# LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE MODUL 5

**Connect to the Internet**

**Oleh:**

**Natalie Grace Katiandagho NIM. 2310817120003**

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS TEKNIK**

**UNIVERSITAS LAMBUNG MANGKURAT**

**JUNI 2025**

# LEMBAR PENGESAHAN
# LAPORAN PRAKTIKUM PEMROGRAMAN I
# MODUL 5

Laporan Praktikum Pemrograman Mobile Modul 5:
Connect to the internet ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Prakitkum ini dikerjakan oleh:

| | |
|---|---|
| Nama Praktikan | : Natalie Grace Katiandagho |
| NIM | : 2310817120003 |

| | |
|---|---|
| Menyetujui, | Mengetahui, |
| Asisten Praktikum | Dosen Penanggung Jawab Praktikum |
| Natalie Grace Katiandagho | Muti`a Maulida S.Kom M.T.I |
| NIM. 2310817120003 | NIP. 19881027 201903 20 13 |

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR GAMBAR

## SOAL 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 bebas, contoh API gratis 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 (misalnya offline-first app, pengaturan dark/light mode, fitur favorite, dll)

f. Gunakan caching strategy pada Room..

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

*Tabel 1. ManhwaDao.kt Modul 5*

```kotlin
package com.example.gracemanhwa_picks.data.local.dao

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import androidx.room.Update
import
com.example.gracemanhwa_picks.data.local.entity.ManhwaE
ntity
import kotlinx.coroutines.flow.Flow

@Dao
interface ManhwaDao {
    @Query("SELECT * FROM manhwas ORDER BY title ASC")
    fun getAllManhwas(): Flow<List<ManhwaEntity>>

    @Query("SELECT * FROM manhwas WHERE id = :id")
    fun getManhwaById(id: Int): Flow<ManhwaEntity?>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAll(manhwas: List<ManhwaEntity>)

    @Update
    suspend fun updateManhwa(manhwa: ManhwaEntity)

    @Query("DELETE FROM manhwas")
    suspend fun deleteAll()
}
```

*Tabel 2. ManhwaEntity.kt Modul 5*

```
        package
com.example.gracemanhwa_picks.data.local.entity

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "manhwas")
data class ManhwaEntity(
    @PrimaryKey
    val id: Int,
    val title: String,
    val author: String,
    val description: String,
    val imageUrl: String,
    val url: String,
    val isFavorite: Boolean = false
)
```

*Tabel 3. ManhwaDatabase.kt Modul 5*

```
package
com.example.gracemanhwa_picks.data.local.entity

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "manhwas")
data class ManhwaEntity(
    @PrimaryKey
    val id: Int,
    val title: String,
    val author: String,
    val description: String,
    val imageUrl: String,
    val url: String,
    val isFavorite: Boolean = false
)
```

*Tabel 4. ManhwaDto.kt Modul 5*

```
package com.example.gracemanhwa_picks.data.model

import kotlinx.serialization.SerialName
import kotlinx.serialization.Serializable

@Serializable
data class ManhwaDto(
    @SerialName("id")
    val id: Int,

    @SerialName("title")
    val title: String,

    @SerialName("author")
    val author: String,

    @SerialName("description")
    val description: String,

    @SerialName("imageUrl")
    val imageUrl: String,

    @SerialName("url")
    val url: String
)
```

*Tabel 5. ManhwaApiService.kt Modul 5*

```
package com.example.gracemanhwa_picks.data.model

import kotlinx.serialization.SerialName
import kotlinx.serialization.Serializable

```

```kotlin
@Serializable
data class ManhwaDto(
    @SerialName("id")
    val id: Int,

    @SerialName("title")
    val title: String,

    @SerialName("author")
    val author: String,

    @SerialName("description")
    val description: String,

    @SerialName("imageUrl")
    val imageUrl: String,

    @SerialName("url")
    val url: String
)
```

*Tabel 6.RetrofitInstance.kt Modul 5*

```kotlin
package com.example.gracemanhwa_picks.data.remote

import
com.example.gracemanhwa_picks.data.remote.api.ManhwaApiServi
ce
import
com.jakewharton.retrofit2.converter.kotlinx.serialization.as
ConverterFactory
import kotlinx.serialization.json.Json
import okhttp3.MediaType.Companion.toMediaType
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit

object RetrofitInstance {

    private const val BASE_URL =
"https://manhwagrzzz.free.beeceptor.com"

    private val json = Json {
        ignoreUnknownKeys = true
    }

    private val logging = HttpLoggingInterceptor().apply {
        level = HttpLoggingInterceptor.Level.BODY
    }

    private val client = OkHttpClient.Builder()
        .addInterceptor(logging)
        .build()

    val api: ManhwaApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .client(client)

.addConverterFactory(json.asConverterFactory("application/js
on".toMediaType()))
            .build()
            .create(ManhwaApiService::class.java)
    }
}
```

*Tabel 7.ManhwaRepository.kt*

```
package com.example.gracemanhwa_picks.data.repository

import
com.example.gracemanhwa_picks.data.local.dao.ManhwaDao
import
com.example.gracemanhwa_picks.data.local.entity.ManhwaE
ntity
import
com.example.gracemanhwa_picks.data.remote.api.ManhwaApi
Service
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.firstOrNull
import java.io.IOException
import android.util.Log

class ManhwaRepository(
    private val apiService: ManhwaApiService,
    private val manhwaDao: ManhwaDao
) {

    fun getAllManhwas(): Flow<List<ManhwaEntity>> =
manhwaDao.getAllManhwas()

    fun getManhwaById(id: Int): Flow<ManhwaEntity?> =
manhwaDao.getManhwaById(id)

    suspend fun refreshManhwas() {
        try {
            val remoteManhwas =
apiService.getAllManhwas()
            val favoriteManhwas =
manhwaDao.getAllManhwas().firstOrNull()?.filter {
it.isFavorite }?.map { it.id }
                    ?: emptyList()

            val manhwaEntities = remoteManhwas.map {
dto ->
                ManhwaEntity(
                    id = dto.id,
                    title = dto.title,
                    author = dto.author,
                    description = dto.description,
                    imageUrl = dto.imageUrl,
                    url = dto.url,
                )
            }

            manhwaDao.insertAll(manhwaEntities)

        } catch (e: IOException) {
            Log.e("ManhwaRepository", "Gagal refresh
manhwas (IO): ", e)
            e.printStackTrace()
        } catch (e: Exception) {
            Log.e("ManhwaRepository", "Gagal refresh
manhwas (Exception): ", e)
            e.printStackTrace()
        }
    }
    suspend fun updateFavoriteStatus(manhwa:
ManhwaEntity, isFavorite: Boolean) {
        manhwaDao.updateManhwa(manhwa.copy(isFavorite =
isFavorite))
    }
}
```

*Tabel 8. Injection.kt Modul 5*

```kotlin
package com.example.gracemanhwa_picks.di

import android.content.Context
import
com.example.gracemanhwa_picks.data.local.ManhwaDatabase
import
com.example.gracemanhwa_picks.data.remote.RetrofitInsta
nce
import
com.example.gracemanhwa_picks.data.repository.ManhwaRep
ository

object Injection {
    fun provideRepository(context: Context):
ManhwaRepository {
        val database =
ManhwaDatabase.getDatabase(context)
        val apiService = RetrofitInstance.api
        return ManhwaRepository(apiService,
database.manhwaDao())
    }
}
```

*Tabel 9. ManhwaCard.kt Modul 5*

```kotlin
package
com.example.gracemanhwa_picks.presentation.components

import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import
androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Favorite
import
androidx.compose.material.icons.outlined.FavoriteBorde
r
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import coil.compose.AsyncImage
import
com.example.gracemanhwa_picks.data.local.entity.Manhwa
Entity

@Composable
fun ManhwaCard(
    manhwa: ManhwaEntity,
    onFavoriteClick: () -> Unit,
    onDetailClick: () -> Unit,
    modifier: Modifier = Modifier
) {
    val context = LocalContext.current

    Card(
        modifier = modifier
            .fillMaxWidth()
            .clickable { onDetailClick() },
        shape = RoundedCornerShape(16.dp),
        elevation =
CardDefaults.cardElevation(defaultElevation = 4.dp)
    ) {
        Box(modifier =
Modifier.height(IntrinsicSize.Min)) {
            Row(
                verticalAlignment =
Alignment.CenterVertically
            ) {
                AsyncImage(
                    model = manhwa.imageUrl,
                    contentDescription = manhwa.title,
                    contentScale = ContentScale.Crop,
                    modifier =
Modifier.width(120.dp).fillMaxHeight()
                )
                Column(
                    modifier = Modifier.padding(16.dp)
                ) {
                    Text(
                        text = manhwa.title,
                        style =
MaterialTheme.typography.titleMedium,
                        fontWeight = FontWeight.Bold,
                        maxLines = 2,
                        overflow =
TextOverflow.Ellipsis
                    )
                    Spacer(modifier =
Modifier.height(4.dp))
                    Text(
                        text = "By ${manhwa.author}",
                        style =
MaterialTheme.typography.bodySmall,
                        color = Color.Gray
                    )
                    Spacer(modifier =
Modifier.height(8.dp))
                    Row(
                        modifier =
Modifier.fillMaxWidth(),
                        horizontalArrangement =
Arrangement.spacedBy(8.dp)
                    ) {
                        OutlinedButton(
                            onClick = onDetailClick,
                            modifier =
Modifier.weight(1f)
                        ) { Text("Detail", fontSize =
12.sp) }

                        Button(
                            onClick = {
                                val intent =
Intent(Intent.ACTION_VIEW, Uri.parse(manhwa.url))
context.startActivity(intent)
                            },
                            modifier =
Modifier.weight(1f)
                        ) { Text("Baca", fontSize =
12.sp) }
                    }
                }
            }
            IconButton(
                onClick = onFavoriteClick,
                modifier = Modifier
                    .align(Alignment.TopEnd)
                    .padding(8.dp)
            ) {
                Icon(
                    imageVector = if
(manhwa.isFavorite) Icons.Filled.Favorite else
Icons.Outlined.FavoriteBorder,
                    contentDescription = "Toggle
Favorite",
                    tint = if (manhwa.isFavorite)
Color.Red else Color.White
                )
            }
        }
    }
}
```

*Tabel 10. Navigation.kt Modul 5*

```
package
com.example.gracemanhwa_picks.presentation.navigation

import androidx.compose.runtime.Composable
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavHostController
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.navArgument
import
com.example.gracemanhwa_picks.presentation.screen.Manhwa
DetailScreen
import
com.example.gracemanhwa_picks.presentation.screen.Manhwa
ListScreen
import
com.example.gracemanhwa_picks.presentation.viewmodel.Man
hwaViewModel
import
```

```
com.example.gracemanhwa_picks.presentation.viewmodel.Man
hwaViewModelFactory

@Composable
fun AppNavigation(
    navController: NavHostController,
    factory: ManhwaViewModelFactory
) {
    val viewModel: ManhwaViewModel = viewModel(factory =
factory)

    NavHost(navController = navController,
startDestination = "list") {
        composable("list") {
            ManhwaListScreen(
                viewModel = viewModel,
                onNavigateToDetail = { manhwaId ->
         navController.navigate("detail/$manhwaId")
                }
            )
        }
        composable(
            route = "detail/{manhwaId}",
            arguments = listOf(navArgument("manhwaId") {
type = NavType.IntType })
        ) { backStackEntry ->
            val manhwaId =
backStackEntry.arguments?.getInt("manhwaId") ?: 0
            ManhwaDetailScreen(
                manhwaId = manhwaId,
                viewModel = viewModel,
                onNavigateBack = {
                    navController.popBackStack()
                }
            )
        }
    }
}
```

*Tabel 11. ManhwaDetailScreen.kt Modul 5*

```
package
com.example.gracemanhwa_picks.presentation.screen
```

```kotlin
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.automirrored.filled.ArrowBack
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
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.unit.dp
import coil.compose.AsyncImage
import com.example.gracemanhwa_picks.presentation.viewmodel.ManhwaViewModel

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun ManhwaDetailScreen(
    manhwaId: Int,
    viewModel: ManhwaViewModel,
    onNavigateBack: () -> Unit
) {
    LaunchedEffect(manhwaId) {
        viewModel.getManhwaById(manhwaId)
    }

    val manhwa by viewModel.selectedManhwa.collectAsState()

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(manhwa?.title ?: "Detail Manhwa") },
                navigationIcon = {
                    IconButton(onClick = onNavigateBack) {
```

```
Icon(Icons.AutoMirrored.Filled.ArrowBack,
contentDescription = "Kembali")
                        }
                    }
                )
            }
        ) { innerPadding ->
            manhwa?.let { item ->
                Column(
                    modifier = Modifier
                        .padding(innerPadding)
                        .fillMaxSize()
                  .verticalScroll(rememberScrollState())
                        .padding(16.dp)
                ) {
                    AsyncImage(
                        model = item.imageUrl,
                        contentDescription = item.title,
                        contentScale = ContentScale.Crop,
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(300.dp)

.clip(RoundedCornerShape(16.dp))
                    )
                    Spacer(modifier =
Modifier.height(16.dp))
                    Text(item.title, style =
MaterialTheme.typography.headlineMedium)
                    Text("By ${item.author}", style =
MaterialTheme.typography.titleMedium)
                    Spacer(modifier =
Modifier.height(8.dp))
                    Divider()
                    Spacer(modifier =
Modifier.height(8.dp))
                    Text(item.description, style =
MaterialTheme.typography.bodyLarge)
                }
            } ?: run {
                Box(
                    modifier = Modifier
                        .padding(innerPadding)
                        .fillMaxSize(),
                    contentAlignment = Alignment.Center
                ) {
                    CircularProgressIndicator()
            }
        }
    }
}


```

*Tabel 12. ManhwaViewModel.kt Modul 5*

```
package
com.example.gracemanhwa_picks.presentation.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import
com.example.gracemanhwa_picks.data.local.entity.ManhwaE
ntity
import
com.example.gracemanhwa_picks.data.repository.ManhwaRep
ository
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.launch

class ManhwaViewModel(private val repository:
ManhwaRepository) : ViewModel() {

    private val _manhwas =
MutableStateFlow<List<ManhwaEntity>>(emptyList())
    val manhwas: StateFlow<List<ManhwaEntity>> =
_manhwas.asStateFlow()

    private val _selectedManhwa =
MutableStateFlow<ManhwaEntity?>(null)
    val selectedManhwa: StateFlow<ManhwaEntity?> =
_selectedManhwa.asStateFlow()

    private val _isLoading = MutableStateFlow(false)
    val isLoading: StateFlow<Boolean> =
_isLoading.asStateFlow()

    init {
        loadManhwas(forceRefresh = true)
    }

    private fun loadManhwas(forceRefresh: Boolean) {
        viewModelScope.launch {
            if (forceRefresh) {
                _isLoading.value = true
                repository.refreshManhwas()
            }
            repository.getAllManhwas().collect {
manhwaList ->
                _manhwas.value = manhwaList
                _isLoading.value = false
            }
        }
    }

    fun getManhwaById(id: Int) {
        viewModelScope.launch {
            repository.getManhwaById(id).collect {
                _selectedManhwa.value = it
            }
        }
    }

    fun toggleFavorite(manhwa: ManhwaEntity) {
        viewModelScope.launch {
            repository.updateFavoriteStatus(manhwa,
!manhwa.isFavorite)
        }
    }
}
```

*Tabel 13. ManhwaViewModelFactory.kt Modul 5*

```
package
com.example.gracemanhwa_picks.presentation.viewmodel

import android.content.Context
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import com.example.gracemanhwa_picks.di.Injection

class ManhwaViewModelFactory(private val context:
Context) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass:
Class<T>): T {
        if
(modelClass.isAssignableFrom(ManhwaViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
            return
ManhwaViewModel(Injection.provideRepository(context))
as T
        }
        throw IllegalArgumentException("Unknown
ViewModel class")
    }
}
```

*Tabel 14. Theme.kt Modul 5*

```
package com.example.gracemanhwa_picks.ui.theme

import android.app.Activity
import android.os.Build
import
androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
import
androidx.compose.material3.dynamicDarkColorScheme
import
androidx.compose.material3.dynamicLightColorScheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.ui.platform.LocalContext

private val DarkColorScheme = darkColorScheme(
    primary = Purple80,
    secondary = PurpleGrey80,
    tertiary = Pink80
)

private val LightColorScheme = lightColorScheme(
    primary = Purple40,
    secondary = PurpleGrey40,
    tertiary = Pink40
)
```

```
@Composable
fun GraceManhwa_picksTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme) dynamicDarkColorScheme(context) else dynamicLightColorScheme(context)
        }

        darkTheme -> DarkColorScheme
        else -> LightColorScheme
    }

    MaterialTheme(
        colorScheme = colorScheme,
        typography = Typography,
        content = content
    )
}
```

*Tabel 15. ConnectivityObserver.kt*

```
package com.example.modul5.util

import kotlinx.coroutines.flow.Flow

interface ConnectivityObserver {
    fun observe(): Flow<Status>

    enum class Status {
        Available, Unavailable, Losing, Lost
    }
}
```

*Tabel 16. MainActivity.kt Modul 5*

```
package com.example.gracemanhwa_picks

```

```kotlin
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.ui.Modifier
import androidx.navigation.compose.rememberNavController
import com.example.gracemanhwa_picks.presentation.navigation.AppNavigation
import com.example.gracemanhwa_picks.presentation.viewmodel.ManhwaViewModelFactory
import com.example.gracemanhwa_picks.ui.theme.GraceManhwa_picksTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            GraceManhwa_picksTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    val navController = rememberNavController()
                    val factory = ManhwaViewModelFactory(this)

                    AppNavigation(navController = navController, factory = factory)
                }
            }
        }
    }
}
```

**B. Output Produk**

**C. Pembahasan**

ManhwaDao.kt berisi interface ManhwaDao yang mendefinisikan fungsi-fungsi akses data lokal menggunakan Room. Fungsi-fungsi seperti getAllManhwas(), getManhwaById(), insertAll(), updateManhwa(), dan deleteAll() digunakan untuk mengambil, menyimpan, dan memperbarui data manhwa secara reaktif melalui Flow. ManhwaEntity.kt berisi data class ManhwaEntity sebagai representasi dari entitas dalam database lokal Room. Setiap entitas menyimpan data manhwa seperti id, title, author, description, imageUrl, url, dan isFavorite. Struktur ini memudahkan penyimpanan dan pengelolaan data secara offline.

ManhwaDto.kt berisi data class ManhwaDto yang berfungsi untuk mendeskripsikan struktur data dari API eksternal. Dengan anotasi @Serializable dan @SerialName, KotlinX Serialization digunakan untuk memetakan data JSON ke dalam objek Kotlin. RetrofitInstance.kt berisi konfigurasi singleton Retrofit untuk komunikasi HTTP. Tabel ini menyiapkan JSON parser yang toleran terhadap field tak dikenal (ignoreUnknownKeys = true), logging interceptor, dan pembuatan objek Retrofit dengan ManhwaApiService.

ManhwaRepository.kt berisi layer repository yang menghubungkan data dari API dan database lokal. Repository ini mengimplementasikan caching, sinkronisasi data, dan error handling. Ia juga menyediakan fungsi refreshManhwas() dan updateFavoriteStatus() untuk sinkronisasi data dan pengelolaan status favorit. Injection.kt berisi class Injection sebagai penyedia dependensi ManhwaRepository secara manual. Fungsi provideRepository() mengambil instance database lokal dan Retrofit untuk membentuk repository.

ManhwaCard.kt sebagai komponen UI berbasis Jetpack Compose yang menampilkan kartu informasi manhwa. Tabel ini memuat elemen visual seperti gambar, judul, nama penulis, tombol detail, tombol baca, dan ikon favorit yang dapat diklik. Navigation.kt berisi konfigurasi NavHost menggunakan Jetpack Compose Navigation. Aplikasi memiliki dua layar: daftar manhwa (list) dan detail manhwa (detail/{manhwaId}), dengan pengelolaan argument dan ViewModel menggunakan factory.

ManhwaDetailScreen.kt menampilkan layar detail dari suatu manhwa tertentu. Menggunakan AsyncImage, Text, dan Scaffold, layar ini menampilkan informasi lengkap manhwa dan memungkinkan kembali ke layar sebelumnya dengan ikon kembali. ManhwaViewModel.kt berisi ViewModel yang mengatur aliran data antara repository dan UI. Menyediakan StateFlow untuk semua manhwa dan manhwa yang dipilih, serta fungsi untuk memuat data awal, mengambil manhwa berdasarkan ID, dan mengubah status favorit.

ManhwaViewModelFactory.kt menampilkan factory class yang digunakan untuk menghasilkan instance ManhwaViewModel dengan dependensi yang tepat dari Injection. Hal ini diperlukan untuk integrasi dengan sistem ViewModel Compose. Theme.kt berisi konfigurasi tema aplikasi. Digunakan untuk menerapkan skema warna terang dan gelap secara otomatis atau dinamis, sesuai pengaturan sistem dan API level perangkat.

ConnectivityObserver.kt menampilkan interface yang memungkinkan observasi status konektivitas jaringan secara reaktif. Interface ini berisi fungsi observe() dan enum Status untuk memantau kondisi seperti Available, Lost, dan lainnya. MainActivity.kt dalamnya dipanggil GraceManhwa_picksTheme, diset Surface, dan digunakan AppNavigation untuk mengelola navigasi antar layar menggunakan NavController.

## Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.

natnutnot/PrakMobile at master + https://github.com/natnutnot/Mobile