LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 5

CONNECT TO THE INTERNET

Oleh:

Damarjati Suryo Laksono **NIM. 2310817210014**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
JUNI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 5

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

Nama Praktikan : Damarjati Suryo Laksono
NIM : 2310817210014

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

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: 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. AppDatabase

```
1  package com.example.myapi_test
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 = [CharacterInfoEntity::class], version =
9  1, exportSchema = false)
```

```
10  abstract class AppDatabase : RoomDatabase() {
11
12      abstract fun characterDao(): CharacterDao
13
14      companion object {
15          @Volatile
16          private var INSTANCE: AppDatabase? = null
17
18          fun getDatabase(context: Context): AppDatabase {
19              return INSTANCE ?: synchronized(this) {
20                  val instance = Room.databaseBuilder(
21                      context.applicationContext,
22                      AppDatabase::class.java,
23                      "character_database"
24                  ).build()
25                  INSTANCE = instance
26                  instance
27              }
28          }
29      }
30  }
```

Tabel 1. 1 Source Code AppDatabase.kt

2. CacheMapper.kt

```
1   package com.example.myapi_test
2
3   fun CharacterInfo.toCharacterInfoEntity():
4   CharacterInfoEntity {
5       return CharacterInfoEntity(
6           characterId = this.characterId,
7           characterName = this.characterName,
8           characterUrl = this.characterUrl,
9           characterImageUrl = this.characterImageUrl,
```

```
        japaneseVoiceActor = this.japaneseVoiceActor,
        englishVoiceActor = this.englishVoiceActor,
        favorites = this.favorites
    )
}

fun CharacterInfoEntity.toCharacterInfo(): CharacterInfo {
    return CharacterInfo(
        characterId = this.characterId,
        characterName = this.characterName,
        characterUrl = this.characterUrl,
        characterImageUrl = this.characterImageUrl,
        japaneseVoiceActor = this.japaneseVoiceActor,
        englishVoiceActor = this.englishVoiceActor,
        favorites = this.favorites
    )
}
```

Tabel 1. 2 Source Code CacheMapper.kt

3. CharacterAdapter.kt

```
package com.example.myapi_test

import android.content.Intent
import android.net.Uri
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.example.myapi_test.databinding.ItemListBinding
import java.text.NumberFormat
import java.util.Locale

class CharacterAdapter(
```

```
    private var characters: List<CharacterInfo>,
    private val onDetailClick: (CharacterInfo) -> Unit
) :
RecyclerView.Adapter<CharacterAdapter.CharacterViewHolder>()
{

    inner class CharacterViewHolder(val binding:
ItemListBinding) :
        RecyclerView.ViewHolder(binding.root) {

        fun bind(character: CharacterInfo) {
            binding.apply {
                textViewCharacterName.text =
character.characterName
                val formattedFavorites =
NumberFormat.getNumberInstance(Locale.US).format(character.f
avorites)
                textViewFavorites.text = formattedFavorites

                val vaJapaneseText = "JP:
${character.japaneseVoiceActor ?: "N/A"}"
                val vaEnglishText = "EN:
${character.englishVoiceActor ?: "N/A"}"
                textViewVoiceActors.text =
"$vaJapaneseText\n$vaEnglishText"

                Glide.with(imageViewCharacter.context)
                    .load(character.characterImageUrl)
                    .into(imageViewCharacter)

                buttonDetail.setOnClickListener {
                    onDetailClick(character)
                }

                buttonUrl.setOnClickListener {
```

```
45                      val intent = Intent(Intent.ACTION_VIEW,
46  Uri.parse(character.characterUrl))
47
    imageViewCharacter.context.startActivity(intent)
48                  }
49              }
50          }
51      }
52
53      override fun onCreateViewHolder(parent: ViewGroup,
54  viewType: Int): CharacterViewHolder {
55          val binding =
56  ItemListBinding.inflate(LayoutInflater.from(parent.context),
57  parent, false)
58          return CharacterViewHolder(binding)
59      }
60
61      override fun getItemCount() = characters.size
62
63      override fun onBindViewHolder(holder:
64  CharacterViewHolder, position: Int) {
65          holder.bind(characters[position])
66      }
67
68      fun setData(newCharacters: List<CharacterInfo>) {
69          characters = newCharacters
70          notifyDataSetChanged()
71      }
72  }
```

Tabel 1. 3 Source Code CharacterAdapter.kt

4. CharacterDao

```
1  package com.example.myapi_test
2
3  import androidx.room.Dao
4  import androidx.room.Insert
5  import androidx.room.OnConflictStrategy
6  import androidx.room.Query
7  import kotlinx.coroutines.flow.Flow
8
9  @Dao
10 interface CharacterDao {
11
12     @Query("SELECT * FROM characters")
13     fun getCharacters(): Flow<List<CharacterInfoEntity>>
14
15     @Insert(onConflict = OnConflictStrategy.REPLACE)
16     suspend fun insertAll(characters:
17 List<CharacterInfoEntity>)
18
19     @Query("DELETE FROM characters")
20     suspend fun clearAll()
21 }
```

Tabel 1. 4 Source Code CharacterDao.kt

5. CharacterInfoEntity

```
1  package com.example.myapi_test
2
3  import androidx.room.Entity
4  import androidx.room.PrimaryKey
5
6  @Entity(tableName = "characters")
7  data class CharacterInfoEntity(
8      @PrimaryKey
9      val characterId: Int,
```

```
    val characterName: String,
    val characterUrl: String,
    val characterImageUrl: String,
    val japaneseVoiceActor: String?,
    val englishVoiceActor: String?,
    val favorites: Int
)
```

Tabel 1. 5 Source Code CharacterInfoEntity.kt

6. CharacterMapper.kt

```
package com.example.myapi_test

fun mapToCharacterInfoList(characterDataList:
List<CharacterListItem>?): List<CharacterInfo> {
    return characterDataList?.map { data ->
        CharacterInfo(
            characterId = data.character.malId,
            characterName = data.character.name ?: "Name not
found",
            characterUrl = data.character.url,
            characterImageUrl =
data.character.images?.jpg?.imageUrl ?: "",

            japaneseVoiceActor = data.voices?.find {
it.language.equals("Japanese", ignoreCase = true)
}?.person?.name,
            englishVoiceActor = data.voices?.find {
it.language.equals("English", ignoreCase = true)
}?.person?.name,
            favorites = data.favorites
        )
    } ?: emptyList()
}
```

Tabel 1. 6 Source Code CharacterMapper.kt

7. CharacterModels.kt

```
1   package com.example.myapi_test
2   import android.os.Parcelable
3   import kotlinx.parcelize.Parcelize
4
5   import com.google.gson.annotations.SerializedName
6
7   @Parcelize
8   data class CharacterInfo(
9       val characterId: Int,
10      val characterName: String,
11      val characterUrl: String,
12      val characterImageUrl: String,
13      val japaneseVoiceActor: String?,
14      val englishVoiceActor: String?,
15      val favorites: Int
16  ) : Parcelable
17
18  data class AnimeCharactersResponse(
19      @SerializedName("data")
20      val data: List<CharacterListItem>
21  )
22
23  data class Character(
24      @SerializedName("mal_id")
25      val malId: Int,
26
27      @SerializedName("url")
28      val url: String,
29
30      @SerializedName("images")
31      val images: Images?,
```

```
32
33      @SerializedName("name")
34      val name: String?
35  )
36
37  data class CharacterListItem(
38      @SerializedName("character")
39      val character: Character,
40
41      @SerializedName("role")
42      val role: String,
43
44      @SerializedName("favorites")
45      val favorites: Int,
46
47      @SerializedName("voice_actors")
48      val voices: List<Voice>?
49  )
50
51  data class CharacterDetailResponse(
52      @SerializedName("data")
53      val data: CharacterDetails
54  )
55
56  data class CharacterDetails(
57      @SerializedName("mal_id")
58      val malId: Int,
59
60      @SerializedName("url")
61      val url: String,
62
63      @SerializedName("images")
64      val images: Images,
65
66      @SerializedName("name")
```

```
67      val name: String,
68
69      @SerializedName("name_kanji")
70      val nameKanji: String?,
71
72      @SerializedName("favorites")
73      val favorites: Int,
74
75      @SerializedName("about")
76      val about: String?,
77
78      @SerializedName("voices")
79      val voices: List<Voice>
80  )
81
82  data class Voice(
83      @SerializedName("person")
84      val person: Person,
85      @SerializedName("language")
86      val language: String
87  )
88
89  data class Person(
90      @SerializedName("mal_id")
91      val malId: Int,
92      @SerializedName("url")
93      val url: String,
94      @SerializedName("images")
95      val images: PersonImages,
96      @SerializedName("name")
97      val name: String
98  )
99
    data class Images(
        @SerializedName("jpg")
```

```
    val jpg: ImageType
)

data class ImageType(
    @SerializedName("image_url")
    val imageUrl: String
)

data class PersonImages(
    @SerializedName("jpg")
    val jpg: ImageType
)
```

Tabel 1. 7 Source Code CharacterModel.kt

8. CharacterRepository

```
1  package com.example.myapi_test
2
3  import kotlinx.coroutines.flow.Flow
4  import kotlinx.coroutines.flow.map
5
6  class CharacterRepository(
7      private val apiService: JikanApiService,
8      private val characterDao: CharacterDao
9  ) {
10
11     fun getAnimeCharacters(): Flow<List<CharacterInfo>> {
12         return characterDao.getCharacters().map { entities -
13 >
14             entities.map { it.toCharacterInfo() }
15         }
16     }
17
18     suspend fun refreshCharacters(animeId: Int) {
```

```
        val response =
apiService.getAnimeCharacters(animeId)
        val characterInfoList =
mapToCharacterInfoList(response.data)

        characterDao.clearAll()
        characterDao.insertAll(characterInfoList.map {
it.toCharacterInfoEntity() })
    }

    suspend fun getCharacterDetails(characterId: Int):
CharacterDetailResponse {
        return apiService.getCharacterDetails(characterId)
    }
}

```

Tabel 1. 8 Source Code CharacterRepository.kt

9. CharacterViewModelFactory

```
package com.example.myapi_test

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

class CharacterViewModelFactory(private val repository:
CharacterRepository) : ViewModelProvider.Factory {

    override fun <T : ViewModel> create(modelClass:
Class<T>): T {
        return when {

modelClass.isAssignableFrom(HomeViewModel::class.java) -> {
                HomeViewModel(repository) as T
            }
```

```
14
15  modelClass.isAssignableFrom(DetailViewModel::class.java) ->
16  {
17                  DetailViewModel(repository) as T
18              }
19              else -> throw IllegalArgumentException("Unknown
20  ViewModel class: ${modelClass.name}")
21          }
22      }
23  }
```

Tabel 1. 9 Source Code CharacterViewModelFactory.kt

10. DetailFragment

```
1   package com.example.myapi_test
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.util.Log
7   import android.view.LayoutInflater
8   import android.view.View
9   import android.view.ViewGroup
10  import androidx.fragment.app.Fragment
11  import androidx.lifecycle.ViewModelProvider
12  import androidx.navigation.fragment.navArgs
13  import com.bumptech.glide.Glide
14  import
15  com.example.myapi_test.databinding.FragmentDetailBinding
16  import java.text.NumberFormat
17  import java.util.Locale
18
19  class DetailFragment : Fragment() {
20
```

```
21      private var _binding: FragmentDetailBinding? = null
22      private val binding get() = _binding!!
23      private val args: DetailFragmentArgs by navArgs()
24
25      private lateinit var viewModel: DetailViewModel
26
27      override fun onCreateView(
28          inflater: LayoutInflater, container: ViewGroup?,
29          savedInstanceState: Bundle?
30      ): View {
31          _binding = FragmentDetailBinding.inflate(inflater,
32  container, false)
33          return binding.root
34      }
35
36      override fun onViewCreated(view: View,
37  savedInstanceState: Bundle?) {
38          super.onViewCreated(view, savedInstanceState)
39
40          val characterDao =
41  AppDatabase.getDatabase(requireContext()).characterDao()
42          val repository =
43  CharacterRepository(RetrofitInstance.api, characterDao)
44          val viewModelFactory =
45  CharacterViewModelFactory(repository)
46
47          viewModel = ViewModelProvider(this,
48  viewModelFactory)[DetailViewModel::class.java]
49
50          bindInitialData(args.character)
51
52          setupObservers()
53
54  viewModel.fetchCharacterDetails(args.character.characterId)
55      }
```

```
56
57      private fun bindInitialData(character: CharacterInfo) {
58          binding.apply {
59              textViewNameDetail.text =
60  character.characterName
61              val formattedFavorites =
62  NumberFormat.getNumberInstance(Locale.US)
63                  .format(character.favorites)
64              textViewFavoritesDetail.text =
65  "$formattedFavorites Favorites"
66              textViewVoiceActorsList.text = "Loading voice
67  actors..."
68
69              Glide.with(this@DetailFragment)
70                  .load(character.characterImageUrl)
71                  .into(imageViewCharacterDetail)
72
73              buttonViewProfileDetail.setOnClickListener {
74                  val intent = Intent(Intent.ACTION_VIEW,
75  Uri.parse(character.characterUrl))
76                  context?.startActivity(intent)
77              }
78          }
79      }
80
81      private fun setupObservers() {
82
83  viewModel.characterDetails.observe(viewLifecycleOwner) {
84  details ->
85              details?.let { bindFullData(it) }
86          }
87
88          viewModel.error.observe(viewLifecycleOwner) {
89  errorMessage ->
90              errorMessage?.let {
```

```
                binding.textViewVoiceActorsList.text = it
                Log.e("DetailFragment", "Error: $it")
            }
        }
    }

    private fun bindFullData(details: CharacterDetails) {
        binding.apply {
            textViewNameDetail.text = details.name
            val formattedFavorites =
NumberFormat.getNumberInstance(Locale.US)
                .format(details.favorites)
            textViewFavoritesDetail.text =
"$formattedFavorites Favorites"

            val voiceActorsText =
details.voices.joinToString("\n") { voice ->
                "${voice.language}: ${voice.person.name}"
            }

            textViewVoiceActorsList.text =
voiceActorsText.ifEmpty { "No voice actor information
available." }
        }
    }

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

Tabel 1. 10 Source CodeDetailFragment.kt

11. DetailViewModel

```
1   package com.example.myapi_test
2
3   import androidx.lifecycle.LiveData
4   import androidx.lifecycle.MutableLiveData
5   import androidx.lifecycle.ViewModel
6   import androidx.lifecycle.viewModelScope
7   import kotlinx.coroutines.launch
8   import java.io.IOException
9
10  class DetailViewModel(private val repository:
11  CharacterRepository) : ViewModel() {
12
13      private val _characterDetails =
14  MutableLiveData<CharacterDetails>()
15      val characterDetails: LiveData<CharacterDetails> get() =
16  _characterDetails
17
18      private val _error = MutableLiveData<String?>()
19      val error: LiveData<String?> get() = _error
20
21      fun fetchCharacterDetails(characterId: Int) {
22          viewModelScope.launch {
23              try {
24                  val response =
25  repository.getCharacterDetails(characterId)
26                  _characterDetails.value = response.data
27                  _error.value = null //
28              } catch (e: IOException) {
29                  _error.value = "Failed to load details due
30  to a network error."
31              } catch (e: Exception) {
32                  _error.value = "An unexpected error occurred
33  while fetching details."
```

```
34              }
35          }
36      }
37  }
```

Tabel 1. 11 Source Code DetailViewModel

12. HomeFragment

```
1   package com.example.myapi_test
2
3   import android.os.Bundle
4   import android.util.Log
5   import android.view.LayoutInflater
6   import android.view.View
7   import android.view.ViewGroup
8   import android.widget.Toast
9   import androidx.fragment.app.Fragment
10  import androidx.lifecycle.ViewModelProvider
11  import androidx.navigation.findNavController
12  import androidx.recyclerview.widget.LinearLayoutManager
13  import
14  com.example.myapi_test.databinding.FragmentHomeBinding
15
16  class HomeFragment : Fragment() {
17
18      private var _binding: FragmentHomeBinding? = null
19      private val binding get() = _binding!!
20
21      private lateinit var characterAdapter: CharacterAdapter
22      private lateinit var viewModel: HomeViewModel
23
24      override fun onCreateView(
25          inflater: LayoutInflater, container: ViewGroup?,
26          savedInstanceState: Bundle?
27      ): View {
```

```
28          _binding = FragmentHomeBinding.inflate(inflater,
29  container, false)
30          return binding.root
31      }
32
33      override fun onViewCreated(view: View,
34  savedInstanceState: Bundle?) {
35          super.onViewCreated(view, savedInstanceState)
36
37          val characterDao =
38  AppDatabase.getDatabase(requireContext()).characterDao()
39
40          val repository =
41  CharacterRepository(RetrofitInstance.api, characterDao)
42          val viewModelFactory =
43  CharacterViewModelFactory(repository)
44
45          viewModel = ViewModelProvider(this,
46  viewModelFactory)[HomeViewModel::class.java]
47
48          setupRecyclerView()
49          setupObservers()
50      }
51
52      private fun setupRecyclerView() {
53          characterAdapter = CharacterAdapter(emptyList()) {
54  character ->
55              val action =
56  HomeFragmentDirections.actionHomeFragmentToDetailFragment(ch
57  aracter)
58
59  requireActivity().findNavController(R.id.nav_host_fragment).
60  navigate(action)
61          }
62          binding.recyclerView.apply {
```

```
            layoutManager = LinearLayoutManager(context)
            adapter = characterAdapter
        }
    }

    private fun setupObservers() {
        viewModel.characters.observe(viewLifecycleOwner) {
characters ->
            if (characters.isNullOrEmpty()) {
                binding.textViewEmptyCache.visibility =
View.VISIBLE
                binding.recyclerView.visibility = View.GONE
            } else {
                binding.textViewEmptyCache.visibility =
View.GONE
                binding.recyclerView.visibility =
View.VISIBLE
                characterAdapter.setData(characters)
            }
        }

        viewModel.isLoading.observe(viewLifecycleOwner) {
isLoading ->
            binding.progressBar.visibility = if (isLoading)
View.VISIBLE else View.GONE
        }

        viewModel.error.observe(viewLifecycleOwner) {
errorMessage ->
            errorMessage?.let {
                Toast.makeText(context, it,
Toast.LENGTH_LONG).show()
                Log.e("HomeFragment", "Error: $it")
            }
        }
```

```
98      }
99
        override fun onDestroyView() {
            super.onDestroyView()
            _binding = null
        }
    }
```

Tabel 1. 12 Source Code HomeFragment.kt

13. HomeViewModel

```
1   package com.example.myapi_test
2
3   import androidx.lifecycle.LiveData
4   import androidx.lifecycle.MutableLiveData
5   import androidx.lifecycle.ViewModel
6   import androidx.lifecycle.asLiveData
7   import androidx.lifecycle.viewModelScope
8   import kotlinx.coroutines.launch
9   import java.io.IOException
10
11  class HomeViewModel(private val repository:
12  CharacterRepository) : ViewModel() {
13
14      val characters: LiveData<List<CharacterInfo>> =
15  repository.getAnimeCharacters().asLiveData()
16
17      private val _isLoading = MutableLiveData<Boolean>()
18      val isLoading: LiveData<Boolean> get() = _isLoading
19
20      private val _error = MutableLiveData<String?>()
21      val error: LiveData<String?> get() = _error
22
23      init {
```

```
24          refreshCharacterData()
25      }
26
27      fun refreshCharacterData() {
28          viewModelScope.launch {
29              _isLoading.value = true
30              try {
31                  repository.refreshCharacters(52991)
32                  _error.value = null
33              } catch (e: IOException) {
34                  _error.value = "Network error. Displaying
35  cached data."
36              } catch (e: Exception) {
37                  _error.value = "An unexpected error occurred
38  during refresh."
39              } finally {
40                  _isLoading.value = false
41              }
42          }
43      }
44  }
```

Tabel 1. 13 Source Code HomeViewModel.kt

14. JikanApiService

```
1   package com.example.myapi_test
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5   import androidx.navigation.NavController
6   import androidx.navigation.fragment.NavHostFragment
7   import
8   androidx.navigation.ui.setupActionBarWithNavController
9   import
10  androidx.core.splashscreen.SplashScreen.Companion.installSp
```

```
11  lashScreen
12  import
    com.example.myapi_test.databinding.ActivityMainBinding
13
14  class MainActivity : AppCompatActivity() {
15      private lateinit var navController: NavController
16      private lateinit var binding: ActivityMainBinding
17
18      override fun onCreate(savedInstanceState: Bundle?) {
19          super.onCreate(savedInstanceState)
20          installSplashScreen()
21          binding =
22  ActivityMainBinding.inflate(layoutInflater)
23          setContentView(binding.root)
24
25          setSupportActionBar(binding.toolbar)

26          val navHostFragment = supportFragmentManager
                .findFragmentById(R.id.nav_host_fragment) as
27  NavHostFragment
28          navController = navHostFragment.navController
29
30          setupActionBarWithNavController(navController)
31      }
32
33      override fun onSupportNavigateUp(): Boolean {
34          return navController.navigateUp() ||
35  super.onSupportNavigateUp()
36      }
37  }
```

Tabel 1. 14 Source Code JikanApiService.kt

15. MainActivity

```
1   package com.example.myapi_test
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5   import androidx.navigation.NavController
6   import androidx.navigation.fragment.NavHostFragment
7   import
8   androidx.navigation.ui.setupActionBarWithNavController
9   import
10  androidx.core.splashscreen.SplashScreen.Companion.installSp
11  lashScreen
12  import
13  com.example.myapi_test.databinding.ActivityMainBinding
14
15  class MainActivity : AppCompatActivity() {
16      private lateinit var navController: NavController
17      private lateinit var binding: ActivityMainBinding
18
19      override fun onCreate(savedInstanceState: Bundle?) {
20          super.onCreate(savedInstanceState)
21          installSplashScreen()
22          binding =
23  ActivityMainBinding.inflate(layoutInflater)
24          setContentView(binding.root)
25
26          setSupportActionBar(binding.toolbar)
27
28          val navHostFragment = supportFragmentManager
29              .findFragmentById(R.id.nav_host_fragment) as
30  NavHostFragment
31          navController = navHostFragment.navController
32
33          setupActionBarWithNavController(navController)
34      }
35
```

```
36      override fun onSupportNavigateUp(): Boolean {
37          return navController.navigateUp() ||
38  super.onSupportNavigateUp()
39      }
40  }
```

Tabel 1. 15 Source Code MainActivity.kt

16. Retrofit Instance

```
1   package com.example.myapi_test
2
3   import retrofit2.Retrofit
4   import retrofit2.converter.gson.GsonConverterFactory
5
6   object RetrofitInstance {
7       private const val BASE_URL =
8   "https://api.jikan.moe/v4/"
9
10      private val retrofit by lazy {
11          Retrofit.Builder()
12              .baseUrl(BASE_URL)
13
14  .addConverterFactory(GsonConverterFactory.create())
15              .build()
16      }
17
18      val api: JikanApiService by lazy {
19          retrofit.create(JikanApiService::class.java)
20      }
21  }
```

Tabel 1. 16 Source Code RetrofitInstance.kt

17. activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3
4  xmlns:android="http://schemas.android.com/apk/res/android"
5      xmlns:app="http://schemas.android.com/apk/res-auto"
6      xmlns:tools="http://schemas.android.com/tools"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      tools:context=".MainActivity"
10
11     android:background="@color/backgroundColor">
12
13     <com.google.android.material.appbar.MaterialToolbar
14         android:id="@+id/toolbar"
           android:layout_width="match_parent"
15         android:layout_height="?attr/actionBarSize"
           android:background="?attr/colorPrimary"
16         app:titleTextColor="?attr/colorOnPrimary"
17         app:layout_constraintTop_toTopOf="parent"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintEnd_toEndOf="parent" />

20     <androidx.fragment.app.FragmentContainerView
21         android:id="@+id/nav_host_fragment"
22
23 android:name="androidx.navigation.fragment.NavHostFragment"
24         android:layout_width="0dp"
25         android:layout_height="0dp"
26         app:layout_constraintLeft_toLeftOf="parent"
27         app:layout_constraintRight_toRightOf="parent"
28         app:layout_constraintTop_toBottomOf="@id/toolbar"
29         app:layout_constraintBottom_toBottomOf="parent"
30         app:defaultNavHost="true"
31         app:navGraph="@navigation/nav_graph" />
32
```

```
33
</androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 1. 17 Source Code activity_main.xml

18. fragment_detail.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ScrollView
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:fillViewport="true"
8      android:background="@color/backgroundColor"
9      tools:context=".DetailFragment">
10
11     <LinearLayout
12         android:layout_width="match_parent"
13         android:layout_height="wrap_content"
14         android:gravity="center_horizontal"
           android:orientation="vertical"
15         android:padding="16dp">

16         <ImageView
17             android:id="@+id/imageViewCharacterDetail"
18             android:layout_width="200dp"
19             android:layout_height="300dp"

20 android:contentDescription="@string/character_image_descrip
21 tion"
22             android:scaleType="centerCrop"
23             tools:src="@tools:sample/avatars" />
24
25         <TextView
26             android:id="@+id/textViewNameDetail"
```

```
27              android:layout_width="wrap_content"
28              android:layout_height="wrap_content"
29              android:layout_marginTop="16dp"
30              android:textSize="24sp"
31              android:textStyle="bold"
32              android:textColor="@color/textColorPrimary"
33              tools:text="Character Name" />
34
35          <TextView
36              android:id="@+id/textViewFavoritesDetail"
37              android:layout_width="wrap_content"
38              android:layout_height="wrap_content"
39              android:layout_marginTop="8dp"
40              android:textSize="16sp"
41              android:textColor="@color/textColorSecondary"
42              tools:text="123,456 Favorites" />
43
44          <TextView
45              android:id="@+id/textViewVoiceActorsList"
46              android:layout_width="wrap_content"
47  android:layout_height="wrap_content"
48              android:layout_marginTop="16dp"
49              android:textAlignment="center"
50              android:textSize="14sp"
51              android:textColor="@color/textColorSecondary"
52              tools:text="Japanese: VA Name\nEnglish: VA
53  Name" />
54
55          <Button
56              android:id="@+id/buttonViewProfileDetail"
57              android:layout_width="wrap_content"
58              android:layout_height="wrap_content"
59              android:layout_marginTop="24dp"
60              android:text="@string/view_profile_button" />
61
```

```
        </LinearLayout>
62  </ScrollView>
```

Tabel 1. 18 Source Code fragment_detail.xml

19. fragment_home.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3
4   xmlns:android="http://schemas.android.com/apk/res/android"
5       xmlns:app="http://schemas.android.com/apk/res-auto"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:layout_width="match_parent"
8       android:layout_height="match_parent"
9       tools:context=".HomeFragment"
10
11      android:background="@color/backgroundColor">
12
13      <androidx.recyclerview.widget.RecyclerView
14          android:id="@+id/recyclerView"
            android:layout_width="0dp"
15          android:layout_height="0dp"
            app:layout_constraintTop_toTopOf="parent"
16          app:layout_constraintBottom_toBottomOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintEnd_toEndOf="parent" />
19
        <ProgressBar
20          android:id="@+id/progressBar"
21          android:layout_width="wrap_content"
22          android:layout_height="wrap_content"
23          android:visibility="gone"
24          app:layout_constraintBottom_toBottomOf="parent"
```

```
25          app:layout_constraintEnd_toEndOf="parent"
26          app:layout_constraintStart_toStartOf="parent"
27          app:layout_constraintTop_toTopOf="parent"
28          tools:visibility="visible" />
29
30      <TextView
31          android:id="@+id/text_view_empty_cache"
32          android:layout_width="wrap_content"
33          android:layout_height="wrap_content"
34          android:text="No cached data available. Please
35  check your network."
36          android:visibility="gone"
37
38          android:textColor="@color/textColorSecondary"
39
40          app:layout_constraintTop_toTopOf="parent"
41          app:layout_constraintBottom_toBottomOf="parent"
42          app:layout_constraintStart_toStartOf="parent"
43          app:layout_constraintEnd_toEndOf="parent" />
44
45  </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 1. 19 Source Code fragment_home.xml

20. item_list.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView

3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="wrap_content"
8       android:layout_margin="8dp"
9       app:cardCornerRadius="12dp"
```

```
    app:cardElevation="4dp"
    app:cardBackgroundColor="@color/cardBackgroundColor">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp">

        <ImageView
            android:id="@+id/imageViewCharacter"
            android:layout_width="100dp"
            android:layout_height="150dp"
            android:scaleType="centerCrop"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:src="@tools:sample/avatars"

android:contentDescription="@string/character_image_descrip
tion" />

        <TextView
            android:id="@+id/textViewCharacterName"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="16dp"
            android:textSize="20sp"
            android:textStyle="bold"
            android:textColor="@color/textColorPrimary"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toEndOf="@id/imageViewCharacter"

app:layout_constraintTop_toTopOf="@id/imageViewCharacter"
            tools:text="Character Name" />

```

```
39          <ImageView
40              android:id="@+id/iconFavorites"
41              android:layout_width="16dp"
                android:layout_height="16dp"
42              android:layout_marginTop="12dp"
43              android:src="@drawable/ic_star_gold"
                app:tint="@color/starColor"
44
45  app:layout_constraintStart_toStartOf="@id/textViewCharacter
46  Name"
47
48  app:layout_constraintTop_toBottomOf="@id/textViewCharacterN
49  ame"
50
51  app:layout_constraintBottom_toBottomOf="@id/textViewFavorit
52  es"
53
54  android:contentDescription="@string/favorites_icon_descript
    ion" />

55          <TextView
56              android:id="@+id/textViewFavorites"
57              android:layout_width="wrap_content"
58              android:layout_height="wrap_content"
59              android:layout_marginStart="6dp"
60              android:textSize="14sp"
61              android:textStyle="bold"
62              android:textColor="@color/textColorSecondary"

    app:layout_constraintTop_toTopOf="@id/iconFavorites"
63
64  app:layout_constraintStart_toEndOf="@id/iconFavorites"
                tools:text="98,425" />
65
66          <TextView
```

```
67              android:id="@+id/textViewVoiceActors"
68              android:layout_width="0dp"
69              android:layout_height="wrap_content"
70              android:layout_marginTop="12dp"
71              android:textSize="14sp"
72              android:textColor="@color/textColorSecondary"
73
74  app:layout_constraintTop_toBottomOf="@id/textViewFavorites"
75
76  app:layout_constraintStart_toStartOf="@id/textViewCharacter
77  Name"
78              app:layout_constraintEnd_toEndOf="parent"
79              tools:text="JP: Voice Actor\nEN: Voice Actor"
80  />
81
82          <Button
83              android:id="@+id/buttonDetail"
84              android:layout_width="wrap_content"
85              android:layout_height="wrap_content"
86              android:layout_marginTop="16dp"
87              android:text="@string/details_button"
88              app:layout_constraintEnd_toEndOf="parent"
89
90  app:layout_constraintTop_toBottomOf="@id/imageViewCharacter
91  "
92
93  app:layout_constraintBottom_toBottomOf="parent"/>
94
95          <Button
96              android:id="@+id/buttonUrl"
97
98  style="@style/Widget.MaterialComponents.Button.TextButton"
99              android:layout_width="wrap_content"
100             android:layout_height="wrap_content"
101             android:layout_marginEnd="8dp"
```

```
102             android:text="@string/view_profile_button"
103
104 app:layout_constraintBaseline_toBaselineOf="@id/buttonDetai
105 l"
106
107 app:layout_constraintEnd_toStartOf="@id/buttonDetail" />
108
109     </androidx.constraintlayout.widget.ConstraintLayout>
110 </androidx.cardview.widget.CardView>
```

Tabel 1. 20 Source Code item_list.xml

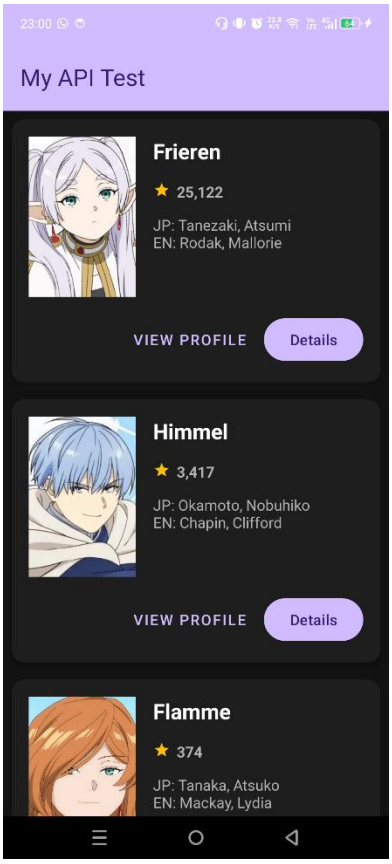
21. nav_graph.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <navigation
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:id="@+id/nav_graph"
7       app:startDestination="@id/homeFragment">
8
9       <fragment
10          android:id="@+id/homeFragment"
11          android:name="com.example.myapi_test.HomeFragment"
12          tools:layout="@layout/fragment_home">
13
14          <action
15
16  android:id="@+id/action_homeFragment_to_detailFragment"
17              app:destination="@id/detailFragment" />
18
19      </fragment>
20
21      <fragment
22          android:id="@+id/detailFragment"
```

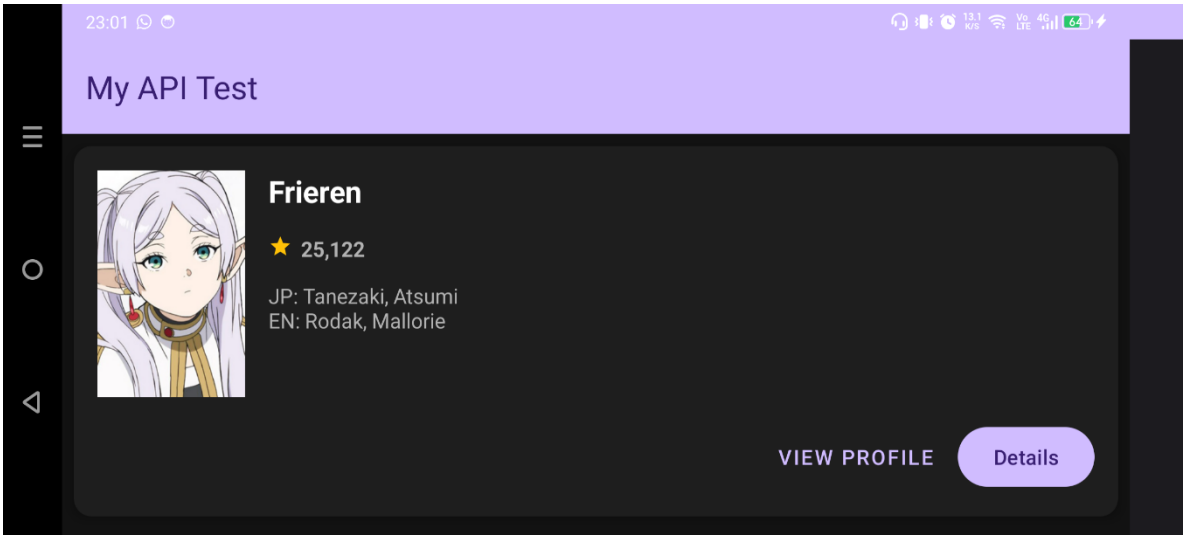
```
23
24  android:name="com.example.myapi_test.DetailFragment"
25          tools:layout="@layout/fragment_detail">
26
27          <argument
28              android:name="character"
29
30  app:argType="com.example.myapi_test.CharacterInfo" />
31      </fragment>
32
33  </navigation>
```

Tabel 1. 21 Source Code nav_graph.xml

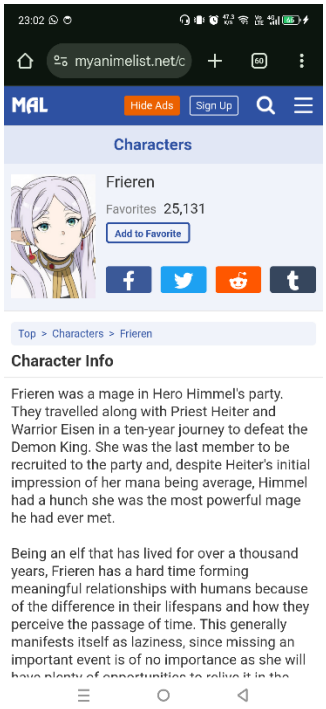
B. Output Program



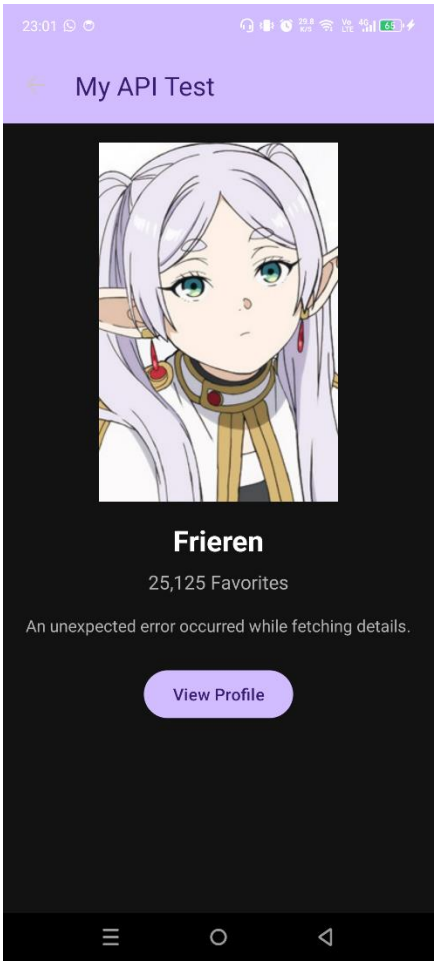
Gambar 1 Screenshot Hasil Jawaban Soal 1



Gambar 2 Screenshot Hasil Jawaban Soal 1



Gambar 3 Screenshot tombol Detail



Gambar 4 Screenshot tombol Info

C. Pembahasan

1. AppDatabase.kt

Digunakan sebagai penghubung utama data base room (local database) dengan cara menyediakan akses ke DAO

2. CacheMapper.kt

Digunakan untuk konversi data misal dari objek CharacterInfo akan di ubah menjadi CharacterInfoEntity dengan tujuan agar bisa di simpan di room.

3. CharacterAdapter.kt

Adapter, dan juga sebagai listener button

4. **CharacterDao.kt**

   DAO buat crud.

5. **CharacterInfoEntity.kt**

   Struktur tabel character untuk database Room

6. **CharacterMapper.kt**

   Konversi data agar bisa dipakai.

7. **CharacterModels.kt**

   Data class utama

8. **CharacterRepository.kt**

   Penguhubung antara remote database, dan local database

9. **CharacterViewModelFactory.kt**

   digunakan untuk membuat instance dari viewmodel yang lain serta untuk memastikan viewmodel memliki depedensi yang di perlukan

10. **DetailFragment.kt**

    Fragment detail yang mengambil data dari navigation, dan view model yang bersangkutan.

11. **DetailViewModel.kt**

    Viewmodel dari detail fragment.

12. **HomeFragment.kt**

HomeFragment, sebagai fragment pertama yang terbuka saat membuka aplikasi

13. HomeViewModel.kt

Viewmodel dari HomeFragment

14. JikanApiService.kt

Wadah untuk membuat request terhadap api

15. MainActivity.kt

Berisi navigasi dan instalasi splash screen

16. RetrofitInstance.kt

Untuk melakukan koneksi terhadap endpoint yang sudah di set.

17. activity_main.xml

Berisi toolbar, dan navhost

18. fragment_detail.xml

Berisi image view, dan text view untuk ui detail fragment

19. fragment_home.xml

Berisi template wadah recylerview yang akan di populate oleh item_list

20. item_list.xml

Card atau ui dari recylerview

21. nav_graph.xml

Navigasi dan data passing

D. Tautan Git

https://github.com/au290/College-Work/tree/main/Semester-4/Pemrogaman-Mobile/Modul-5