LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 5

CONNECT TO THE INTERNET

Oleh:

Noviana Nur Aisyah NIM. 2310817120005

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
JUNI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE
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	: Noviana Nur Aisyah
NIM	: 2310817120005

Menyetujui,	Mengetahui,
Asisten Praktikum	Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar	Muti`a Maulida S.Kom M.T.I
NIM. 2210817210026	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 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
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. 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 dapat mempertahankan fitur-fitur yang sudah dibuat pada modul sebelumnya.

A. Source Code

1. MainActivity.kt

Tabel 1. Source Code Jawaban Soal 1

```
package com.example.movielist.presentation.ui.activity

import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.app.AppCompatDelegate
import androidx.lifecycle.Observer
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.room.Room
import
com.example.movielist.data.local.MovieAppPreferences
import
com.example.movielist.data.local.database.AppDatabase
import
com.example.movielist.data.remote.api.RetrofitClient
import
com.example.movielist.data.repository.MovieRepositoryImpl
import
com.example.movielist.databinding.ActivityMainBinding
import
com.example.movielist.domain.usecase.GetPopularMoviesUseC
ase
import
com.example.movielist.presentation.ui.adapter.MovieAdapte
r
import
com.example.movielist.presentation.viewmodel.MovieViewMod
el
import
com.example.movielist.presentation.viewmodel.ViewModelFac
tory
```

```
import com.example.movielist.utils.Result

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding
    private lateinit var movieAdapter: MovieAdapter
    private lateinit var movieAppPreferences:
MovieAppPreferences

    private val movieViewModel: MovieViewModel by
viewModels {
        val apiService = RetrofitClient.tmdbApiService
        val database = Room.databaseBuilder(
            applicationContext,
            AppDatabase::class.java,
            AppDatabase.DATABASE_NAME
        ).build()
        val movieDao = database.movieDao()

        val tmdbApiKey =
"efa2ab3869af63c9dd27712409e6737d"
        movieAppPreferences.saveApiKey(tmdbApiKey)

        val movieRepositoryImpl =
MovieRepositoryImpl(apiService, movieDao, tmdbApiKey)
        val getPopularMoviesUseCase =
GetPopularMoviesUseCase(movieRepositoryImpl)
        ViewModelFactory(getPopularMoviesUseCase)
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding =
ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        movieAppPreferences = MovieAppPreferences(this)

        setupRecyclerView()
        observeViewModel()
        setupDarkModeToggle()

        binding.btnRetry.setOnClickListener {
            movieViewModel.fetchPopularMovies()
        }
    }

    private fun setupRecyclerView() {
        movieAdapter = MovieAdapter()
        binding.rvMovies.apply {
            layoutManager =
LinearLayoutManager(this@MainActivity)
            adapter = movieAdapter
        }

        movieAdapter.onItemClick = { movie ->
            val intent = Intent(this,
DetailActivity::class.java).apply {
                putExtra(DetailActivity.EXTRA_MOVIE,
movie)
            }
            startActivity(intent)
        }
    }

    private fun observeViewModel() {
        movieViewModel.popularMovies.observe(this,
Observer { result ->
            when (result) {
                is Result.Loading -> {
                    binding.progressBar.visibility =
View.VISIBLE
                    binding.tvError.visibility =
View.GONE
                    binding.btnRetry.visibility =
View.GONE
                    binding.rvMovies.visibility =
View.GONE
                }
                is Result.Success -> {
                    binding.progressBar.visibility =
View.GONE
                    binding.tvError.visibility =
View.GONE
                    binding.btnRetry.visibility =
View.GONE
                    binding.rvMovies.visibility =
View.VISIBLE
                    movieAdapter.submitList(result.data)
                }
                is Result.Error -> {
                    binding.progressBar.visibility =
View.GONE
```

```
                    binding.rvMovies.visibility =
View.GONE
                    binding.tvError.visibility =
View.VISIBLE
                    binding.btnRetry.visibility =
View.VISIBLE
                    binding.tvError.text = "Error:
${result.exception.message}"
                    Toast.makeText(this, "Error:
${result.exception.message}", Toast.LENGTH_LONG).show()
                }
            }
        })
    }

    private fun setupDarkModeToggle() {
        binding.switchDarkMode.isChecked =
movieAppPreferences.getDarkModeState()


applyTheme(movieAppPreferences.getDarkModeState())

        binding.switchDarkMode.setOnCheckedChangeListener
{ _, isChecked ->

movieAppPreferences.saveDarkModeState(isChecked)
            applyTheme(isChecked)
        }
    }

    private fun applyTheme(isDarkMode: Boolean) {
        if (isDarkMode) {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.M
ODE_NIGHT_YES)
        } else {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.M
ODE_NIGHT_NO)
        }
    }
}
```

2. **MovieDao.kt**

Tabel 2. Source Code Jawaban Soal 1

```
package com.example.movielist.data.local.dao

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import
com.example.movielist.data.local.entities.MovieEntity

@Dao
interface MovieDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAllMovies(movies: List<MovieEntity>)

    @Query("SELECT * FROM movies ORDER BY popularity DESC")
    suspend fun getAllMovies(): List<MovieEntity>

```

```kotlin
    @Query("DELETE FROM movies")
    suspend fun clearAllMovies()
}
```

3. **AppDatabase.kt**

Tabel 3. Source Code Jawaban Soal 1

```kotlin
package com.example.movielist.data.local.database

import androidx.room.Database
import androidx.room.RoomDatabase
import com.example.movielist.data.local.dao.MovieDao
import
com.example.movielist.data.local.entities.MovieEntity

@Database(entities = [MovieEntity::class], version = 1,
exportSchema = false)
abstract class AppDatabase : RoomDatabase() {
    abstract fun movieDao(): MovieDao

    companion object {
        const val DATABASE_NAME = "tmdb_app_db"
    }
}
```

4. **MovieAppPreferences.kt**

Tabel 4. Source Code Jawaban Soal 1

```kotlin
package com.example.movielist.data.local

import android.content.Context
import android.content.SharedPreferences

```

```
class MovieAppPreferences(context: Context) {

    private val sharedPreferences: SharedPreferences =
        context.getSharedPreferences("tmdb_app_prefs",
Context.MODE_PRIVATE)

    companion object {
        private const val KEY_API_KEY = "api_key"
        private const val KEY_DARK_MODE = "dark_mode"
    }

    fun saveApiKey(apiKey: String) {
        sharedPreferences.edit().putString(KEY_API_KEY,
apiKey).apply()
    }

    fun getApiKey(): String? {
        return sharedPreferences.getString(KEY_API_KEY,
null)
    }

    fun saveDarkModeState(isDarkMode: Boolean) {

sharedPreferences.edit().putBoolean(KEY_DARK_MODE,
isDarkMode).apply()
    }

    fun getDarkModeState(): Boolean {
        return
sharedPreferences.getBoolean(KEY_DARK_MODE, false)
    }
}
```

5. RetrofitClient.kt

Tabel 5. Source Code Jawaban Soal 1

```
package com.example.movielist.data.remote.api

import
com.jakewharton.retrofit2.converter.kotlinx.serialization.asConverter
import kotlinx.serialization.json.Json
import okhttp3.MediaType.Companion.toMediaType
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import java.util.concurrent.TimeUnit

object RetrofitClient {

    private const val BASE_URL = "https://api.themoviedb.org/3/"

    private val json = Json {
        ignoreUnknownKeys = true
        prettyPrint = true
    }

    private val okHttpClient: OkHttpClient by lazy {
        val logging = HttpLoggingInterceptor()
        logging.setLevel(HttpLoggingInterceptor.Level.BODY)

        OkHttpClient.Builder()
            .addInterceptor(logging)
            .connectTimeout(30, TimeUnit.SECONDS)
```

```
                .readTimeout(30, TimeUnit.SECONDS)
                .writeTimeout(30, TimeUnit.SECONDS)
                .build()
    }

    val tmdbApiService: TmdbApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .client(okHttpClient)

.addConverterFactory(json.asConverterFactory("application/json".toMed
                .build()
                .create(TmdbApiService::class.java)
    }
}
```

6. TmdbApiService.kt

Tabel 6. Source Code Jawaban Soal 1

```
package com.example.movielist.data.remote.api

import
com.example.movielist.data.remote.models.MovieListRespons
e
import retrofit2.Response
import retrofit2.http.GET
import retrofit2.http.Query

interface TmdbApiService {

    @GET("movie/popular")
    suspend fun getPopularMovies(
        @Query("api_key") apiKey: String,
        @Query("language") language: String = "en-US",
        @Query("page") page: Int = 1
    ): Response<MovieListResponse>
}
```

7. **MovieDto.kt**

Tabel 7. Source Code Jawaban Soal 1

```
package com.example.movielist.data.remote.models

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

@Serializable
data class MovieDto(
    val adult: Boolean,
    @SerialName("backdrop_path")
    val backdropPath: String?,
    @SerialName("genre_ids")
    val genreIds: List<Int>,
    val id: Int,
    @SerialName("original_language")
    val originalLanguage: String,
    @SerialName("original_title")
    val originalTitle: String,
    val overview: String,
    val popularity: Double,
    @SerialName("poster_path")
    val posterPath: String?,
    @SerialName("release_date")
    val releaseDate: String,
    val title: String,
    val video: Boolean,
    @SerialName("vote_average")
    val voteAverage: Double,
    @SerialName("vote_count")
    val voteCount: Int
)
```

8. MovieDtoExtension.kt

Tabel 8. Source Code Jawaban Soal 1

```
package com.example.movielist.data.remote.models

import com.example.movielist.domain.model.Movie
import
com.example.movielist.data.local.entities.MovieEntity

fun MovieDto.toDomainMovie(): Movie {
    return Movie(
        id = id,
        title = title,
        overview = overview,
        posterPath = posterPath,
        releaseDate = releaseDate,
        voteAverage = voteAverage
    )
}

fun MovieDto.toMovieEntity(): MovieEntity {
    return MovieEntity(
        id = id,
        title = title,
        overview = overview,
        posterPath = posterPath,
        releaseDate = releaseDate,
        voteAverage = voteAverage,
        popularity = popularity
    )
}
```

9. MovieListResponse.kt

Tabel 9. Source Code Jawaban Soal 1

```
package com.example.movielist.data.remote.models

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

@Serializable
data class MovieListResponse(
    val page: Int,
    val results: List<MovieDto>,
    @SerialName("total_pages")
    val totalPages: Int,
    @SerialName("total_results")
    val totalResults: Int
)
```

10. MovieRepository.kt

Tabel 10. Source Code Jawaban Soal 1

```
package com.example.movielist.data.repository

import com.example.movielist.data.local.dao.MovieDao

```

```kotlin
import
com.example.movielist.data.remote.api.TmdbApiService
import
com.example.movielist.data.remote.models.toDomainMovie
import
com.example.movielist.data.remote.models.toMovieEntity
import com.example.movielist.domain.model.Movie
import com.example.movielist.utils.Result
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.flow
import retrofit2.HttpException
import java.io.IOException

interface MovieRepository {
    fun getPopularMovies(): Flow<Result<List<Movie>>>
}

class MovieRepositoryImpl(
    private val apiService: TmdbApiService,
    private val movieDao: MovieDao,
    private val apiKey: String
) : MovieRepository {

    override fun getPopularMovies():
Flow<Result<List<Movie>>> = flow {
        emit(Result.Loading)

        val cachedMovies = movieDao.getAllMovies().map {
it.toDomainMovie() }
        if (cachedMovies.isNotEmpty()) {
            emit(Result.Success(cachedMovies))
        }

        try {
            val response = apiService.getPopularMovies(apiKey = apiKey)
            if (response.isSuccessful) {
                val movieDtos = response.body()?.results ?: emptyList()
                val domainMovies = movieDtos.map { it.toDomainMovie() }

                movieDao.clearAllMovies()
                movieDao.insertAllMovies(movieDtos.map { it.toMovieEntity() })

                emit(Result.Success(domainMovies))
            } else {
                emit(Result.Error(Exception("API Error: ${response.code()} ${response.message()}")))
            }
        } catch (e: HttpException) {
            emit(Result.Error(Exception("Network Error (HTTP ${e.code()}): ${e.message()}")))
        } catch (e: IOException) {
            emit(Result.Error(Exception("No Internet Connection or API Timeout: ${e.message}")))
        } catch (e: Exception) {
            emit(Result.Error(Exception("An unexpected error occurred: ${e.localizedMessage}")))
        }
    }
```

```
}
```

11. Movie.kt

Tabel 11. Source Code Jawaban Soal 1

```kotlin
package com.example.movielist.domain.model

import android.os.Parcelable
import kotlinx.parcelize.Parcelize

@Parcelize
data class Movie(
    val id: Int,
    val title: String,
    val overview: String,
    val posterPath: String?,
    val releaseDate: String,
    val voteAverage: Double
) : Parcelable
```

12. GetPopularMoviesUseCase.kt

Tabel 12. Source Code Jawaban Soal 1

```kotlin
package com.example.movielist.domain.usecase

import com.example.movielist.domain.model.Movie
import
com.example.movielist.data.repository.MovieRepositoryImp
l
import com.example.movielist.utils.Result
import kotlinx.coroutines.flow.Flow

class GetPopularMoviesUseCase (
    private val movieRepository: MovieRepositoryImpl
) {
    operator fun invoke(): Flow<Result<List<Movie>>> {
        return movieRepository.getPopularMovies()
    }
}
```

13. DetailActivity.kt

Tabel 13. Source Code Jawaban Soal 1

```kotlin
package com.example.movielist.presentation.ui.activity

import android.os.Build
import android.os.Bundle
import android.view.MenuItem
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.bumptech.glide.Glide
import
com.example.movielist.databinding.ActivityDetailBinding
import com.example.movielist.domain.model.Movie

class DetailActivity : AppCompatActivity() {

    private lateinit var binding: ActivityDetailBinding

    companion object {
        const val EXTRA_MOVIE = "extra_movie"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityDetailBinding.inflate(layoutInflater)
        setContentView(binding.root)



supportActionBar?.setDisplayHomeAsUpEnabled(true)


        val movie = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            intent.getParcelableExtra(EXTRA_MOVIE, Movie::class.java)
        } else {
            @Suppress("DEPRECATION")
            intent.getParcelableExtra(EXTRA_MOVIE)
        }

        movie?.let {

            supportActionBar?.title = it.title

            binding.apply {
                tvDetailTitle.text = it.title
                tvDetailReleaseDate.text = "Release Date: ${it.releaseDate}"
                tvDetailVoteAverage.text = "Rating: ${String.format("%.1f", it.voteAverage)}"
                tvDetailOverview.text = it.overview


                val imageUrl =
"https://image.tmdb.org/t/p/w500${it.posterPath}"
                Glide.with(this@DetailActivity)
                    .load(imageUrl)
                    .centerCrop()
                    .into(ivDetailPoster)
            }
        } ?: run {
            Toast.makeText(this, "Film tidak
ditemukan.", Toast.LENGTH_SHORT).show()
            finish()
        }
    }


    override fun onOptionsItemSelected(item: MenuItem):
Boolean {
        if (item.itemId == android.R.id.home) {
            onBackPressedDispatcher.onBackPressed()
            return true
        }
        return super.onOptionsItemSelected(item)
    }
}
```

14. MovieAdapter.kt

Tabel 14. Source Code Jawaban Soal 1

```
package com.example.movielist.presentation.ui.adapter

import android.view.LayoutInflater
import android.view.ViewGroup
```

```kotlin
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.example.movielist.databinding.ItemMovieBinding
import com.example.movielist.domain.model.Movie

class MovieAdapter : ListAdapter<Movie,
MovieAdapter.MovieViewHolder>(MovieDiffCallback()) {

    var onItemClick: ((Movie) -> Unit)? = null

    override fun onCreateViewHolder(parent: ViewGroup,
viewType: Int): MovieViewHolder {
        val binding =
ItemMovieBinding.inflate(LayoutInflater.from(parent.conte
xt), parent, false)
        return MovieViewHolder(binding)
    }

    override fun onBindViewHolder(holder:
MovieViewHolder, position: Int) {
        val movie = getItem(position)
        holder.bind(movie)
    }

    inner class MovieViewHolder(private val binding:
ItemMovieBinding) :
        RecyclerView.ViewHolder(binding.root) {

        init {
            binding.btnDetail.setOnClickListener {
onItemClick?.invoke(getItem(adapterPosition))
            }
        }

        fun bind(movie: Movie) {
            binding.apply {
                tvMovieTitle.text = movie.title
                tvReleaseDate.text = "Release Date: ${movie.releaseDate}"
                tvVoteAverage.text = "Rating: ${String.format("%.1f", movie.voteAverage)}"
                tvOverview.text = movie.overview

                val imageUrl = "https://image.tmdb.org/t/p/w500${movie.posterPath}"

                Glide.with(itemView.context)
                    .load(imageUrl)
                    .centerCrop()
                    .into(ivPoster)
            }
        }
    }

    class MovieDiffCallback : DiffUtil.ItemCallback<Movie>() {
        override fun areItemsTheSame(oldItem: Movie, newItem: Movie): Boolean {
            return oldItem.id == newItem.id
        }

        override fun areContentsTheSame(oldItem: Movie,
newItem: Movie): Boolean {
            return oldItem == newItem
        }
    }
}
```

15. MovieViewModel.kt

Tabel 15. Source Code Jawaban Soal 1

```
package com.example.movielist.presentation.viewmodel

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.movielist.domain.model.Movie
import
com.example.movielist.domain.usecase.GetPopularMoviesUseCase
import com.example.movielist.utils.Result
import kotlinx.coroutines.launch

class MovieViewModel (
    private val getPopularMoviesUseCase:
GetPopularMoviesUseCase
) : ViewModel() {

    private val _popularMovies =
MutableLiveData<Result<List<Movie>>>()

```

```
    val popularMovies: LiveData<Result<List<Movie>>> =
_popularMovies

    init {
        fetchPopularMovies()
    }

    fun fetchPopularMovies() {
        viewModelScope.launch {
            getPopularMoviesUseCase().collect { result ->
                _popularMovies.value = result
            }
        }
    }
}
```

16. ViewModelFactory.kt

Tabel 16. Source Code Jawaban Soal 1

```
package com.example.movielist.presentation.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import
com.example.movielist.domain.usecase.GetPopularMoviesUseCase

class ViewModelFactory(
    private val getPopularMoviesUseCase:
GetPopularMoviesUseCase
) : ViewModelProvider.Factory {


```

```kotlin
    override fun <T : ViewModel> create(modelClass:
Class<T>): T {
        if
(modelClass.isAssignableFrom(MovieViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
            return MovieViewModel(getPopularMoviesUseCase)
as T
        }
        throw IllegalArgumentException("Unknown ViewModel
class")
    }
}
```

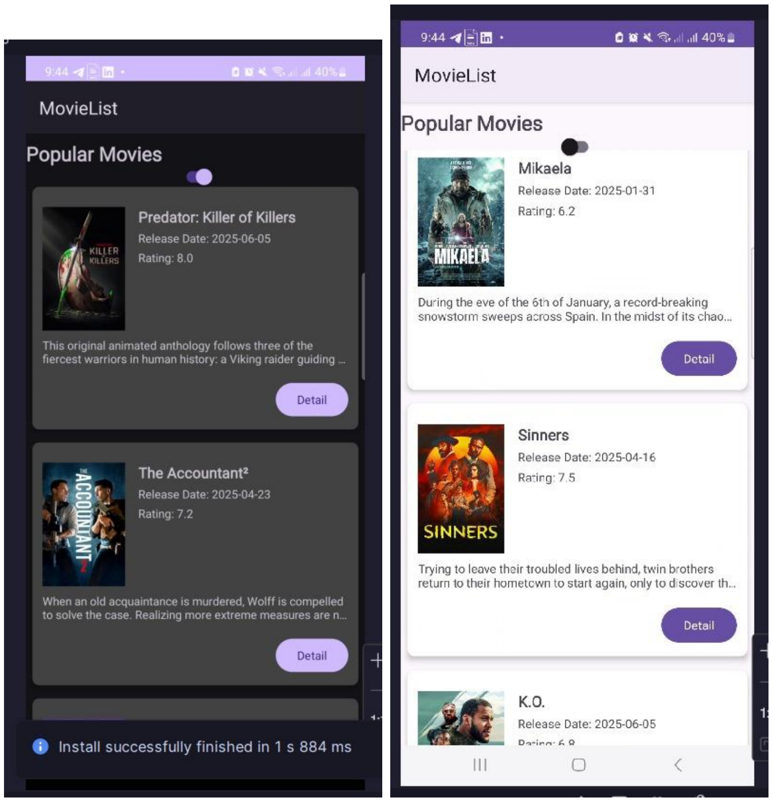
17. Result.kt

Tabel 17. Source Code Jawaban Soal 1

```kotlin
package com.example.movielist.utils

sealed class Result<out T> {
    object Loading : Result<Nothing>()
    data class Success<out T>(val data: T) : Result<T>()
    data class Error(val exception: Exception) :
Result<Nothing>()
}
```

B. Output Program


MovieList

Popular Movies

Predator: Killer of Killers
Release Date: 2025-06-05
Rating: 8.0
This original animated anthology follows three of the fiercest warriors in human history: a Viking raider guiding ...
Detail

The Accountant²
Release Date: 2025-04-23
Rating: 7.2
When an old acquaintance is murdered, Wolff is compelled to solve the case. Realizing more extreme measures are n...
Detail

Install successfully finished in 1 s 884 ms

MovieList

Popular Movies

Mikaela
Release Date: 2025-01-31
Rating: 6.2
During the eve of the 6th of January, a record-breaking snowstorm sweeps across Spain. In the midst of its chao...
Detail

Sinners
Release Date: 2025-04-16
Rating: 7.5
Trying to leave their troubled lives behind, twin brothers return to their hometown to start again, only to discover th...
Detail

K.O.
Release Date: 2025-06-05

Gambar 1. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt

- Inisialisasi ViewModel:
  Menggunakan viewModels dan ViewModelFactory untuk membuat MovieViewModel. MovieRepositoryImpl menggabungkan API dan Room database. API key disimpan via MovieAppPreferences.
- onCreate():
  Inflate layout dengan ViewBinding (ActivityMainBinding). Inisialisasi MovieAppPreferences. Panggil fungsi untuk setup RecyclerView, ViewModel observer, dan Dark Mode. Tombol "Retry" untuk memuat ulang data jika gagal.
- setupRecyclerView():
  Inisialisasi MovieAdapter dan pasang ke RecyclerView. Saat item diklik, buka DetailActivity dengan data film.
- observeViewModel():

Mengamati LiveData dari ViewModel (popularMovies). Tampilkan loading, data, atau error berdasarkan status (Result.Loading, Result.Success, Result.Error).

- setupDarkModeToggle():
  Toggle switch untuk mengaktifkan/menonaktifkan Dark Mode. Status disimpan di MovieAppPreferences.
- applyTheme():
  Menerapkan tema sesuai status Dark Mode (YES/NO).

2. MovieDao.kt

a) insertAllMovies(movies)
Menyimpan daftar film ke database.
Jika ada konflik (misalnya ID sama), data lama akan diganti (REPLACE).

b) getAllMovies()
Mengambil semua film dari tabel movies.
Diurutkan berdasarkan popularity secara menurun.

c) clearAllMovies()
Menghapus seluruh data film dari tabel movies.

3. AppDatabase.kt

a) @Database(...)
Memberi tahu Room bahwa ini adalah database yang menyimpan data MovieEntity (yaitu data film).
version = 1: versi database-nya.
exportSchema = false: kita tidak perlu ekspor skema database ke file.

b) abstract fun movieDao(): MovieDao
Fungsi ini memberi akses ke DAO (Data Access Object) bernama MovieDao, yang isinya perintah buat masukin, ngambil, dan hapus data film.

c) companion object
Menyimpan nama database, yaitu "tmdb_app_db".

4. MovieAppPreferences.kt

a) saveApiKey(apiKey: String)
   Menyimpan API key ke penyimpanan lokal (biar nggak perlu diketik ulang setiap kali buka aplikasi).

b) getApiKey()
   Mengambil kembali API key yang sudah disimpan.

c) saveDarkModeState(isDarkMode: Boolean)
   Menyimpan status Dark Mode (apakah aktif atau tidak).

5. RetrofitClient.kt

a) BASE_URL
   Alamat dasar dari API TMDB:
   "https://api.themoviedb.org/3/".

b) Json
   Mengatur supaya parsing JSON dari API bisa mengabaikan data yang tidak dikenal dan tampil rapi saat di-log.

c) OkHttpClient
   Dipakai oleh Retrofit untuk kirim-permintaan ke server.
   Ada interceptor log untuk lihat request/response di Logcat.
   Ada timeout 30 detik biar nggak nunggu selamanya kalau server lambat.

d) tmdbApiService
   Objek yang dibuat Retrofit untuk mengakses fungsi-fungsi di TmdbApiService.
   Sudah siap dipakai di mana saja dalam aplikasi.

6. TmdbApiService.kt

a) @GET("movie/popular")
   Menandakan bahwa ini adalah request GET ke endpoint movie/popular.
   getPopularMovies(...)
   Fungsi yang akan dipanggil untuk mengambil data film populer.

b) Parameter:

apiKey: kunci API TMDB (wajib).

language: bahasa hasil (default: "en-US").

c) page: halaman data (default: 1).

Response<MovieListResponse>

Data dari server akan dikembalikan dalam bentuk objek MovieListResponse, dibungkus dalam Response Retrofit.

7. MovieDto.kt

MovieDto adalah model data yang mencerminkan satu item film yang diterima dari API TMDB (dalam format JSON).

8. MovieDtoExtension.kt

File ini berisi fungsi ekstensi untuk mengubah MovieDto (data dari API) menjadi dua bentuk lain, yaitu Movie → untuk ditampilkan di UI dan MovieEntity → untuk disimpan di database lokal.

9. MovieListResponse.kt

a) page: halaman saat ini dari data yang ditampilkan.
b) results: daftar film dalam bentuk List<MovieDto>.
c) totalPages: jumlah total halaman yang tersedia.
d) totalResults: total seluruh film yang tersedia.

10. MovieRepository.kt

File ini mengatur pengambilan data film populer, baik dari API maupun database lokal, lalu mengirimkannya ke ViewModel dalam bentuk Flow<Result<List<Movie>>>.

11. Movie.kt

Movie adalah model data di lapisan domain, yaitu bentuk film yang digunakan di dalam logika aplikasi dan UI.

12. GetPopularMovieUseCase.kt

Merupakan komponen di arsitektur bersih (clean architecture) yang mengatur satu tugas spesifik, yaitu mengambil daftar film populer dari repository.

13. DetailActivitiy.kt

DetailActivity adalah halaman detail film yang muncul setelah pengguna klik salah satu item film di daftar utama.

14. MovieAdaptor.kt

MovieAdapter adalah adapter untuk RecyclerView yang menampilkan daftar film dalam bentuk item (card) satu per satu.

15. MovieViewModel.kt

MovieViewModel adalah penghubung antara UI (tampilan) dan data film. Ia bertugas mengambil data dari use case dan memberikannya ke UI.

16. ViewModelFactory.kt

ViewModelFactory adalah pembuat khusus untuk MovieViewModel. Karena MovieViewModel butuh parameter (use case), kita nggak bisa pakai cara biasa (ViewModelProvider(this)[...]). Nah, di sinilah ViewModelFactory membantu.

17. Result.kt

Result adalah cara aplikasi ini mengemas hasil dari proses yang bisa berhasil atau gagal (misalnya saat ambil data dari internet).

Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/Noviana21/Pemrograman-Mobile