LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 5

CONNECT TO THE INTERNET

Oleh:

Rifky Putra Mahardika NIM. 2310817210023

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
JUNI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE
MODUL 5

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

Nama Praktikan : Rifky Putra Mahardika
NIM : 2310817210023

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar NIM. 2210817210026	Muti`a Maulida S.Kom M.T.I NIP. 198810272019032013

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

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.

Aplikasi harus mempertahankan fitur-fitur yang dibuat pada modul sebelumnya.

A. Source Code

1. MainActivity.kt

```
1   package com.presca.modul5
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import androidx.activity.ComponentActivity
7   import androidx.activity.compose.setContent
8   import androidx.activity.viewModels
9   import androidx.compose.runtime.Composable
10  import androidx.compose.runtime.collectAsState
11  import androidx.compose.runtime.getValue
12  import androidx.compose.ui.platform.LocalContext
13  import androidx.lifecycle.ViewModel
```

```
14  import androidx.lifecycle.ViewModelProvider
15  import androidx.navigation.compose.NavHost
16  import androidx.navigation.compose.composable
17  import androidx.navigation.compose.rememberNavController
18  import com.presca.modul5.data.local.AppDatabase
19  import com.presca.modul5.data.remote.RetrofitInstance
20  import
    com.presca.modul5.data.repository.CountryRepositoryImpl
21  import
    com.presca.modul5.presentation.screens.CountryDetailScreen
22  import
    com.presca.modul5.presentation.screens.CountryListScreen
23  import com.presca.modul5.ui.theme.Modul5Theme
24  import com.presca.modul5.presentation.theme.ThemeViewModel
25  import
    com.presca.modul5.presentation.viewmodel.CountryViewModel
26  import
    com.presca.modul5.presentation.viewmodel.CountryViewModelFac
    tory
27
28  class MainActivity : ComponentActivity() {
29      private val db by lazy {
30          AppDatabase.getDatabase(this)
31      }
32      private val repository by lazy {
33          CountryRepositoryImpl(RetrofitInstance.api, db)
34      }
35      private val countryViewModel: CountryViewModel by
    viewModels {
36          CountryViewModelFactory(repository)
37      }
38      private val themeViewModel: ThemeViewModel by viewModels
    {
39          object : ViewModelProvider.Factory {
40              override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
41                  if
    (modelClass.isAssignableFrom(ThemeViewModel::class.java)) {
42                      @Suppress("UNCHECKED_CAST")
43                      return
    ThemeViewModel(applicationContext) as T
44                  }
45                  throw IllegalArgumentException("Unknown
    ViewModel class")
46              }
47          }
48      }
49
50      override fun onCreate(savedInstanceState: Bundle?) {
51          super.onCreate(savedInstanceState)
52          setContent {
53              val isDarkTheme by
```

```
    themeViewModel.isDarkTheme.collectAsState()
54
55              Modul5Theme(darkTheme = isDarkTheme) {
56                  AppNavigation(
57                      viewModel = countryViewModel,
58                      themeViewModel = themeViewModel
59                  )
60              }
61          }
62      }
63  }
64
65  @Composable
66  fun AppNavigation(
67      viewModel: CountryViewModel,
68      themeViewModel: ThemeViewModel
69  ) {
70      val navController = rememberNavController()
71      val context = LocalContext.current
72      val state by viewModel.state.collectAsState()
73      val isDarkTheme by
    themeViewModel.isDarkTheme.collectAsState()
74
75      NavHost(
76          navController = navController,
77          startDestination = "country_list"
78      ) {
79          composable("country_list") {
80              CountryListScreen(
81                  state = state,
82                  onRefresh = { viewModel.refreshCountries()
    },
83                  onClickDetail = { country ->

84  navController.navigate("detail/${country.id}")
85                  },
86                  onClickInfo = { url ->
87                      try {
88                          context.startActivity(
89                              Intent(Intent.ACTION_VIEW,
    Uri.parse(url))
90                          )
91                      } catch (e: Exception) {
92                          e.printStackTrace()
93                      }
94                  },
95                  onToggleTheme = {
    themeViewModel.toggleTheme() },
96                  isDarkTheme = isDarkTheme
97              )
98          }
99          composable("detail/{id}") { backStackEntry ->
```

```
            val id =
backStackEntry.arguments?.getString("id")?.toLongOrNull()

            when (val currentState = state) {
                is CountryViewModel.CountryState.Success ->
{
                    currentState.countries.find { it.id ==
id }?.let { country ->
                        CountryDetailScreen(
                            country = country,
                            navController = navController,
                            onClickInfo = {
                                context.startActivity(
                                    Intent(
                                        Intent.ACTION_VIEW,

Uri.parse(country.externalUrl)
                                    )
                                )
                            }
                        ) ?: run {
                            println("Error: Country with ID $id
not found in current state.")
                        }
                    }
                    else -> {
                        println("Error: Cannot display detail
screen in state: $currentState")
                    }
                }
            }
        }
}
```

Tabel 1. Source Code Jawaban Soal 1 MainActivity.kt

2. data/local/dao/CountryDao.kt

```
package com.presca.modul5.data.local.dao

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import com.presca.modul5.data.local.entity.CountryEntity

@Dao
interface CountryDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
```

```
    suspend fun insertAll(countries: List<CountryEntity>)

    @Query("SELECT * FROM countries")
    suspend fun getAllCountries(): List<CountryEntity>

    @Query("DELETE FROM countries")
    suspend fun clearAll()

    @Query("SELECT COUNT(*) FROM countries")
    suspend fun count(): Int
}
```

Tabel 2. Source Code Jawaban Soal 1 CountryDao.kt

3. data/local/entity/CountryEntity.kt

```
package com.presca.modul5.data.local.entity

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

@Entity(tableName = "countries")
data class CountryEntity(
    @PrimaryKey val id: Long,
    val name: String,
    val officialName: String,
    val flagUrl: String,
    val region: String,
    val capital: String,
    val description: String,
    val externalUrl: String,
    val lastUpdated: Long = System.currentTimeMillis()
)
```

Tabel 3. Source Code Jawaban Soal 1 CountryEntityc.kt

4. data/local/AppDatabase.kt

```
package com.presca.modul5.data.local

import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
import android.content.Context
import com.presca.modul5.data.local.dao.CountryDao
import com.presca.modul5.data.local.entity.CountryEntity

```

```
@Database(
    entities = [CountryEntity::class],
    version = 1,
    exportSchema = false
)
abstract class AppDatabase : RoomDatabase() {
    abstract fun countryDao(): CountryDao

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null

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

```

Tabel 4. Source Code Jawaban Soal 1 AppDatabase.kt

5. data/mapper/CountryMapper.kt

```
package com.presca.modul5.data.mapper

import com.presca.modul5.data.local.entity.CountryEntity
import com.presca.modul5.data.remote.response.Country
import com.presca.modul5.domain.model.CountryInfo
import java.text.DecimalFormat

object CountryMapper {
    fun mapResponseToEntity(country: Country, index: Long): CountryEntity {
        return CountryEntity(
            id = index,
            name = country.name.common,
            officialName = country.name.official ?: country.name.common,
            flagUrl = country.flags.png,
            region = country.region ?: "Unknown",
            capital = country.capital?.firstOrNull() ?: "Unknown",
            description = buildCountryDescription(country),
```

```
18              externalUrl =
    generateWikipediaUrl(country.name.common)
19          )
20      }
21
22      fun mapEntityToDomain(entity: CountryEntity):
    CountryInfo {
23          return CountryInfo(
24              id = entity.id,
25              name = entity.name,
26              officialName = entity.officialName,
27              flagUrl = entity.flagUrl,
28              region = entity.region,
29              capital = entity.capital,
30              description = entity.description,
31              externalUrl = entity.externalUrl
32          )
33      }
34
35      private fun buildCountryDescription(country: Country):
    String {
36          return """
37              |Nama Resmi: ${country.name.official ?:
    country.name.common}
38              |Ibukota: ${country.capital?.firstOrNull() ?:
    "Unknown"}
39              |Region: ${country.region ?: "Unknown"}
40              |Subregion: ${country.subregion ?: "Unknown"}
41              |Populasi:
    ${country.population?.formatWithCommas() ?: "Unknown"} jiwa
42              |Bahasa Resmi:
    ${country.languages?.values?.joinToString(", ") ?:
    "Unknown"}
43          """.trimMargin()
44      }
45
46      private fun Long.formatWithCommas(): String {
47          return DecimalFormat("#,###").format(this)
48      }
49
50      private fun generateWikipediaUrl(countryName: String):
    String {
51          return
    "https://en.wikipedia.org/wiki/${countryName.replace(" ",
    "_")}"
52      }
53  }
54
```

Tabel 5. Source Code Jawaban Soal 1 CountryMapper.kt

6. data/remote/response/Country.kt

```
1   package com.presca.modul5.data.remote.response
2
3   import kotlinx.serialization.SerialName
4   import kotlinx.serialization.Serializable
5
6   @Serializable
7   data class Country(
8       val name: Name,
9       val flags: Flags,
10      val region: String? = null,
11      val subregion: String? = null,
12      val capital: List<String>? = null,
13      val population: Long? = null,
14      val languages: Map<String, String>? = null,
15      val timezones: List<String>? = null
16  )
17
18  @Serializable
19  data class Name(
20      val common: String,
21      val official: String? = null,
22      @SerialName("nativeName")
23      val nativeNames: Map<String, NativeName>? = null
24  )
25
26  @Serializable
27  data class NativeName(
28      val official: String? = null,
29      val common: String? = null
30  )
31
32  @Serializable
33  data class Flags(
34      val png: String,
35      val svg: String? = null,
36      val alt: String? = null
37  )
```

Tabel 6. Source Code Jawaban Soal 1 Country.kt

7. data/remote/CountryApiService.kt

```
1   package com.presca.modul5.data.remote
2
3   import com.presca.modul5.data.remote.response.Country
4   import retrofit2.http.GET
5
```

```
6   interface CountryApiService {

7   @GET("v3.1/all?fields=name,flags,region,subregion,capital,p
    opulation,languages,timezones")
8       suspend fun getAllCountries(): List<Country>
9   }
```

Tabel 7. Source Code Jawaban Soal 1 CountryApiService.kt

8. data/remote/RetrofitInstance.kt

```
1   package com.presca.modul5.data.remote
2
3   import
    com.jakewharton.retrofit2.converter.kotlinx.serialization.a
    sConverterFactory
4   import kotlinx.serialization.json.Json
5   import okhttp3.MediaType.Companion.toMediaType
6   import okhttp3.OkHttpClient
7   import okhttp3.logging.HttpLoggingInterceptor
8   import retrofit2.Retrofit
9   import java.util.concurrent.TimeUnit
10
11  object RetrofitInstance {
12      private val json = Json {
13          ignoreUnknownKeys = true
14          isLenient = true
15          explicitNulls = false
16      }
17
18      private val loggingInterceptor =
    HttpLoggingInterceptor().apply {
19          level = HttpLoggingInterceptor.Level.BODY
20      }
21
22      private val httpClient = OkHttpClient.Builder()
23          .addInterceptor(loggingInterceptor)
24          .connectTimeout(30, TimeUnit.SECONDS)
25          .readTimeout(30, TimeUnit.SECONDS)
26          .build()
27
28      val api: CountryApiService by lazy {
29          Retrofit.Builder()
30              .baseUrl("https://restcountries.com/")
31              .client(httpClient)

32  .addConverterFactory(json.asConverterFactory("application/j
    son".toMediaType()))
33              .build()
34              .create(CountryApiService::class.java)
```

```
35      }
36  }
```

Tabel 8. Source Code Jawaban Soal 1 RetrofitInstance.kt

9. data/repository/CountryRepositoryImpl.kt

```
1   package com.presca.modul5.data.repository
2
3   import com.presca.modul5.data.local.AppDatabase
4   import com.presca.modul5.data.mapper.CountryMapper
5   import com.presca.modul5.data.remote.CountryApiService
6   import com.presca.modul5.domain.model.CountryInfo
7   import
    com.presca.modul5.domain.repository.CountryRepository
8   import kotlinx.coroutines.Dispatchers
9   import kotlinx.coroutines.flow.Flow
10  import kotlinx.coroutines.flow.flow
11  import kotlinx.coroutines.flow.flowOn
12  import kotlinx.coroutines.withContext
13
14  class CountryRepositoryImpl constructor(
15      private val api: CountryApiService,
16      private val db: AppDatabase
17  ) : CountryRepository {
18
19      override fun fetchCountries(): Flow<List<CountryInfo>>
    = flow {
20          try {
21              val cachedCountries =
    withContext(Dispatchers.IO) {
22                  db.countryDao().getAllCountries()
23              }
24
25              if (cachedCountries.isNotEmpty()) {
26                  emit(cachedCountries.map {
    CountryMapper.mapEntityToDomain(it) })
27              }
28
29              val countries = api.getAllCountries()
30              val entities = countries.mapIndexed { index,
    country ->
31                  CountryMapper.mapResponseToEntity(country,
    index.toLong())
32              }
33
34              withContext(Dispatchers.IO) {
35                  db.countryDao().clearAll()
36                  db.countryDao().insertAll(entities)
37              }
```

```
38
39              emit(entities.map {
    CountryMapper.mapEntityToDomain(it) })
40          } catch (e: Exception) {
41              val cachedCountries =
    withContext(Dispatchers.IO) {
42                  db.countryDao().getAllCountries()
43              }
44              if (cachedCountries.isNotEmpty()) {
45                  emit(cachedCountries.map {
    CountryMapper.mapEntityToDomain(it) })
46              } else {
47                  throw e
48              }
49          }
50      }.flowOn(Dispatchers.IO)
51
52      override suspend fun refreshCountries() {
53          try {
54              val countries = api.getAllCountries()
55              val entities = countries.mapIndexed { index,
    country ->
56                  CountryMapper.mapResponseToEntity(country,
    index.toLong())
57              }
58              db.countryDao().clearAll()
59              db.countryDao().insertAll(entities)
60          } catch (e: Exception) {
61              throw e
62          }
63      }
64  }
```

Tabel 9. Source Code Jawaban Soal 1 CountryRepositoryImpl.kt

10. domain/model/CountryInfo.kt

```
1   package com.presca.modul5.domain.model
2
3   data class CountryInfo(
4       val id: Long,
5       val name: String,
6       val officialName: String,
7       val flagUrl: String,
8       val region: String,
9       val capital: String,
10      val description: String,
11      val externalUrl: String
12  )
```

Tabel 10. Source Code Jawaban Soal 1 CountryInfo.kt

11. domain/repository/CountryRepository.kt

```
1   package com.presca.modul5.domain.repository
2
3   import com.presca.modul5.domain.model.CountryInfo
4   import kotlinx.coroutines.flow.Flow
5
6   interface CountryRepository {
7       fun fetchCountries(): Flow<List<CountryInfo>>
8       suspend fun refreshCountries()
9   }
```

Tabel 11. Source Code Jawaban Soal 1 CountryRepository.kt

12. presentation/screens/CountryDetailScreen.kt

```
1   package com.presca.modul5.presentation.screens
2
3   import androidx.compose.foundation.layout.*
4   import androidx.compose.foundation.rememberScrollState
5   import androidx.compose.foundation.verticalScroll
6   import androidx.compose.material.icons.Icons
7   import androidx.compose.material.icons.filled.ArrowBack
8   import androidx.compose.material.icons.filled.Info
9   import androidx.compose.material3.*
10  import androidx.compose.runtime.Composable
11  import androidx.compose.ui.Modifier
12  import androidx.compose.ui.draw.clip
13  import androidx.compose.ui.unit.dp
14  import androidx.navigation.NavController
15  import
    com.bumptech.glide.integration.compose.ExperimentalGlideCom
    poseApi
16  import com.bumptech.glide.integration.compose.GlideImage
17  import com.presca.modul5.domain.model.CountryInfo
18
19  @OptIn(ExperimentalGlideComposeApi::class,
    ExperimentalMaterial3Api::class)
20  @Composable
21  fun CountryDetailScreen(
22      country: CountryInfo,
23      navController: NavController,
24      onClickInfo: () -> Unit
25  ) {
26      Scaffold(
27          topBar = {
```

```
28              TopAppBar(
29                  title = { Text(country.name) },
30                  navigationIcon = {
31                      IconButton(onClick = {
    navController.popBackStack() }) {
32                          Icon(Icons.Default.ArrowBack,
    contentDescription = "Kembali")
33                      }
34                  },
35                  actions = {
36                      IconButton(onClick = onClickInfo) {
37                          Icon(Icons.Default.Info,
    contentDescription = "Info")
38                      }
39                  }
40              )
41          }
42      ) { padding ->
43          Column(
44              modifier = Modifier
45                  .padding(padding)
46                  .padding(16.dp)
47                  .verticalScroll(rememberScrollState())
48          ) {
49              GlideImage(
50                  model = country.flagUrl,
51                  contentDescription = "Bendera
    ${country.name}",
52                  modifier = Modifier
53                      .fillMaxWidth()
54                      .height(200.dp)
55                      .clip(MaterialTheme.shapes.medium)
56              )
57
58              Spacer(modifier = Modifier.height(24.dp))
59
60              Text(
61                  text = "Nama Resmi:",
62                  style = MaterialTheme.typography.labelLarge
63              )
64              Text(
65                  text = country.officialName,
66                  style = MaterialTheme.typography.bodyLarge,
67                  modifier = Modifier.padding(bottom = 16.dp)
68              )
69
70              Text(
71                  text = "Informasi:",
72                  style = MaterialTheme.typography.labelLarge
73              )
74              Text(
75                  text = country.description,
```

```
                    style = MaterialTheme.typography.bodyLarge,
                    modifier = Modifier.padding(bottom = 16.dp)
                )

                Button(
                    onClick = onClickInfo,
                    modifier = Modifier.fillMaxWidth()
                ) {
                    Text("Buka Info Lengkap")
                }
            }
        }
}
```

Tabel 12. Source Code Jawaban Soal 1 CountryDetailScreen.kt

13. presentation/screens/CountryListScreen.kt

```
package com.presca.modul5.presentation.screens

import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.*
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import
com.bumptech.glide.integration.compose.ExperimentalGlideCom
poseApi
import com.bumptech.glide.integration.compose.GlideImage
import com.presca.modul5.domain.model.CountryInfo
import
com.presca.modul5.presentation.viewmodel.CountryViewModel

@OptIn(ExperimentalGlideComposeApi::class,
ExperimentalMaterial3Api::class)
@Composable
fun CountryListScreen(
    state: CountryViewModel.CountryState,
    onRefresh: () -> Unit,
    onClickDetail: (CountryInfo) -> Unit,
    onClickInfo: (String) -> Unit,
    onToggleTheme: () -> Unit,
```

```
    isDarkTheme: Boolean
) {
    Scaffold(
        topBar = {
            CenterAlignedTopAppBar(
                title = { Text("Negara di Dunia") },
                navigationIcon = {
                    IconButton(onClick = { /* TODO */ }) {
                        Icon(Icons.Filled.Public,
contentDescription = null)
                    }
                },
                actions = {
                    IconButton(onClick = onToggleTheme) {
                        Icon(
                            imageVector = if (isDarkTheme)
Icons.Filled.LightMode else Icons.Filled.DarkMode,
                            contentDescription = "Toggle
Theme"
                        )
                    }
                }
            )
        }
    ) { innerPadding ->
        Box(modifier = Modifier.padding(innerPadding)) {
            when (state) {
                is CountryViewModel.CountryState.Loading ->
LoadingView()
                is CountryViewModel.CountryState.Error ->
ErrorView(state.message, onRefresh)
                is CountryViewModel.CountryState.Success ->
                    CountryListView(
                        countries = state.countries,
                        onClickDetail = onClickDetail,
                        onClickInfo = onClickInfo
                    )
            }
        }
    }
}

@Composable
fun LoadingView() {
    Box(
        modifier = Modifier.fillMaxSize(),
        contentAlignment = Alignment.Center
    ) {
        CircularProgressIndicator()
    }
}

```

```
@Composable
fun ErrorView(message: String, onRefresh: () -> Unit) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(message)
        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = onRefresh) {
            Text("Coba Lagi")
        }
    }
}

@OptIn(ExperimentalGlideComposeApi::class)
@Composable
fun CountryListView(
    countries: List<CountryInfo>,
    onClickDetail: (CountryInfo) -> Unit,
    onClickInfo: (String) -> Unit
) {
    LazyColumn(
        modifier = Modifier.fillMaxSize(),
        contentPadding = PaddingValues(8.dp)
    ) {
        items(countries) { country ->
            CountryCard(
                country = country,
                onClickDetail = { onClickDetail(country) },
                onClickInfo = {
onClickInfo(country.externalUrl) }
            )
        }
    }
}

@OptIn(ExperimentalGlideComposeApi::class)
@Composable
fun CountryCard(
    country: CountryInfo,
    onClickDetail: () -> Unit,
    onClickInfo: () -> Unit
) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 8.dp),
        shape = RoundedCornerShape(12.dp)
    ) {
        Column(modifier = Modifier.padding(12.dp)) {
            Row(verticalAlignment =
```

```
    Alignment.CenterVertically) {
127                 GlideImage(
128                     model = country.flagUrl,
129                     contentDescription = country.name,
130                     modifier = Modifier
131                         .size(80.dp)
132                         .clip(RoundedCornerShape(8.dp))
133                 )
134                 Spacer(modifier = Modifier.width(16.dp))
135                 Column(modifier = Modifier.weight(1f)) {
136                     Text(country.name, fontSize = 18.sp)
137                     Text(
138                         text = "Lokasi: ${country.region}",
139                         fontSize = 14.sp
140                     )
141                 }
142             }
143             Spacer(modifier = Modifier.height(12.dp))
144             Row(
145                 modifier = Modifier.fillMaxWidth(),
146                 horizontalArrangement =
    Arrangement.spacedBy(8.dp)
147             ) {
148                 Button(
149                     onClick = onClickInfo,
150                     modifier = Modifier.weight(1f)
151                 ) {
152                     Text("Info")
153                 }
154                 Button(
155                     onClick = onClickDetail,
156                     modifier = Modifier.weight(1f)
157                 ) {
158                     Text("Detail")
159                 }
160             }
161         }
162     }
163 }
```

Tabel 13. Source Code Jawaban Soal 1 CountryListScreen.kt

14. presentation/theme/Theme.kt

```
1   package com.presca.modul5.ui.theme
2
3   import androidx.compose.foundation.isSystemInDarkTheme
4   import androidx.compose.material3.MaterialTheme
5   import androidx.compose.material3.darkColorScheme
6   import androidx.compose.material3.lightColorScheme
```

```
7   import androidx.compose.runtime.Composable
8
9   private val DarkColorScheme = darkColorScheme(
10      primary = LightNavyBlue,
11      secondary = PurpleGrey80,
12      tertiary = Pink80
13  )
14
15  private val LightColorScheme = lightColorScheme(
16      primary = NavyBlue,
17      secondary = PurpleGrey40,
18      tertiary = Pink40
19  )
20
21  @Composable
22  fun Modul5Theme(
23      darkTheme: Boolean = isSystemInDarkTheme(),
24      content: @Composable () -> Unit
25  ) {
26      val colorScheme = if (darkTheme) DarkColorScheme else
    LightColorScheme
27
28      MaterialTheme(
29          colorScheme = colorScheme,
30          typography = Typography,
31          content = content
32      )
33  }
34
```

Tabel 14. Source Code Jawaban Soal 1 Theme.kt

15. presentation/theme/ThemeViewModel.kt

```
1   package com.presca.modul5.presentation.theme
2
3   import android.content.Context
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.viewModelScope
6   import androidx.datastore.core.DataStore
7   import androidx.datastore.preferences.core.Preferences
8   import
    androidx.datastore.preferences.core.booleanPreferencesKey
9   import androidx.datastore.preferences.core.edit
10  import androidx.datastore.preferences.preferencesDataStore
11  import kotlinx.coroutines.flow.MutableStateFlow
12  import kotlinx.coroutines.flow.StateFlow
13  import kotlinx.coroutines.flow.asStateFlow
14  import kotlinx.coroutines.flow.map
15  import kotlinx.coroutines.launch
```

```
16
17  private val Context.dataStore: DataStore<Preferences> by
    preferencesDataStore(name = "theme_preferences")
18
19  class ThemeViewModel(private val applicationContext:
    Context) : ViewModel() {
20
21      private val IS_DARK_THEME_KEY =
    booleanPreferencesKey("is_dark_theme")
22
23      private val _isDarkTheme = MutableStateFlow(false)
24      val isDarkTheme: StateFlow<Boolean> =
    _isDarkTheme.asStateFlow()
25
26      init {
27          viewModelScope.launch {
28              applicationContext.dataStore.data
29                  .map { preferences ->
30                      preferences[IS_DARK_THEME_KEY] ?: false
31                  }
32                  .collect { isDark ->
33                      _isDarkTheme.value = isDark
34                  }
35          }
36      }
37
38      fun toggleTheme() {
39          viewModelScope.launch {
40              val currentTheme = _isDarkTheme.value
41              applicationContext.dataStore.edit { preferences
    ->
42                  preferences[IS_DARK_THEME_KEY] =
    !currentTheme
43              }
44          }
45      }
46  }
47
```

Tabel 15. Source Code Jawaban Soal 1 ThemeViewModel.kt

16. presentation/theme/CountryViewModel.kt

```
1   package com.presca.modul5.presentation.viewmodel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.viewModelScope
5   import com.presca.modul5.domain.model.CountryInfo
6   import
    com.presca.modul5.domain.repository.CountryRepository
```

```
7  import kotlinx.coroutines.flow.MutableStateFlow
8  import kotlinx.coroutines.flow.StateFlow
9  import kotlinx.coroutines.flow.asStateFlow
10 import kotlinx.coroutines.launch
11
12 class CountryViewModel(private val repository:
   CountryRepository) : ViewModel() {
13     sealed class CountryState {
14         object Loading : CountryState()
15         data class Success(val countries:
   List<CountryInfo>) : CountryState()
16         data class Error(val message: String) :
   CountryState()
17     }
18
19     private val _state =
   MutableStateFlow<CountryState>(CountryState.Loading)
20     val state: StateFlow<CountryState> =
   _state.asStateFlow()
21
22     init {
23         fetchCountries()
24     }
25
26     fun fetchCountries() {
27         viewModelScope.launch {
28             _state.value = CountryState.Loading
29             try {
30                 repository.fetchCountries().collect {
   countries ->
31                     _state.value = if (countries.isEmpty())
   {
32                         CountryState.Error("Tidak ada data
   negara yang ditemukan")
33                     } else {
34                         CountryState.Success(countries)
35                     }
36                 }
37             } catch (e: Exception) {
38                 _state.value = CountryState.Error(
39                     "Gagal memuat data: ${e.message ?:
   "Terjadi kesalahan"}"
40                 )
41             }
42         }
43     }
44
45     fun refreshCountries() {
46         viewModelScope.launch {
47             _state.value = CountryState.Loading
48             try {
49                 repository.refreshCountries()
```

```
50                  fetchCountries()
51              } catch (e: Exception) {
52                  _state.value = CountryState.Error(
53                      "Gagal menyegarkan data: ${e.message ?:
    "Terjadi kesalahan"}"
54                  )
55              }
56          }
57      }
58  }
```

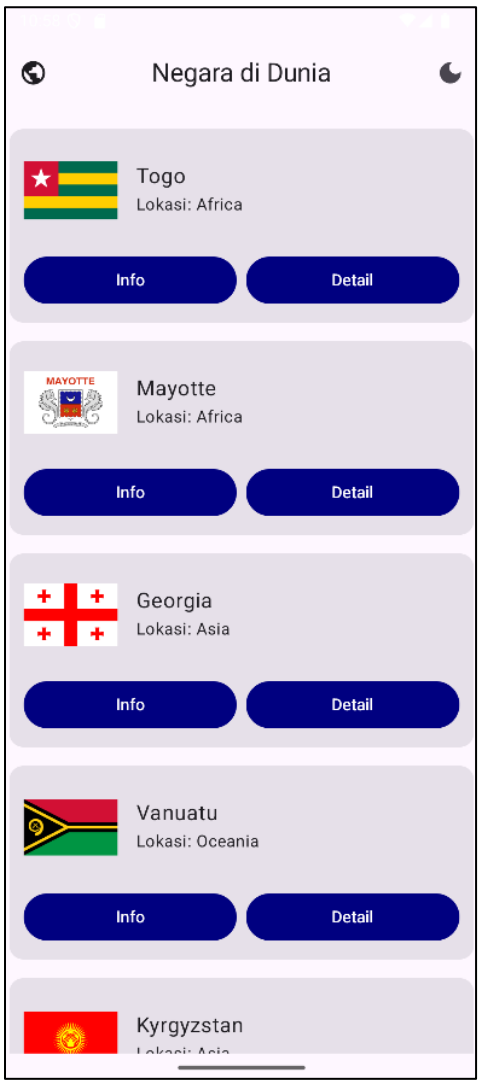
Tabel 16. Source Code Jawaban Soal 1 CountryViewModel.kt

17. presentation/theme/CountryViewModelFactory.kt

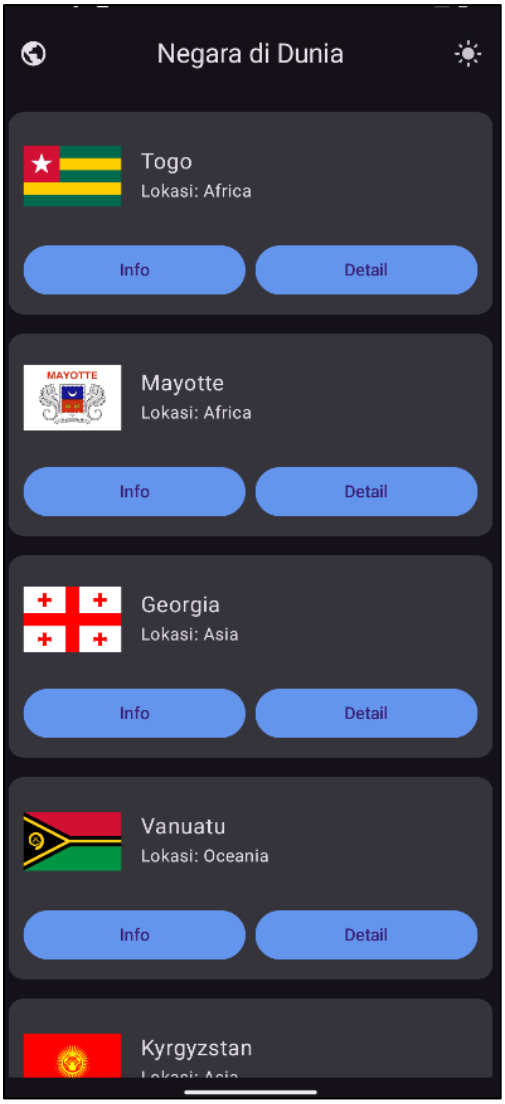
```
1   package com.presca.modul5.presentation.viewmodel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5   import
    com.presca.modul5.domain.repository.CountryRepository
6
7   class CountryViewModelFactory(
8       private val repository: CountryRepository
9   ) : ViewModelProvider.Factory {
10      override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
11          if
    (modelClass.isAssignableFrom(CountryViewModel::class.java))
    {
12              @Suppress("UNCHECKED_CAST")
13              return CountryViewModel(repository) as T
14          }
15          throw IllegalArgumentException("Unknown ViewModel
    class")
16      }
17  }
18
```

Tabel 17. Source Code Jawaban Soal 1 CountryViewModelFactory.kt

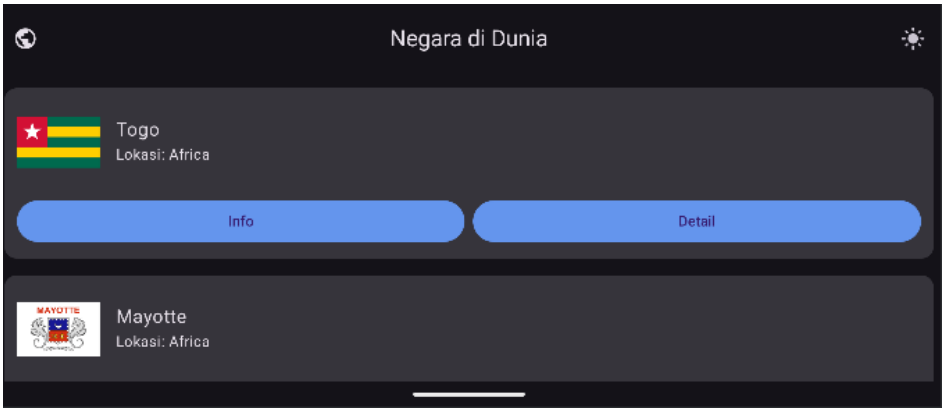
B. Output Program



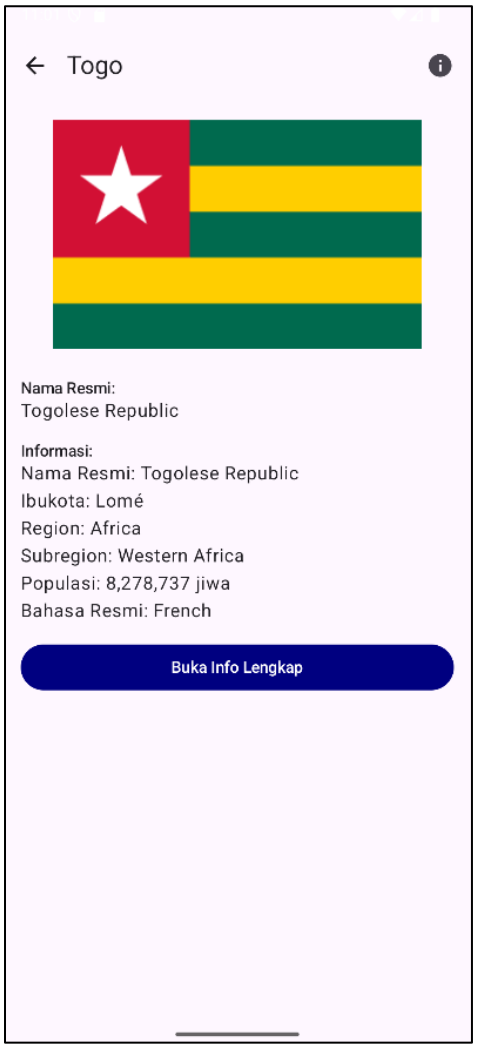
Gambar 1. Screenshot Hasil Jawaban Soal 1 Tampilan List



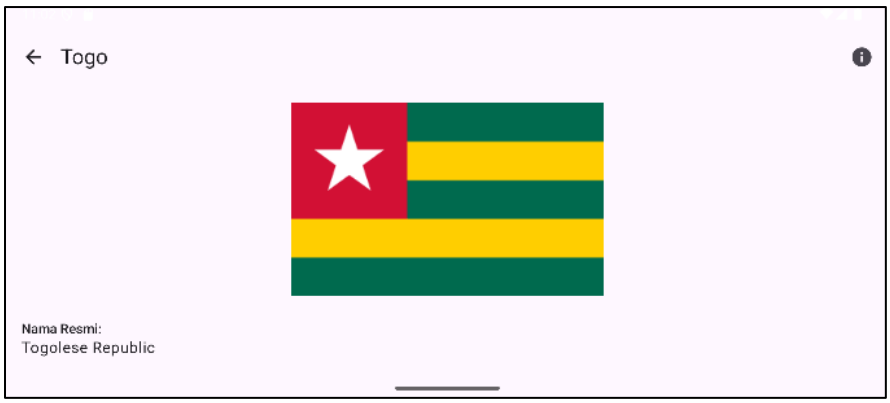
Gambar 2. Screenshot Hasil Jawaban Soal 1 Tampilan List Dark Mode



Gambar 3. Screenshot Hasil Jawaban Soal 1 Tampilan List Landscape

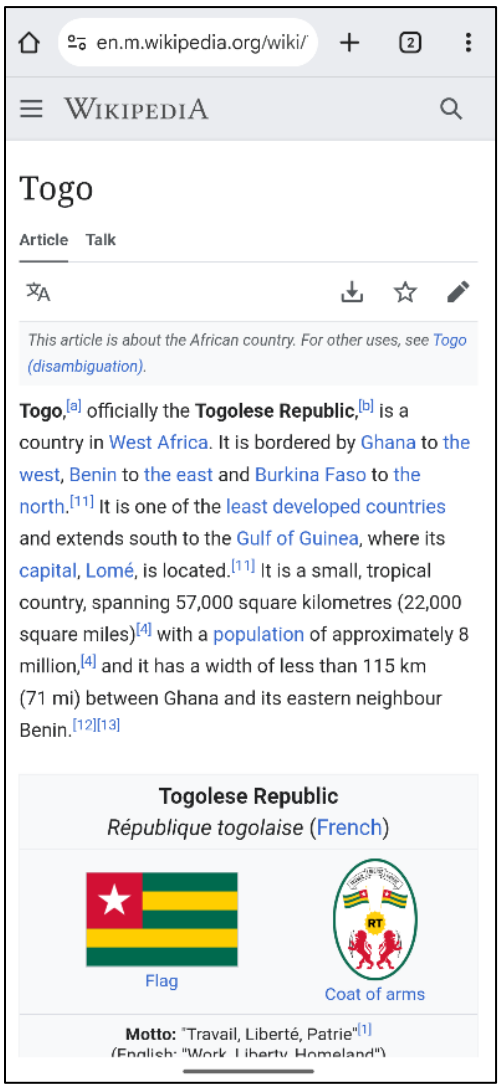


Gambar 4. Screenshot Hasil Jawaban Soal 1 Tampilan Detail



Gambar 5. Screenshot Hasil Jawaban Soal 1 Tampilan Detail Landscape

Gambar 6. Screenshot Hasil Jawaban Soal 1 Tampilan Detail Dark Mode



Gambar 7. Screenshot Hasil Jawaban Soal 1 Hasil Tombol Info

C. Pembahasan

Berikut adalah penjelasan untuk soal nomor 1:

1. MainActivity.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul5`

- Pada baris [3] hingga [26], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [28], `class MainActivity : ComponentActivity()` ini digunakan sebagai titik awal aplikasi yang akan mengatur tampilan aplikasi.
- Pada baris [29], `private val db by lazy { AppDatabase.getDatabase(this)` digunakan untuk mendeklarasikan properti db yang akan menampung instance database Room (`AppDatabase`)
- Pada baris [32], `private val repository by lazy { CountryRepositoryImpl(RetrofitInstance.api, db)` digunakan untuk mendeklarasikan properti repository yang menampung instance dari `CountryRepositoryImpl`.
- Pada baris [35], `private val themeViewModel: ThemeViewModel by viewModels` digunakan untuk mendeklarasikan properti `themeViewModel.`
- Pada baris [65] dan [66], `@Composable` dan `fun AppNavigation` pada bagian ini merupakan fungsi Composable yang bertanggung jawab untuk mendefinisikan grafik navigasi pada aplikasi.
- Pada baris [71], `val context = LocalContext.current` digunakan untuk mendapatkan `Context` saat ini dalam lingkungan Composable.
- Pada baris [72], `val state by viewModel.state.collectAsState()` digunakan untuk mengamati state data negara dari `CountryViewModel`.
- Pada baris [75], `NavHost` adalah komponen inti dari navigasi Compose yang bertanggung jawab untuk menampilkan Composable yang sesuai dengan rute navigasi saat ini.
- Pada baris [79], `composable("country_list")` pada bagian digunakan untuk mendefinisikan sebuah "tujuan" navigasi (destination) untuk layar daftar negara.

- Pada baris [99], `composable("detail/{id}") { backStackEntry -> ... }` pada bagian ini digunakan untuk mendefinisikan tujuan navigasi untuk layar detail negara.

2. CountryDao.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul5.data.local.dao`
- Pada baris [3] hingga [7], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [9], `@Dao` adalah menandakan bahwa CountryDao adalah antarmuka Dao Room yang dimana Room akan memproses antarmuka ini dan menggenerasikan kode yang diperlukan untuk interaksi dengan database.
- Pada baris [11], `@Insert` adalah menandakan bahwa nantinya pada bagian ini akan menyisipkan data pada bagian ini.
- Pada baris [14], `@Query` merupakan argument berupa string SQL, yang dimana pada bagian ini untuk mengambil semua kolom bernama `countries`.
- Pada baris [17], Query digunakan untuk mengapus semua baris tabel `countries`.
- Pada baris [20], Query digunakan untuk menghitung semua baris tabel `countries`.

3. CountryEntity.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul5.data.local.entity`
- Pada baris [3] dan [4], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.

- Pada baris [6], `@Entity` digunakan untuk memberitahu Room bahwa kelas ini adalah sebuah entitas database (yaitu, akan dipetakan ke sebuah tabel).
- Pada baris [7], `data class CountryEntity` digunakan untuk mendeklarasikan sebuah data class Kotlin.
- Pada baris [8], `@PrimaryKey` pada bagian ini menandai bahwa properti id sebagai Primary Key untuk tabel `countries`.
- Pada baris [9] hingga [16], berbagai property ditetapkan di dalam tabel `countries`.

4. AppDatabase.kt:

- Pada intinya, `AppDatabase` ini akan mengonfigurasi database, mengaitkan entitas dengan tabel, menyediakan akses ke DAO, dan memastikan bahwa hanya ada satu instance database yang berjalan di seluruh aplikasi, yang merupakan praktik terbaik untuk efisiensi dan mencegah masalah database.
- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul5.data.local`
- Pada baris [3] hingga [8], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [10], `@Database` digunakan untuk konfigurasi utama untuk database Room.
- Pada baris [15], `abstract class AppDatabase : RoomDatabase()` digunakan untuk mendefinisikan kelas `AppDatabase` sebagai kelas `abstract`.
- Pada baris [18], `companion object` mendefinisikan bahwa semua anggota yang dideklarasikan di dalam `companion object` dapat diakses langsung menggunakan nama kelas, tanpa perlu membuat instance kelas tersebut.
- Pada baris [22], `fun getDatabase(context: Context): AppDatabase {…}` ini dgunakan untuk mendapatkan satu-satunya instance dari `AppDatabase`.

5. CountryMapper.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul5.data.local`
- Pada baris [3] hingga [6], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [8], `object CountryMapper {…}` ini digunakan untuk Deklarasi object dalam Kotlin membuat sebuah singleton, yang dimana hanya akan ada satu instance dari `CountryMapper` di seluruh aplikasi.
- Pada baris [9], `fun mapResponseToEntity(country: Country, index: Long): CountryEntity { ..}` digunakan untuk mengkonversi objek `Country` (dari API) menjadi objek `CountryEntity` (untuk database Room).
- Pada baris [22], `fun mapEntityToDomain(entity: CountryEntity): CountryInfo {...}` digunakan untuk mengkonversi objek `CountryEntity` (dari database) menjadi objek `CountryInfo` (untuk lapisan domain/UI).
- Pada baris [35], `private fun buildCountryDescription(country: Country): String {...}` digunakan untuk membuat string deskripsi yang diformat dari objek `Country` yang lebih lengkap.
- Pada baris [46], `private fun Long.formatWithCommas(): String {…}` pada bagian ini digunakan untuk memformat angka Long (seperti populasi) dengan pemisah ribuan.
- Pada baris [50], `private fun generateWikipediaUrl(countryName: String): String {…}` digunakan untuk membuat URL Wikipedia berdasarkan nama negara.

6. **Country.kt:**
    - Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.data.remote.response`.
    - Pada baris [3] dan [4], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
    - Pada baris [6], `@Serializable` digunakan untuk memberitahu KotlinX Serialization bahwa `Country` adalah kelas yang dapat diserialisasi. Room akan menggenerasikan kode untuk mengkonversi objek `Country` ke/dari representasi JSON.
    - Pada baris [7], `data class Country(...)` merepresentasikan struktur data utama untuk satu negara yang diterima dari API.
    - Pada baris [19], `data class Name(…)` digunakan untuk merepresentasikan berbagai format nama negara.
    - Pada baris [27], `data class NativeName(...)` digunakan untuk merepresentasikan nama asli dalam format resmi dan umum.
    - Pada baris [33], `data class Flags(...)` Merepresentasikan berbagai format URL bendera dan teks alternatif.

7. **CountryApiService.kt:**
    - Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.data.remote`.
    - Pada baris [3] dan [4], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
    - Pada baris [6], `interface CountryApiService` digunakan untuk mendefinisikan antarmuka `CountryApiService`.

- Pada baris [7], `@GET("v3.1/all?fields=name,flags,region,subregion,capital,population,languages,timezones")` digunakan untuk mengirim permintaan HTTP GET ke server sesuai dengan yang diminta (misal bagian ini akan meminta nama, flags, region, hingga timezones).
- Pada baris [8], `suspend fun getAllCountries(): List<Country>` merupakan metode yang akan melakukan panggilan API.

8. RetrofitInstance.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.data.remote`.
- Pada baris [3] hingga [9], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [11], `object RetrofitInstance {...}` mendefinisikan bahwa hanya ada satu instance dari RetrofitInstance yang akan dibuat di seluruh aplikasi.
- Pada baris [12], `private val json = Json { ... }` merupakan dari bagian konfigurasi untuk bagaimana JSON akan diurai (parse).
- Pada baris [18], `private val loggingInterceptor = HttpLoggingInterceptor().apply { level = HttpLoggingInterceptor.Level.BODY }` digunakan untuk mengatur tingkat logging dan mencatat seluruh body permintaan dan respons.
- Pada baris [22], `private val httpClient = OkHttpClient.Builder().addInterceptor(loggingInterceptor) .connectTimeout(30, TimeUnit.SECONDS) .readTimeout(30, TimeUnit.SECONDS) .build()` digunakan untuk membangun instance OkHttpClient yang akan digunakan oleh Retrofit.

- Pada baris [28], `val api: CountryApiService by lazy {...}` digunakan untuk mendefinisikan properti API (termasuk endpoint) yang akan menyediakan instance `CountryApiService` yang siap pakai.

9. CountryrepositoryImpt.kt:

- Pada intinya, CountryRepositoryImpl.kt adalah implementasi konkret dari antarmuka CountryRepository (yang ada di lapisan domain) dan berfungsi untuk mengabstraksi asal-usul data (apakah dari jaringan, database lokal, atau memori), menerapkan logika bisnis seperti strategi caching, dan menyediakan data yang konsisten ke lapisan di atasnya (ViewModel).

10. CountryInfo.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.domain.model`
- Pada baris [3], `data class CountryInfo(...)` digunakan untuk mendefinisikan struktur data sebuah objek `CountryInfo` dan secara otomatis menyediakan fungsi-fungsi dasar yang sangat sering dibutuhkan saat bekerja dengan data.

11. CountryRepository.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.domain.repository`
- Pada baris [6], `interface CountryRepository {}` digunakan untuk mendefinisikan metode yang fungsionalitas yang berkaitan dengan data negara, tanpa menyediakan implementasi (detail bagaimana fungsionalitas itu bekerja).

12. CountryDetailScreen.kt:

- Pada intinya, `CountryDetailScreen.kt` memiliki berbagai fungsi `@Composable` yang mendefinisikan tampilan antarmuka pengguna (UI) untuk layar detail negara. Bagian ini menampilkan informasi rinci tentang sebuah negara, seperti bendera, nama resmi, dan deskripsi pada tiap-tiap negara.

13. CountryListScreen.kt:

- Pada intinya, `CountryListScreen.kt` memiliki berbagai fungsi `@Composable` yang mendefinisikan tampilan antarmuka pengguna (UI) untuk layar daftar (list) negara. Bagian ini bertanggung jawab untuk menampilkan daftar negara, menunjukkan status loading atau error, dan memungkinkan interaksi seperti menyegarkan data atau beralih tema.

14. CountryViewModel.kt:

- File `CountryViewModel.kt` ini berguna untuk mengelola state UI terkait data negara, berinteraksi dengan `CountryRepository` untuk mengambil data, dan menyediakan state ini ke UI melalui `StateFlow` agar UI dapat bereaksi secara reaktif terhadap perubahan data, loading, atau error.

15. CountryViewModelFactory.kt:

- File `CountryViewModelFactory.kt` ini berfungsi sebagai pabrik (factory) untuk membuat instance dari `CountryViewModel. ViewModelFactory` menjadi sangat penting ketika `ViewModel` memiliki dependensi di konstruktornya (misalnya, `CountryRepository`)
- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.presentation.viewmodel`

- Pada baris [3] hingga [5], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [8], `(private val repository: CountryRepository)` digunakan sebagai konstruktor utama untuk `CountryViewModelFactory` yang digunakan untuk menerima sebuah instance dari `CountryRepository`.
- Pada baris [10], create dalam `ViewModelProvider.Factory` berfungsi sebagai jantung dari proses inisialisasi `ViewModel` yang memiliki dependensi kustom, ketika sistem Android perlu membuat instance `ViewModel` baru (misalnya, saat Activity pertama kali dibuat atau setelah rotasi layar), ia akan memanggil metode create ini, meneruskan objek Class dari `ViewModel` yang diinginkan (misalnya, `CountryViewModel::class.java`) sebagai parameter `modelClass`.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/Prescaa/Kuliah/tree/master/Praktikum%20Mobile/MODUL%205