LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 5

CONNECT TO THE INTERNET

Oleh:

Farisa Adelia **NIM. 2110817120010**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
JUNI 2024

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE I
MODUL 5

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

Nama Praktikan : Farisa Adelia
NIM : 2110817120010

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar NIM. 2210817210026	Muti`a Maulida S.Kom M.T.I NIP. 19881027 201903 20 13

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

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

A. Source Code

Tabel 1. Source Code MainActivity.kt

```
package com.example.modul5connecttotheinternetcopy

import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.Card
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.core.net.toUri
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
```

```
import
com.example.modul5connecttotheinternetcopy.model.MovieItem
import
com.example.modul5connecttotheinternetcopy.ui.theme.Modul3Scrol
lableListTheme
import
com.example.modul5connecttotheinternetcopy.viewmodel.MovieViewM
odel
import
com.example.modul5connecttotheinternetcopy.viewmodel.MovieViewM
odelFactory
import
com.example.modul5connecttotheinternetcopy.viewmodel.UserViewMo
del
import
com.example.modul5connecttotheinternetcopy.viewmodel.UserViewMo
delFactory


class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Modul3ScrollableListTheme {
                Surface(modifier = Modifier.fillMaxSize(),
color = MaterialTheme.colorScheme.background) {
                    val movieViewModel: MovieViewModel =
viewModel(factory = MovieViewModelFactory())
                    val userViewModel: UserViewModel =
viewModel(factory = UserViewModelFactory(application))
                    MainScreen(movieViewModel, userViewModel)
                }
            }
        }
    }
}
@Composable
fun ProfileScreen(userViewModel: UserViewModel) {
    val user by userViewModel.user.collectAsState()
    val context = LocalContext.current

    LaunchedEffect(Unit) {
        userViewModel.fetchUser(true)
    }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
    ) {
        Text("Profile", style =
MaterialTheme.typography.titleLarge)
```

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

        if (user != null) {
            Text("Name: ${user?.name?.title}
${user?.name?.first} ${user?.name?.last}")
            Text("Email: ${user?.email}")
            Text("Phone: ${user?.phone}")
            Text("Cell: ${user?.cell}")
            Text("Address: ${user?.location?.street?.number}
${user?.location?.street?.name}, ${user?.location?.city},
${user?.location?.state}, ${user?.location?.country},
${user?.location?.postcode}")
        } else {
            Text("Loading user data or no data available
offline...")
        }

        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = {
            userViewModel.fetchUser(true)
        }) {
            Text("Refresh User")
        }
        Spacer(modifier = Modifier.height(8.dp))
        Button(onClick = {  }) {
            Text("Logout")
        }
    }
}

@Composable
fun MainScreen(
    viewModel: MovieViewModel,
    userViewModel: UserViewModel
) {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination =
"list") {
        composable("list") {
            MovieListScreen(navController, viewModel)
        }
        composable("detail") {
            val movie by
viewModel.selectedMovie.collectAsState()
            movie?.let {
                DetailScreen(it)
            }
        }
        composable("profile") {
            ProfileScreen(userViewModel)
        }
    }
```

```
}


@Composable
fun MovieListScreen(navController: NavController, viewModel:
MovieViewModel) {
    val context = LocalContext.current
    val movieList by viewModel.movies.collectAsState()

    Column(modifier = Modifier.fillMaxSize()) {
        Button(
            onClick = { navController.navigate("profile") },
            modifier = Modifier
                .padding(8.dp)
                .fillMaxWidth()
        ) {
            Text("Go to Profile")
        }

        LazyColumn(modifier =
Modifier.padding(8.dp).weight(1f)) {
            items(movieList) { movie ->
                Card(
                    shape = RoundedCornerShape(12.dp),
                    modifier = Modifier
                        .padding(vertical = 8.dp)
                        .fillMaxWidth()
                ) {
                    Column(modifier = Modifier.padding(12.dp))
{
                        Image(
                            painter = painterResource(id =
movie.imageResId),
                            contentDescription = movie.title,
                            contentScale = ContentScale.Crop,
                            modifier = Modifier
                                .fillMaxWidth()
                                .height(180.dp)
                                .clip(RoundedCornerShape(8.dp))
                        )
                        Spacer(modifier =
Modifier.height(8.dp))
                        Row(Modifier.fillMaxWidth(),
horizontalArrangement = Arrangement.SpaceBetween) {
                            Text(movie.title, fontWeight =
FontWeight.Bold)
                            Text(movie.year)
                        }
                        Spacer(modifier =
Modifier.height(4.dp))
                        Text("Plot: ${movie.plot}", maxLines =
3)
```

```
                            Spacer(modifier =
Modifier.height(8.dp))
                            Row(Modifier.fillMaxWidth(),
horizontalArrangement = Arrangement.SpaceEvenly) {
                                Button(onClick = {
                                    viewModel.onImdbClick(movie)
                                    val intent =
Intent(Intent.ACTION_VIEW, movie.imdbLink.toUri())
                                    context.startActivity(intent)
                                }) {
                                    Text("IMDB")
                                }
                                Button(onClick = {
                                    viewModel.selectMovie(movie)
                                    viewModel.onDetailClick(movie)

navController.navigate("detail")
                                }) {
                                    Text("Detail")
                                }
                            }
                        }
                    }
                }
            }
        }


@Composable
fun DetailScreen(movie: MovieItem) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
    ) {
        Image(
            painter = painterResource(id = movie.imageResId),
            contentDescription = movie.title,
            contentScale = ContentScale.Crop,
            modifier = Modifier
                .fillMaxWidth()
                .height(250.dp)
                .clip(RoundedCornerShape(12.dp))
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(movie.title, style =
MaterialTheme.typography.titleLarge)
        Text(movie.year, style =
MaterialTheme.typography.labelMedium)
        Spacer(modifier = Modifier.height(8.dp))
        Text("Plot:", fontWeight = FontWeight.Bold)
```

```
        Text(movie.plot)
    }
}
```

Tabel 2. Source Code AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET"
/>

    <application
        android:allowBackup="true"

android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Modul3ScrollableList"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:theme="@style/Theme.Modul3ScrollableList">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Tabel 3. Source Code MovieItem.kt

```
package com.example.modul4viewmodelanddebugging.model

data class MovieItem(
    val id: Int,
    val title: String,
    val year: String,
```

```
    val plot: String,
    val imageResId: Int,
    val imdbLink: String
)
```

Tabel 4. Source Code MovieViewModel.kt

```
package com.example.modul4viewmodelanddebugging.viewmodel

import android.util.Log
import androidx.lifecycle.ViewModel
import com.example.modul4viewmodelanddebugging.R
import com.example.modul4viewmodelanddebugging.model.MovieItem
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow

class MovieViewModel : ViewModel() {

    private val _movies =
MutableStateFlow<List<MovieItem>>(emptyList())
    val movies: StateFlow<List<MovieItem>> = _movies

    private val _selectedMovie =
MutableStateFlow<MovieItem?>(null)
    val selectedMovie: StateFlow<MovieItem?> = _selectedMovie

    init {
        val sample = listOf(
            MovieItem(1, "Pengabdi Setan 2: Communion", "2022",
"When the heavy storm hits...", R.drawable.pengabdi,
"https://www.imdb.com"),
            MovieItem(2, "Siksa Kubur", "2024", "Tells about
the punishment of the grave...", R.drawable.siksa,
"https://www.imdb.com"),
            MovieItem(3, "Pengepungan di Bukit Duri", "2025",
"A special school for troubled children...",
R.drawable.bukitduri, "https://www.imdb.com")
        )

        Log.d("MovieViewModel", "Data masuk ke dalam list:
${sample.size} item")
        _movies.value = sample
    }

    fun selectMovie(movie: MovieItem) {
        Log.d("MovieViewModel", "Data yang dipilih untuk
detail: $movie")
        _selectedMovie.value = movie
    }

    fun onImdbClick(movie: MovieItem) {
```

```
        Log.d("MovieViewModel", "Tombol IMDB ditekan:
${movie.title}")
    }

    fun onDetailClick(movie: MovieItem) {
        Log.d("MovieViewModel", "Tombol Detail ditekan:
${movie.title}")
    }
}
```

Tabel 5. Source Code MovieViewModelFactory.kt

```
package com.example.modul4viewmodelanddebugging.viewmodel

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

class MovieViewModelFactory : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>):
T {
        return MovieViewModel() as T
    }
}
```

Tabel 6. Source Code User.kt

```
package com.example.modul5connecttotheinternetcopy.model

import androidx.room.Embedded
import androidx.room.Entity
import androidx.room.PrimaryKey
import kotlinx.serialization.Serializable

@Serializable
@Entity(tableName = "users")
data class User(
    @PrimaryKey val email: String,
    val gender: String,
    @Embedded val name: Name,
    @Embedded val location: Location,
    val phone: String,
    val cell: String,
    @Embedded val picture: Picture
)

@Serializable
data class Name(
    val title: String,
    val first: String,
    val last: String
```

```
)

@Serializable
data class Location(
    @Embedded val street: Street,
    val city: String,
    val state: String,
    val country: String,
    val postcode: Int
)

@Serializable
data class Street(
    val number: Int,
    val name: String
)

@Serializable
data class Picture(
    val large: String,
    val medium: String,
    val thumbnail: String
)
```

Tabel 7. Source Code UserResponse.kt

```
package com.example.modul5connecttotheinternetcopy.model

import com.example.modul5connecttotheinternetcopy.model.User
import kotlinx.serialization.Serializable

@Serializable
data class UserResponse(
    val results: List<User>
)
```

Tabel 8. Source Code UserRepository.kt

```
package com.example.modul5connecttotheinternetcopy.data

import com.example.modul5connecttotheinternetcopy.model.User
import com.example.modul5connecttotheinternetcopy.ApiService
import com.example.modul5connecttotheinternetcopy.data.UserDao
import kotlinx.coroutines.flow.Flow

class UserRepository(private val apiService: ApiService,
private val userDao: UserDao) {

    fun getUser(): Flow<User?> {
        return userDao.getUser()
```

```
    }

    suspend fun refreshUser() {
        try {
            val response = apiService.getRandomUser()
            val user = response.results.firstOrNull()
            if (user != null) {
                userDao.deleteAllUsers()
                userDao.insertUser(user)
            }
        } catch (e: Exception) {
            println("Error refreshing user: ${e.message}")
            throw e
        }
    }
}
```

Tabel 9. Source Code UserViewModel.kt

```
package com.example.modul5connecttotheinternetcopy.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import
com.example.modul5connecttotheinternetcopy.data.UserRepository
import com.example.modul5connecttotheinternetcopy.model.User
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.stateIn
import kotlinx.coroutines.flow.SharingStarted
import kotlinx.coroutines.launch

class UserViewModel(private val repository: UserRepository) :
ViewModel() {
    val user: StateFlow<User?> = repository.getUser()
        .stateIn(viewModelScope,
SharingStarted.WhileSubscribed(5000), null)

    init {
        fetchUser(true)
    }

    fun fetchUser(forceRefresh: Boolean = false) {
        viewModelScope.launch {
            if (forceRefresh) {
                try {
                    repository.refreshUser()
                } catch (e: Exception) {
                    println("Error fetching user from network:
${e.message}")
                }
            }
```

```
        }
    }
}
```

Tabel 10. Source Code UserViewModelFactory.kt

```
package com.example.modul5connecttotheinternetcopy.viewmodel

import android.app.Application
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import
com.example.modul5connecttotheinternetcopy.data.AppDatabase
import
com.example.modul5connecttotheinternetcopy.RetrofitInstance
import
com.example.modul5connecttotheinternetcopy.data.UserRepository
import
com.example.modul5connecttotheinternetcopy.viewmodel.UserViewMo
del

class UserViewModelFactory(private val application:
Application) : ViewModelProvider.Factory {
    @Suppress("UNCHECKED_CAST")
    override fun <T : ViewModel> create(modelClass: Class<T>):
T {
        if
(modelClass.isAssignableFrom(UserViewModel::class.java)) {
            val database = AppDatabase.getDatabase(application)
            val userDao = database.userDao()
            val apiService = RetrofitInstance.api
            val repository = UserRepository(apiService,
userDao)
            return UserViewModel(repository) as T
        }
        throw IllegalArgumentException("Unknown ViewModel
class")
    }
}
```

Tabel 11. Source Code ApiService

```
package com.example.modul5connecttotheinternetcopy

import
com.example.modul5connecttotheinternetcopy.model.UserResponse
import retrofit2.http.GET

interface ApiService {
    @GET("api/")
```

```
    suspend fun getRandomUser(): UserResponse
}
```

Tabel 12. AppDatabase

```
package com.example.modul5connecttotheinternetcopy.data

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
import com.example.modul5connecttotheinternetcopy.model.User
import com.example.modul5connecttotheinternetcopy.data.UserDao

@Database(entities = [User::class], version = 1, exportSchema =
false)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "app_database"
                ).build()
                INSTANCE = instance
                instance
            }
        }
    }
}
```

Tabel 13. Source Code RetrofitInstance

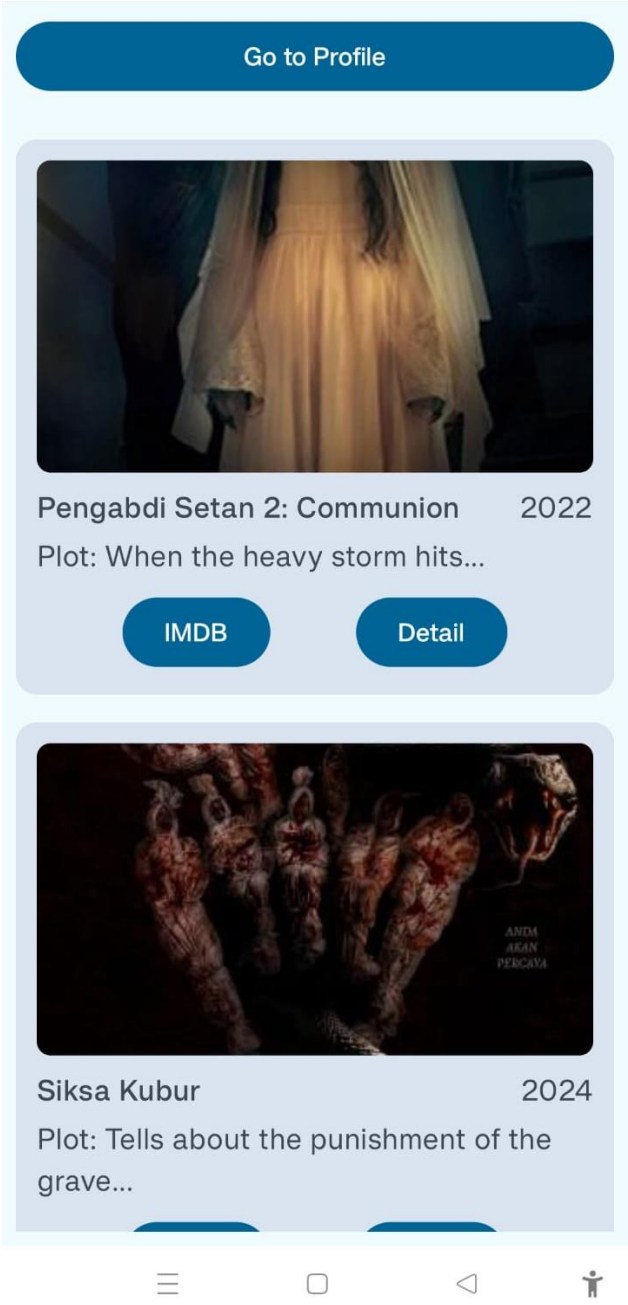
```
package com.example.modul5connecttotheinternetcopy

import
com.jakewharton.retrofit2.converter.kotlinx.serialization.asCo
nverterFactory
import kotlinx.serialization.json.Json
import retrofit2.Retrofit
import okhttp3.MediaType.Companion.toMediaType

object RetrofitInstance {
    private val json = Json { ignoreUnknownKeys = true }
```

```
    private val retrofit by lazy {
        Retrofit.Builder()
            .baseUrl("https://randomuser.me/")

.addConverterFactory(json.asConverterFactory("application/json
".toMediaType()))
            .build()
    }

    val api: ApiService by lazy {
        retrofit.create(ApiService::class.java)
    }
}
```

Tabel 14. Source Code UserDao

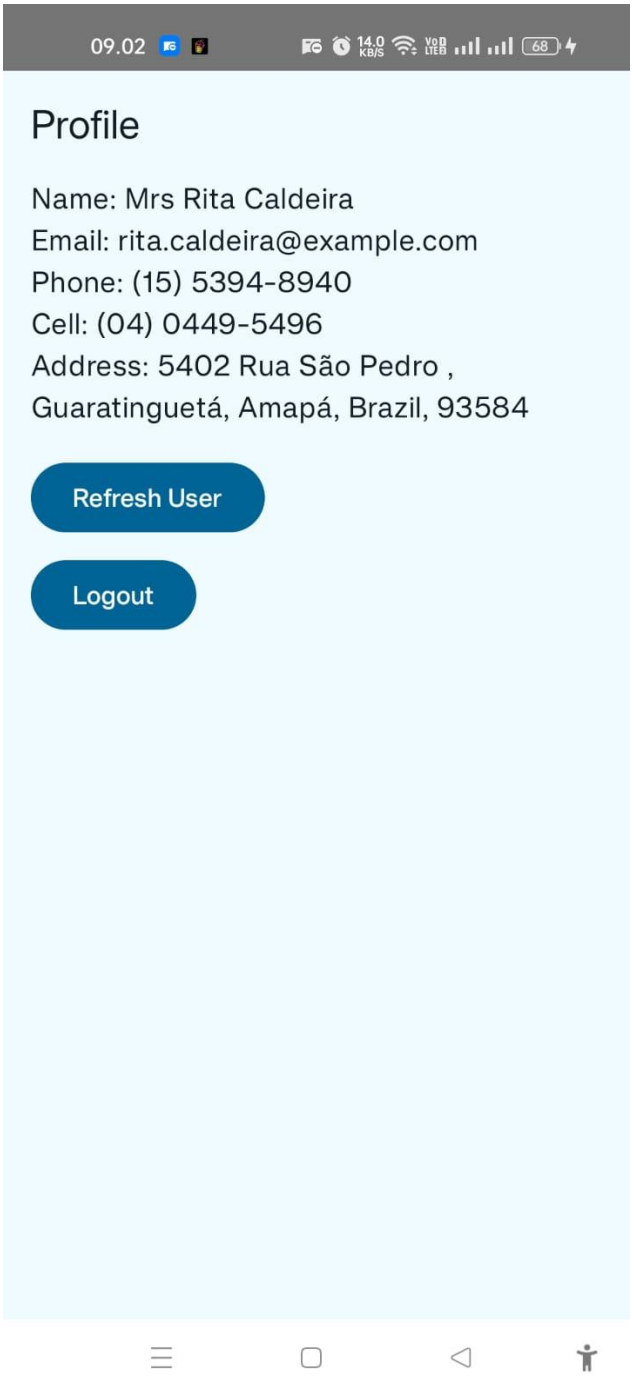
```
package com.example.modul5connecttotheinternetcopy.data

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import com.example.modul5connecttotheinternetcopy.model.User
import kotlinx.coroutines.flow.Flow

@Dao
interface UserDao {
    @Query("SELECT * FROM users LIMIT 1")
    fun getUser(): Flow<User?>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)

    @Query("DELETE FROM users")
    suspend fun deleteAllUsers()
}
```

B. Output Program



Gambar 1. Screenshot Hasil Soal 1

Gambar 2. Screenshot Hasil Soal 1



Gambar 3. Screenshot Hasil Soal 1

C. Pembahasan

- Networking dengan Retrofit dan KotlinX Serialization
  - Aplikasi menggunakan Retrofit sebagai library utama untuk mengambil data dari remote API.
  - Untuk parsing JSON, digunakan KotlinX Serialization dengan konfigurasi converter di Retrofit:

```
val retrofit = Retrofit.Builder()
    .baseUrl("https://randomuser.me/")
addConverterFactory(json.asConverterFactory("application
/json".toMediaType()))
    .build()
```

  - Struktur response dibungkus dalam generic response handler untuk menampilkan status loading, success, dan error.
  - Pengambilan data dari ViewModel dilakukan melalui Flow untuk streaming data: `val userList: StateFlow<ApiResponse<List<User>>> = _userList`
- Image Loading dengan Coil
  - Aplikasi menggunakan library Coil untuk memuat gambar profil pengguna yang diambil dari API.
  - Coil diintegrasikan langsung di Jetpack Compose:

```
Image(
    painter = rememberImagePainter(data =
user.picture.large),
    contentDescription = null,
    modifier = Modifier.size(64.dp).clip(CircleShape),
    contentScale = ContentScale.Crop
)
```

- Navigasi ke Detail
  - Navigasi menggunakan NavController menuju halaman detail dengan menyertakan ID atau parameter unik pengguna
  - Detail pengguna diambil berdasarkan parameter tersebut dari repository atau local database.
  - StateFlow digunakan dengan collectAsState() dalam Jetpack Compose agar UI otomatis merespons perubahan data.
- Data Persistence

    - Aplikasi menerapkan offline-first strategy dengan menyimpan data dari API ke Room database.
    - Jika API tidak dapat dijangkau, data tetap ditampilkan dari cache lokal.
- Caching Strategy dengan Room
    - Data yang diambil dari API akan disimpan ke dalam Room agar tetap bisa diakses saat offline.
    - Saat aplikasi diluncurkan, data akan dimuat dari Room terlebih dahulu dan kemudian disegarkan dari API jika koneksi tersedia.

LINK GITHUB : https://github.com/FarisaAdelia/Pemrograman-Mobile