LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 5

Connect to the Internet

Oleh:

Muhammad Ibnu Sina NIM. 2310817210009

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 : Muhammad Ibnu Sina
NIM : 2310817210009

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

LEMBAR PENGESAHAN 2
DAFTAR ISI 3
DAFTAR GAMBAR 4
DAFTAR TABEL 5
SOAL 1 6
A. Source Code 7
B. Output Program 26
C. Pembahasan 28
D. Tautan Git 33

DAFTAR GAMBAR

Gambar 2 Hasil Tampilan UI List Soal 1 26
Gambar 3 Hasil Tampilan UI Detail Soal 1 27

DAFTAR TABEL

Tabel 1 Source Code Jawaban Soal 1 7
Tabel 2 Source Code Jawaban Soal 1 7
Tabel 3 Source Code Jawaban Soal 1 9
Tabel 4 Source Code Jawaban Soal 1 10
Tabel 5 Source Code Jawaban Soal 1 12
Tabel 6 Source Code Jawaban Soal 1 13
Tabel 7 Source Code Jawaban Soal 1 15
Tabel 8 Source Code Jawaban Soal 1 16
Tabel 9 Source Code Jawaban Soal 1 16
Tabel 10 Source Code Jawaban Soal 1 19
Tabel 11 Source Code Jawaban Soal 1 20
Tabel 12 Source Code Jawaban Soal 1 21
Tabel 13 Source Code Jawaban Soal 1 22
Tabel 14 Source Code Jawaban Soal 1 22
Tabel 15 Source Code Jawaban Soal 1 23
Tabel 16 Source Code Jawaban Soal 1 24
Tabel 17 Source Code Jawaban Soal 1 25

SOAL 1

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

1. MainActivity.kt

Tabel 1 Source Code Jawaban Soal 1

```
1   package com.example.modul5
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5
6   class MainActivity : AppCompatActivity() {
7       override fun onCreate(savedInstanceState: Bundle?) {
8           super.onCreate(savedInstanceState)
9           setContentView(R.layout.activity_main)
10      }
11  }
```

2. DetailFragment.kt

Tabel 2 Source Code Jawaban Soal 1

```
1   package com.example.modul5.ui
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.view.LayoutInflater
7   import android.view.View
8   import android.view.ViewGroup
9   import android.widget.Toast
10  import androidx.core.view.isVisible
11  import androidx.fragment.app.Fragment
12  import androidx.fragment.app.activityViewModels
13  import androidx.lifecycle.lifecycleScope
14  import coil.load
15  import com.example.modul5.data.MovieDetails
16  import com.example.modul5.databinding.FragmentDetailBinding
17  import kotlinx.coroutines.flow.collectLatest
18  import kotlinx.coroutines.launch
19
20  class DetailFragment : Fragment() {
21
22      private var _binding: FragmentDetailBinding? = null
23      private val binding get() = _binding!!
24
25      private val viewModel: MovieViewModel by activityViewModels {
26          MovieViewModelFactory(requireActivity().application)
27      }
28
29      override fun onCreateView(
30          inflater: LayoutInflater, container: ViewGroup?,
31          savedInstanceState: Bundle?
```

```
    ): View {
        _binding = FragmentDetailBinding.inflate(inflater, container,
false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?)
{
        super.onViewCreated(view, savedInstanceState)

        viewLifecycleOwner.lifecycleScope.launch {
            viewModel.movieDetails.collectLatest { details ->
                // Tampilkan atau sembunyikan loading
                binding.progressBar.isVisible = details == null
                binding.contentGroup.isVisible = details != null

                details?.let {
                    bindMovieDetails(it)
                }
            }
        }
    }

    private fun bindMovieDetails(details: MovieDetails) {
        with(binding) {

moviePoster.load("https://image.tmdb.org/t/p/w500${details.posterPath}")
{
                crossfade(true)
            }
            movieTitle.text = details.title
            movieOverview.text = details.overview

            // Info tambahan
            releaseYearText.text = "Tahun Rilis:
${details.releaseDate?.substring(0, 4) ?: "N/A"}"

            val director = details.credits.crew.find { it.job ==
"Director" }
            directorText.text = "Sutradara: ${director?.name ?: "N/A"}"

            val actors = details.credits.cast.take(3).joinToString(", ")
{ it.name }
            actorsText.text = "Aktor: $actors"

            // Logika Tombol Trailer
            val trailer = details.videos.results.find { it.site ==
"YouTube" && it.type == "Trailer" }
            if (trailer != null) {
                playTrailerButton.visibility = View.VISIBLE
                playTrailerButton.setOnClickListener {
                    val intent = Intent(Intent.ACTION_VIEW,
```

```
76  Uri.parse("https://www.youtube.com/watch?v=${trailer.key}"))
77                      startActivity(intent)
78                  }
79              } else {
80                  playTrailerButton.visibility = View.GONE
81              }
82          }
83      }
84
85      override fun onDestroyView() {
86          super.onDestroyView()
87          _binding = null
88      }
89  }
```

3. ListFragment.kt

Tabel 3 Source Code Jawaban Soal 1

```
1   package com.example.modul5.ui
2
3   import android.os.Bundle
4   import android.view.LayoutInflater
5   import android.view.View
6   import android.view.ViewGroup
7   import androidx.fragment.app.Fragment
8   import androidx.fragment.app.activityViewModels
9   import androidx.lifecycle.Observer // Import Observer secara eksplisit
10  import androidx.navigation.fragment.findNavController
11  import androidx.recyclerview.widget.GridLayoutManager
12  import com.example.modul5.R
13  import com.example.modul5.data.Movie
14  import com.example.modul5.databinding.FragmentListBinding
15
16  class ListFragment : Fragment() {
17
18      private var _binding: FragmentListBinding? = null
19      private val binding get() = _binding!!
20
21      private val viewModel: MovieViewModel by activityViewModels {
22          MovieViewModelFactory(requireActivity().application)
23      }
24
25      override fun onCreateView(
26          inflater: LayoutInflater, container: ViewGroup?,
27          savedInstanceState: Bundle?
28      ): View {
          _binding = FragmentListBinding.inflate(inflater, container,
false)
          return binding.root
```

```
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?)
{
        super.onViewCreated(view, savedInstanceState)

        // Tentukan tipe lambda secara eksplisit
        val adapter = MovieAdapter { movie: Movie ->
            viewModel.selectMovie(movie)

findNavController().navigate(R.id.action_listFragment_to_detailFragment)
        }

        binding.recyclerView.layoutManager =
GridLayoutManager(requireContext(), 2)
        binding.recyclerView.adapter = adapter

        // Tentukan tipe data yang di-observe
        viewModel.movies.observe(viewLifecycleOwner,
Observer<List<Movie>> { movies ->
            // Gunakan submitList untuk ListAdapter
            adapter.submitList(movies)
        })
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}

```

4. MovieAdapter.kt

Tabel 4 Source Code Jawaban Soal 1

```
package com.example.modul5.ui

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import coil.load
import com.example.modul5.data.Movie
import com.example.modul5.databinding.ItemMovieBinding

class MovieAdapter(private val onClick: (Movie) -> Unit) :
    ListAdapter<Movie, MovieAdapter.MovieViewHolder>(MovieDiffCallback)
{

```

```
15      // Inner class ViewHolder
16      class MovieViewHolder(private val binding: ItemMovieBinding) :
17  RecyclerView.ViewHolder(binding.root) {
18          fun bind(movie: Movie, onClick: (Movie) -> Unit) {
19              binding.movieTitle.text = movie.title

20  binding.moviePoster.load("https://image.tmdb.org/t/p/w500${movie.poster
21  Path}") {
22                  crossfade(true)
23                  placeholder(android.R.drawable.ic_menu_gallery)
24                  error(android.R.drawable.ic_menu_close_clear_cancel)
25              }
26              // Gunakan binding.root untuk setOnClickListener
27              binding.root.setOnClickListener {
28                  onClick(movie)
29              }
30          }
31      }
32
33      override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
34  MovieViewHolder {
35          val binding =
36  ItemMovieBinding.inflate(LayoutInflater.from(parent.context), parent,
37  false)
38          return MovieViewHolder(binding)
39      }
40
        override fun onBindViewHolder(holder: MovieViewHolder, position:
41  Int) {
42          val movie = getItem(position)
43          holder.bind(movie, onClick)
44      }
    }
45
    // Object untuk DiffUtil di luar kelas Adapter
46  object MovieDiffCallback : DiffUtil.ItemCallback<Movie>() {
47      override fun areItemsTheSame(oldItem: Movie, newItem: Movie):
48  Boolean {
49          return oldItem.id == newItem.id
50      }
51
        override fun areContentsTheSame(oldItem: Movie, newItem: Movie):
    Boolean {
            return oldItem == newItem
        }
    }
```

5. Movie.kt

Tabel 5 Source Code Jawaban Soal 1

```
1   @file:OptIn(InternalSerializationApi::class)
2
3   package com.example.modul5.data
4
5   import androidx.room.Entity
6   import androidx.room.PrimaryKey
7   import kotlinx.serialization.InternalSerializationApi
8   import kotlinx.serialization.SerialName
9   import kotlinx.serialization.Serializable
10  import kotlin.OptIn
11
12  // Kelas untuk daftar film (tetap sama)
13  @Serializable
14  @Entity(tableName = "movies")
15  data class Movie(
16      @PrimaryKey
17      val id: Int,
18      val title: String,
19      @SerialName("poster_path")
20      val posterPath: String?,
21      @SerialName("overview")
22      val overview: String
23  )
24
25  // Wrapper untuk daftar film populer
26  @Serializable
27  data class MovieResponse(
28      val results: List<Movie>
29  )
30
31  //--- KELAS-KELAS BARU UNTUK DETAIL FILM ---
32
33  // Kelas utama untuk detail film
34  @Serializable
35  data class MovieDetails(
36      val id: Int,
37      val title: String,
38      val overview: String,
39      @SerialName("poster_path")
40      val posterPath: String?,
41      @SerialName("release_date")
42      val releaseDate: String?,
43      val credits: Credits,
44      val videos: VideoResponse
45  )
46
47  // Kelas untuk kredit (aktor dan kru)
48  @Serializable
49  data class Credits(
50      val cast: List<CastMember>,
```

```
    val crew: List<CrewMember>
)

// Kelas untuk anggota cast (aktor)
@Serializable
data class CastMember(
    val name: String,
    val character: String
)

// Kelas untuk anggota kru (sutradara, dll)
@Serializable
data class CrewMember(
    val name: String,
    val job: String
)

// Wrapper untuk daftar video/trailer
@Serializable
data class VideoResponse(
    val results: List<Video>
)

// Kelas untuk satu video/trailer
@Serializable
data class Video(
    val key: String,
    val site: String,
    val type: String
)
```

6. MovieView Model.kt

Tabel 6 Source Code Jawaban Soal 1

```
package com.example.modul5.ui

import android.app.Application
import androidx.lifecycle.LiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.asLiveData
import androidx.lifecycle.viewModelScope
import com.example.modul5.data.AppDatabase
import com.example.modul5.data.Movie
import com.example.modul5.data.MovieDetails
import com.example.modul5.data.MovieRepository
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch
```

```
16
17  class MovieViewModel(application: Application) :
18  ViewModel() {
19
20      private val repository: MovieRepository
21      val movies: LiveData<List<Movie>>
22
23      // StateFlow untuk film yang dipilih dari daftar
24      private val _selectedMovie =
    MutableStateFlow<Movie?>(null)
25      val selectedMovie: StateFlow<Movie?> = _selectedMovie
26      // --- STATEFLOW BARU UNTUK MENAMPUNG DETAIL LENGKAP -
27  --
28      private val _movieDetails =
    MutableStateFlow<MovieDetails?>(null)
29      val movieDetails: StateFlow<MovieDetails?> =
30  _movieDetails
31
32      init {
            val movieDao =
    AppDatabase.getDatabase(application).movieDao()
33          repository = MovieRepository(movieDao)
34          movies = repository.movies.asLiveData()
35          refreshDataFromRepository()
        }
36
37
38      private fun refreshDataFromRepository() {
39          viewModelScope.launch {
40              repository.refreshMovies()
41          }
42      }
43
44      fun selectMovie(movie: Movie) {
45          _selectedMovie.value = movie
46          // Saat film dipilih, langsung ambil detail
    lengkapnya
47          fetchMovieDetails(movie.id)
48      }
49
50      // --- FUNGSI BARU UNTUK MENGAMBIL DETAIL ---
51      private fun fetchMovieDetails(movieId: Int) {
52          viewModelScope.launch {
53              // Set null dulu agar UI menampilkan loading
    (jika ada)
54              _movieDetails.value = null
55              // Panggil repository untuk mendapatkan detail
56
```

```
57              val details =
58  repository.getMovieDetails(movieId)
                _movieDetails.value = details
            }
        }
    }
```

7. MovieView ModelFactory.kt

Tabel 7 Source Code Jawaban Soal 1

```
1   package com.example.modul5.ui
2
3   import android.app.Application
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.ViewModelProvider
6
7   class MovieViewModelFactory(private val application:
8   Application) : ViewModelProvider.Factory {
9       override fun <T : ViewModel> create(modelClass:
10  Class<T>): T {
            if
11  (modelClass.isAssignableFrom(MovieViewModel::class.java))
    {
12              @Suppress("UNCHECKED_CAST")
13              return MovieViewModel(application) as T
            }
14          throw IllegalArgumentException("Unknown ViewModel
15  class")
        }
    }
```

8. activity_main.xml

Tabel 8 Source Code Jawaban Soal 1

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <FrameLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:background="@color/background">
7
8      <androidx.fragment.app.FragmentContainerView
9          android:id="@+id/nav_host_fragment"
10
11 android:name="androidx.navigation.fragment.NavHostFragment"
           android:layout_width="match_parent"
12         android:layout_height="match_parent"
13         app:defaultNavHost="true"
14         app:navGraph="@navigation/nav_graph" />
15 </FrameLayout>
```

9. Fragment_detail.xml

Tabel 9 Source Code Jawaban Soal 1

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ScrollView
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:background="#F5F5F5"
9      tools:context=".ui.DetailFragment">
10
11     <androidx.constraintlayout.widget.ConstraintLayout
12         android:layout_width="match_parent"
13         android:layout_height="wrap_content"
14         android:padding="16dp">
15
16         <!-- Progress Bar untuk loading -->
17         <ProgressBar
18             android:id="@+id/progress_bar"
19             android:layout_width="wrap_content"
20             android:layout_height="wrap_content"
21             app:layout_constraintTop_toTopOf="parent"
22             app:layout_constraintBottom_toBottomOf="parent"
23             app:layout_constraintStart_toStartOf="parent"
```

```
24              app:layout_constraintEnd_toEndOf="parent"
25              android:visibility="visible" />
26
27          <!-- Grup konten yang akan ditampilkan setelah loading
    selesai -->
28          <androidx.constraintlayout.widget.Group
29              android:id="@+id/content_group"
30              android:layout_width="wrap_content"
31              android:layout_height="wrap_content"
32              android:visibility="gone"
33
    app:constraint_referenced_ids="movie_poster,movie_title,play_tra
    iler_button,label_overview,movie_overview,label_info,info_contai
    ner"
34              tools:visibility="visible"/>
35
36          <com.google.android.material.card.MaterialCardView
37              android:id="@+id/movie_poster_card"
38              android:layout_width="150dp"
39              android:layout_height="225dp"
40              app:cardCornerRadius="12dp"
41              app:cardElevation="8dp"
42              app:layout_constraintStart_toStartOf="parent"
43              app:layout_constraintTop_toTopOf="parent">
44
45              <ImageView
46                  android:id="@+id/movie_poster"
47                  android:layout_width="match_parent"
48                  android:layout_height="match_parent"
49                  android:scaleType="centerCrop"
50                  tools:src="@tools:sample/avatars" />
51
52          </com.google.android.material.card.MaterialCardView>
53
54          <TextView
55              android:id="@+id/movie_title"
56              android:layout_width="0dp"
57              android:layout_height="wrap_content"
58              android:layout_marginStart="16dp"
59
    android:textAppearance="?attr/textAppearanceHeadline6"
60              android:textColor="@android:color/black"
61              android:textStyle="bold"
62              app:layout_constraintEnd_toEndOf="parent"
63
    app:layout_constraintStart_toEndOf="@id/movie_poster_card"
64
    app:layout_constraintTop_toTopOf="@id/movie_poster_card"
65              tools:text="Judul Film yang Sangat Panjang" />
66
67          <Button
68              android:id="@+id/play_trailer_button"
```

```
69              style="@style/Widget.MaterialComponents.Button.Icon"
70              android:layout_width="160dp"
71              android:layout_height="100dp"
72              android:text="Tonton Trailer"
73              app:icon="@drawable/ic_play_arrow"
74
    app:layout_constraintBottom_toBottomOf="@id/movie_poster_card"
75
    app:layout_constraintStart_toStartOf="@id/movie_title" />
76
77          <TextView
78              android:id="@+id/label_info"
79              android:layout_width="wrap_content"
80              android:layout_height="wrap_content"
81              android:text="Informasi"
82              android:layout_marginTop="24dp"
83
    android:textAppearance="?attr/textAppearanceTitleMedium"
84              android:textSize="20sp"
85              android:textStyle="bold"
86              android:textColor="@android:color/black"
87              app:layout_constraintStart_toStartOf="parent"
88
    app:layout_constraintTop_toBottomOf="@id/movie_poster_card"/>
89
90          <LinearLayout
91              android:id="@+id/info_container"
92              android:layout_width="0dp"
93              android:layout_height="wrap_content"
94              android:orientation="vertical"
95              android:layout_marginTop="8dp"
96              android:padding="12dp"
97              android:background="@drawable/rounded_background"
98              app:layout_constraintTop_toBottomOf="@id/label_info"
99              app:layout_constraintStart_toStartOf="parent"
100             app:layout_constraintEnd_toEndOf="parent">
101
102             <TextView android:id="@+id/release_year_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" tools:text="Tahun Rilis:
    2025" android:textColor="@android:color/black"/>
103             <TextView android:id="@+id/director_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" tools:text="Sutradara: John
    Doe" android:layout_marginTop="4dp"
    android:textColor="@android:color/black"/>
104             <TextView android:id="@+id/actors_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" tools:text="Aktor: A, B, C"
    android:layout_marginTop="4dp"
    android:textColor="@android:color/black"/>
105
```

```
106         </LinearLayout>
107
108         <TextView
109             android:id="@+id/label_overview"
110             android:layout_width="wrap_content"
111             android:layout_height="wrap_content"
112             android:text="Sinopsis"
113             android:layout_marginTop="16dp"
114
    android:textAppearance="?attr/textAppearanceTitleMedium"
115             android:textSize="20sp"
116             android:textStyle="bold"
117             android:textColor="@android:color/black"
118             app:layout_constraintStart_toStartOf="parent"
119
    app:layout_constraintTop_toBottomOf="@id/info_container"/>
120
121         <TextView
122             android:id="@+id/movie_overview"
123             android:layout_width="0dp"
124             android:layout_height="wrap_content"
125             android:layout_marginTop="8dp"
126             android:textAppearance="?attr/textAppearanceBody2"
127             android:textColor="@android:color/black"
128             app:layout_constraintEnd_toEndOf="parent"
129             app:layout_constraintStart_toStartOf="parent"
130
    app:layout_constraintTop_toBottomOf="@id/label_overview"
131             tools:text="Ini adalah deskripsi film yang sangat
    panjang..." />
132
        </androidx.constraintlayout.widget.ConstraintLayout>
133 </ScrollView>
```

10. Fragment_list.xml

Tabel 10 Source Code Jawaban Soal 1

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3       xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="@color/background">
8
9       <androidx.recyclerview.widget.RecyclerView
10          android:id="@+id/recyclerView"
11          android:layout_width="match_parent"
12          android:layout_height="match_parent"
```

```
13          android:clipToPadding="false"
14          android:padding="8dp"

15  app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"

16  android:contentDescription="@string/recycler_content_description"
17          app:layout_constraintBottom_toBottomOf="parent"
18          app:layout_constraintEnd_toEndOf="parent"
19          app:layout_constraintStart_toStartOf="parent"
20          app:layout_constraintTop_toTopOf="parent" />
21
22      <TextView
23          android:id="@+id/emptyView"
24          android:layout_width="match_parent"
25          android:layout_height="500dp"
26          android:gravity="center"
27          android:text="@string/empty_list_message"
28          android:visibility="gone"
29          app:layout_constraintBottom_toBottomOf="parent"
30          app:layout_constraintEnd_toEndOf="parent"
31          app:layout_constraintStart_toStartOf="parent"
32          app:layout_constraintTop_toTopOf="parent" />
33  </androidx.constraintlayout.widget.ConstraintLayout>
```

11. item_movie.xml

Tabel 11 Source Code Jawaban Soal 1

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <com.google.android.material.card.MaterialCardView
3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="wrap_content"
8       android:layout_margin="8dp"
9       app:cardCornerRadius="12dp"
10      app:cardElevation="4dp">
11
12      <LinearLayout
13          android:layout_width="match_parent"
14          android:layout_height="wrap_content"
15          android:orientation="vertical">
16
17          <ImageView
18              android:id="@+id/movie_poster"
19              android:layout_width="match_parent"
20              android:layout_height="250dp"
```

```
            android:scaleType="centerCrop"
21
    android:contentDescription="@string/recycler_content_description"
22          tools:srcCompat="@tools:sample/backgrounds/scenic" />
23
24      <TextView
25          android:id="@+id/movie_title"
26          android:layout_width="match_parent"
27          android:layout_height="wrap_content"
28          android:padding="12dp"
29
    android:textAppearance="?attr/textAppearanceTitleMedium"
30          android:gravity="center"
31          android:maxLines="2"
32          android:ellipsize="end"
33          android:textStyle="bold"
34          android:textColor="@color/white"
35          android:textSize="20sp"
36          tools:text="Judul Film yang Sangat Panjang Sekali
    Contohnya" />
37
38      </LinearLayout>
39
40  </com.google.android.material.card.MaterialCardView>
```

12. nav_graph.xml

Tabel 12 Source Code Jawaban Soal 1

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <navigation
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:id="@+id/nav_graph"
6       app:startDestination="@id/listFragment">
7
8       <fragment
9           android:id="@+id/listFragment"
10          android:name="com.example.modul5.ui.ListFragment"
11          android:label="Popular Movies"
12          tools:layout="@layout/fragment_list" >
13      <action
14          android:id="@+id/action_listFragment_to_detailFragment"
15          app:destination="@id/detailFragment" />
16      </fragment>
17
18      <fragment
19          android:id="@+id/detailFragment"
20          android:name="com.example.modul5.ui.DetailFragment"
```

```
21          android:label="Movie Detail"
22          tools:layout="@layout/fragment_detail" />
23
24  </navigation>
```

13. AppDatabase.kt

Tabel 13 Source Code Jawaban Soal 1

```
1   package com.example.modul5.data
2
    import android.content.Context
3   import androidx.room.Database
4   import androidx.room.Room
5   import androidx.room.RoomDatabase
6
7   @Database(entities = [Movie::class], version = 1, exportSchema =
8   false)
9   abstract class AppDatabase : RoomDatabase() {
10      abstract fun movieDao(): MovieDao
11
12      companion object {
13          @Volatile
14          private var INSTANCE: AppDatabase? = null
15
16          fun getDatabase(context: Context): AppDatabase {
17              return INSTANCE ?: synchronized(this) {
18                  val instance = Room.databaseBuilder(
19                      context.applicationContext,
20                      AppDatabase::class.java,
21                      "movie_database"
22                  ).build()
23                  INSTANCE = instance
24                  instance
25              }
26          }
27      }
28  }
```

14. MovieDao.kt

Tabel 14 Source Code Jawaban Soal 1

```
1   package com.example.modul5.data
2
3   import androidx.room.Dao
4   import androidx.room.Insert
5   import androidx.room.OnConflictStrategy
6   import androidx.room.Query
```

```
7   import kotlinx.coroutines.flow.Flow
8
9   @Dao
10  interface MovieDao {
11      @Query("SELECT * FROM movies")
12      fun getAllMovies(): Flow<List<Movie>>
13
14      @Insert(onConflict = OnConflictStrategy.REPLACE)
15      suspend fun insertAll(movies: List<Movie>)
16
17      @Query("DELETE FROM movies")
18      suspend fun deleteAll()
19  }
```

15. MovieRepository.kt

Tabel 15 Source Code Jawaban Soal 1

```
1   package com.example.modul5.data
2
3   import android.util.Log
4   import com.example.modul5.networking.ApiClient
5   import kotlinx.coroutines.flow.Flow
6
7   class MovieRepository(private val movieDao: MovieDao) {
8
9       val movies: Flow<List<Movie>> = movieDao.getAllMovies()
10
11      suspend fun refreshMovies() {
12          try {
13              val response = ApiClient.instance.getPopularMovies()
14              movieDao.deleteAll()
15              movieDao.insertAll(response.results)
16          } catch (e: Exception) {
17              Log.e("MovieRepository", "Error refreshing movies:
    ${e.message}")
18          }
19      }
20
21      // --- FUNGSI BARU UNTUK MENGAMBIL DETAIL DARI INTERNET ---
22      suspend fun getMovieDetails(movieId: Int): MovieDetails? {
23          return try {
24              ApiClient.instance.getMovieDetails(movieId)
25          } catch (e: Exception) {
26              Log.e("MovieRepository", "Error getting movie
27  details: ${e.message}")
                null
28          }
29      }
30  }
```

16. ApiClient.kt

Tabel 16 Source Code Jawaban Soal 1

```
1   // File: networking/ApiClient.kt
2   package com.example.modul5.networking
3
4   import
    com.jakewharton.retrofit2.converter.kotlinx.serialization.asConvert
    erFactory
5   import kotlinx.serialization.json.Json
6   import okhttp3.Interceptor
7   import okhttp3.MediaType.Companion.toMediaType
8   import okhttp3.OkHttpClient
9   import retrofit2.Retrofit
10
11  object ApiClient {
12      private const val BASE_URL = "https://api.themoviedb.org/3/"
13
14      // PENTING: Masukkan API Read Access Token Anda di sini
        private const val API_READ_ACCESS_TOKEN =
    "eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIxMzY5MDE0ZDJjYjkxNjIzOWY1NjdkMWUwZ
    WZmZDUzNiIsIm5iZiI6MTc0OTcwNjcyNi4yMiwic3ViIjoiNjg0YTY3ZTYyNGMzOWVm
    NGU1MWViYTIyIiwic2NvcGVzIjpbImFwaV9yZWFkIl0sInZlcnNpb24iOjF9.Aojt3C
    Y11RJ5kjNKHHHGxKedOWNHzbvnqJ_1LBJ34TY"
15
16      private val json = Json {
17          ignoreUnknownKeys = true
18      }
19
20      val instance: ApiService by lazy {
21          // 1. Buat Interceptor untuk menambahkan header secara
    otomatis
22          val authInterceptor = Interceptor { chain ->
23              val originalRequest = chain.request()
24              val newRequest = originalRequest.newBuilder()
25                  .header("Authorization", "Bearer
    $API_READ_ACCESS_TOKEN")
26                  .header("accept", "application/json")
27                  .build()
28              chain.proceed(newRequest)
29          }
30
31          // 2. Buat OkHttpClient dan tambahkan interceptor
32          val okHttpClient = OkHttpClient.Builder()
33              .addInterceptor(authInterceptor)
34              .build()
35
36          // 3. Buat instance Retrofit dengan OkHttpClient yang sudah
    dimodifikasi
37          val retrofit = Retrofit.Builder()
38              .baseUrl(BASE_URL)
```

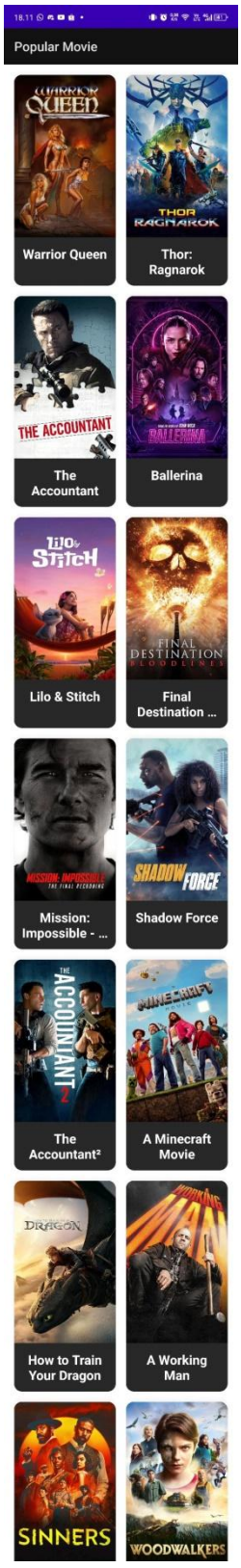
```
            .client(okHttpClient) // Gunakan client custom kita
            .addConverterFactory(json.asConverterFactory("application/json".toMediaType()))
            .build()

        retrofit.create(ApiService::class.java)
    }
}
```

17. ApiService.kt

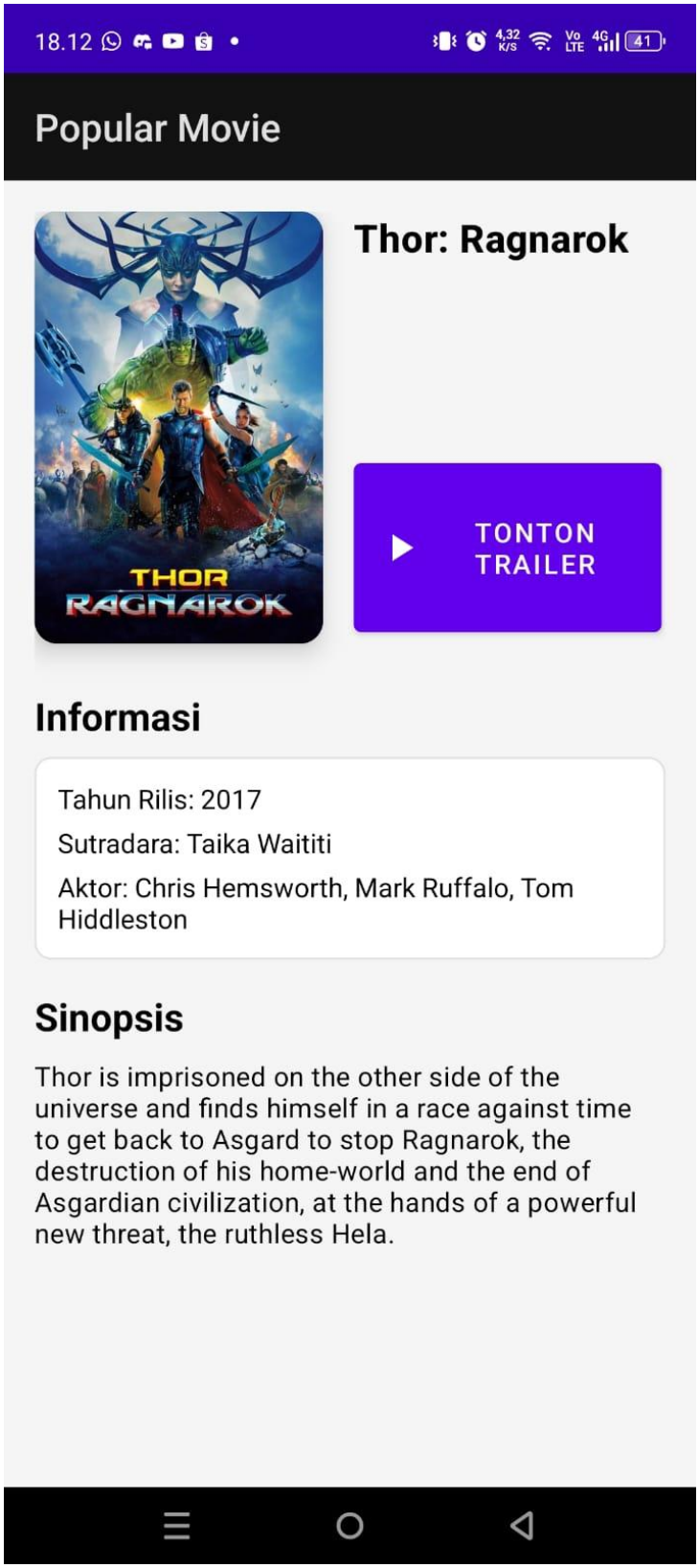
Tabel 17 Source Code Jawaban Soal 1

```
package com.example.modul5.networking

import com.example.modul5.data.MovieDetails
import com.example.modul5.data.MovieResponse
import retrofit2.http.GET
import retrofit2.http.Path
import retrofit2.http.Query

interface ApiService {
    // Endpoint untuk daftar film populer (tetap sama)
    @GET("movie/popular")
    suspend fun getPopularMovies(
        @Query("language") language: String = "en-US",
        @Query("page") page: Int = 1
    ): MovieResponse

    // --- FUNGSI BARU UNTUK MENDAPATKAN DETAIL FILM ---
    @GET("movie/{movie_id}")
    suspend fun getMovieDetails(
        @Path("movie_id") movieId: Int,
        // 'append_to_response' adalah trik API TMDB untuk mendapatkan
        // data kredit (aktor) dan video (trailer) dalam satu panggilan
        @Query("append_to_response") appendToResponse: String = "videos,credits"
    ): MovieDetails
}
```

B. Output Program



Gambar 2 Hasil Tampilan UI List Soal 1



Gambar 3 Hasil Tampilan UI Detail Soal 1

C. Pembahasan

1. MainActivity.kt

MainActivity adalah titik masuk utama aplikasi. Fungsinya sangat sederhana namun krusial, yaitu sebagai *host* untuk semua *fragment* dalam aplikasi ini. Dalam metode onCreate(), ia hanya mengatur *layout* utama dari activity_main.xml [320]. *Layout* ini berisi sebuah FragmentContainerView yang berfungsi sebagai wadah untuk NavHostFragment, komponen dari Jetpack Navigation yang mengelola semua transisi antar *fragment* (seperti dari ListFragment ke DetailFragment) sesuai dengan yang didefinisikan di nav_graph.xml

2. DetailFragment.kt

DetailFragment menampilkan informasi rinci dari satu film yang dipilih. Ia menerima id film melalui Safe Args dari Jetpack Navigation. Sama seperti ListFragment, ia membuat ViewModel sendiri untuk mengambil detail film. Dalam onViewCreated, ia mengobservasi state dari ViewModel (yang kemungkinan memuat data film tunggal berdasarkan ID). Setelah data diterima, ia mengisi semua View yang relevan: judul (movieTitle), sinopsis (movieOverview), tanggal rilis (movieReleaseDate), dan rating (movieRating). Library Glide juga digunakan di sini untuk memuat gambar poster film ke ImageView di bagian atas layar.

3. ListFragment.kt:

ListFragment adalah layar utama yang menampilkan daftar film. Ia menggunakan View Binding untuk mengakses view. Di dalam onViewCreated, ia menginisialisasi MovieRepository dan MovieViewModelFactory untuk membuat MovieViewModel. RecyclerView diatur dengan LinearLayoutManager dan MovieAdapter. Bagian terpenting adalah blok viewLifecycleOwner.lifecycleScope.launch, di mana ia mengobservasi viewModel.movies menggunakan collectLatest. Setiap kali ada data baru, data tersebut akan dikirim ke adapter (adapter.submitList(movies)). Fragment ini juga menangani navigasi ke DetailFragment saat sebuah film diklik, dengan mengirimkan id film yang dipilih melalui NavDirections.

4. MovieAdapter.kt

jembatan antara data film dan RecyclerView. Ia menggunakan ListAdapter yang merupakan subclass dari RecyclerView.Adapter yang dioptimalkan untuk

daftar yang dapat berubah, berkat penggunaan DiffUtil. Class internal MovieViewHolder bertanggung jawab untuk binding data satu objek Movie ke tampilan dalam item_movie.xml. Di dalam fungsi bind(), ia mengatur judul film dan tanggal rilis ke TextView yang sesuai, dan yang paling penting, ia menggunakan library Glide untuk memuat gambar poster dari URL (movie.poster_path) ke dalam ImageView, lengkap dengan placeholder dan gambar error. Adapter ini juga menangani klik pada setiap item dengan meneruskan aksi klik ke lambda onItemClick yang diinisialisasi dari ListFragment.

5. **Movie.kt**

mendefinisikan model data utama aplikasi, yaitu **data class Movie**. *Class* ini berfungsi sebagai representasi dari satu entitas film dan dirancang untuk bekerja dengan *database* Room dan respons dari API. Setiap objek Movie memiliki properti seperti id (integer, sebagai kunci utama atau @PrimaryKey), title (judul film), overview (sinopsis), poster_path (URL untuk gambar poster), release_date (tanggal rilis), dan vote_average (rata-rata rating). Anotasi **@Entity(tableName = "movies")** menandakan bahwa *class* ini adalah sebuah tabel dalam *database* Room dengan nama "movies". Selain itu, anotasi **@SerializedName** digunakan pada setiap properti untuk memetakan nama *field* dari JSON yang diterima dari API (misalnya, poster_path) ke nama properti di dalam *class* Kotlin, memastikan deserialisasi data dari jaringan berjalan dengan benar.

Fungsi getById() disediakan untuk mencari OOTD berdasarkan ID-nya, sangat berguna ketika aplikasi perlu menampilkan detail item tertentu saat pengguna memilih dari daftar.

6. **MovieViewModel.kt**

MovieViewModel adalah kelas yang bertanggung jawab untuk menyediakan data ke UI dan mempertahankan state-nya dari perubahan konfigurasi. Kelas ini mengambil MovieRepository sebagai dependensi pada konstruktornya. Properti utamanya adalah movies, yang merupakan StateFlow<List<Movie>>. Properti ini diinisialisasi dengan memanggil repository.getMovies() dan mengubahnya menjadi StateFlow menggunakan stateIn(). Ini memungkinkan Fragment untuk mengobservasi daftar film secara reaktif. Terdapat juga fungsi refreshMovies() yang dipanggil di dalam blok init. Fungsi ini diluncurkan dalam viewModelScope dan memanggil repository.refreshMovies() untuk memastikan data film diambil dari API saat ViewModel pertama kali dibuat. Penanganan error sederhana juga disertakan dalam blok try-catch.

7. **MovieViewModelFactory.kt**

MovieViewModelFactory adalah kelas pabrik yang krusial untuk membuat instance dari MovieViewModel. Karena MovieViewModel memiliki dependensi (MovieRepository), kita tidak bisa membiarkan sistem membuatnya secara otomatis.

Factory ini mengimplementasikan ViewModelProvider.Factory dan mengambil MovieRepository sebagai parameter. Di dalam metode create(), ia memeriksa apakah modelClass yang diminta adalah MovieViewModel::class.java. Jika benar, ia akan mengembalikan instance baru dari MovieViewModel dengan menyuntikkan repository yang telah diterimanya. Pola ini sangat penting untuk Dependency Injection dan pengujian (testing).

8. activity_main.xml

kerangka dasar untuk MainActivity. Satu-satunya elemen penting di dalamnya adalah <androidx.fragment.app.FragmentContainerView>. Komponen ini dikonfigurasi sebagai NavHostFragment melalui atribut android:name. Atribut app:navGraph="@navigation/nav_graph" menghubungkannya dengan grafik navigasi aplikasi, dan app:defaultNavHost="true" menjadikannya sebagai target utama untuk navigasi dan penanganan tombol "kembali" sistem.

9. fragment_detail.xml

Layout ini mendesain layar detail film. Biasanya menggunakan <ScrollView> sebagai elemen akar agar konten bisa digulir jika tidak muat di layar. Di dalamnya, ConstraintLayout atau LinearLayout digunakan untuk menyusun berbagai elemen UI seperti ImageView besar di bagian atas untuk poster film, diikuti oleh beberapa TextView untuk menampilkan judul, rating, tanggal rilis, dan sinopsis film yang bisa jadi cukup panjang.

10. fragment_list.xml

layout untuk ListFragment. Komponen utamanya adalah <androidx.recyclerview.widget.RecyclerView>, yang akan diisi dengan daftar film. Selain itu, biasanya terdapat juga komponen lain seperti <ProgressBar> untuk menunjukkan status loading saat data sedang diambil dari jaringan, dan mungkin sebuah <TextView> yang berfungsi sebagai emptyView untuk memberitahu pengguna jika tidak ada data yang dapat ditampilkan.

11. item_movie.xml

File ini mendefinisikan tampilan untuk satu baris atau satu item di dalam RecyclerView pada ListFragment. Umumnya menggunakan CardView sebagai kontainer untuk memberikan tampilan yang rapi dengan bayangan dan sudut melengkung. Di dalam card, terdapat ImageView untuk menampilkan poster film dan beberapa TextView untuk menampilkan informasi ringkas seperti judul dan tanggal rilis film.

12. nav_graph.xml

File ini adalah jantung dari komponen Navigasi Jetpack. Ia secara visual dan deklaratif mendefinisikan semua tujuan navigasi (yaitu, fragment) dan aksi yang menghubungkannya. Di sini, <fragment android:id="@+id/listFragment"> didefinisikan sebagai tujuan awal (app:startDestination). Terdapat sebuah <action> di dalam listFragment yang mendefinisikan transisi ke <fragment android:id="@+id/detailFragment">. Pentingnya, detailFragment didefinisikan dengan sebuah <argument> (misalnya, movieId dengan tipe integer), yang memungkinkan pengiriman data (ID film) dari ListFragment ke DetailFragment dengan cara yang aman dan terjamin tipenya (type-safe

13. AppDatabase.kt

mendefinisikan kelas abstrak yang mewarisi RoomDatabase. Kelas ini berfungsi sebagai "pemegang" utama database dan titik akses utama ke data yang tersimpan. Anotasi @Database digunakan untuk mengkonfigurasi database, dengan menyebutkan class Movie sebagai satu-satunya entity (entities = [Movie::class]) dan mengatur nomor versi database (version = 1). Di dalamnya, terdapat sebuah fungsi abstrak movieDao() yang mengembalikan instance dari MovieDao, sehingga bagian lain dari aplikasi (seperti Repository) dapat mengakses metode-metode query yang telah didefinisikan. Terdapat juga sebuah companion object yang mengimplementasikan pola Singleton untuk memastikan hanya ada satu instance AppDatabase yang dibuat di seluruh aplikasi, mencegah masalah konkurensi dan menjaga konsistensi data.

14. MovieDao.kt

adalah interface MovieDao (Data Access Object), yang merupakan komponen inti dari Room Persistence Library. Interface ini mendefinisikan semua operasi database yang dibutuhkan oleh aplikasi. Anotasi @Dao memberitahu Room bahwa ini adalah sebuah DAO. Di dalamnya, terdapat beberapa fungsi yang diberi anotasi sesuai fungsinya: @Query("SELECT * FROM movies") pada fungsi getMovies() untuk mengambil semua data film dari tabel sebagai Flow<List<Movie>>, yang memungkinkan UI untuk mengobservasi perubahan data secara real-time. Fungsi insertAll() dengan anotasi @Insert(onConflict = OnConflictStrategy.REPLACE) digunakan untuk menyimpan daftar film ke dalam database; jika film dengan id yang sama sudah ada, data lama akan diganti dengan yang baru. Terakhir, @Query("DELETE FROM movies") pada fungsi deleteAll() menyediakan cara untuk menghapus semua data dari tabel.

15. MovieRepository.kt

MovieRepository adalah kelas yang berperan sebagai perantara antara sumber data (database lokal dan jaringan) dengan seluruh bagian aplikasi lainnya

(terutama ViewModel). Ini adalah implementasi dari Repository Pattern dan berfungsi sebagai Single Source of Truth (Satu Sumber Kebenaran). Repository ini memiliki dependensi ke ApiService (untuk mengambil data dari jaringan) dan MovieDao (untuk mengakses database lokal). Fungsi utamanya adalah getMovies(), yang mengembalikan Flow<List<Movie>> langsung dari MovieDao. Metode refreshMovies() bertugas untuk mengambil data film terbaru dari API melalui apiService.getMovies(). Jika panggilan API berhasil, data baru tersebut akan disimpan ke database lokal dengan memanggil movieDao.deleteAll() terlebih dahulu lalu movieDao.insertAll(). Dengan cara ini, UI selalu menampilkan data dari database, sementara repository menangani pembaruan data di latar belakang.

16. **ApiClient.kt**

Object ApiClient bertanggung jawab untuk membuat dan mengkonfigurasi instance Retrofit yang akan digunakan di seluruh aplikasi. Ini adalah implementasi dari pola Singleton. Di dalamnya, sebuah base URL untuk API (misalnya, The Movie Database API) didefinisikan. Kemudian, instance Retrofit dibuat menggunakan Retrofit.Builder(), yang dikonfigurasi dengan URL dasar tersebut dan sebuah converter factory (GsonConverterFactory.create()). Converter ini bertugas untuk mengubah respons JSON dari API menjadi objek-objek Kotlin (data class) secara otomatis. Akhirnya, instance ApiService dibuat dengan memanggil retrofit.create(ApiService::class.java), yang siap digunakan untuk melakukan panggilan jaringan.

17. **ApiService.kt**

Interface ApiService adalah tempat di mana semua endpoint API didefinisikan menggunakan Retrofit. Di sini, hanya ada satu fungsi, yaitu getMovies(), yang diberi anotasi @GET("movie/popular"). Anotasi ini memberitahu Retrofit untuk membuat sebuah permintaan HTTP GET ke endpoint movie/popular relatif terhadap URL dasar yang dikonfigurasi di ApiClient. Fungsi ini juga menyertakan parameter api_key melalui anotasi @Query, yang akan ditambahkan ke URL sebagai parameter kueri (misalnya, ...?api_key=YOUR_API_KEY). Fungsi ini dideklarasikan sebagai suspend fun, yang berarti ia dapat dipanggil dari dalam sebuah coroutine tanpa memblokir thread utama, dan akan mengembalikan sebuah objek MovieResponse (sebuah data class yang mungkin membungkus daftar film dari API).

D. Tautan Git

https://github.com/Ibnuuuuuuuuuuuuuuuuuuuuuuuuuuuu/PEMROGRAMAN-MOBILE