# LAPORAN PRAKTIKUM
# PEMROGRAMAN MOBILE
# MODUL 5


## CONNECT TO THE INTERNET

**Oleh:**

**Firda Khoirunisa** **NIM. 2310817220025**

**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 Praktikum ini dikerjakan oleh:

Nama Praktikan : Firda Khoirunisa
NIM : 2310817220025

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

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# SOAL 1

**Soal Praktikum:**

1. Lanjutkan aplikasi Android yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi sesuai ketentuan berikut:

   a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.
   b. Gunakan KotlinX Serialization sebagai library JSON.
   c. Gunakan library seperti Coil atau Glide untuk image loading.
   d. API yang digunakan pada modul ini bebas, contoh API gratis The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: 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. AppDatabase.kt

```
1  package com.example.modul5
2
3  import android.content.Context
4  import androidx.room.Database
5  import androidx.room.Room
6  import androidx.room.RoomDatabase
7
8  @Database(entities = [FavoriteTvShow::class], version = 1,
9  exportSchema = false)
10 abstract class AppDatabase : RoomDatabase() {
11     abstract fun favoriteDao(): FavoriteDao
12
13     companion object {
14         @Volatile
15         private var INSTANCE: AppDatabase? = null
16
```

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

*Tabel 1. Source Code Jawaban Soal 1*

## 2. BindingAdapter.kt

```
package com.example.modul5

import android.widget.ImageView
import androidx.databinding.BindingAdapter
import com.bumptech.glide.Glide

@BindingAdapter("imageUrl")
fun loadImage(view: ImageView, url: String?) {
    if (!url.isNullOrEmpty()) {
        Glide.with(view.context)
            .load(url)
            .into(view)
    }
}
```

*Tabel 2. Source Code Jawaban Soal 1*

## 3. Detailfragment.kt

```
package com.example.modul5

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.fragment.app.viewModels
import androidx.lifecycle.lifecycleScope
import com.example.modul5.databinding.FragmentDetailBinding
import kotlinx.coroutines.launch

class DetailFragment : Fragment() {

    private lateinit var binding: FragmentDetailBinding
```

```
    private val favoriteViewModel: FavoriteViewModel by viewModels()

    private var currentTvShow: TvShow? = null

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View {
        binding = FragmentDetailBinding.inflate(inflater, container, false)
        binding.lifecycleOwner = viewLifecycleOwner

        val tvShow = arguments?.getParcelable<TvShow>("tvshow")
        tvShow?.let {
            currentTvShow = it
            binding.tvShow = it
            binding.executePendingBindings()

            binding.openLinkButton.setOnClickListener {
                currentTvShow?.let { tv ->
                    val intent = Intent(Intent.ACTION_VIEW, Uri.parse("https://www.themoviedb.org/tv/${tv.id}"))
                    startActivity(intent)
                }
            }

            binding.backButton.setOnClickListener {

requireActivity().supportFragmentManager.popBackStack()
            }

            lifecycleScope.launch {
                val isFav = favoriteViewModel.isFavorite(it)
                updateFavIcon(isFav)

                binding.favIconButton.setOnClickListener {
                    favoriteViewModel.toggleFavorite(tvShow)
                    lifecycleScope.launch {

updateFavIcon(favoriteViewModel.isFavorite(tvShow))
                    }
                }
            }
        }

        return binding.root
    }

    private fun updateFavIcon(isFav: Boolean) {
        binding.favIconButton.setImageResource(
            if (isFav) R.drawable.ic_favorite else R.drawable.ic_favorite_border
        )
```

```
69          val descRes = if (isFav) R.string.remove_from_favorite
70  else                                    R.string.add_to_favorite
71          binding.favIconButton.contentDescription              =
72  getString(descRes)
73      }
74  }
```

*Tabel 3. Source Code Jawaban Soal 1*

## 4. DramaViewModel

```
1   package                                          com.example.modul5
2
3   import                                             android.util.Log
4   import                                  androidx.lifecycle.ViewModel
5   import                                androidx.lifecycle.viewModelScope
6   import                         kotlinx.coroutines.flow.MutableStateFlow
7   import                                kotlinx.coroutines.flow.StateFlow
8   import                                        kotlinx.coroutines.launch
9
10  class DramaViewModel(private val repository: TmdbRepository) :
11  ViewModel()                                                          {
12
13      private              val              _tvShows              =
14  MutableStateFlow<List<TvShow>>(emptyList())
15      val    tvShows:    StateFlow<List<TvShow>>    =    _tvShows
16
17      init                                                             {
18          loadTvShows()
19      }
20
21      private            fun            loadTvShows()                 {
22          viewModelScope.launch                                        {
23              try                                                      {
24                  val response = repository.getChineseTvShows()
25                  _tvShows.value        =        response.results
26                  Log.d("DramaViewModel",                  "Loaded
27  ${response.results.size}           Chinese           dramas.")
28              }        catch        (e:        Exception)        {
29                  Log.e("DramaViewModel", "Failed to load data",
30  e)
31              }
32          }
33      }
34  }
```

*Tabel 4. Source Code Jawaban Soal 1*

## 5. DramaViewModelFactory

```
1   package                                          com.example.modul5
2
3   import                                  androidx.lifecycle.ViewModel
4   import                        androidx.lifecycle.ViewModelProvider
```

```
5
6   class DramaViewModelFactory(private val repository:
7   TmdbRepository) : ViewModelProvider.Factory {
8       override fun <T : ViewModel> create(modelClass: Class<T>):
9   T {
10          if
11  (modelClass.isAssignableFrom(DramaViewModel::class.java)) {
12              return DramaViewModel(repository) as T
13          }
14          throw IllegalArgumentException("Unknown ViewModel
15  class")
16      }
17  }
```

*Tabel 5. Source Code Jawaban Soal 1*

### 6. FavoriteDao.kt

```
1   package com.example.modul5
2
3   import androidx.room.*
4   import kotlinx.coroutines.flow.Flow
5
6   @Dao
7   interface FavoriteDao {
8
9       @Insert(onConflict = OnConflictStrategy.REPLACE)
10      suspend fun addToFavorite(tvShow: FavoriteTvShow)
11
12      @Delete
13      suspend fun removeFromFavorite(tvShow: FavoriteTvShow)
14
15      @Query("SELECT * FROM favorite_tv_shows")
16      fun getAllFavorites(): Flow<List<FavoriteTvShow>>
17
18      @Query("SELECT * FROM favorite_tv_shows WHERE id = :id LIMIT
19  1")
20      suspend fun getFavoriteById(id: Int): FavoriteTvShow?
21  }
```

*Tabel 6. Source Code Jawaban Soal 1*

### 7. FavoriteFragment,kt

```
1   package com.example.modul5
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.viewModels
9   import androidx.lifecycle.ViewModelProvider
10  import androidx.lifecycle.lifecycleScope
```

```
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.modul5.databinding.FragmentFavoriteBinding
import kotlinx.coroutines.flow.collectLatest
import kotlinx.coroutines.launch

class FavoriteFragment : Fragment() {

    private lateinit var binding: FragmentFavoriteBinding
    private val favoriteViewModel: FavoriteViewModel by
viewModels {

ViewModelProvider.AndroidViewModelFactory.getInstance(requireA
ctivity().application)
    }

    private lateinit var adapter: ListDramaAdapter

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?
    ): View {
        binding = FragmentFavoriteBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        adapter = ListDramaAdapter(emptyList()) { selected ->
            val fragment = DetailFragment().apply {
                arguments = Bundle().apply {
                    putParcelable("tvshow", selected)
                }
            }
            parentFragmentManager.beginTransaction()
                .replace(R.id.fragment_container, fragment)
                .addToBackStack(null)
                .commit()
        }

        binding.recyclerView.apply {
            layoutManager =
LinearLayoutManager(requireContext())
            adapter = this@FavoriteFragment.adapter
        }

        binding.backButton.setOnClickListener {
            parentFragmentManager.popBackStack()
        }

        viewLifecycleOwner.lifecycleScope.launch {
            favoriteViewModel.favorites.collectLatest {
```

```
favoriteList ->
                val tvShowList = favoriteList.map {
it.toTvShow() }
                adapter.updateData(tvShowList)
            }
        }
    }
}
```

*Tabel 7. Source Code Jawaban Soal 1*

## 8. FavoriteRepository

```
package com.example.modul5

import android.util.Log
import kotlinx.coroutines.flow.Flow

class FavoriteRepository(private val dao: FavoriteDao) {

    fun getFavorites(): Flow<List<FavoriteTvShow>> =
dao.getAllFavorites()

    suspend fun isFavorite(id: Int): Boolean =
dao.getFavoriteById(id) != null

    suspend fun add(tvShow: FavoriteTvShow) {
        dao.addToFavorite(tvShow)
    }

    suspend fun remove(tvShow: FavoriteTvShow) {
        dao.removeFromFavorite(tvShow)
    }
}
```

*Tabel 8. Source Code Jawaban Soal 1*

## 9. FavoriteTvShow.kt

```
package com.example.modul5

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

@Entity(tableName = "favorite_tv_shows")
data class FavoriteTvShow(
    @PrimaryKey val id: Int,
    val name: String,
    val posterPath: String?,
    val firstAirDate: String,
    val rating: Double,
    val overview: String
)
fun FavoriteTvShow.toTvShow(): TvShow {
```

```
    return                                                  TvShow(
        id                      =                       this.id,
        name                    =                     this.name,
        posterPath              =                this.posterPath,
        rating                  =                    this.rating,
        firstAirDate            =              this.firstAirDate,
        overview                 =                  this.overview
    )
}
```

*Tabel 9. Source Code Jawaban Soal 1*

### 10. FavoriteViewModel.kt

```
package                                         com.example.modul5

import                                      android.app.Application
import                                            android.util.Log
import                          androidx.lifecycle.AndroidViewModel
import                            androidx.lifecycle.viewModelScope
import                                    kotlinx.coroutines.flow.*
import                                    kotlinx.coroutines.launch

class     FavoriteViewModel(application:     Application)     :
AndroidViewModel(application)                                    {

    private   val   db   =   AppDatabase.getDatabase(application)
    private             val             repository             =
FavoriteRepository(db.favoriteDao())

    val     favorites:     StateFlow<List<FavoriteTvShow>>     =
repository.getFavorites()
        .stateIn(viewModelScope,
SharingStarted.WhileSubscribed(5000),               emptyList())

    fun         toggleFavorite(tvShow:         TvShow)         {
        viewModelScope.launch                                   {
            val          fav          =          FavoriteTvShow(
                id                  =                  tvShow.id,
                name                =                tvShow.name,
                posterPath          =          tvShow.posterPath,
                firstAirDate        =        tvShow.firstAirDate,
                rating              =              tvShow.rating,
                overview             =            tvShow.overview
            )
            if       (repository.isFavorite(tvShow.id))       {
                repository.remove(fav)
            }                         else                      {
                repository.add(fav)
            }
        }
    }

```

```
40      suspend fun isFavorite(tvShow: TvShow): Boolean {
41          return repository.isFavorite(tvShow.id)
42      }
43  }
```

*Tabel 10. Source Code Jawaban Soal 1*

### 11. HomeFragment.kt

```
1   package com.example.modul5
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.viewModels
9   import androidx.lifecycle.lifecycleScope
10  import androidx.recyclerview.widget.LinearLayoutManager
11  import com.example.modul5.databinding.FragmentHomeBinding
12  import kotlinx.coroutines.flow.collectLatest
13  import kotlinx.coroutines.launch
14
15  class HomeFragment : Fragment() {
16
17      private lateinit var binding: FragmentHomeBinding
18      private val viewModel: DramaViewModel by viewModels {
19
20  DramaViewModelFactory(TmdbRepository(NetworkModule.apiService))
21      }
22      private lateinit var adapter: ListDramaAdapter
23
24      override fun onCreateView(inflater: LayoutInflater,
25  container: ViewGroup?, savedInstanceState: Bundle?): View {
26          binding = FragmentHomeBinding.inflate(inflater,
27  container, false)
28          return binding.root
29      }
30
31      override fun onViewCreated(view: View, savedInstanceState:
32  Bundle?) {
33          adapter = ListDramaAdapter(emptyList()) { selectedTvShow
34  ->
35              val fragment = DetailFragment().apply {
36                  arguments = Bundle().apply {
37                      putParcelable("tvshow", selectedTvShow)
38                  }
39              }
40              parentFragmentManager.beginTransaction()
41                  .replace(R.id.fragment_container, fragment)
42                  .addToBackStack(null)
43                  .commit()
44          }
```

```
45
46          binding.recyclerView.layoutManager                          =
47  LinearLayoutManager(requireContext())
48          binding.recyclerView.adapter          =          adapter
49
50          viewLifecycleOwner.lifecycleScope.launch                    {
51              viewModel.tvShows.collectLatest    {    tvList    ->
52                  adapter.updateData(tvList)
53              }
54          }
55
56          binding.buttonFavorite.setOnClickListener                   {
57              parentFragmentManager.beginTransaction()
58                  .replace(R.id.fragment_container,
59  FavoriteFragment())
60                  .addToBackStack(null)
61                  .commit()
62          }
63      }
64  }
```

*Tabel 11. Source Code Jawaban Soal 1*

**12. ListDramaAdapter.kt**

```
1   package                                          com.example.modul5
2
3   import                                       android.content.Intent
4   import                                              android.net.Uri
5   import                                    android.view.LayoutInflater
6   import                                       android.view.ViewGroup
7   import                  androidx.recyclerview.widget.RecyclerView
8   import           com.example.modul5.databinding.ItemDramaBinding
9
10  class                                               ListDramaAdapter(
11      private           var           list:           List<TvShow>,
12      private    val    onDetailClick:    (TvShow)    ->    Unit
13  )   :   RecyclerView.Adapter<ListDramaAdapter.MyViewHolder>()   {
14
15      inner class MyViewHolder(val binding: ItemDramaBinding) :
16  RecyclerView.ViewHolder(binding.root)
17
18      override fun onCreateViewHolder(parent: ViewGroup, viewType:
19  Int):                        MyViewHolder                        {
20          val                          binding                        =
21  ItemDramaBinding.inflate(LayoutInflater.from(parent.context),
22  parent,                                                    false)
23          return                                  MyViewHolder(binding)
24      }
25
26      override    fun    getItemCount():    Int    =    list.size
27
28      override    fun    onBindViewHolder(holder:    MyViewHolder,
```

```
29 position:                          Int)                          {
30         val          tvShow          =          list[position]
31         val      context      =      holder.binding.root.context
32
33         holder.binding.tvShow              =              tvShow
34
35         holder.binding.buttonDetail.setOnClickListener        {
36             onDetailClick(tvShow)
37         }
38
39         holder.binding.buttonLink.setOnClickListener          {
40             val     intent     =     Intent(Intent.ACTION_VIEW,
41 Uri.parse("https://www.themoviedb.org/tv/${tvShow.id}"))
42             context.startActivity(intent)
43         }
44     }
45
46     fun        updateData(newList:        List<TvShow>)        {
47         list                        =                        newList
48         notifyDataSetChanged()
49     }
50 }
```

*Tabel 12. Source Code Jawaban Soal No. 1*

**13. MainActivity.kt**

```
1  package                                          com.example.modul5
2
3  import                                            android.os.Bundle
4  import                  androidx.appcompat.app.AppCompatActivity
5  import       com.example.modul5.databinding.ActivityMainBinding
6
7  class      MainActivity      :      AppCompatActivity()      {
8
9      private   lateinit   var   binding:   ActivityMainBinding
10
11     override   fun   onCreate(savedInstanceState:   Bundle?)   {
12         super.onCreate(savedInstanceState)
13         binding   =   ActivityMainBinding.inflate(layoutInflater)
14         setContentView(binding.root)
15
16         if       (savedInstanceState      ==      null)       {
17             supportFragmentManager.beginTransaction()
18                 .replace(R.id.fragment_container,
19 HomeFragment())
20                 .commit()
21         }
22     }
23 }
```

*Tabel 13. Source Code Jawaban Soal No. 1*

**14. NetworkModule.kt**

```
1  package com.example.modul5
2
3  import
4  com.jakewharton.retrofit2.converter.kotlinx.serialization.asCo
5  nverterFactory
6  import kotlinx.serialization.json.Json
7  import okhttp3.MediaType.Companion.toMediaType
8  import retrofit2.Retrofit
9
10 object NetworkModule {
11
12     private const val BASE_URL = "https://api.themoviedb.org/3/"
13
14     val apiService: TmdbApiService by lazy {
15         val contentType = "application/json".toMediaType()
16
17         val json = Json {
18             ignoreUnknownKeys = true
19         }
20
21         Retrofit.Builder()
22             .baseUrl(BASE_URL)
23
24 .addConverterFactory(json.asConverterFactory(contentType))
25             .build()
26             .create(TmdbApiService::class.java)
27     }
28 }
```

*Tabel 14. Source Code Jawaban Soal No. 1*

### 15. TmbdApiService.kt

```
1  package com.example.modul5
2
3  import retrofit2.http.GET
4  import retrofit2.http.Query
5
6  interface TmdbApiService {
7      @GET("discover/tv")
8      suspend fun getChineseTvShows(
9          @Query("api_key") apiKey: String,
10         @Query("with_original_language") language: String =
11 "zh",
12         @Query("sort_by") sortBy: String = "popularity.desc",
13         @Query("first_air_date.gte") startDate: String = "2020-
14 01-01",
15         @Query("first_air_date.lte") endDate: String = "2025-05-
16 31"
17     ): TvShowResponse
18 }
```

*Tabel 15. Source Code Jawaban Soal No. 1*

### 16. TmbdRepository.kt

```
1  package com.example.modul5
2
3  class TmdbRepository(private val apiService: TmdbApiService) {
4      suspend fun getChineseTvShows(): TvShowResponse {
5          return apiService.getChineseTvShows(
6              apiKey = "36854acbbff7b761e0f196e36d1fc1a9",
7              startDate = "2023-01-01",
8              endDate = "2025-05-31"
9          )
10     }
11 }
```

*Tabel 16. Source Code Jawaban Soal No. 1*

### 17. TvShow.kt

```
1  package com.example.modul5
2
3  import android.os.Parcelable
4  import kotlinx.parcelize.Parcelize
5  import kotlinx.serialization.SerialName
6  import kotlinx.serialization.Serializable
7
8  @Parcelize
9  @Serializable
10 data class TvShow(
11     val id: Int,
12     val name: String,
13     @SerialName("poster_path") val posterPath: String?,
14     @SerialName("vote_average") val rating: Double,
15     @SerialName("first_air_date") val firstAirDate: String,
16     val overview: String
17 ) : Parcelable
```

*Tabel 17. Source Code Jawaban Soal No. 1*

### 18. TvShowResponse.kt

```
1  package com.example.modul5
2
3  import kotlinx.serialization.Serializable
4
5  @Serializable
6  data class TvShowResponse(
7      val results: List<TvShow>
8  )
```

*Tabel 18. Source Code Jawaban Soal No. 1*

### 19. activity_main.xml

```
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       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent">
8
9       <FrameLayout
10          android:id="@+id/fragment_container"
11          android:layout_width="match_parent"
12          android:layout_height="match_parent"                    />
13  </androidx.constraintlayout.widget.ConstraintLayout>
```

*Tabel 19. Source Code Jawaban Soal No. 1*

**20. fragment_detail.xml**

```
1   <?xml              version="1.0"              encoding="utf-8"?>
2   <layout
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
7       <data>
8           <variable
9               name="tvShow"
10              type="com.example.modul5.TvShow"                    />
11      </data>
12
13      <ScrollView
14          android:layout_width="match_parent"
15          android:layout_height="match_parent"
16          tools:context=".DetailFragment">
17
18          <LinearLayout
19              android:orientation="vertical"
20              android:padding="16dp"
21              android:layout_width="match_parent"
22              android:layout_height="wrap_content"
23              android:gravity="center_horizontal"
24              android:background="@android:color/white">
25
26              <ImageView
27                  android:id="@+id/detail_image"
28                  android:layout_width="match_parent"
29                  android:layout_height="500dp"
30                  android:scaleType="centerCrop"
31
32  android:contentDescription="@string/detail_image"
33                  app:imageUrl='@{tvShow.posterPath  !=  null  ?
34  "https://image.tmdb.org/t/p/w500" + tvShow.posterPath : null}'
35  />
36
37              <TextView
38                  android:id="@+id/title_text"
```

```
39                  android:layout_width="wrap_content"
40                  android:layout_height="wrap_content"
41                  android:text="@{tvShow.name}"
42                  android:textSize="22sp"
43                  android:textStyle="bold"
44                  android:paddingTop="12dp"
45                  android:layout_gravity="center_horizontal"  />
46
47              <TextView
48                  android:id="@+id/year_genre_text"
49                  android:layout_width="wrap_content"
50                  android:layout_height="wrap_content"
51                  android:text="@{tvShow.firstAirDate}"
52                  android:textSize="16sp"
53                  android:paddingTop="4dp"
54                  android:layout_gravity="center_horizontal"  />
55
56              <TextView
57                  android:id="@+id/rating_text"
58                  android:layout_width="wrap_content"
59                  android:layout_height="wrap_content"
60                  android:text='@{"Rating:          "          +
61  String.valueOf(tvShow.rating)}'
62                  android:textSize="16sp"
63                  android:paddingTop="4dp"
64                  android:layout_gravity="center_horizontal"  />
65
66              <ImageButton
67                  android:id="@+id/favIconButton"
68                  android:layout_width="60dp"
69                  android:layout_height="60dp"
70                  android:layout_marginTop="12dp"
71                  android:layout_gravity="center_horizontal"
72
73  android:background="?attr/selectableItemBackgroundBorderless"
74
75  android:contentDescription="@string/add_to_favorite"
76                  android:src="@drawable/ic_favorite_border"  />
77
78              <LinearLayout
79                  android:layout_width="wrap_content"
80                  android:layout_height="wrap_content"
81                  android:orientation="horizontal"
82                  android:gravity="center"
83                  android:layout_marginTop="16dp">
84
85                  <Button
86                      android:id="@+id/openLinkButton"
87                      android:layout_width="wrap_content"
88                      android:layout_height="wrap_content"
89                      android:text="@string/watch_here"
90                      android:textColor="@android:color/white"
```

```
                    android:backgroundTint="@color/primary"
                    android:layout_marginEnd="8dp"          />

                <Button
                    android:id="@+id/backButton"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/button_back"
                    android:textColor="@android:color/white"
                    android:backgroundTint="@color/primary" />
            </LinearLayout>
        </LinearLayout>
    </ScrollView>
</layout>
```

*Tabel 20. Source Code Jawaban Soal No. 1*

**21. fragment_favorite.xml**

```
<layout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <data>
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:id="@+id/headerBar"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:padding="16dp"
            android:background="#2196F3"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent">

            <Button
                android:id="@+id/backButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/button_back"
                android:textColor="@android:color/white"
                android:background="@android:color/transparent"
/>

            <TextView
                android:id="@+id/titleText"
                android:layout_width="180dp"
```

```
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_marginStart="16dp"
                android:text="@string/favorite_title"
                android:textColor="@android:color/white"
                android:textSize="18sp"
                android:textStyle="bold"                    />
        </LinearLayout>

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/recyclerView"
            android:layout_width="0dp"
            android:layout_height="0dp"
            app:layout_constraintTop_toBottomOf="@id/headerBar"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"/>

    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

*Tabel 21. Source Code Jawaban Soal No. 1*

**22. fragment_home.xml**

```
<?xml              version="1.0"              encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/headerBar"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="#2196F3"
        android:padding="16dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent">

        <TextView
            android:id="@+id/appTitleText"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/app_name"
            android:textSize="18sp"
            android:textStyle="bold"
            android:textColor="@android:color/white"        />

```

```
        <Button
            android:id="@+id/buttonFavorite"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/go_to_favorites"
            android:textColor="@android:color/white"
            android:background="@android:color/transparent"  />
    </LinearLayout>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toBottomOf="@id/headerBar"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"            />
</androidx.constraintlayout.widget.ConstraintLayout>
```

*Tabel 22. Source Code Jawaban Soal No. 1*

**23. item_drama.xml**

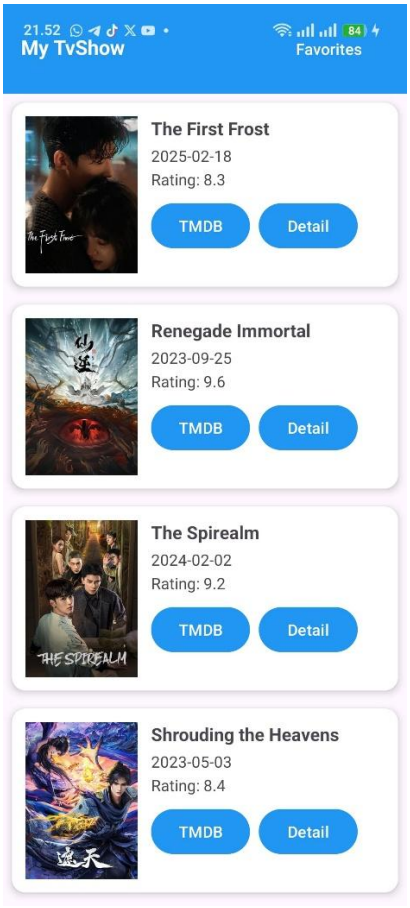
```
<?xml             version="1.0"             encoding="utf-8"?>
<layout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="tvShow"
            type="com.example.modul5.TvShow"               />
    </data>

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="8dp"
        app:cardCornerRadius="12dp"
        app:cardElevation="4dp">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="12dp">

            <ImageView
                android:id="@+id/imageView"
                android:layout_width="100dp"
                android:layout_height="140dp"
                android:scaleType="centerCrop"

```

```
31  android:contentDescription="@string/detail_image"
32
33  app:imageUrl='@{"https://image.tmdb.org/t/p/w500"            +
34  tvShow.posterPath}'
35                  app:layout_constraintStart_toStartOf="parent"
36                  app:layout_constraintTop_toTopOf="parent"
37
38  app:layout_constraintBottom_toBottomOf="parent"             />
39
40              <TextView
41                  android:id="@+id/titleText"
42                  android:layout_width="0dp"
43                  android:layout_height="wrap_content"
44                  android:text="@{tvShow.name}"
45                  android:textStyle="bold"
46                  android:textSize="16sp"
47
48  app:layout_constraintTop_toTopOf="@id/imageView"
49
50  app:layout_constraintStart_toEndOf="@id/imageView"
51                  app:layout_constraintEnd_toEndOf="parent"
52                  android:layout_marginStart="12dp"           />
53
54              <TextView
55                  android:id="@+id/yearText"
56                  android:layout_width="0dp"
57                  android:layout_height="wrap_content"
58                  android:text="@{tvShow.firstAirDate}"
59                  android:textSize="14sp"
60
61  app:layout_constraintTop_toBottomOf="@id/titleText"
62
63  app:layout_constraintStart_toStartOf="@+id/titleText"
64                  app:layout_constraintEnd_toEndOf="parent"
65                  android:layout_marginTop="4dp"              />
66
67              <TextView
68                  android:id="@+id/ratingText"
69                  android:layout_width="0dp"
70                  android:layout_height="wrap_content"
71                  android:text='@{"Rating:          "          +
72  String.valueOf(tvShow.rating)}'
73
74  app:layout_constraintTop_toBottomOf="@id/yearText"
75
76  app:layout_constraintStart_toStartOf="@+id/titleText"
77                  app:layout_constraintEnd_toEndOf="parent"
78                  android:layout_marginTop="2dp"              />
79
80              <Button
81                  android:id="@+id/buttonLink"
82                  android:layout_width="wrap_content"
```

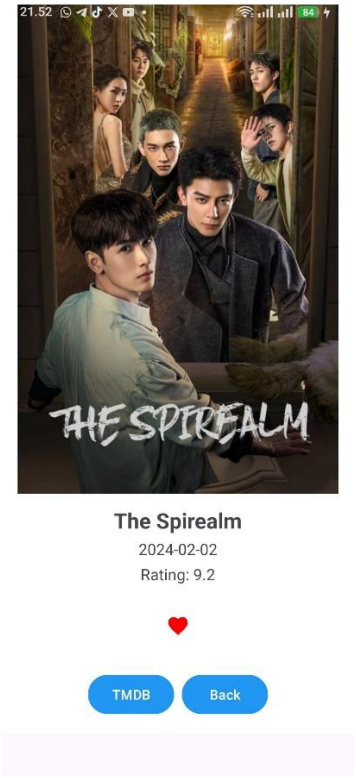
```
                android:layout_height="wrap_content"
                android:text="@string/watch_here"
                android:textColor="@android:color/white"
                android:backgroundTint="@color/primary"

app:layout_constraintTop_toBottomOf="@id/ratingText"

app:layout_constraintStart_toStartOf="@+id/titleText"
                android:layout_marginTop="8dp"              />

            <Button
                android:id="@+id/buttonDetail"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/title_detail"
                android:textColor="@android:color/white"
                android:backgroundTint="@color/primary"

app:layout_constraintTop_toTopOf="@id/buttonLink"

app:layout_constraintStart_toEndOf="@id/buttonLink"
                android:layout_marginStart="8dp"            />

        </androidx.constraintlayout.widget.ConstraintLayout>
    </androidx.cardview.widget.CardView>
</layout>
```

*Tabel 23. Source Code Jawaban Soal No. 1*

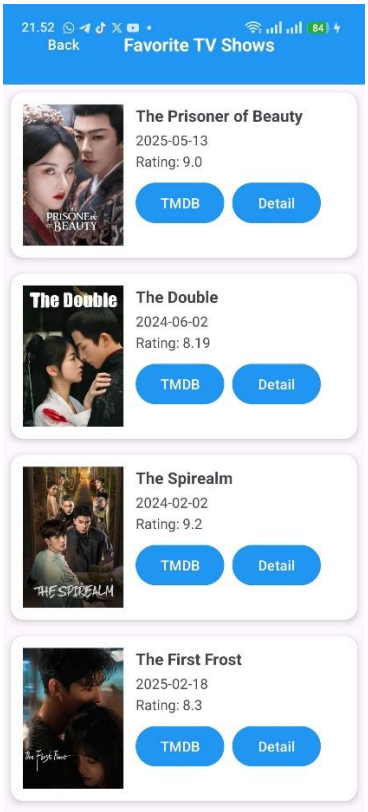
## B. Output Program




*Gambar 1. Screenshot Hasil Jawaban Soal No. 1*




*Gambar 2. Screenshot Hasil Jawaban Soal No. 1*




*Gambar 3. Screenshot Hasil Jawaban Soal No. 1*

## C. Pembahasan

- **AppDatabase.kt:**

AppDatabase ini intinya adalah cetak biru untuk database lokal sebuah aplikasi. Di sinilah ditentukan data apa saja yang akan disimpan, seperti daftar acara TV favorit, dan juga versi database-nya. Ada juga bagian yang bertugas sebagai "penjaga" database, yang disebut DAO (Data Access Object); dia ini yang punya semua perintah untuk menyimpan, menghapus, atau mencari data. Nah, yang penting lagi, AppDatabase ini memastikan cuma ada satu database yang aktif di seluruh aplikasi. Ini penting banget agar aplikasi efisien dan tidak bentrok saat banyak bagian program mau pakai database bersamaan. Jadi, jika akses ke database dibutuhkan, tinggal panggil saja fungsi khusus yang sudah disediakan, dan dia akan otomatis memberikan akses atau bahkan membuat database baru jika belum ada.

- **BindingAdapter.kt**

Secara sederhana, kode ini membuat sebuah aturan khusus di mana setiap kali ada elemen ImageView di tata letak XML yang memiliki atribut imageUrl (yang dibuat sendiri), maka fungsi loadImage ini akan dipanggil.Di dalam fungsi loadImage, ia menerima dua hal: view (yaitu ImageView itu sendiri) dan url (alamat gambar dari internet). Kode ini memeriksa apakah url gambar tidak kosong. Jika ada url, ia akan menggunakan pustaka Glide untuk mengunduh gambar dari url tersebut dan langsung menampilkannya ke ImageView. Ini artinya, pengembang tidak perlu lagi menulis kode Java atau Kotlin yang rumit untuk memuat gambar dari internet setiap kali mereka ingin menampilkan gambar; cukup dengan menambahkan atribut imageUrl di XML.

- **DetailFragment.kt**

DetailFragment adalah komponen antarmuka pengguna (UI) yang dirancang untuk menampilkan informasi lengkap dari sebuah acara TV. Saat fragment ini dimuat, ia akan mengambil data acara TV yang diterima, lalu menampilkan semua detailnya, seperti judul, deskripsi, dan gambar, ke tampilan. Fragment ini memiliki beberapa fitur utama: ada tombol untuk membuka tautan yang akan mengarahkan pengguna ke halaman acara TV di situs The

Movie Database melalui browser, tombol kembali untuk navigasi ke layar sebelumnya, serta tombol favorit. Tombol favorit ini sangat interaktif, karena DetailFragment berkomunikasi dengan FavoriteViewModel untuk mengecek status favorit acara TV tersebut. Berdasarkan status ini, ikon hati akan berubah menjadi penuh jika sudah difavoritkan atau berongga jika belum. Jika pengguna mengklik ikon hati, status favorit acara TV akan diubah (ditambah atau dihapus dari daftar favorit), dan ikon akan langsung diperbarui secara otomatis. Jadi, DetailFragment ini bertanggung jawab penuh dalam menampilkan detail acara TV, membuka tautan terkait, memfasilitasi navigasi, dan mengelola status favorit acara TV tersebut.

- **DramaViewmodel.kt**

DramaViewModel adalah sebuah ViewModel yang bertanggung jawab untuk mengambil dan mengelola daftar acara TV drama Tiongkok. ViewModel ini dirancang agar data tetap ada meskipun terjadi perubahan konfigurasi pada perangkat.Saat DramaViewModel pertama kali dibuat, ia akan langsung memanggil fungsi loadTvShows(). Fungsi ini bekerja di balik layar, menggunakan viewModelScope untuk menjalankan operasi jaringan. Ia akan memanggil repository (yang fungsinya berhubungan dengan data dari The Movie Database) untuk mendapatkan daftar drama Tiongkok. Jika berhasil, daftar acara TV tersebut akan disimpan dan siap ditampilkan ke antarmuka pengguna. Namun, jika terjadi kesalahan saat mengambil data, pesan kesalahan akan dicatat. Jadi, DramaViewModel ini adalah jembatan antara tampilan aplikasi dan data drama Tiongkok yang diambil dari internet, memastikan data tersedia dan diperbarui dengan lancar.

- **DramaViewmodelFactory.kt**

DramaViewModelFactory ini adalah sebuah pabrik khusus yang tugasnya adalah membuat dan menyediakan DramaViewModel dengan benar. Karena DramaViewModel membutuhkan TmdbRepository (yang bertugas mengambil data dari internet), pabrik ini memastikan bahwa setiap kali DramaViewModel diminta, ia akan dibuat dan langsung dilengkapi dengan TmdbRepository yang dibutuhkan. Jadi, DramaViewModelFactory ini berfungsi seperti manajer produksi yang memastikan DramaViewModel selalu siap pakai dengan semua komponen yang diperlukan.

- **FavoriteDao.kt**

FavoriteDao adalah sebuah antarmuka (interface) yang berfungsi sebagai Data Access Object (DAO) untuk database Room aplikasi. Ini adalah "penjaga" yang berisi semua perintah untuk mengelola data acara TV favorit. Di sini, didefinisikan beberapa operasi dasar: ada fungsi addToFavorite yang bisa menambahkan acara TV ke daftar favorit (atau menggantinya jika sudah ada), removeFromFavorite untuk menghapus acara TV dari daftar favorit, getAllFavorites yang akan memberikan semua daftar acara TV favorit dan akan terus diperbarui secara otomatis jika ada perubahan, serta getFavoriteById untuk mencari apakah ada acara TV tertentu berdasarkan ID-nya di daftar favorit. Semua fungsi ini dirancang untuk berinteraksi dengan database secara aman dan efisien.

- **FavoriteFragment.kt**

FavoriteFragment adalah bagian antarmuka pengguna yang dirancang untuk menampilkan daftar semua acara TV yang sudah ditandai sebagai favorit. Fragment ini menggunakan RecyclerView untuk menampilkan daftar tersebut secara efisien. Ketika fragment ini dibuat, ia akan menyiapkan tampilan dan adapter untuk RecyclerView agar bisa menampilkan data. Fragment ini juga berinteraksi dengan FavoriteViewModel untuk terus memantau daftar acara TV favorit. Setiap kali ada perubahan pada daftar favorit, tampilan akan otomatis diperbarui. Jika pengguna mengklik salah satu acara TV di daftar favorit, aplikasi akan membuka DetailFragment untuk menampilkan informasi lebih lanjut tentang acara TV yang dipilih tersebut. Ada juga tombol kembali yang memungkinkan pengguna untuk dengan mudah menavigasi ke layar sebelumnya.

- **FavoriteRepository.kt**

FavoriteRepository ini adalah penghubung antara FavoriteViewModel dan database lokal yang sebenarnya, yang diwakili oleh FavoriteDao. Tugas utamanya adalah mengelola semua operasi terkait acara TV favorit. Repository ini menyediakan beberapa fungsi penting: ada getFavorites() yang akan memberikan daftar semua acara TV yang sudah difavoritkan secara terus-menerus (jika ada perubahan, daftar ini otomatis terbarui), isFavorite() untuk mengecek apakah sebuah acara TV sudah ada di daftar favorit berdasarkan ID-nya, add() untuk menambahkan acara TV ke daftar favorit, dan remove() untuk menghapus acara TV dari

daftar tersebut. Dengan adanya repository ini, ViewModel tidak perlu tahu detail bagaimana data disimpan atau diambil dari database; ia cukup memanggil fungsi-fungsi yang disediakan oleh FavoriteRepository ini.

- **FavoriteTvShow.kt**

FavoriteTvShow adalah sebuah kelas data yang mewakili satu baris data di dalam database Room aplikasi. Kelas ini ditandai sebagai @Entity dengan nama tabel "favorite_tv_shows", yang berarti setiap objek dari kelas ini akan disimpan sebagai entri terpisah di tabel tersebut. Setiap acara TV favorit memiliki id unik sebagai kunci utama, name, posterPath (jalur gambar poster), firstAirDate (tanggal tayang perdana), rating, dan overview (ringkasan). Selain itu, ada juga fungsi ekstensi toTvShow() yang sangat praktis. Fungsi ini memungkinkan untuk dengan mudah mengubah objek FavoriteTvShow menjadi objek TvShow biasa, yang kemungkinan digunakan di bagian lain aplikasi yang menampilkan data acara TV secara umum.

- **FavoriteViewModel.kt**

FavoriteViewModel adalah sebuah ViewModel yang bertugas mengelola data acara TV favorit di aplikasi. ViewModel ini berkomunikasi dengan AppDatabase untuk mendapatkan akses ke FavoriteRepository, yang selanjutnya akan berinteraksi langsung dengan database. Tugas utamanya adalah menyediakan daftar acara TV favorit secara real-time kepada tampilan, sehingga setiap kali ada perubahan di database, tampilan akan otomatis diperbarui. Selain itu, ViewModel ini juga memiliki fungsi untuk mengubah status favorit sebuah acara TV (menambahkannya ke favorit atau menghapusnya) dan memeriksa apakah suatu acara TV sudah difavoritkan atau belum, semuanya dilakukan dengan efisien di latar belakang tanpa menghambat antarmuka pengguna.

- **HomeFragment.kt**

HomeFragment adalah bagian utama antarmuka pengguna aplikasi yang menampilkan daftar acara TV drama. Fragment ini bertugas untuk menyiapkan tampilan dan RecyclerView yang akan menampilkan daftar drama tersebut. Ia menggunakan DramaViewModel untuk mendapatkan data acara TV, dan setiap kali data baru tersedia atau diperbarui, tampilan daftar

akan otomatis ikut berubah. Jika pengguna mengetuk salah satu acara TV dalam daftar, aplikasi akan langsung berpindah ke DetailFragment untuk menunjukkan informasi lebih lengkap tentang acara TV yang dipilih. Selain itu, ada juga tombol khusus yang memungkinkan pengguna untuk dengan mudah berpindah ke layar FavoriteFragment, tempat daftar acara TV favorit disimpan.

- **ListDramaAdapter.kt**

ListDramaAdapter ini adalah komponen penting yang bertugas menampilkan daftar acara TV dalam bentuk daftar yang bisa digulir di aplikasi. Ia mengambil daftar acara TV dan mengatur bagaimana setiap item dalam daftar tersebut akan terlihat. Untuk setiap acara TV, adapter ini akan membuat tampilannya, mengisi data seperti judul dan poster, lalu menyiapkan dua tombol interaktif. Tombol pertama, "Detail", saat diklik akan memicu aksi untuk menampilkan detail lengkap acara TV tersebut di layar lain. Tombol kedua, "Link", akan membuka halaman acara TV terkait di situs The Movie Database melalui browser web perangkat. Adapter ini juga dilengkapi dengan fungsi updateData yang memungkinkan daftar acara TV diperbarui secara efisien kapan pun ada data baru, sehingga tampilan akan selalu segar dan sesuai.

- **MainActivity.kt**

MainActivity adalah aktivitas utama atau bisa dibilang titik masuk pertama kali aplikasi dijalankan. Saat aplikasi dibuka, MainActivity ini akan bertanggung jawab untuk mengatur tata letak utama aplikasi dan secara otomatis menampilkan HomeFragment sebagai tampilan awal. Jika aplikasi ini dibuka lagi setelah sebelumnya ditutup, MainActivity akan memastikan bahwa HomeFragment tidak dibuat ulang secara berlebihan, sehingga pengalaman pengguna tetap lancar.

- **NetworkModule.kt**

NetworkModule adalah sebuah objek yang bertugas untuk menyiapkan segala sesuatu yang dibutuhkan aplikasi agar bisa berkomunikasi dengan API The Movie Database (TMDB) di internet. Modul ini menentukan alamat dasar (BASE_URL) dari API tersebut. Bagian utamanya adalah apiService yang dibuat secara lazy, artinya ia hanya akan dibuat saat

pertama kali dibutuhkan. Di dalamnya, digunakan pustaka Retrofit untuk membangun koneksi ke API TMDB dan Kotlinx Serialization untuk mengubah data dari JSON yang diterima API menjadi objek data yang bisa dimengerti aplikasi. Pengaturan ignoreUnknownKeys = true juga ditambahkan untuk memastikan aplikasi tidak akan error jika ada data yang tidak dikenal dari API. Jadi, NetworkModule ini adalah jembatan utama yang memungkinkan aplikasi mengambil data acara TV dari internet.

- **TmdbApiService.kt**

TmdbApiService ini adalah sebuah antarmuka yang menjelaskan bagaimana aplikasi akan berkomunikasi dengan API The Movie Database (TMDB) untuk mengambil data acara TV. Di sini, didefinisikan satu fungsi utama bernama getChineseTvShows. Fungsi ini adalah permintaan untuk mendapatkan daftar acara TV Tiongkok (with_original_language=zh) yang diurutkan berdasarkan popularitas (popularity.desc), serta sudah tayang antara awal tahun 2020 hingga akhir Mei 2025. Yang terpenting, setiap permintaan harus menyertakan api_key untuk otentikasi. Jadi, TmdbApiService ini seperti sebuah "menu" yang mendefinisikan permintaan apa saja yang bisa aplikasi kirimkan ke server TMDB.

- **TmdbRepository.kt**

TmdbRepository ini adalah sebuah kelas yang berfungsi sebagai jembatan antara ViewModel aplikasi dan TmdbApiService yang berkomunikasi dengan API The Movie Database. Tugas utamanya adalah menyediakan data acara TV yang bersih dan mudah digunakan untuk ViewModel, tanpa perlu ViewModel tahu bagaimana cara sebenarnya data itu diambil dari internet. Di sini, ada fungsi getChineseTvShows() yang saat dipanggil akan langsung meminta data drama Tiongkok dari API. Fungsi ini sudah dilengkapi dengan apiKey yang dibutuhkan untuk akses ke API, serta menentukan rentang tanggal tayang perdana yaitu dari 1 Januari 2023 hingga 31 Mei 2025. Jadi, TmdbRepository ini memastikan bahwa data dari internet bisa didapatkan dengan mudah dan terorganisir untuk bagian aplikasi lainnya.

- **TvShow.kt**

TvShow adalah sebuah kelas data yang merepresentasikan informasi detail dari satu acara TV. Kelas ini dibuat agar mudah dikirim antar komponen Android, berkat anotasi

@Parcelize. Selain itu, dengan anotasi @Serializable, data TvShow ini bisa dengan mudah diubah dari atau ke format seperti JSON, yang umum digunakan saat berkomunikasi dengan API. Setiap objek TvShow memiliki beberapa properti penting seperti id (pengenal unik), name (judul), posterPath (lokasi gambar poster), rating (rata-rata nilai ulasan), firstAirDate (tanggal tayang perdana), dan overview (deskripsi singkat). Anotasi @SerialName digunakan untuk mencocokkan nama properti dengan nama yang diterima dari API, seperti poster_path yang menjadi posterPath di kode.

- **TvShowResponse.kt**

TvShowResponse adalah sebuah kelas data yang berfungsi sebagai wadah untuk menampung hasil dari permintaan API yang berisi daftar acara TV. Kelas ini ditandai dengan @Serializable, yang berarti ia bisa dengan mudah diubah dari atau ke format seperti JSON. Properti utamanya adalah results, yang akan berisi daftar objek TvShow. Jadi, ketika aplikasi meminta data acara TV dari API, respons yang diterima akan langsung dimasukkan ke dalam objek TvShowResponse ini, dan daftar acara TV yang sebenarnya bisa diakses melalui properti results.

- **activity_main.xml**

File activity_main.xml ini adalah tata letak utama (layout) untuk MainActivity di aplikasi. Secara sederhana, ia berfungsi sebagai bingkai kosong yang akan diisi oleh komponen UI yang lebih kecil, yaitu Fragment. Di dalamnya, ada FrameLayout dengan ID fragment_container yang ukurannya memenuhi seluruh layar. FrameLayout ini adalah tempat di mana berbagai Fragment (seperti HomeFragment atau DetailFragment) akan "diletakkan" dan ditampilkan secara dinamis oleh aplikasi. Jadi, activity_main.xml ini adalah fondasi visual yang akan menampung semua tampilan utama aplikasi.

- **fragment_detail.xml**

File fragment_detail.xml ini adalah tata letak (layout) untuk tampilan detail sebuah acara TV yang akan ditampilkan di DetailFragment. Layout ini menggunakan Data Binding, ditandai dengan tag <layout> dan <data>, yang memungkinkan data dari objek TvShow langsung ditampilkan ke elemen-elemen UI tanpa perlu banyak kode di Java/Kotlin. Seluruh konten

detail acara TV diletakkan di dalam ScrollView agar bisa digulir jika informasinya terlalu panjang. Di dalamnya, ada ImageView untuk menampilkan poster acara TV, TextView untuk menampilkan judul, tanggal tayang, dan rating, serta ImageButton untuk tombol favorit. Ada juga dua Button yaitu "Open Link" untuk membuka halaman web acara TV, dan "Back" untuk kembali ke tampilan sebelumnya. Semua elemen ini disusun secara vertikal dan dipusatkan agar mudah dilihat.

- **fragment_favorite.xml**

File fragment_favorite.xml ini adalah tata letak (layout) untuk tampilan daftar acara TV favorit yang akan muncul di FavoriteFragment. Layout ini didesain dengan sebuah bilah judul (header bar) di bagian atas yang berisi tombol kembali dan judul "Favorit". Di bawah bilah judul tersebut, ada RecyclerView yang akan digunakan untuk menampilkan daftar semua acara TV yang sudah ditandai sebagai favorit dalam bentuk daftar yang bisa digulir. Jadi, fragment_favorite.xml ini bertugas untuk mengatur bagaimana tampilan daftar favorit tersebut akan terlihat di layar aplikasi.

- **fragment_home.xml**

File fragment_home.xml ini adalah tata letak (layout) utama untuk tampilan beranda aplikasi yang akan muncul di HomeFragment. Bagian atas layout ini terdapat bilah judul (header bar) berwarna biru yang berisi judul aplikasi dan sebuah tombol. Tombol ini berfungsi untuk mengarahkan pengguna ke layar daftar acara TV favorit. Di bawah bilah judul, terdapat RecyclerView yang akan menampilkan daftar drama Tiongkok yang bisa digulir. Jadi, fragment_home.xml ini bertanggung jawab untuk mengatur bagaimana tampilan beranda aplikasi, lengkap dengan daftar drama dan tombol navigasi ke favorit, akan terlihat di layar.

- **item_drama.xml**

File item_drama.xml ini adalah tata letak (layout) yang digunakan untuk menampilkan setiap satu item dalam daftar acara TV di RecyclerView. Setiap item ditampilkan di dalam sebuah CardView yang memberikan tampilan efek kartu dengan sudut membulat dan sedikit bayangan, membuatnya terlihat lebih modern. Layout ini menggunakan Data Binding, memungkinkan data dari objek TvShow langsung ditampilkan ke elemen UI yang relevan.

Di dalam setiap kartu, terdapat ImageView untuk menampilkan poster acara TV, beberapa TextView untuk judul, tahun tayang, dan rating. Selain itu, ada dua tombol: "Watch Here" yang mengarahkan pengguna ke tautan eksternal, dan "Detail" yang akan membuka layar detail lengkap acara TV tersebut. Ini membuat setiap item daftar menjadi interaktif dan informatif.

## D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

Pemrograman-Mobile/MODUL5 at main · firdakhrns/Pemrograman-Mobile