
11	@Serializable
12	data class NetworkFilm(
13	val id: Int,
14	val title: String,
15	val overview: String,
16	@SerialName("poster_path") val posterPath:
17	String?,
18	@SerialName("backdrop_path") val
19	backdropPath: String?,
20	@SerialName("release_date") val releaseDate:
21	String?
22)

Table 1. 14 Source Code NetworkFilm.kt Jetpack Compose

PreferenceManager.kt

1	package com.example.scrollablelist
2	
3	import android.content.Context
4	import android.content.SharedPreferences
5	import kotlinx.coroutines.flow.Flow
6	import kotlinx.coroutines.flow.MutableStateFlow

```

7
8 class PreferenceManager(context: Context) {
9
10     private val prefs: SharedPreferences =
11
12     context.getSharedPreferences("app_prefs",
13     Context.MODE_PRIVATE)
14
15     private val _isDarkMode =
16     MutableStateFlow(prefs.getBoolean(KEY_DARK_MODE
17     , false))
18     val isDarkMode: Flow<Boolean> = _isDarkMode
19
20     fun setDarkMode(enabled: Boolean) {
21         prefs.edit().putBoolean(KEY_DARK_MODE,
22         enabled).apply()
23         _isDarkMode.value = enabled
24     }
25
26     companion object {
27         private const val KEY_DARK_MODE =
28         "key_dark_mode"
29     }
30 }

```

Tabel 1. 15 Source Code PreferenceManager.kt Jetpack Compose

Result.kt

```

1 package com.example.scrollablelist
2
3
4 sealed class Result<out T> {
5     object Loading : Result<Nothing>()
6     data class Success<T>(val data: T) :
7     Result<T>()
8     data class Error(val exception: Throwable) :
9     Result<Nothing>()
10 }

```

Tabel 1. 16 Source Code Result.kt Jetpack Compose

RetrofitInstance.kt

```

1 package com.example.scrollablelist.network
2
3 import
4 com.jakewharton.retrofit2.converter.kotlinx.ser
5 ialization.asConverterFactory
6 import kotlinx.serialization.json.Json
7 import okhttp3.MediaType.Companion.toMediaType
8 import okhttp3.OkHttpClient
9 import retrofit2.Retrofit
10
11 object RetrofitInstance {
12
13     private val json = Json {

```

```

14         ignoreUnknownKeys = true // biar bisa
15 ignore field yg gak dipakai
16     }
17
18     private val client = OkHttpClient.Builder()
19         .build()
20
21     private val retrofit by lazy {
22         Retrofit.Builder()
23
24         .baseUrl("https://api.themoviedb.org/3/") //
25         base URL TMDb API
26         .client(client)
27
28         .addConverterFactory(json.asConverterFactory("a
29 pplication/json".toMediaType()))
30         .build()
31     }
32
33     val api: TmdbApi by lazy {
34         retrofit.create(TmdbApi::class.java)
35     }
36 }

```

Tabel 1. 17 Source Code RetrofitInstance.kt Jetpack Compose

SharedPreferencesViewModel.kt

```

1 package com.example.scrollablelist
2
3 import androidx.lifecycle.ViewModel
4 import kotlinx.coroutines.flow.StateFlow
5 import kotlinx.coroutines.flow.SharingStarted
6 import kotlinx.coroutines.flow.stateIn
7 import kotlinx.coroutines.flow.map
8 import kotlinx.coroutines.CoroutineScope
9 import kotlinx.coroutines.Dispatchers
10
11 class SharedPreferencesViewModel(private val
12 preferenceManager: PreferenceManager) :
13 ViewModel() {
14
15     val isDarkMode: StateFlow<Boolean> =
16 preferenceManager.isDarkMode
17     .stateIn(
18         CoroutineScope(Dispatchers.Main),
19         SharingStarted.Eagerly,
20         false
21     )
22
23     fun toggleDarkMode() {
24         val current = isDarkMode.value
25         preferenceManager.setDarkMode(!current)
26     }
27 }

```

Tabel 1. 18 Source Code SharedPreferencesViewModel.kt Jetpack Compose

SharedPreferenceViewModelFactory.kt

```
1 package com.example.scrollablelist
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5
6 class SharedPreferenceViewModelFactory(
7     private val preferenceManager:
8     PreferenceManager
9 ) : ViewModelProvider.Factory {
10
11     override fun <T : ViewModel>
12     create(modelClass: Class<T>): T {
13         if
14         (modelClass.isAssignableFrom(SharedPreferenceVi
15         ewModel::class.java)) {
16             @Suppress("UNCHECKED_CAST")
17             return
18             SharedPreferenceViewModel(preferenceManager) as
19             T
20         }
21         throw IllegalArgumentException("Unknown
22         ViewModel class")
23     }
24 }
```

Table 1. 19 Source Code SharedPreferenceViewModelFactory.kt Jetpack Compose

TMDBApi.kt

```
1 package com.example.scrollablelist.network
2
3 import
4 com.example.scrollablelist.model.MovieResponse
5 import retrofit2.http.GET
6 import retrofit2.http.Query
7
8 interface TmdbApi {
9     @GET("movie/popular")
10     suspend fun getPopularMovies(
11         @Query("api_key") apiKey: String
12     ): MovieResponse
13 }
```

Table 1. 20 Source Code TMDBApi.kt Jetpack Compose

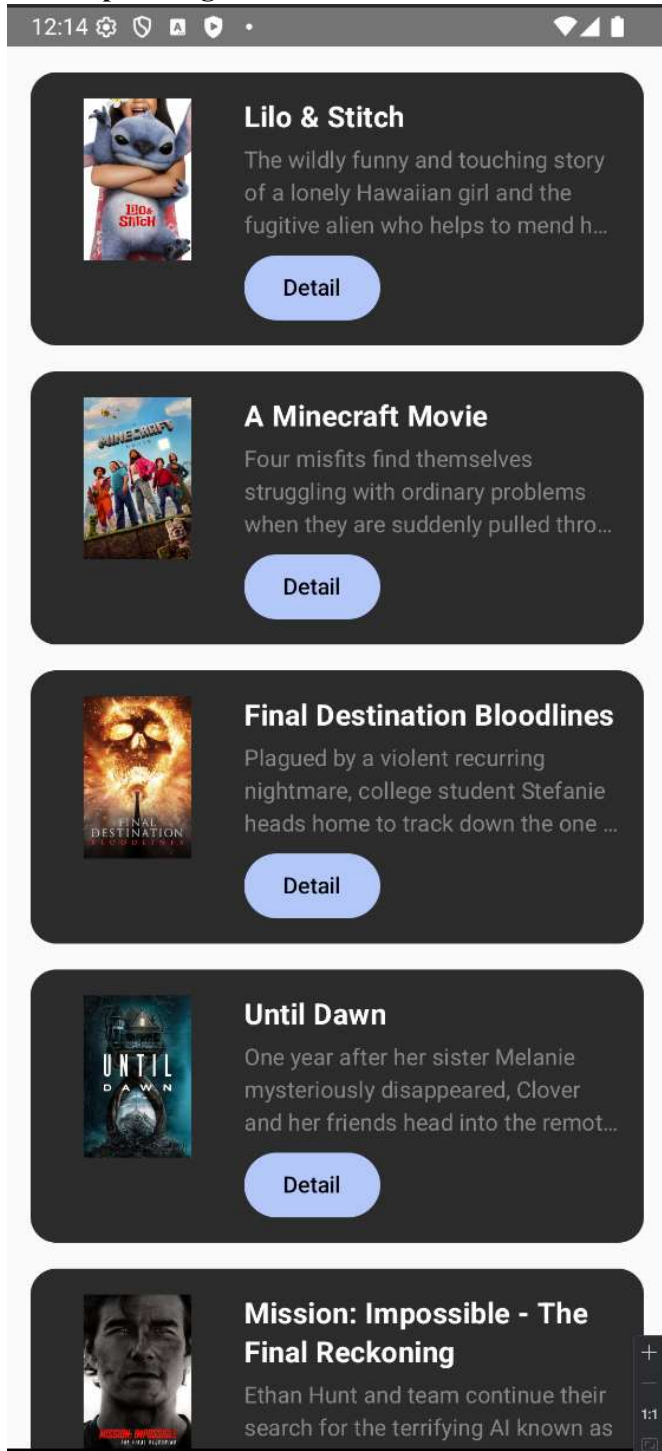
SettingsScreem.kt

```
1 package com.example.scrollablelist.screens
2
3 import androidx.compose.material3.Switch
4 import androidx.compose.material3.Text
5 import androidx.compose.foundation.layout.Row
6 import
7 androidx.compose.foundation.layout.padding
8 import androidx.compose.runtime.Composable
```

9	import androidx.compose.runtime.collectAsState
10	import androidx.compose.ui.Modifier
11	import androidx.compose.ui.unit.dp
12	import
13	com.example.scrollablelist.SharedPreferenceView
14	Model
15	
16	@Composable
17	fun SettingsScreen(viewModel:
18	SharedPreferenceViewModel) {
19	val isDarkMode =
20	viewModel.isDarkMode.collectAsState()
21	
22	Row(modifier = Modifier.padding(16.dp)) {
23	Text(text = "Dark Mode")
24	Switch(
25	checked = isDarkMode.value,
26	onCheckedChange = {
27	viewModel.toggleDarkMode() },
28	modifier = Modifier.padding(start =
29	8.dp)
30)
31	}
32	}

Table 1. 21 Source Code SettingScreen.kt Jetpack Compose

B. Output Program



Gambar 1. 1 Screenshot Tampilan List Menggunakan API

C. Pembahasan

MainActivity.kt Jetpack Compose:

- MainActivity adalah titik awal aplikasi.
- setContent digunakan untuk menampilkan UI berbasis **Jetpack Compose**.
- ScrollableListTheme { NavGraph() } memastikan bahwa seluruh UI mengikuti tema tertentu dan menggunakan sistem navigasi terpisah (NavGraph) untuk berpindah antar halaman (screen).
- apiKey disimpan secara lokal, sebaiknya diletakkan di BuildConfig atau local.properties untuk keamanan.

ApiService.kt

- Menggunakan **Retrofit** untuk mengambil data dari endpoint movie/popular di TMDB.
- Fungsi suspend artinya dapat dipanggil dalam coroutine (asynchronous).
- Menggunakan query api_key untuk otorisasi.
- Return type adalah MovieResponse, yaitu response model JSON dari TMDB.

Film.kt Jetpack Compose :

- Film adalah model yang digunakan dalam **lapisan domain** (digunakan oleh UI dan repository).
- Disusun agar bersih dari detail jaringan (network response), memudahkan testing dan pemeliharaan.
- Dipisahkan dari model response agar modular dan tidak terikat pada struktur response TMDB.

FilmRepository.kt Jetpack Compose :

- Menghubungkan data dari api ke dalam model Film.
- api.getPopularMovies(apiKey) adalah panggilan ke Retrofit.
- Kemudian, hasilnya dikonversi dari response model ke domain model Film.

DetailScreen.kt

- UI detail film, menampilkan **judul, gambar, dan deskripsi** film.
- Gunakan **Coil** (rememberAsyncImagePainter) untuk memuat gambar dari URL TMDB.
- Menggunakan Scaffold dan SmallTopAppBar untuk UI yang konsisten dan material-based.
- onBackPressed digunakan agar tombol back bisa kembali ke screen sebelumnya dengan BackHandler.

FilmViewModel.kt

Fungsi Utama:

- Mengelola **state film** menggunakan StateFlow agar Compose bisa observe data secara reaktif.
- Menangani navigasi dan aksi user dengan event handler berbasis sealed class (Event).

Penjelasan Teknis:

- `_filmList` menyimpan daftar film dan diamati dari UI menggunakan `collectAsState()`.
- `_event` digunakan sebagai kanal komunikasi satu arah (event-driven) ke UI, seperti `NavigateToDetail` atau `OpenWebUrl`.
- `viewModelScope.launch {}` memungkinkan coroutine berjalan lifecycle-aware (tidak memory leak).
- `loadFilmsFromApi()` memanggil repository dan menangani error log dengan `Timber`.
-

FilmViewModelFactory.kt

Fungsi Utama:

- Membuat instance dari `FilmViewModel` dengan repository sebagai dependensi eksternal (dependency injection manual).

Penjelasan Teknis:

- Diperlukan karena `FilmViewModel` tidak memiliki constructor kosong.
- Digunakan di `HomeScreen` untuk membuat `ViewModel` secara benar.

HomeScreen.kt

Fungsi Utama:

- Menampilkan daftar film populer.
- Meng-handle klik tombol Detail dan Web dengan Event dari `ViewModel`.
- Menginisialisasi repository dan `ViewModel` (biasanya bisa dikelola oleh DI seperti Hilt).

Penjelasan Teknis:

- `LaunchedEffect(event)` digunakan untuk menjalankan navigasi satu kali saat event terjadi.
- Navigasi dilakukan dengan `navController.navigate(...)`.
- Gambar dimuat menggunakan `Coil (rememberAsyncImagePainter)`.
- List film ditampilkan menggunakan `LazyColumn`.

MainScreen.kt

Fungsi Utama:

- Menampilkan daftar film seperti `HomeScreen`, namun tidak memuat tombol dan event detail.
- Tampaknya merupakan screen awal atau versi sederhana dari tampilan utama.

Penjelasan Teknis:

- Menggunakan `LazyColumn` untuk menampilkan list.
- Tidak ada aksi klik di `FilmListItem` (komentar placeholder).
- Gunakan `rememberAsyncImagePainter` untuk gambar (via `Coil`).

Mappers.kt

Fungsi Utama:

- Mengonversi dari `NetworkFilm` (model response API) ke `Film` (domain model).

Penjelasan Teknis:

- Memisahkan model dari API dengan model domain merupakan praktik Clean Architecture agar layer UI dan repository tidak terikat dengan detail dari API.
- Digunakan di `FilmRepository`.

MovieResponse.kt

Penjelasan:

- Fungsi ekstensi untuk mengonversi `NetworkFilm` (response dari API) menjadi `Film` (model domain).
- Ini penting karena kamu memisahkan data yang didapat dari internet (`NetworkFilm`) dengan data yang digunakan di aplikasi (`Film`), sebuah prinsip Clean Architecture.

MyApplication.kt

Isinya:

- Inisialisasi global seperti `FilmRepository` dan `PreferenceManager`.
- Konfigurasi Retrofit menggunakan Kotlin Serialization.
- Setup Timber untuk logging.

Penjelasan:

- File ini menjalankan `Application`, yang hanya dibuat sekali saat app dijalankan.
- `filmRepository` dan `preferenceManager` disiapkan di sini agar bisa digunakan di mana saja, misalnya disuntikkan ke `ViewModel`.
- `Json { ignoreUnknownKeys = true }` membuat deserializer lebih fleksibel terhadap perubahan API.

NetworkFilm.kt

Penjelasan:

- `@Serializable`: Mengaktifkan serialisasi/deserialisasi dengan KotlinX Serialization.
- `@SerializedName(...)`: Menghubungkan nama di JSON dengan properti Kotlin.
- Ini adalah representasi data mentah dari TMDb API.

Navigation.kt

Isinya:

- Menentukan flow navigasi antara dua screen: `HomeScreen` dan `DetailScreen`.
- Gunakan `NavHost` dari Jetpack Compose Navigation.

Penjelasan:

- `NavHost` mendefinisikan struktur screen.
- Data dari `HomeScreen` ke `DetailScreen` dikirim lewat path parameter (`detail/{title}/{imageUrl}/{fullDescription}`).
- `Uri.encode(...)` di `HomeScreen` penting agar teks tidak rusak saat dikirim melalui URL.

PreferenceManager.kt

Isinya:

- Membungkus `SharedPreferences` agar bisa digunakan secara reaktif menggunakan `StateFlow`.
- Menyimpan dan membaca mode gelap/terang (dark mode).

Penjelasan:

- `MutableStateFlow` memungkinkan UI bereaksi jika user mengaktifkan atau mematikan dark mode.
- Menyediakan API `setDarkMode(enabled)` dan `isDarkMode: Flow<Boolean>`.

Result.kt

Fungsi:

- Membungkus status dari operasi async (misal: memanggil API).

Penjelasan:

`sealed class Result<out T>` memiliki tiga kemungkinan:

- `Loading`: Proses sedang berlangsung.
- `Success<T>(val data: T)`: Berhasil dan menyimpan hasil.
- `Error(val exception: Throwable)`: Gagal dan menyimpan error-nya.

RetrofitInstance.kt

Fungsi:

- Singleton untuk membuat instance Retrofit dan `TmdbApi`.

Penjelasan:

- `Json { ignoreUnknownKeys = true }`: Menghindari crash jika JSON API punya field yang tidak didefinisikan di model.
- `by lazy`: Menunda inisialisasi sampai digunakan.

SharedPreferenceViewModel.kt

Fungsi:

- Mengelola dan mengekspos status dark mode dari `PreferenceManager`.

Penjelasan:

- `stateIn(...)`: Mengubah Flow jadi StateFlow, sehingga bisa digunakan di Compose dan Live UI update.
- `toggleDarkMode()`: Membalik status dark mode dan menyimpannya.

SharedPreferenceViewModelFactory.kt

Fungsi:

- Membuat instance `SharedPreferenceViewModel` dengan constructor yang membutuhkan parameter (`PreferenceManager`).

Penjelasan:

- Karena `ViewModelProvider` default hanya bisa buat `ViewModel` tanpa argumen, kamu butuh factory ini.

TmdbApi.kt

Fungsi:

- Interface untuk endpoint Retrofit ke API TMDb.

Penjelasan:

- Endpoint: `GET https://api.themoviedb.org/3/movie/popular?api_key=...`
- Hasil dikembalikan sebagai `MovieResponse` (yang berisi list `NetworkFilm`).

SettingsScreen.kt

Fungsi:

- Tampilan UI untuk mengubah mode gelap/terang dengan Switch.

Penjelasan:

- Menggunakan `collectAsState()` agar UI auto update saat state `isDarkMode` berubah.

- Ketika Switch diklik, `viewModel.toggleDarkMode()` dipanggil.

E. Tautan GIT

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