LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 5

CONNECT TO THE INTERNET

Oleh:

Muhammad Azwin Hakim **NIM. 2310817310012**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE
MODUL 5

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

Nama Praktikan : Muhammad Azwin Hakim
NIM : 2310817310012

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

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

Lanjutkan aplikasi Android 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.ultraman
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
6  import androidx.compose.foundation.layout.padding
7  import androidx.compose.material3.Scaffold
8  import androidx.compose.runtime.Composable
9  import androidx.compose.ui.Modifier
10 import androidx.lifecycle.viewmodel.compose.viewModel
11 import androidx.navigation.NavHostController
12 import androidx.navigation.NavType
13 import androidx.navigation.compose.*
14 import androidx.navigation.navArgument
15 import com.example.ultraman.ui.ViewModel.UltramanViewModel
16 import
   com.example.ultraman.ui.ViewModel.UltramanViewModelFactory
17 import com.example.ultraman.ui.components.BottomBarNavigation
18 import com.example.ultraman.ui.screens.DetailScreen
```

```
19 import com.example.ultraman.ui.screens.FavoriteScreen
20 import com.example.ultraman.ui.screens.ListScreen
21 import com.example.ultraman.ui.theme.UltramanTheme
22
23 class MainActivity : ComponentActivity() {
24     override fun onCreate(savedInstanceState: Bundle?) {
25         super.onCreate(savedInstanceState)
26         setContent {
27             UltramanTheme {
28                 val navController = rememberNavController()
29                 val viewModel: UltramanViewModel = viewModel(
30                     factory                                          =
   UltramanViewModelFactory(applicationContext)
31                 )
32                 UltramanApp(navController, viewModel)
33             }
34         }
35     }
36 }
37
38 @Composable
39 fun UltramanApp(
40     navController: NavHostController,
41     viewModel: UltramanViewModel
42 ) {
43     Scaffold(
44         bottomBar = {
45             BottomBarNavigation(navController = navController)
46         }
47     ) { innerPadding ->
48         NavHost(
49             navController = navController,
50             startDestination = "list",
51             modifier = Modifier.padding(innerPadding)
52         ) {
53             composable("list") {
54                 ListScreen(navController, viewModel)
55             }
56             composable("favorite") {
57                 FavoriteScreen(navController, viewModel)
58             }
59
60             composable(
61                 "detail/{itemId}",
62                 arguments = listOf(navArgument("itemId") { type
   = NavType.IntType })
63             ) { backStackEntry ->
64                 val                    itemId                    =
   backStackEntry.arguments?.getInt("itemId")
65                 DetailScreen(itemId, viewModel)
66             }
67         }
```

```
    }
}
```

2. Model/UltramanItem.kt

Tabel 2. Source Code Jawaban Soal 1

```
package com.example.ultraman.model

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

@Serializable
@Entity(tableName = "ultraman")
data class UltramanItem(
    @PrimaryKey val id: Int,
    val title: String,

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

    @SerialName("poster_path")
    val posterPath: String? = null,

    @SerialName("release_date")
    val releaseDate: String? = null,

    @SerialName("first_air_date")
    val firstAirDate: String? = null,

    val isFavorite: Boolean = false
) {
    val year: String
        get() = releaseDate?.take(4) ?: firstAirDate?.take(4) ?: "????"

    fun getPosterUrl(): String? = posterPath?.let {
        "https://image.tmdb.org/t/p/w500$it"
    }

    fun getTmdbLink(): String {
        return if (!releaseDate.isNullOrBlank()) {
            "https://www.themoviedb.org/movie/$id"
        } else {
            "https://www.themoviedb.org/tv/$id"
        }
    }
}
```

3. Model/SearchResponse.kt

Tabel 3. Source Code Jawaban Soal 1

```
1  package com.example.ultraman.model
2
3  import kotlinx.serialization.SerialName
4  import kotlinx.serialization.Serializable
5
6  @Serializable
7  data class SearchResponse(
8      val results: List<UltramanResponse>
9  )
```

4. Model/UltramanResponse.kt

Tabel 4. Source Code Jawaban Soal 1

```
1  package com.example.ultraman.model
2
3  import kotlinx.serialization.SerialName
4  import kotlinx.serialization.Serializable
5
6  @Serializable
7  data class UltramanResponse(
8      val id: Int,
9      @SerialName("name") val name: String? = null,
10     @SerialName("title") val title: String? = null,
11     val overview: String,
12     @SerialName("poster_path") val posterPath: String?,
13
14     @SerialName("release_date") val releaseDate: String? = null,
15     @SerialName("first_air_date") val firstAirDate: String? =
   null
16 )
```

5. Model/UltramanMapper.kt

Tabel 5. Source Code Jawaban Soal 1

```
1  package com.example.ultraman.model
2
3  fun UltramanResponse.toUltramanItem(): UltramanItem {
4      return UltramanItem(
5          id = id,
6          title = title ?: name ?: "Unknown Title",
7          description = overview,
8          posterPath = posterPath,
9          releaseDate = releaseDate,
10         firstAirDate = firstAirDate
11     )
```

```
}
```

6. Network/ TmdbApiService.kt

Tabel 6. Source Code Jawaban Soal 1

```
package com.example.ultraman.network

import com.example.ultraman.model.SearchResponse
import com.example.ultraman.model.UltramanResponse
import retrofit2.http.GET
import retrofit2.http.Path
import retrofit2.http.Query

interface TmdbApiService {

    @GET("tv/{id}")
    suspend fun getTvDetails(
        @Path("id") id: Int,
        @Query("api_key") apiKey: String
    ): UltramanResponse

    @GET("movie/{id}")
    suspend fun getMovieDetails(
        @Path("id") id: Int,
        @Query("api_key") apiKey: String
    ): UltramanResponse

    @GET("search/multi")
    suspend fun searchUltraman(
        @Query("query") query: String = "Ultraman",
        @Query("api_key") apiKey: String,
        @Query("page") page: Int = 1
    ): SearchResponse

}
```

7. Network/ RetrofitInstance.kt

Tabel 7. Source Code Jawaban Soal 1

```
package com.example.ultraman.network

import
com.jakewharton.retrofit2.converter.kotlinx.serialization.
asConverterFactory
import kotlinx.serialization.json.Json
import okhttp3.MediaType.Companion.toMediaType
import okhttp3.OkHttpClient
import retrofit2.Retrofit

```

```
object RetrofitInstance {
    private const val BASE_URL =
"https://api.themoviedb.org/3/"

    private val json = Json {
        ignoreUnknownKeys = true
    }

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

    val api: TmdbApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(json.asConverterFactory(
"application/json".toMediaType()))
            .client(client)
            .build()
            .create(TmdbApiService::class.java)
    }
}
```

8. Data/ApiResponse.kt

Tabel 8. Source Code Jawaban Soal 1

```
package com.example.ultraman.data

sealed class ApiResponse<out T> {
    data class Success<T>(val data: T) : ApiResponse<T>()
    data class Error(val message: String) :
ApiResponse<Nothing>()
    object Loading : ApiResponse<Nothing>()
}
```

9. Data/ UltramanDao.kt

Tabel 9. Source Code Jawaban Soal 1

```
package com.example.ultraman.data

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import com.example.ultraman.model.UltramanItem
import kotlinx.coroutines.flow.Flow

@Dao
interface UltramanDao {
    @Query("SELECT * FROM ultraman")
    fun getAll(): Flow<List<UltramanItem>>
```

```
14
15      @Insert(onConflict = OnConflictStrategy.REPLACE)
16      suspend fun insertAll(items: List<UltramanItem>)
17
18      @Query("UPDATE ultraman SET isFavorite = :isFav WHERE id =
    :id")
19      suspend fun setFavorite(id: Int, isFav: Boolean)
20
21      @Query("SELECT * FROM ultraman WHERE isFavorite = 1")
22      fun getFavorites(): Flow<List<UltramanItem>>
23
24      @Query("SELECT * FROM ultraman")
25      suspend fun getAllOnce(): List<UltramanItem>
26  }
```

10. Data/UltramanDatabase.kt

Tabel 10. Source Code Jawaban Soal 1

```
1   package com.example.ultraman.data
2
3   import android.content.Context
4   import androidx.room.Database
5   import androidx.room.Room
6   import androidx.room.RoomDatabase
7   import com.example.ultraman.model.UltramanItem
8
9   @Database(entities = [UltramanItem::class], version = 3,
    exportSchema = false)
10  abstract class UltramanDatabase : RoomDatabase() {
11      abstract fun ultramanDao(): UltramanDao
12
13      companion object {
14          @Volatile
15          private var INSTANCE: UltramanDatabase? = null
16
17          fun getDatabase(context: Context): UltramanDatabase {
18              return INSTANCE ?: synchronized(this) {
19                  val instance = Room.databaseBuilder(
20                      context.applicationContext,
21                      UltramanDatabase::class.java,
22                      "ultraman_db"
23                  ).fallbackToDestructiveMigration()
24                  .build()
25                  INSTANCE = instance
26                  instance
27              }
28          }
29      }
30  }
```

11. Repository/UltramanRepository.kt

Tabel 11. Source Code Jawaban Soal 1

```
1   package com.example.ultraman.repository
2
3   import android.util.Log
4   import com.example.ultraman.data.ApiResponse
5   import com.example.ultraman.data.UltramanDao
6   import com.example.ultraman.model.UltramanItem
7   import com.example.ultraman.model.toUltramanItem
8   import com.example.ultraman.network.RetrofitInstance
9   import kotlinx.coroutines.flow.Flow
10  import kotlinx.coroutines.flow.flow
11
12  class UltramanRepository(private val dao: UltramanDao) {
13
14      val ultramanItems: Flow<List<UltramanItem>> = dao.getAll()
15      val      favoriteItems:      Flow<List<UltramanItem>>      =
    dao.getFavorites()
16
17      fun               refreshData(apiKey:               String):
    Flow<ApiResponse<List<UltramanItem>>> = flow {
18          emit(ApiResponse.Loading)
19
20          try {
21              val                 searchResponse                =
    RetrofitInstance.api.searchUltraman(apiKey = apiKey)
22              val newItems = searchResponse.results.mapNotNull {
    it.toUltramanItem() }
23
24              val oldItems = dao.getAllOnce()
25              val mergedItems = newItems.map { newItem ->
26                  val old = oldItems.find { it.id == newItem.id }
27                  newItem.copy(isFavorite  =  old?.isFavorite  ?:
    false)
28              }
29
30              dao.insertAll(mergedItems.take(20))
31              emit(ApiResponse.Success(mergedItems.take(20)))
32
33          } catch (e: Exception) {
34              Log.e("UltramanRepository", "Error: ${e.message}")
35              emit(ApiResponse.Error(e.message    ?:    "Terjadi
    kesalahan"))
36          }
37      }
38
39      suspend fun toggleFavorite(id: Int, currentValue: Boolean)
    {
40          dao.setFavorite(id, !currentValue)
41      }
```

```
42
43      fun getFavorites(): Flow<List<UltramanItem>> {
44          return dao.getFavorites()
45      }
46  }
```

12. ui/ViewModel/UltramanViewModel.kt

Tabel 12. Source Code Jawaban Soal 1

```
1   package com.example.ultraman.ui.ViewModel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.viewModelScope
5   import com.example.ultraman.data.ApiResponse
6   import com.example.ultraman.model.UltramanItem
7   import com.example.ultraman.repository.UltramanRepository
8   import kotlinx.coroutines.flow.MutableStateFlow
9   import kotlinx.coroutines.flow.StateFlow
10  import kotlinx.coroutines.flow.collectLatest
11  import kotlinx.coroutines.launch
12
13  class UltramanViewModel(private val repository:
    UltramanRepository) : ViewModel() {
14
15      private val _ultramanList =
    MutableStateFlow<List<UltramanItem>>(emptyList())
16      val ultramanList: StateFlow<List<UltramanItem>> =
    _ultramanList
17
18      private val _favoriteList =
    MutableStateFlow<List<UltramanItem>>(emptyList())
19      val favoriteList: StateFlow<List<UltramanItem>> =
    _favoriteList
20
21      private val _uiState = MutableStateFlow<ApiResponse
    <List<UltramanItem>>>(ApiResponse.Loading)
22      val uiState: StateFlow<ApiResponse<List
    <UltramanItem>>> = _uiState
23
24      private val _favoriteState =
    MutableStateFlow<ApiResponse<List<UltramanItem>>>
    (ApiResponse.Loading)
25      val favoriteState:
    StateFlow<ApiResponse<List<UltramanItem>>> = _favoriteState
26
27      init {
28          refreshData()
29          observeLocalData()
30      }
31
```

```
32      private fun observeLocalData() {
33          viewModelScope.launch {
34              repository.ultramanItems.collectLatest {
35                  _ultramanList.value = it
36                  _uiState.value = ApiResponse.Success(it)
37              }
38          }
39
40          viewModelScope.launch {
41              try {
42                  repository.favoriteItems.collectLatest {
43                      _favoriteList.value = it
44                      _favoriteState.value =
    ApiResponse.Success(it)
45                  }
46              } catch (e: Exception) {
47                  _favoriteState.value =
    ApiResponse.Error(e.message ?: "Unknown error")
48              }
49          }
50      }
51
52      fun refreshData(apiKey: String =
    "5747e2247c220f5aac1c7654bfa77d4b") {
53          viewModelScope.launch {
54              repository.refreshData(apiKey).collectLatest {
55                  _uiState.value = it
56              }
57          }
58      }
59
60      fun getItemById(id: Int): UltramanItem? {
61          return _ultramanList.value.find { it.id == id }
62      }
63
64      fun toggleFavorite(item: UltramanItem) {
65          viewModelScope.launch {
66              repository.toggleFavorite(item.id, item.isFavorite)
67          }
68      }
69  }
```

13. ui/ViewModel/UltramanViewModelFactory.kt

Tabel 13. Source Code Jawaban Soal 1

```
1   package com.example.ultraman.ui.ViewModel
2
3   import android.content.Context
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.ViewModelProvider
```

```
6  import com.example.ultraman.data.UltramanDatabase
7  import com.example.ultraman.repository.UltramanRepository
8
9  class UltramanViewModelFactory(private val context: Context) :
   ViewModelProvider.Factory {
10     override fun <T : ViewModel> create(modelClass: Class<T>):
   T {
11         val                          dao                          =
   UltramanDatabase.getDatabase(context).ultramanDao()
12         val repository = UltramanRepository(dao)
13         @Suppress("UNCHECKED_CAST")
14         return UltramanViewModel(repository) as T
15     }
16 }
```

14. ui/screens/ListScreen.kt

Tabel 14. Source Code Jawaban Soal 1

```
1  package com.example.ultraman.ui.screens
2
3  import android.content.Intent
4  import androidx.compose.foundation.background
5  import androidx.compose.foundation.layout.*
6  import androidx.compose.foundation.lazy.LazyColumn
7  import androidx.compose.foundation.lazy.items
8  import androidx.compose.foundation.shape.CircleShape
9  import androidx.compose.foundation.shape.RoundedCornerShape
10 import androidx.compose.material.icons.Icons
11 import androidx.compose.material.icons.filled.Favorite
12 import androidx.compose.material.icons.filled.FavoriteBorder
13 import androidx.compose.material3.*
14 import androidx.compose.runtime.Composable
15 import androidx.compose.runtime.collectAsState
16 import androidx.compose.ui.Alignment
17 import androidx.compose.ui.Modifier
18 import androidx.compose.ui.draw.clip
19 import androidx.compose.ui.graphics.Color
20 import androidx.compose.ui.layout.ContentScale
21 import androidx.compose.ui.text.SpanStyle
22 import androidx.compose.ui.text.buildAnnotatedString
23 import androidx.compose.ui.text.font.FontWeight
24 import androidx.compose.ui.text.style.TextOverflow
25 import androidx.compose.ui.text.withStyle
26 import androidx.compose.ui.unit.dp
27 import androidx.core.net.toUri
28 import androidx.navigation.NavHostController
29 import com.example.ultraman.ui.ViewModel.UltramanViewModel
30 import com.example.ultraman.data.ApiResponse
31
32
```

```
@Composable
fun ListScreen(
    navController: NavHostController,
    viewModel: UltramanViewModel,
    modifier: Modifier = Modifier
) {
    val uiState = viewModel.uiState.collectAsState().value

    when (uiState) {
        is ApiResponse.Loading -> {
            Box(
                modifier = Modifier.fillMaxSize(),
                contentAlignment = Alignment.Center
            ) {
                CircularProgressIndicator()
            }
        }

        is ApiResponse.Error -> {
            Box(
                modifier                                        =
Modifier.fillMaxSize().padding(16.dp),
                contentAlignment = Alignment.Center
            ) {
                Text("Terjadi  kesalahan:  ${uiState.message}",
color = Color.Red)
            }
        }

        is ApiResponse.Success -> {
            val ultramanList = uiState.data
            LazyColumn(
                modifier = modifier
                    .fillMaxSize()
                    .padding(8.dp)
            ) {
                items(ultramanList) { item ->
                    Card(
                        shape = RoundedCornerShape(16.dp),
                        elevation                                   =
CardDefaults.cardElevation(8.dp),
                        modifier = Modifier
                            .fillMaxWidth()
                            .padding(vertical = 8.dp)
                    ) {
                        Row(
                            modifier = Modifier
                                .padding(16.dp)
                                .fillMaxWidth()
                        ) {
                            Box(
                                modifier = Modifier
```

```
82                                      .height(150.dp)
83                                      .width(100.dp)
84                              ) {
85                                  GlideImage(
86                                      imageUrl                =
    item.getPosterUrl().toString(),
87                                      contentDescription      =
    item.title,
88                                      modifier = Modifier
89                                          .matchParentSize()
90
    .clip(RoundedCornerShape(12.dp)),
91                                      contentScale            =
    ContentScale.Crop
92                                  )
93
94                                  IconButton(
95                                      onClick        =        {
    viewModel.toggleFavorite(item) },
96                                      modifier = Modifier
97
    .align(Alignment.TopStart)
98                                          .padding(4.dp)
99
    .background(Color.White.copy(alpha    =    0.5f),    shape    =
    CircleShape)
100                                 ) {
101                                     Icon(
102                                         imageVector    =    if
    (item.isFavorite)          Icons.Default.Favorite          else
    Icons.Default.FavoriteBorder,
103                                         contentDescription   =
    "Favorite",
104                                         tint = Color.Red
105                                     )
106                                 }
107                             }
108
109                             Spacer(modifier                 =
    Modifier.width(16.dp))
110
111                             Column(
112                                 modifier = Modifier.weight(1f)
113                             ) {
114                                 Row(
115                                     verticalAlignment       =
    Alignment.CenterVertically,
116                                     horizontalArrangement   =
    Arrangement.SpaceBetween,
117                                     modifier                =
    Modifier.fillMaxWidth()
118                                 ) {
```

```
119                                     Text(
120                                         text                    =
    item.title.take(20) + if (item.title.length > 20) "…" else "",
121                                         style                   =
    MaterialTheme.typography.titleMedium,
122                                         fontWeight              =
    FontWeight.Bold
123                                     )
124                                     Text(
125                                         text = item.year,
126                                         style                   =
    MaterialTheme.typography.bodySmall,
127                                         color = Color.Gray
128                                     )
129                                 }
130
131                                 Spacer(modifier                 =
    Modifier.height(6.dp))
132
133                                 Row(
134                                     modifier                    =
    Modifier.fillMaxWidth()
135                                 ) {
136                                     Text(
137                                         text = "Info: ",
138                                         style                   =
    MaterialTheme.typography.bodySmall,
139                                         fontWeight              =
    FontWeight.Bold
140                                     )
141                                     Text(
142                                         text                    =
    item.description,
143                                         style                   =
    MaterialTheme.typography.bodySmall,
144                                         maxLines = 4,
145                                         overflow                =
    TextOverflow.Ellipsis
146                                     )
147                                 }
148
149                                 Spacer(modifier                 =
    Modifier.height(12.dp))
150
151                                 Row(
152                                     horizontalArrangement       =
    Arrangement.spacedBy(30.dp)
153                                 ) {
154                                     Button(
155                                         onClick = {
156                                             val    intent       =
    Intent(Intent.ACTION_VIEW, item.getTmdbLink().toUri())
```

```
157
    navController.context.startActivity(intent)
158                                         },
159                                         modifier              =
    Modifier.weight(1f)
160                                     ) {
161                                         Text("More info")
162                                     }
163
164                                     Button(
165                                         onClick = {
166
    navController.navigate("detail/${item.id}")
167                                         },
168                                         modifier              =
    Modifier.weight(1f)
169                                     ) {
170                                         Text("Detail")
171                                     }
172                                 }
173                             }
174                         }
175                     }
176                 }
177             }
178         }
179     }
180 }
```

15. ui/screens/DetailScreen.kt

Tabel 15. Source Code Jawaban Soal 1

```
1   package com.example.ultraman.ui.screens
2
3   import android.util.Log
4   import androidx.compose.foundation.layout.*
5   import androidx.compose.foundation.rememberScrollState
6   import androidx.compose.foundation.shape.RoundedCornerShape
7   import androidx.compose.foundation.verticalScroll
8   import androidx.compose.material3.MaterialTheme
9   import androidx.compose.material3.Text
10  import androidx.compose.runtime.Composable
11  import androidx.compose.ui.Modifier
12  import androidx.compose.ui.draw.clip
13  import androidx.compose.ui.layout.ContentScale
14  import androidx.compose.ui.text.font.FontWeight
15  import androidx.compose.ui.text.style.TextAlign
16  import androidx.compose.ui.unit.dp
17  import com.example.ultraman.ui.ViewModel.UltramanViewModel
18
```

```
19 @Composable
20 fun DetailScreen(itemId: Int?, viewModel: UltramanViewModel) {
21     val item = itemId?.let { viewModel.getItemById(it) }
22     Log.d("DetailScreen", "Navigasi ke DetailScreen untuk ID:
   $itemId - ${item?.title}")
23
24     item?.let {
25         Column(
26             modifier = Modifier
27                 .fillMaxSize()
28                 .padding(16.dp)
29
   .padding(WindowInsets.systemBars.asPaddingValues())
30                 .verticalScroll(rememberScrollState())
31         ) {
32             it.getPosterUrl()?.let { posterUrl ->
33                 GlideImage(
34                     imageUrl = posterUrl,
35                     contentDescription = it.title,
36                     modifier = Modifier
37                         .fillMaxWidth()
38                         .height(600.dp)
39                         .clip(RoundedCornerShape(16.dp)),
40                     contentScale = ContentScale.Crop
41                 )
42             }
43
44             Spacer(modifier = Modifier.height(12.dp))
45
46             Text(
47                 text = it.title,
48                 style = MaterialTheme.typography.headlineSmall,
49                 fontWeight = FontWeight.Bold
50             )
51             Spacer(modifier = Modifier.height(8.dp))
52
53             Text( text = "Tahun: ",
54                   fontWeight = FontWeight.Bold
55             )
56             Text(it.year)
57             Text(
58                 text = "Sinopsis:",
59                 style = MaterialTheme.typography.titleSmall,
60                 fontWeight = FontWeight.Bold
61             )
62             Text(
63                 text = it.description,
64                 style = MaterialTheme.typography.bodyMedium,
65                 textAlign = TextAlign.Justify
66             )
67         }
68     } ?: run {
```

```
69          Column(
70              modifier = Modifier
71                  .fillMaxSize()
72                  .padding(16.dp),
73              verticalArrangement = Arrangement.Center
74          ) {
75              Text(
76                  text = "Data tidak ditemukan.",
77                  style = MaterialTheme.typography.bodyLarge,
78                  modifier = Modifier.fillMaxWidth(),
79                  textAlign = TextAlign.Center
80              )
81          }
82      }
83  }
```

16. ui/screens/favoritescreen.kt

Tabel 16. Source Code Jawaban Soal 1

```
1   package com.example.ultraman.ui.screens
2
3   import android.content.Intent
4   import androidx.compose.foundation.background
5   import androidx.compose.foundation.layout.*
6   import androidx.compose.foundation.lazy.LazyColumn
7   import androidx.compose.foundation.lazy.items
8   import androidx.compose.foundation.shape.CircleShape
9   import androidx.compose.foundation.shape.RoundedCornerShape
10  import androidx.compose.material.icons.Icons
11  import androidx.compose.material.icons.filled.Favorite
12  import androidx.compose.material.icons.filled.FavoriteBorder
13  import androidx.compose.material3.*
14  import androidx.compose.runtime.Composable
15  import androidx.compose.runtime.collectAsState
16  import androidx.compose.ui.Alignment
17  import androidx.compose.ui.Modifier
18  import androidx.compose.ui.draw.clip
19  import androidx.compose.ui.graphics.Color
20  import androidx.compose.ui.layout.ContentScale
21  import androidx.compose.ui.text.font.FontWeight
22  import androidx.compose.ui.text.style.TextOverflow
23  import androidx.compose.ui.unit.dp
24  import androidx.core.net.toUri
25  import androidx.navigation.NavHostController
26  import com.example.ultraman.ui.ViewModel.UltramanViewModel
27  import com.example.ultraman.data.ApiResponse
28  import com.example.ultraman.ui.screens.GlideImage
29
30  @Composable
31  fun FavoriteScreen(
```

```
32      navController: NavHostController,
33      viewModel: UltramanViewModel,
34      modifier: Modifier = Modifier
35  ) {
36      val                          uiState                                =
    viewModel.favoriteState.collectAsState().value
37
38      when (uiState) {
39          is ApiResponse.Loading -> {
40              Box(
41                  modifier = Modifier.fillMaxSize(),
42                  contentAlignment = Alignment.Center
43              ) {
44                  CircularProgressIndicator()
45              }
46          }
47
48          is ApiResponse.Error -> {
49              Box(
50                  modifier                                               =
    Modifier.fillMaxSize().padding(16.dp),
51                  contentAlignment = Alignment.Center
52              ) {
53                  Text("Terjadi  kesalahan:  ${uiState.message}",
    color = Color.Red)
54              }
55          }
56
57          is ApiResponse.Success -> {
58              val favoriteList = uiState.data
59
60              if (favoriteList.isEmpty()) {
61                  Box(
62                      modifier = Modifier.fillMaxSize(),
63                      contentAlignment = Alignment.Center
64                  ) {
65                      Text("Belum ada yang difavoritkan", color
    = Color.Gray)
66                  }
67              } else {
68                  LazyColumn(
69                      modifier = modifier
70                          .fillMaxSize()
71                          .padding(8.dp)
72                  ) {
73                      items(favoriteList) { item ->
74                          Card(
75                              shape = RoundedCornerShape(16.dp),
76                              elevation                              =
    CardDefaults.cardElevation(8.dp),
77                              modifier = Modifier
78                                  .fillMaxWidth()
```

```
79                                      .padding(vertical = 8.dp)
80                              ) {
81                                  Row(
82                                      modifier = Modifier
83                                          .padding(16.dp)
84                                          .fillMaxWidth()
85                                  ) {
86                                      Box(
87                                          modifier = Modifier
88                                              .height(150.dp)
89                                              .width(100.dp)
90                                      ) {
91                                          GlideImage(
92                                              imageUrl              =
    item.getPosterUrl().toString(),
93                                              contentDescription   =
    item.title,
94                                              modifier = Modifier
95                                                  .matchParentSize()
96
    .clip(RoundedCornerShape(12.dp)),
97                                              contentScale         =
    ContentScale.Crop
98                                          )
99
100                                         IconButton(
101                                             onClick      =      {
    viewModel.toggleFavorite(item) },
102                                             modifier = Modifier
103 .align(Alignment.TopStart)
                                                    .padding(4.dp)
104
105 .background(Color.White.copy(alpha    =    0.5f),    shape    =
    CircleShape)
106                                         ) {
107                                             Icon(
108                                                 imageVector  =  if
    (item.isFavorite)          Icons.Default.Favorite          else
    Icons.Default.FavoriteBorder,
109                                                 contentDescription
    = "Favorite",
110                                                 tint = Color.Red
111                                             )
112                                         }
113                                     }
114
115                                     Spacer(modifier              =
    Modifier.width(16.dp))
116
117                                     Column(
```

```
118                                     modifier                =
    Modifier.weight(1f)
119                                 ) {
120                                     Row(
121                                         verticalAlignment    =
    Alignment.CenterVertically,
122                                         horizontalArrangement
    = Arrangement.SpaceBetween,
123                                         modifier             =
    Modifier.fillMaxWidth()
124                                     ) {
125                                         Text(
126                                             text             =
    item.title.take(20) + if (item.title.length > 20) "…" else "",
127                                             style            =
    MaterialTheme.typography.titleMedium,
128                                             fontWeight       =
    FontWeight.Bold
129                                         )
130                                         Text(
131                                             text = item.year,
132                                             style            =
    MaterialTheme.typography.bodySmall,
133                                             color = Color.Gray
134                                         )
135                                     }
136
137                                     Spacer(modifier          =
    Modifier.height(6.dp))
138
139                                     Row(
140                                         modifier             =
    Modifier.fillMaxWidth()
141                                     ) {
142                                         Text(
143                                             text = "Info: ",
144                                             style            =
    MaterialTheme.typography.bodySmall,
145                                             fontWeight       =
    FontWeight.Bold
146                                         )
147                                         Text(
148                                             text             =
    item.description,
149                                             style            =
    MaterialTheme.typography.bodySmall,
150                                             maxLines = 4,
151                                             overflow         =
    TextOverflow.Ellipsis
152                                         )
153                                     }
154
```

```
155                                     Spacer(modifier          =
    Modifier.height(12.dp))
156
157                                     Row(
158                                         horizontalArrangement
    = Arrangement.spacedBy(30.dp)
159                                     ) {
160                                         Button(
161                                             onClick = {
162                                                 val  intent  =
    Intent(Intent.ACTION_VIEW, item.getTmdbLink().toUri())
163
    navController.context.startActivity(intent)
164                                             },
165                                             modifier         =
    Modifier.weight(1f)
166                                         ) {
167                                             Text("More info")
168                                         }
169
170                                         Button(
171                                             onClick = {
172
    navController.navigate("detail/${item.id}")
173                                             },
174                                             modifier         =
    Modifier.weight(1f)
175                                         ) {
176                                             Text("Detail")
177                                         }
178                                     }
179                                 }
180                             }
181                         }
182                     }
183                 }
184             }
185         }
186     }
187 }
```

17. ui/screens/GlideImage.kt

Tabel 17. Source Code Jawaban Soal 1

```
1  package com.example.ultraman.ui.screens
2
3  import android.widget.ImageView
4  import androidx.compose.runtime.Composable
5  import androidx.compose.ui.Modifier
6  import androidx.compose.ui.platform.LocalContext
```

```
7  import androidx.compose.ui.viewinterop.AndroidView
8  import com.bumptech.glide.Glide
9  import androidx.compose.ui.layout.ContentScale
10
11 @Composable
12 fun GlideImage(
13     imageUrl: String,
14     contentDescription: String?,
15     modifier: Modifier = Modifier,
16     contentScale: ContentScale = ContentScale.Crop
17 ) {
18     val context = LocalContext.current
19     AndroidView(
20         factory = {
21             ImageView(context).apply {
22                 scaleType = when (contentScale) {
23                     ContentScale.Crop                          ->
   ImageView.ScaleType.CENTER_CROP
24                     ContentScale.Fit                           ->
   ImageView.ScaleType.FIT_CENTER
25                     else -> ImageView.ScaleType.CENTER_CROP
26                 }
27                 contentDescription?.let                         {
   this.contentDescription = it }
28             }
29         },
30         update = {
31             Glide.with(context)
32                 .load(imageUrl)
33                 .into(it)
34         },
35         modifier = modifier
36     )
37 }
```

18. ui/components/BottomBarNavigation.kt

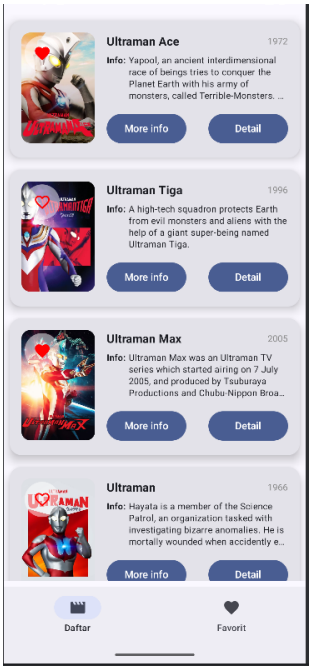
Tabel 18. Source Code Jawaban Soal 1

```
1  package com.example.ultraman.ui.components
2
3  import androidx.compose.material.icons.Icons
4  import androidx.compose.material.icons.filled.Favorite
5  import androidx.compose.material.icons.filled.Movie
6  import androidx.compose.material3.*
7  import androidx.compose.runtime.Composable
8  import androidx.compose.ui.graphics.vector.ImageVector
9  import androidx.navigation.NavController
10 import androidx.navigation.compose.currentBackStackEntryAsState
11
12 data class BottomNavItem(
```

```
13      val title: String,
14      val icon: ImageVector,
15      val route: String
16  )
17
18  val bottomNavItems = listOf(
19      BottomNavItem("Daftar", Icons.Default.Movie, "list"),
20      BottomNavItem("Favorit",            Icons.Default.Favorite,
    "favorite")
21  )
22
23  @Composable
24  fun BottomBarNavigation(navController: NavController) {
25      NavigationBar {
26          val                  navBackStackEntry                  =
    navController.currentBackStackEntryAsState().value
27          val                    currentRoute                     =
    navBackStackEntry?.destination?.route
28
29          bottomNavItems.forEach { item ->
30              NavigationBarItem(
31                  selected = currentRoute == item.route,
32                  onClick = {
33                      if (currentRoute != item.route) {
34                          navController.navigate(item.route) {
35                              popUpTo("list") { inclusive = false
    }
36                              launchSingleTop = true
37                          }
38                      }
39                  },
40                  icon   =   {   Icon(imageVector   =   item.icon,
    contentDescription = item.title) },
41                  label = { Text(item.title) }
42              )
43          }
44      }
45  }
```

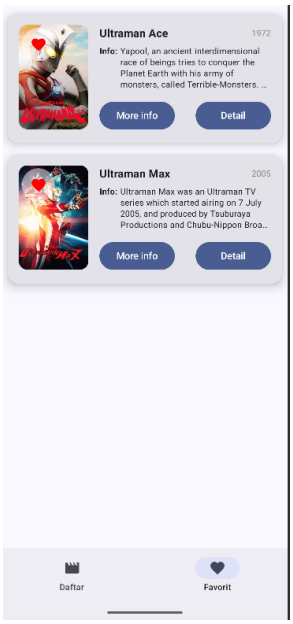
B. Output Program

- Tampilan halaman daftar film ultraman :



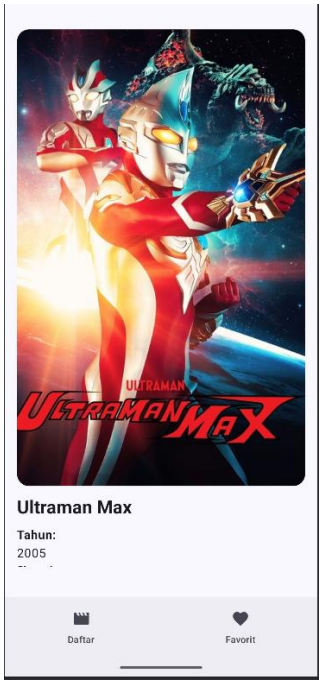
Gambar 1. Screenshot Hasil Jawaban Soal 1

- Tampilan Halaman film favorit (menampilkan film ultraman yang ditandai favorit) :



Gambar 2. Screenshot Hasil Jawaban Soal 1

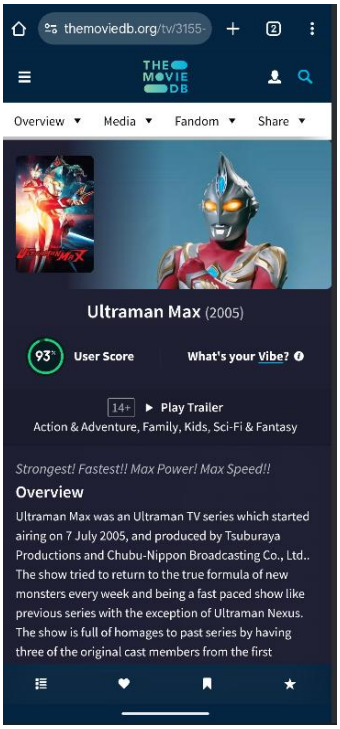
- Tampilan Halaman Detail :



Gambar 3. Screenshot Hasil Jawaban Soal 1

Gambar 4. Screenshot Hasil Jawaban Soal 1

- Ketika Klik More Info :



Gambar 5. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt

Di dalam MainActivity.kt, saya membuat struktur utama aplikasi Android bertema Ultraman dengan menggunakan Jetpack Compose dan sistem navigasi berbasis NavController. Pertama, saya extend ComponentActivity, lalu di dalam onCreate, saya memanggil setContent untuk menyusun UI utama aplikasi dengan menggunakan UltramanTheme yang telah saya definisikan sebelumnya.

Untuk kebutuhan navigasi, saya menggunakan rememberNavController() dan saya siapkan UltramanViewModel lewat viewModel() dengan custom factory UltramanViewModelFactory agar bisa mengakses context dari aplikasi (diperlukan untuk Room database). Selanjutnya, saya panggil fungsi UltramanApp() dan mengoperkan navController dan viewModel ke dalamnya.

Di dalam fungsi UltramanApp, saya menggunakan Scaffold untuk menyusun layout dasar Compose dengan BottomBarNavigation sebagai komponen bottomBar-nya. Dengan begitu, pengguna bisa berpindah antar halaman utama aplikasi menggunakan bottom navigation.

Kemudian, saya atur navigasi menggunakan NavHost. Start destination-nya saya tetapkan ke "list" agar aplikasi menampilkan daftar Ultraman saat pertama kali dibuka. Saya definisikan tiga rute utama:

- "list" Menampilkan ListScreen, yaitu daftar semua serial dan film Ultraman.
- "favorite" Menampilkan FavoriteScreen, yang berisi daftar tontonan yang ditandai sebagai favorit oleh pengguna.
- "detail/{itemId}" Menampilkan DetailScreen berdasarkan ID item yang diklik. Rute ini menggunakan parameter itemId bertipe Int, dan saya ambil nilainya dari backStackEntry.arguments.

2. UltramanItem.kt

Pada file UltramanItem.kt, saya mendefinisikan sebuah data class bernama UltramanItem yang merepresentasikan satu entitas Ultraman, baik itu film maupun serial TV, dan ini digunakan di seluruh aplikasi saya. Kelas ini saya tandai dengan anotasi @Entity dari Room karena datanya juga saya simpan secara lokal ke database, dan saya beri nama tabelnya "ultraman". Saya menggunakan @PrimaryKey untuk menetapkan id sebagai kunci utama, sesuai dengan ID dari TMDB.

Karena saya juga mengambil data dari TMDB API, saya menggunakan anotasi @Serializable dari Kotlinx Serialization, dan saya map field JSON seperti "overview", "poster_path", "release_date", dan "first_air_date" ke properti Kotlin menggunakan @SerialName, agar bisa langsung diparsing otomatis.

Properti isFavorite saya tambahkan sebagai penanda apakah item ini sudah ditandai favorit oleh user atau belum, dengan default false. Lalu, saya buat properti turunan year untuk mengambil tahun rilis, baik dari releaseDate (jika itu film) atau firstAirDate (jika itu serial), dan kalau dua-duanya kosong, akan tampil "????".

Selain itu, saya juga buat dua fungsi utilitas: getPosterUrl() yang menghasilkan URL lengkap ke gambar poster dari TMDB, dan getTmdbLink() yang menentukan link ke halaman TMDB dari item tersebut, membedakan antara movie dan TV berdasarkan apakah releaseDate ada atau tidak.

3. SearchResponse.kt

Pada file SearchResponse.kt, saya membuat sebuah data class bernama SearchResponse yang digunakan untuk memetakan respons JSON dari API pencarian TMDB. Saya beri anotasi @Serializable agar bisa langsung diparsing menggunakan Kotlinx Serialization.

Respons dari TMDB untuk endpoint pencarian biasanya berupa objek JSON yang memiliki properti results, yang isinya adalah daftar hasil pencarian dalam bentuk array. Di sini saya map properti results tersebut ke properti results di Kotlin, yang bertipe List<UltramanResponse>, karena masing-masing item hasil pencarian nantinya akan direpresentasikan oleh kelas UltramanResponse.kt.

Dengan begini, saat saya melakukan request pencarian, saya bisa langsung parsing respons JSON-nya menjadi objek SearchResponse, dan mengambil daftar hasil pencariannya lewat properti results.

4. UltramanResponse.kt

Pada file UltramanResponse.kt, saya membuat data class UltramanResponse sebagai representasi dari satu item hasil respons API TMDB, baik itu berupa film maupun serial. Saya beri anotasi @Serializable agar data ini bisa langsung diparsing dari JSON menggunakan Kotlinx Serialization.

Properti id adalah ID unik dari TMDB yang selalu tersedia. Karena TMDB membedakan antara film (title) dan serial TV (name), saya siapkan kedua field tersebut sebagai properti opsional (String?) dan memetakannya ke JSON dengan @SerialName. Dengan begitu, saya bisa menangani kedua jenis konten tanpa harus membuat dua data class terpisah.

Field overview adalah deskripsi dari film atau serial, sedangkan poster_path saya map ke posterPath yang berisi path gambar poster. Lalu saya juga menambahkan dua tanggal berbeda: release_date (untuk film) dan first_air_date (untuk serial), yang masing-masing saya map ke releaseDate dan firstAirDate.

Data class ini saya gunakan sebagai struktur mentah yang langsung dihasilkan dari API sebelum diubah ke model utama aplikasi saya, yaitu UltramanItem. Biasanya proses konversinya dilakukan melalui mapper.

5. UltramanMapper.kt

Pada file ini, saya membuat sebuah ekstensi fungsi bernama toUltramanItem() bertugas mengonversi objek UltramanResponse yang merupakan hasil mentah dari API TMDB menjadi UltramanItem, yaitu model utama yang saya gunakan di seluruh aplikasi, termasuk untuk penyimpanan di Room dan penampilan di UI.

Dalam fungsi ini, saya mengambil properti id, overview, posterPath, releaseDate, dan firstAirDate langsung dari UltramanResponse dan meneruskannya apa adanya ke dalam UltramanItem. Untuk bagian title, karena TMDB kadang menggunakan title (untuk film) dan kadang name (untuk serial), saya prioritaskan nilai dari title, lalu fallback ke name, dan jika keduanya tidak tersedia, saya isi dengan string default "Unknown Title".

Fungsi ini saya buat agar proses konversi data menjadi lebih bersih, modular, dan bisa digunakan berulang kali setiap kali saya menerima data dari API. Dengan begitu, saya bisa menjaga agar UI dan database hanya berinteraksi dengan satu model data yang konsisten, yaitu UltramanItem.

6. TmdbApiService.kt

Di file TmdbApiService.kt, saya mendefinisikan interface TmdbApiService sebagai kontrak komunikasi antara aplikasi saya dan TMDB API menggunakan Retrofit. Interface ini berisi deklarasi beberapa endpoint yang saya butuhkan untuk mengambil data terkait Ultraman, baik itu berupa film, serial, maupun hasil pencarian gabungan.

Fungsi getTvDetails() dan getMovieDetails() masing-masing digunakan untuk mengambil detail lengkap dari sebuah TV show atau film berdasarkan ID-nya. Saya gunakan anotasi @GET("tv/{id}") dan @GET("movie/{id}") dengan parameter dinamis {id} yang diisi lewat anotasi @Path. Selain itu, karena TMDB memerlukan API key, saya sertakan parameter @Query("api_key") agar bisa dikirimkan sebagai query parameter dalam setiap request. Respons dari keduanya akan langsung dipetakan ke objek UltramanResponse.

Sementara itu, fungsi searchUltraman() saya buat untuk melakukan pencarian dengan query default "Ultraman", namun tetap fleksibel karena bisa diganti dengan kata kunci lain. Saya pakai endpoint search/multi karena endpoint ini bisa mencari baik film maupun TV sekaligus. Parameter page saya beri default 1 untuk mendukung pagination nanti kalau dibutuhkan. Respons dari fungsi ini saya petakan ke SearchResponse, yang berisi daftar UltramanResponse.

Interface ini menjadi pondasi utama komunikasi jaringan di aplikasi saya, dan akan diimplementasikan lewat instance Retrofit untuk mengambil data dari TMDB.

7. RetrofitInstance.kt

Di file RetrofitInstance.kt, saya membuat sebuah singleton object bernama RetrofitInstance yang berfungsi untuk menyediakan instance Retrofit yang sudah dikonfigurasi dengan baik untuk berkomunikasi dengan TMDB API. Tujuannya supaya saya bisa menggunakan satu instance Retrofit secara global di seluruh aplikasi tanpa perlu membuat ulang setiap kali.

Pertama, saya tentukan BASE_URL ke "https://api.themoviedb.org/3/", yaitu base URL resmi dari TMDB API versi 3. Kemudian, saya buat konfigurasi Json dari Kotlinx Serialization dengan opsi ignoreUnknownKeys = true. Pengaturan ini penting agar aplikasi tidak error saat TMDB mengirim field yang tidak saya definisikan dalam model Kotlin — ini membuat parsing jadi lebih fleksibel.

Untuk client-nya, saya gunakan OkHttpClient default tanpa konfigurasi tambahan. Tapi kalau nanti perlu logging atau interceptor untuk debugging, saya bisa tambahkan di sini.

Lalu, saya buat instance Retrofit menggunakan Retrofit.Builder(), di mana saya pasang baseUrl, tambahkan converter dari Kotlinx Serialization (asConverterFactory()), dan set client sebagai HTTP client-nya. Setelah itu, saya panggil .create(TmdbApiService::class.java) untuk menghasilkan implementasi dari interface API yang sudah saya buat sebelumnya.

Akhirnya, properti api dideklarasikan menggunakan by lazy, jadi instance Retrofit hanya dibuat saat pertama kali digunakan. Dengan pendekatan ini, saya bisa dengan mudah mengakses RetrofitInstance.api di mana pun untuk memanggil endpoint TMDB secara efisien dan terpusat.

8. ApiResponse.kt

Pada file ApiResponse.kt, saya membuat sebuah kelas sealed bernama ApiResponse yang digunakan sebagai pembungkus hasil dari operasi jaringan (network request) di aplikasi saya. Tujuan saya membuat ini adalah untuk menangani status data secara reaktif baik saat data sedang dimuat (Loading), berhasil diambil (Success), atau gagal (Error) dalam satu struktur yang konsisten.

Karena ini adalah sealed class, saya bisa memastikan bahwa hanya ada tiga kemungkinan bentuk respons:

- Success<T> berisi data yang berhasil diambil, dibungkus dalam properti data. Ini bertipe generic agar bisa digunakan untuk berbagai jenis data, misalnya List<UltramanItem> atau objek tunggal seperti UltramanItem.
- Error digunakan ketika terjadi kegagalan saat mengambil data dari API. Saya hanya butuh pesan error-nya, jadi saya simpan dalam properti message.
- Loading merepresentasikan status saat request masih berjalan — biasanya saya pakai ini untuk menampilkan indikator loading di UI.

Dengan pola ini, saya bisa mengelola UI berdasarkan state secara lebih bersih di ViewModel dan Composable, misalnya menampilkan progress bar saat ApiResponse.Loading, menampilkan daftar saat Success, dan pesan kesalahan saat Error.

Pendekatan ini juga mempermudah penggunaan Flow atau LiveData karena hanya perlu mengobservasi satu objek respons yang mencakup semua kemungkinan.

9. UltramanDao.kt

Pada file UltramanDao.kt, saya mendefinisikan interface UltramanDao yang menjadi data access object (DAO) untuk Room Database. Interface ini berisi fungsi-fungsi yang saya gunakan untuk mengelola data UltramanItem yang disimpan secara lokal di SQLite melalui Room. Dengan kata lain, inilah jembatan antara database dan layer ViewModel/repository.

Pertama, getAll() saya gunakan untuk mengambil semua data Ultraman dari tabel ultraman dan membungkus hasilnya dalam Flow<List<UltramanItem>>, sehingga saya bisa mengobservasi data ini secara reaktif di UI, cocok untuk penggunaan Jetpack Compose.

Fungsi insertAll() dipakai untuk menyimpan daftar Ultraman ke database. Saya beri strategi OnConflictStrategy.REPLACE, artinya kalau ada data dengan ID yang sama, maka akan ditimpa. Ini penting untuk sinkronisasi dengan API, supaya data lama yang sudah kadaluarsa bisa diperbarui.

Kemudian setFavorite() adalah fungsi yang saya gunakan untuk mengubah status favorit dari sebuah item berdasarkan ID-nya. Fungsi ini penting untuk fitur “tandai sebagai favorit” yang ada di aplikasi.

Selanjutnya, getFavorites() mengambil hanya data yang isFavorite = 1, dan membungkus hasilnya dalam Flow juga, agar UI bisa menampilkan daftar favorit secara real-time.

Terakhir, getAllOnce() adalah versi non-reaktif dari getAll(), hasilnya langsung berupa List<UltramanItem> biasa. Ini saya gunakan ketika hanya butuh snapshot satu kali, misalnya saat sinkronisasi awal, tanpa harus mengamati data terus-menerus.

10. UltramanDatabase.kt

Di file UltramanDatabase.kt, saya membuat kelas abstrak UltramanDatabase yang merupakan implementasi dari RoomDatabase, dan menjadi inti dari database lokal aplikasi saya. Di sinilah saya mendefinisikan skema database dan menyediakan akses ke DAO (UltramanDao).

Anotasi @Database saya gunakan untuk memberitahu Room bahwa ini adalah database dan saya tentukan:

- entities = [UltramanItem::class], artinya hanya ada satu tabel, yaitu tabel ultraman yang berbasis dari data class UltramanItem.
- version = 3, yang berarti ini versi ketiga dari database. Kalau ada perubahan skema, versi ini harus ditingkatkan.
- exportSchema = false, karena saya tidak butuh Room menghasilkan file JSON skema.

Fungsi abstrak ultramanDao() akan diimplementasikan otomatis oleh Room, dan memungkinkan saya mengakses semua fungsi yang saya definisikan di UltramanDao.

Lalu di dalam companion object, saya buat singleton pattern agar hanya ada satu instance UltramanDatabase selama aplikasi berjalan. Saya gunakan @Volatile dan synchronized untuk memastikan instance ini thread-safe, penting agar tidak terjadi race condition ketika database pertama kali diakses dari berbagai thread.

Metode getDatabase(context: Context) memeriksa apakah instance sudah ada. Jika belum, saya buat dengan Room.databaseBuilder() menggunakan application context, lalu memberi nama database "ultraman_db". Saya tambahkan .fallbackToDestructiveMigration() supaya jika ada perbedaan versi database, Room akan mereset database (menghapus dan membuat ulang) daripada crash karena konflik migrasi.

11. UltramanRepository.kt

Di file UltramanRepository.kt, saya membuat kelas UltramanRepository yang berfungsi sebagai jembatan antara layer data (DAO dan Retrofit) dengan ViewModel. Ini adalah tempat saya menyatukan logika pengambilan data dari jaringan (TMDB API) dan penyimpanan lokal menggunakan Room. Dengan cara ini, ViewModel saya tetap bersih dan hanya fokus pada penyajian data.

Pertama, saya mendeklarasikan dua properti:

- ultramanItems: mengambil semua data dari database lokal dalam bentuk Flow, supaya UI bisa menampilkan data secara real-time.
- favoriteItems: mengambil semua item favorit, juga sebagai Flow, untuk ditampilkan di halaman favorit.

Fungsi refreshData(apiKey: String) adalah fungsi inti yang saya pakai untuk memuat data terbaru dari TMDB. Saya buat dalam bentuk Flow<ApiResponse<...>>, sehingga ViewModel dan UI bisa merespons statusnya (Loading, Success, atau Error).

Dalam fungsi tersebut, saya memulai dengan emit(ApiResponse.Loading) agar UI tahu sedang ada proses pengambilan data. Lalu saya memanggil searchUltraman() dari Retrofit, yang akan mengembalikan hasil pencarian Ultraman dari TMDB. Hasil results saya konversi satu per satu ke UltramanItem melalui ekstensi toUltramanItem().

Setelah itu, saya ambil data lama dari database menggunakan getAllOnce() untuk mengecek status favorit sebelumnya. Ini penting karena data dari API tidak menyimpan informasi favorit. Jadi saya cocokkan data baru dengan data lama, dan kalau id cocok, saya salin nilai isFavorite agar tidak hilang.

Saya kemudian simpan data baru ke database dengan dao.insertAll(...), dibatasi hanya 20 item pertama agar tidak membebani aplikasi. Terakhir, saya emit hasil sukses berupa ApiResponse.Success(...).

Kalau ada error saat proses, misalnya koneksi internet gagal, saya tangani dengan catch, log error-nya, dan emit ApiResponse.Error.

Selain itu, saya juga buat fungsi toggleFavorite() untuk membalik status favorit sebuah item berdasarkan id dan nilai saat ini. Fungsi ini dipanggil dari UI ketika pengguna menekan ikon favorit.

Terakhir, fungsi getFavorites() hanya mengembalikan ulang dao.getFavorites(), disediakan agar ViewModel bisa mengaksesnya langsung bila dibutuhkan.

12. UltramanViewModel.kt

Pada file UltramanViewModel.kt, saya membangun ViewModel yang berfungsi sebagai penghubung antara UI dan UltramanRepository. Di sini, saya kelola data Ultraman secara

reaktif menggunakan StateFlow, agar UI bisa merespons perubahan data secara real-time dan tetap konsisten meskipun terjadi perubahan konfigurasi (seperti rotasi layar).

Pertama-tama, saya deklarasikan beberapa MutableStateFlow sebagai state internal:

- _ultramanList menyimpan daftar seluruh data Ultraman dari database lokal.
- _favoriteList menyimpan daftar yang telah ditandai favorit.
- _uiState menyimpan status API saat pengambilan data dari internet (Loading, Success, atau Error).
- _favoriteState menyimpan status pengambilan data favorit, dengan struktur respons yang sama seperti _uiState.

Untuk setiap MutableStateFlow, saya expose versinya yang StateFlow ke luar agar hanya bisa dibaca, bukan dimodifikasi langsung oleh UI.

Di dalam init, saya langsung memanggil refreshData() untuk mengambil data awal dari API TMDB, dan juga observeLocalData() agar bisa terus memantau data dari database secara live.

Dalam fungsi observeLocalData(), saya meluncurkan dua collectLatest coroutine:

- Pertama, untuk mengambil semua data dari repository.ultramanItems, lalu menyimpannya ke _ultramanList dan sekaligus memperbarui _uiState menjadi sukses.
- Kedua, untuk data favorit, saya lakukan hal serupa. Saya juga bungkus dengan try-catch agar bila terjadi error saat pengambilan data favorit, UI tetap mendapatkan ApiResponse.Error.

Fungsi refreshData() bisa dipanggil dari UI untuk memaksa ambil data terbaru dari TMDB. Ia memanggil fungsi dari repository dan mengalirkan hasilnya langsung ke _uiState.

Untuk kebutuhan seperti menampilkan detail, saya sediakan fungsi getItemById(id) yang mencari satu item berdasarkan ID dari daftar yang sudah dimuat.

Terakhir, untuk menangani aksi favorit dari pengguna, saya buat fungsi toggleFavorite(item). Fungsi ini akan memanggil repository agar membalik nilai isFavorite di database. Karena perubahan disimpan ke database, UI akan otomatis merespons berkat collectLatest yang berjalan di awal tadi.

13. UltramanViewModelFactory.kt

Pada file UltramanViewModelFactory.kt, saya membuat sebuah ViewModelProvider.Factory yang digunakan untuk menciptakan instance dari UltramanViewModel dengan dependensi yang tidak bisa disediakan secara default oleh Android (yaitu Repository yang memerlukan Dao, dan Dao yang butuh Context untuk mengakses Room database).

Konstruktor UltramanViewModelFactory menerima parameter Context, yang akan digunakan untuk menginisialisasi UltramanDatabase. Di dalam fungsi create(), saya panggil UltramanDatabase.getDatabase(context) untuk mendapatkan instance Room database, lalu saya ambil ultramanDao()-nya.

Dengan Dao tersebut, saya buat instance UltramanRepository, dan dari repository ini saya akhirnya membentuk instance UltramanViewModel.

Karena fungsi create() mengembalikan tipe generik T, saya perlu melakukan casting ke T dengan anotasi @Suppress("UNCHECKED_CAST") agar tidak terjadi error saat kompilasi. Ini adalah pendekatan standar saat menggunakan ViewModel yang membutuhkan konstruktor khusus, dan memastikan integrasi dengan ViewModelProvider berjalan lancar saat membuat ViewModel di dalam Activity atau Composable.

14. ListScreen.kt

Pada file ListScreen.kt, saya membuat tampilan utama yang menampilkan daftar seluruh data Ultraman dari API dan database lokal. Komponen utama yang saya gunakan adalah LazyColumn untuk menampilkan setiap item dalam bentuk kartu. Di dalam setiap kartu, saya tampilkan poster Ultraman dengan GlideImage, judul, tahun rilis, deskripsi singkat, dan dua tombol: satu untuk membuka tautan TMDB di browser (Intent.ACTION_VIEW) dan satu lagi untuk navigasi ke halaman detail dengan navController.

Saya juga menambahkan tombol favorit berupa ikon hati di pojok kiri atas gambar poster. Tombol ini bisa diklik untuk mengubah status favorit item tersebut melalui fungsi viewModel.toggleFavorite(item). Untuk menjaga tampilan tetap responsif dan aman dari error, saya membungkus state data dengan ApiResponse yang memiliki tiga kondisi: Loading, Error, dan Success. Jika data sedang dimuat, akan muncul CircularProgressIndicator. Jika terjadi error, muncul pesan error. Dan jika sukses, data ditampilkan dalam daftar.

15. DetailScreen.kt

File DetailScreen.kt berfungsi untuk menampilkan detail dari salah satu item Ultraman. Data detail ini saya ambil berdasarkan itemId yang dikirim dari halaman sebelumnya melalui navigasi. Saya memanggil fungsi viewModel.getItemById(itemId) untuk mengambil objek Ultraman yang sesuai. Jika data ditemukan, saya tampilkan poster dalam ukuran besar di bagian atas dengan GlideImage, lalu dilanjutkan dengan judul, tahun rilis, dan sinopsis atau deskripsi lengkapnya. Semua informasi saya bungkus dalam Column yang bisa discroll menggunakan verticalScroll, agar tampilan tetap rapi di berbagai ukuran layar.

Jika itemId tidak valid atau data tidak ditemukan, maka saya tampilkan teks "Data tidak ditemukan" di tengah layar sebagai fallback.

16. FavoriteScreen.kt

FavoriteScreen.kt memiliki struktur yang hampir sama dengan ListScreen, tapi hanya menampilkan data yang telah ditandai sebagai favorit oleh pengguna. Saya menggunakan state viewModel.favoriteState untuk mengelola datanya, yang juga dibungkus dengan ApiResponse agar dapat menangani loading, error, maupun data kosong dengan baik. Jika tidak ada item favorit, saya tampilkan pesan abu-abu di tengah layar agar pengguna tahu bahwa belum ada data yang difavoritkan.

Setiap item favorit ditampilkan dalam bentuk kartu, sama seperti di ListScreen, lengkap dengan gambar poster, judul, tahun, dan deskripsi singkat. Saya juga menyertakan dua tombol: "More Info" untuk membuka link TMDB, dan "Detail" untuk navigasi ke halaman detail. Ikon favorit di pojok kiri atas gambar tetap tersedia agar pengguna bisa menghapus item dari favorit langsung dari layar ini.

17. GlideImage.kt

Pada file GlideImage.kt, saya membuat sebuah komponen @Composable khusus bernama GlideImage yang fungsinya adalah untuk menampilkan gambar menggunakan Glide, library populer dari Android untuk memuat gambar dari internet. Karena Glide sendiri tidak bisa digunakan langsung di Jetpack Compose (karena dia berbasis View), maka saya gunakan AndroidView untuk menyisipkan ImageView ke dalam dunia Compose.

Saya menerima beberapa parameter: imageUrl untuk URL gambar, contentDescription untuk aksesibilitas, modifier untuk styling Compose seperti ukuran dan padding, serta contentScale untuk mengatur cara gambar di-scale. Nilai contentScale ini saya konversikan ke ImageView.ScaleType supaya tetap konsisten antara Compose dan View klasik.

Di dalam AndroidView, saya buat ImageView baru dan atur scaleType-nya berdasarkan contentScale. Kemudian di bagian update, saya panggil Glide.with(context).load(imageUrl).into(it) untuk memuat gambar ke ImageView tersebut. Dengan cara ini, saya bisa tetap menggunakan kemampuan Glide dalam proyek Compose tanpa harus migrasi ke library lain seperti Coil.

18. BottomBarNavigation.kt

Pada file ini, saya membuat komponen navigasi bawah (BottomBarNavigation) menggunakan Material 3 di Jetpack Compose. Tujuannya adalah agar pengguna bisa berpindah antar layar utama, yaitu "Daftar" dan "Favorit", dengan mudah melalui bottom bar.

Pertama, saya mendefinisikan sebuah data class bernama BottomNavItem yang menyimpan informasi dasar setiap item navigasi: judul (title), ikon (icon), dan rute (route) yang akan digunakan untuk navigasi. Kemudian, saya membuat daftar item bottom bar dalam variabel bottomNavItems, di mana "Daftar" menggunakan ikon film (Icons.Default.Movie) dan rutenya "list", sementara "Favorit" menggunakan ikon hati (Icons.Default.Favorite) dengan rute "favorite".

Dalam fungsi BottomBarNavigation, saya menggunakan NavigationBar dari Material 3. Di dalamnya, saya ambil rute aktif saat ini dari navController.currentBackStackEntryAsState() untuk menentukan item mana yang sedang aktif (dipilih). Lalu, saya melakukan iterasi pada bottomNavItems dan menampilkan masing-masing sebagai NavigationBarItem.

Setiap item akan dianggap terpilih (selected) jika currentRoute cocok dengan item.route. Saat pengguna mengetuk item, saya melakukan navigasi ke rute tersebut hanya jika belum berada di rute yang sama, agar tidak membuat tumpukan navigasi menumpuk. Saya juga menggunakan popUpTo("list") agar kembali ke awal saat perlu, dan launchSingleTop = true agar tidak membuat layar ganda jika sudah berada di rute itu.

Dengan pendekatan ini, saya memastikan bottom bar tetap sederhana, responsif, dan intuitif untuk berpindah antar halaman utama aplikasi.

TAUTAN GIT

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/Omelette719/Pemrograman-Mobile.git