LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 5

CONNECT TO THE INTERNET

Oleh:

Muhammad Fauzan Ahsani **NIM. 2310817310009**

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 Prakitkum ini dikerjakan oleh:

Nama Praktikan : Muhammad Fauzan Ahsani
NIM : 2310817310009

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 menambagkan 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 lubrary 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-nya: https://developer.themoviedb.org/docs/getting-started
e. Implementasikan konsep data persistence (misalnya, offline-first app, pengaturan dark/light node, fitur favourite, dll).
f. Gunakan caching strategy pada Room.
g. Untuk Modul 5, bebas memilih UI yang inigin 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 MainActivity.kt

```
 1  package com.example.treblewinner
 2
 3  import android.os.Bundle
 4  import androidx.activity.ComponentActivity
 5  import androidx.activity.compose.setContent
 6  import
    com.example.treblewinner.presentation.navigation.Navigation
 7  import
    com.example.treblewinner.presentation.theme.TrebleWinnerTheme
 8  import dagger.hilt.android.AndroidEntryPoint
 9
10  @AndroidEntryPoint
11  class MainActivity : ComponentActivity() {
12      override fun onCreate(savedInstanceState: Bundle?) {
13          super.onCreate(savedInstanceState)
14          setContent {
15              TrebleWinnerTheme() {
16                  Navigation()
17              }
```

```
        }
    }
}
```

2. TrebleWinnerApplication.kt

Tabel 2. Source Code TrebleWinnerApplication.kt

```
package com.example.treblewinner

import android.app.Application
import dagger.hilt.android.HiltAndroidApp

@HiltAndroidApp
class TrebleWinnerApplication : Application()
```

3. AndroidManifest.xml

Tabel 3. Source Code AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/androi
d"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission
android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:name=".TrebleWinnerApplication"

android:dataExtractionRules="@xml/data_extraction_rules"
        android:enableOnBackInvokedCallback="true"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.TrebleWinner"
        android:usesCleartextTraffic="true"

        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.TrebleWinner">
```

```
25              <intent-filter>
26                  <action
    android:name="android.intent.action.MAIN" />
27
28                  <category
    android:name="android.intent.category.LAUNCHER" />
29              </intent-filter>
30          </activity>
31      </application>
32
33  </manifest>
```

Package: data

4. RepositoryModule.kt

Tabel 4. Source Code RepositoryModule.kt

```
 1  package com.example.treblewinner.data.di
 2
 3  import
    com.example.treblewinner.data.repository.TrebleWinnerRe
    positoryImpl
 4  import
    com.example.treblewinner.domain.repository.TrebleWinner
    Repository
 5  import dagger.Binds
 6  import dagger.Module
 7  import dagger.hilt.InstallIn
 8  import dagger.hilt.components.SingletonComponent
 9  import javax.inject.Singleton
10
11  @Module
12  @InstallIn(SingletonComponent::class)
13  abstract class RepositoryModule {
14
15      @Binds
16      @Singleton
17      abstract fun bindTrebleWinnerRepository(
18          impl: TrebleWinnerRepositoryImpl
19      ): TrebleWinnerRepository
20  }
```

5. Converters.kt

Tabel 5. Source Code Converters.kt

```
 1 package com.example.treblewinner.data.local.converters
 2
 3 import androidx.room.TypeConverter
 4 import com.google.gson.Gson
 5 import com.google.gson.reflect.TypeToken
 6
 7 class Converters {
 8     private val gson = Gson()
 9
10     @TypeConverter
11     fun fromStringList(value: List<String>): String {
12         return gson.toJson(value)
13     }
14
15     @TypeConverter
16     fun toStringList(value: String): List<String> {
17         val listType = object :
   TypeToken<List<String>>() {}.type
18         return gson.fromJson(value, listType)
19     }
20 }
```

6. TrebleWinnerDao.kt

Tabel 6. Source Code TrebleWinnerDao.kt

```
 1 package com.example.treblewinner.data.local.dao
 2
 3 import androidx.room.Dao
 4 import androidx.room.Insert
 5 import
   androidx.room.OnConflictStrategy.Companion.IGNORE
 6 import androidx.room.Query
 7 import androidx.room.Transaction
 8 import androidx.room.Update
 9 import
   com.example.treblewinner.data.local.entity.ClubEntity
10 import
   com.example.treblewinner.data.local.entity.CompetitionE
   ntity
11 import
   com.example.treblewinner.data.local.relation.ClubWithCo
   mpetitions
12 import kotlinx.coroutines.flow.Flow
```

```
13
14 @Dao
15 interface TrebleWinnerDao {
16     @Query("SELECT * FROM clubs ORDER BY confederation
   ASC")
17     fun getAllClubs(): Flow<List<ClubEntity>>
18
19     @Query("SELECT * FROM competitions ORDER BY
   confederation ASC")
20     fun getAllCompetitions():
   Flow<List<CompetitionEntity>>
21
22     @Transaction
23     @Query("SELECT * FROM clubs WHERE id = :clubId")
24     suspend fun getClubWithCompetitions(clubId:
   String): ClubWithCompetitions?
25
26     @Insert(onConflict = IGNORE)
27     suspend fun insertClubs(clubs: List<ClubEntity>)
28
29     @Insert(onConflict = IGNORE)
30     suspend fun insertCompetitions(competitions:
   List<CompetitionEntity>)
31
32     @Update
33     suspend fun updateClub(club: ClubEntity)
34
35
36     @Query("SELECT * FROM clubs WHERE is_bookmarked =
   1")
37     fun getBookmarkedClubs(): Flow<List<ClubEntity>>
38
39     @Query("DELETE FROM clubs WHERE is_bookmarked = 0")
40     suspend fun deleteAllNonBookmarked()
41
42     @Query("SELECT EXISTS(SELECT * FROM clubs WHERE id
   = :clubId AND is_bookmarked = 1)")
43     suspend fun isClubsBookmarked(clubId: String):
   Boolean
44
45     @Query("UPDATE clubs SET is_bookmarked = NOT
   is_bookmarked WHERE id = :clubId")
46     suspend fun toggleClubBookmark(clubId: String)
47 }
```

7. AppDatabase.kt

Tabel 7. Source Code AppDatabase.kt

```
 1  package com.example.treblewinner.data.local.db
 2
 3  import androidx.room.Database
 4  import androidx.room.RoomDatabase
 5  import androidx.room.TypeConverters
 6  import
    com.example.treblewinner.data.local.converters.Converte
    rs
 7  import
    com.example.treblewinner.data.local.dao.TrebleWinnerDao
 8  import
    com.example.treblewinner.data.local.entity.ClubEntity
 9  import
    com.example.treblewinner.data.local.entity.CompetitionE
    ntity
10  import
    com.example.treblewinner.data.local.entity.ClubCompetit
    ionCrossRef
11
12  @Database(
13      entities = [
14          ClubEntity::class,
15          CompetitionEntity::class,
16          ClubCompetitionCrossRef::class
17      ],
18      version = 1,
19      exportSchema = false
20  )
21
22  @TypeConverters(Converters::class)
23  abstract class AppDatabase : RoomDatabase() {
24      abstract fun trebleWinnerDao(): TrebleWinnerDao
25  }
```

8. DatabaseModule.kt

Tabel 8. Source Code DatabaseModule.kt

```
 1  package com.example.treblewinner.data.local.di
 2
 3  import android.content.Context
 4  import androidx.room.Room
 5  import
    com.example.treblewinner.data.local.dao.TrebleWinnerDao
```

```
 6 import
   com.example.treblewinner.data.local.db.AppDatabase
 7 import dagger.Module
 8 import dagger.Provides
 9 import dagger.hilt.InstallIn
10 import
   dagger.hilt.android.qualifiers.ApplicationContext
11 import dagger.hilt.components.SingletonComponent
12 import javax.inject.Singleton
13
14 @Module
15 @InstallIn(SingletonComponent::class)
16 object DatabaseModule {
17
18     @Provides
19     @Singleton
20     fun provideAppDatabase(@ApplicationContext context:
   Context): AppDatabase {
21         return Room.databaseBuilder(
22             context,
23             AppDatabase::class.java,
24             "treble_winner_db"
25         ).build()
26     }
27
28     @Provides
29     fun provideTrebleWinnerDao(appDatabase:
   AppDatabase): TrebleWinnerDao {
30         return appDatabase.trebleWinnerDao()
31     }
32 }
```

9. ClubCompetitionCrossRef.kt

Tabel 9. Source Code ClubCompetitionCrossRed.kt

```
 1 package com.example.treblewinner.data.local.entity
 2
 3 import androidx.room.Entity
 4 import androidx.room.ForeignKey
 5 import androidx.room.Index
 6 import androidx.room.RoomDatabase
 7 import
   com.example.treblewinner.data.local.dao.TrebleWinnerDao
 8
 9 @Entity(
10     tableName = "club_competition_cross_ref",
```

```
11      primaryKeys = ["clubId", "competitionId"],
12      foreignKeys = [
13          ForeignKey(entity = ClubEntity::class,
    parentColumns = ["id"], childColumns = ["clubId"],
    onDelete = ForeignKey.CASCADE),
14          ForeignKey(entity = CompetitionEntity::class,
    parentColumns = ["id"], childColumns =
    ["competitionId"], onDelete = ForeignKey.CASCADE)
15      ],
16      indices = [Index("clubId"), Index("competitionId")]
17  )
18
19  data class ClubCompetitionCrossRef(
20      val clubId: String,
21      val competitionId: String
22  )
23
24  abstract class TrebleWinnerDatabase : RoomDatabase() {
25      abstract fun trebleWinnerDao(): TrebleWinnerDao
26  }
```

10. ClubEntity.kt

Tabel 10. Source Code ClubEntity.kt

```
 1  package com.example.treblewinner.data.local.entity
 2
 3  import androidx.room.ColumnInfo
 4  import androidx.room.Entity
 5  import androidx.room.PrimaryKey
 6
 7  @Entity(tableName = "clubs")
 8  data class ClubEntity(
 9      @PrimaryKey
10      @ColumnInfo(name = "id")
11      val id: String,
12
13      @ColumnInfo(name = "name")
14      val name: String,
15
16      @ColumnInfo(name = "country")
17      val country: String,
18
19      @ColumnInfo(name = "confederation")
20      val confederation: String,
21
22      @ColumnInfo(name = "treble_years")
```

```
    val trebleYears: List<String>,

    @ColumnInfo(name = "competition_ids")
    val competitionIds: List<String>,

    @ColumnInfo(name = "logo_url")
    val logoUrl: String,

    @ColumnInfo(name = "web_url")
    val webUrl: String,

    @ColumnInfo(name = "description")
    val description: String,

    @ColumnInfo(name = "is_bookmarked")
    val isBookmarked: Boolean = false
)
```

11. CompetitionEntity.kt

Tabel 11. Source Code CompetitionEntity.kt

```
package com.example.treblewinner.data.local.entity

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

@Entity(tableName = "competitions")
data class CompetitionEntity(
    @PrimaryKey
    @ColumnInfo(name = "id")
    val id: String,

    @ColumnInfo(name = "name")
    val name: String,

    @ColumnInfo(name = "country")
    val country: String,

    @ColumnInfo(name = "confederation")
    val confederation: String,

    @ColumnInfo(name = "logo_url")
    val logoUrl: String,

    @ColumnInfo(name = "logo_url_dark")
```

```
26      val logoUrlDark: String
27  )
```

12. ClubWithCompetitions.kt

Tabel 12. Source Code ClubWithCompetitions.kt

```
 1  package com.example.treblewinner.data.local.relation
 2
 3  import androidx.room.Embedded
 4  import androidx.room.Junction
 5  import androidx.room.Relation
 6  import
    com.example.treblewinner.data.local.entity.ClubCompetit
    ionCrossRef
 7  import
    com.example.treblewinner.data.local.entity.ClubEntity
 8  import
    com.example.treblewinner.data.local.entity.CompetitionE
    ntity
 9
10  class ClubWithCompetitions(
11      @Embedded val club: ClubEntity,
12
13      @Relation(
14          parentColumn = "id",
15          entityColumn = "id",
16          associateBy = Junction(
17              value = ClubCompetitionCrossRef::class,
18              parentColumn = "clubId",
19              entityColumn = "competitionId"
20          )
21      )
22      val competitions: List<CompetitionEntity>
23  )
```

13. DtoMapper.kt

Tabel 13. Source Code DtoMapper.kt

```
 1  package com.example.treblewinner.data.mapper
 2
 3  import
    com.example.treblewinner.data.local.entity.ClubEntity
 4  import
    com.example.treblewinner.data.local.entity.CompetitionE
    ntity
```

```
 5 import com.example.treblewinner.data.remote.dto.ClubDto
 6 import
   com.example.treblewinner.data.remote.dto.CompetitionDto
 7
 8 import com.example.treblewinner.domain.model.Club
 9 import
   com.example.treblewinner.domain.model.Competition
10 import
   com.example.treblewinner.data.local.relation.ClubWithCo
   mpetitions
11
12 object DtoMapper {
13     fun toClubEntity(dto: ClubDto): ClubEntity {
14         return ClubEntity(
15             id = dto.id,
16             name = dto.name,
17             country = dto.country,
18             confederation = dto.confederation,
19             trebleYears = dto.trebleYears,
20             competitionIds = dto.competitionIds,
21             logoUrl = dto.logoUrl,
22             webUrl = dto.webUrl,
23             description = dto.description,
24             isBookmarked = false
25         )
26     }
27
28     fun toCompetitionEntity(dto: CompetitionDto):
   CompetitionEntity {
29         return CompetitionEntity(
30             id = dto.id,
31             name = dto.name,
32             country = dto.country,
33             confederation = dto.confederation,
34             logoUrl = dto.logoUrl,
35             logoUrlDark = dto.logoUrlDark
36         )
37     }
38
39     fun toClubEntityList(dtos: List<ClubDto>):
   List<ClubEntity> {
40         return dtos.map { toClubEntity(it) }
41     }
42
43     fun toCompetitionEntityList(dtos:
   List<CompetitionDto>): List<CompetitionEntity> {
44         return dtos.map { toCompetitionEntity(it) }
```

```
45      }
46
47      fun mapClubEntityToDomain(entity: ClubEntity): Club
    {
48          return Club(
49              id = entity.id,
50              name = entity.name,
51              country = entity.country,
52              confederation = entity.confederation,
53              trebleYears = entity.trebleYears,
54              competitionIds = entity.competitionIds,
55              logoUrl = entity.logoUrl,
56              webUrl = entity.webUrl,
57              description = entity.description,
58              isBookmarked = entity.isBookmarked
59          )
60      }
61
62      fun mapCompetitionEntityToDomain(entity:
    CompetitionEntity): Competition {
63          return Competition(
64              id = entity.id,
65              name = entity.name,
66              country = entity.country,
67              confederation = entity.confederation,
68              logoUrl = entity.logoUrl,
69              logoUrlDark = entity.logoUrlDark
70          )
71      }
72
73      fun mapClubWithCompetitionsToDomain(data:
    ClubWithCompetitions): Club {
74          return Club(
75              id = data.club.id,
76              name = data.club.name,
77              country = data.club.country,
78              confederation = data.club.confederation,
79              trebleYears = data.club.trebleYears,
80              competitionIds = data.club.competitionIds,
81              logoUrl = data.club.logoUrl,
82              webUrl = data.club.webUrl,
83              description = data.club.description,
84              isBookmarked = data.club.isBookmarked
85          )
86      }
87  }
```

14. RemoteDataSource.kt

Tabel 14. Source Code RemoteDataSource.kt

```
 1 package com.example.treblewinner.data.remote
 2
 3 import android.util.Log
 4 import
   com.example.treblewinner.data.remote.api.TrebleWinnerAPI
 5 import com.example.treblewinner.data.remote.dto.ClubDto
 6 import
   com.example.treblewinner.data.remote.dto.CompetitionDto
 7 import
   com.example.treblewinner.data.remote.mapper.ResponseMapp
   er
 8 import
   com.example.treblewinner.data.remote.response.ClubRespon
   se
 9 import
   com.example.treblewinner.data.remote.response.Competitio
   nResponse
10 import kotlinx.coroutines.flow.Flow
11 import kotlinx.coroutines.flow.flow
12 import retrofit2.HttpException
13 import java.io.IOException
14 import javax.inject.Inject
15
16 class RemoteDataSource @Inject constructor(
17     private val api: TrebleWinnerAPI
18 ) {
19     fun getClubs(): Flow<List<ClubDto>> = flow {
20         try {
21             val response: List<ClubResponse> =
   api.getClubs()
22             val clubsDto: List<ClubDto> = response.map {
   ResponseMapper.toClubDto(it) }
23
24             Log.d("RemoteDataSource", "API getClubs
   response size: ${response.size}")
25             Log.d("RemoteDataSource", "API getClubs
   response: $response")
26
27             Log.d("RemoteDataSource", "Mapped clubsDto
   size: ${clubsDto.size}")
28
29             emit(clubsDto)
30         } catch (e: IOException){
31             e.printStackTrace()
```

```
32              emit(emptyList())
33          } catch (e: HttpException){
34              e.printStackTrace()
35              emit(emptyList())
36          }
37      }
38
39      fun getCompetitions(): Flow<List<CompetitionDto>> =
    flow {
40          try {
41              val response: List<CompetitionResponse> =
    api.getCompetitions()
42              val competitionsDto: List<CompetitionDto> =
    response.map { ResponseMapper.toCompetitionDto(it) }
43
44              emit(competitionsDto)
45          } catch (e: IOException){
46              e.printStackTrace()
47              emit(emptyList())
48          } catch (e: HttpException){
49              e.printStackTrace()
50              emit(emptyList())
51          }
52      }
53  }
```

15. TrebleWinnerAPI.kt

Tabel 15. Source Code TrebleWinnerAPI.kt

```
 1  package com.example.treblewinner.data.remote.api
 2
 3  import retrofit2.http.GET
 4  import
    com.example.treblewinner.data.remote.response.ClubRespon
    se
 5  import
    com.example.treblewinner.data.remote.response.Competitio
    nResponse
 6  import retrofit2.http.Path
 7
 8  interface TrebleWinnerAPI {
 9      @GET("/clubs")
10      suspend fun getClubs(): List<ClubResponse>
11
12      @GET("/competitions")
```

```
13      suspend fun getCompetitions():
    List<CompetitionResponse>
14
15      @GET("/clubs/{id}")
16      suspend fun getClubDetail(@Path("id") clubId:
    String): ClubResponse
17
18      @GET("/competitions/{id}")
19      suspend fun getCompetitionDetail(@Path("id")
    competitionId: String): CompetitionResponse
20  }
```

16. NetworkModule.kt

Tabel 16. Source Code NetworkModule.kt

```
 1  package com.example.treblewinner.data.remote.di
 2
 3  import
    com.example.treblewinner.data.remote.api.TrebleWinnerAPI
 4  import dagger.Module
 5  import dagger.Provides
 6  import dagger.hilt.InstallIn
 7  import dagger.hilt.components.SingletonComponent
 8  import okhttp3.OkHttpClient
 9  import okhttp3.logging.HttpLoggingInterceptor
10  import retrofit2.Retrofit
11  import retrofit2.converter.gson.GsonConverterFactory
12  import javax.inject.Singleton
13
14  @Module
15  @InstallIn(SingletonComponent::class)
16  object NetworkModule {
17
18
19      // Use this for AVD
20      // private const val BASE_URL =
    "http://10.0.2.2:8000"
21
22      // Use this for physical device
23      private const val BASE_URL =
    "http://192.168.100.141:8000"
24
25      @Provides
26      @Singleton
27      fun provideOkHttpClient(): OkHttpClient {
28          val logging = HttpLoggingInterceptor().apply {
```

```
29              level = HttpLoggingInterceptor.Level.BODY
30          }
31
32          return OkHttpClient.Builder()
33              .addInterceptor(logging)
34              .build()
35      }
36
37      @Provides
38      @Singleton
39      fun provideRetrofit(client: OkHttpClient): Retrofit
    {
40          return Retrofit.Builder()
41              .baseUrl(BASE_URL)
42              .client(client)
43
    .addConverterFactory(GsonConverterFactory.create())
44              .build()
45      }
46
47      @Provides
48      @Singleton
49      fun provideTrebleWinnerAPI(retrofit: Retrofit):
    TrebleWinnerAPI {
50          return
    retrofit.create(TrebleWinnerAPI::class.java)
51      }
52  }
```

17. ClubDto.kt

Tabel 17. Source Code ClubDto.kt

```
 1  package com.example.treblewinner.data.remote.dto
 2
 3  data class ClubDto(
 4      val id: String,
 5      val name: String,
 6      val country: String,
 7      val confederation: String,
 8      val trebleYears: List<String>,
 9      val competitionIds: List<String>,
10      val logoUrl: String,
11      val webUrl: String,
12      val description: String
13  )
```

18. CompetitionDto.kt

Tabel 18. Source Code CompetitionDto.kt

```
 1 package com.example.treblewinner.data.remote.dto
 2
 3 data class CompetitionDto(
 4     val id: String,
 5     val name: String,
 6     val country: String,
 7     val confederation: String,
 8     val logoUrl: String,
 9     val logoUrlDark: String
10 )
```

19. ResponseMapper.kt

Tabel 19. Source Code ResponseMapper.kt

```
 1 package com.example.treblewinner.data.remote.mapper
 2
 3 import com.example.treblewinner.data.remote.dto.ClubDto
 4 import
   com.example.treblewinner.data.remote.dto.CompetitionDto
 5 import
   com.example.treblewinner.data.remote.response.ClubRespon
   se
 6 import
   com.example.treblewinner.data.remote.response.Competitio
   nResponse
 7
 8 object ResponseMapper {
 9     fun toClubDto(response: ClubResponse): ClubDto {
10         return ClubDto (
11             id = response.id,
12             name = response.name,
13             country = response.country,
14             confederation = response.confederation,
15             trebleYears = response.trebleYears,
16             competitionIds = response.competitionIds,
17             logoUrl = response.logoUrl,
18             webUrl = response.webUrl,
19             description = response.description
20         )
21     }
22
23     fun toCompetitionDto(response: CompetitionResponse):
   CompetitionDto {
```

```
24          return CompetitionDto (
25              id = response.id,
26              name = response.name,
27              country = response.country,
28              confederation = response.confederation,
29              logoUrl = response.logoUrl,
30              logoUrlDark = response.logoUrlDark
31          )
32      }
33  }
```

20. ClubResponse.kt

Tabel 20. Source Code ClubResponse.kt

```
 1  package com.example.treblewinner.data.remote.response
 2
 3  import com.google.gson.annotations.SerializedName
 4
 5  data class ClubResponse(
 6      val id: String,
 7      val name: String,
 8      val country: String,
 9      val confederation: String,
10
11      @SerializedName("treble_years")
12      val trebleYears: List<String>,
13
14      @SerializedName("competition_ids")
15      val competitionIds: List<String>,
16
17      @SerializedName("logo_url")
18      val logoUrl: String,
19
20      @SerializedName("web_url")
21      val webUrl: String,
22
23      val description: String
24  )
```

21. CompetitionResponse.kt

Tabel 21. Source Code CompetionResponse.kt

```
 1  package com.example.treblewinner.data.remote.response
 2
 3  import com.google.gson.annotations.SerializedName
```

```
 4
 5  data class CompetitionResponse(
 6      val id: String,
 7      val name: String,
 8      val country: String,
 9      val confederation: String,
10
11      @SerializedName("logo_url")
12      val logoUrl: String,
13
14      @SerializedName("logo_url_dark")
15      val logoUrlDark: String
16  )
```

22. TrebleWinnerRepositoryImpl.kt

Tabel 22. Source Code TrebleWinnerRepositoryImpl.kt

```
 1  package com.example.treblewinner.data.repository
 2
 3  import
    com.example.treblewinner.data.local.dao.TrebleWinnerDao
 4  import
    com.example.treblewinner.data.local.relation.ClubWithCo
    mpetitions
 5  import com.example.treblewinner.data.mapper.DtoMapper
 6  import
    com.example.treblewinner.data.remote.RemoteDataSource
 7  import com.example.treblewinner.domain.model.Club
 8  import
    com.example.treblewinner.domain.model.Competition
 9  import
    com.example.treblewinner.domain.repository.TrebleWinner
    Repository
10  import kotlinx.coroutines.flow.Flow
11  import kotlinx.coroutines.flow.map
12  import javax.inject.Inject
13
14  class TrebleWinnerRepositoryImpl @Inject constructor(
15      private val remoteDataSource: RemoteDataSource,
16      private val trebleWinnerDao: TrebleWinnerDao
17  ): TrebleWinnerRepository {
18      override fun getAllClubs(): Flow<List<Club>> {
19          return trebleWinnerDao.getAllClubs()
20              .map { clubEntities ->
21                  clubEntities.map {
    DtoMapper.mapClubEntityToDomain(it) }
22              }
```

```
23      }
24
25      override suspend fun getClubDetail(id: String):
    Club? {
26          val clubWithCompetitions: ClubWithCompetitions?
    = trebleWinnerDao.getClubWithCompetitions(id)
27          return clubWithCompetitions?.let {
    DtoMapper.mapClubWithCompetitionsToDomain(it) }
28      }
29
30      override fun getBookmarkedClubs(): Flow<List<Club>>
    {
31          return trebleWinnerDao.getBookmarkedClubs()
32              .map { bookmarkedEntities ->
33                  bookmarkedEntities.map {
    DtoMapper.mapClubEntityToDomain(it) }
34              }
35      }
36
37      override suspend fun toggleClubBookmark(id: String)
    {
38          trebleWinnerDao.toggleClubBookmark(id)
39      }
40
41
42      override suspend fun fetchAndCacheData() {
43          remoteDataSource.getClubs().collect { clubDtos
    ->
44              val clubEntities =
    DtoMapper.toClubEntityList(clubDtos)
45              trebleWinnerDao.insertClubs(clubEntities)
46          }
47          remoteDataSource.getCompetitions().collect {
    competitionDtos ->
48              val competitionEntities =
    DtoMapper.toCompetitionEntityList(competitionDtos)
49
    trebleWinnerDao.insertCompetitions(competitionEntities)
50          }
51      }
52
53
54      override fun getAllCompetitions():
    Flow<List<Competition>> {
55          return trebleWinnerDao.getAllCompetitions()
56              .map { competitionEntities ->
```

```
57                  competitionEntities.map {
    DtoMapper.mapCompetitionEntityToDomain(it) }
58              }
59      }
60
61  }
```

Package: domain

23. Club.kt

Tabel 23. Source Code Club.kt

```
 1  package com.example.treblewinner.domain.model
 2
 3  data class Club(
 4      val id: String,
 5      val name: String,
 6      val country: String,
 7      val confederation: String,
 8      val trebleYears: List<String>,
 9      val competitionIds: List<String>,
10      val logoUrl: String,
11      val webUrl: String,
12      val description: String,
13      val isBookmarked: Boolean
14  )
```

24. Competition.kt

Tabel 24. Source Code Competition.kt

```
 1  package com.example.treblewinner.domain.model
 2
 3
 4  data class Competition (
 5      val id: String,
 6      val name: String,
 7      val country: String,
 8      val confederation: String,
 9      val logoUrl: String,
10      val logoUrlDark: String
11  )
```

25. TrebleWinnerRepository.kt

Tabel 25. Source Code TrebleWinnerRepository.kt

```
 1 package com.example.treblewinner.domain.repository
 2
 3 import com.example.treblewinner.domain.model.Club
 4 import
   com.example.treblewinner.domain.model.Competition
 5 import kotlinx.coroutines.flow.Flow
 6
 7 interface TrebleWinnerRepository {
 8     fun getAllClubs(): Flow<List<Club>>
 9     suspend fun getClubDetail(id: String): Club?
10     fun getBookmarkedClubs(): Flow<List<Club>>
11     suspend fun toggleClubBookmark(id: String)
12     suspend fun fetchAndCacheData()
13     fun getAllCompetitions(): Flow<List<Competition>>
14 }
```

26. FetchAndCacheDataUseCase.kt

Tabel 26. Source Code FetchAndCacheDataUseCase.kt

```
1 package com.example.treblewinner.domain.usecase
2
3 import
  com.example.treblewinner.domain.repository.TrebleWinnerRe
  pository
4 import javax.inject.Inject
5
6 class FetchAndCacheDataUseCase @Inject
  constructor(private val repository:
  TrebleWinnerRepository) {
7     suspend operator fun invoke() =
  repository.fetchAndCacheData()
8 }
```

27. GetAllClubsUseCase.kt

Tabel 27. Source Code GetAllClubsUseCase.kt

```
 1 package com.example.treblewinner.domain.usecase
 2
 3 import com.example.treblewinner.domain.model.Club
```

```
 4 import
   com.example.treblewinner.domain.repository.TrebleWinnerR
   epository
 5 import kotlinx.coroutines.flow.Flow
 6 import javax.inject.Inject
 7
 8 class GetAllClubsUseCase @Inject constructor(private val
   repository: TrebleWinnerRepository) {
 9     operator fun invoke(): Flow<List<Club>> =
   repository.getAllClubs()
10 }
```

28. GetAllCompetitionsUseCase.kt

Tabel 28. Source Code GetAllCompetitonsUseCase.kt

```
 1 package com.example.treblewinner.domain.usecase
 2
 3 import com.example.treblewinner.domain.model.Competition
 4 import
   com.example.treblewinner.domain.repository.TrebleWinnerR
   epository
 5 import kotlinx.coroutines.flow.Flow
 6 import javax.inject.Inject
 7
 8 class GetAllCompetitionsUseCase @Inject constructor
   (private val repository: TrebleWinnerRepository) {
 9     operator fun invoke(): Flow<List<Competition>> =
   repository.getAllCompetitions()
10 }
```

29. GetBookmarkedClubUseCase.kt

Tabel 29. Source Code GetBookmarkedClubUseCase.kt

```
 1 package com.example.treblewinner.domain.usecase
 2
 3 import com.example.treblewinner.domain.model.Club
 4 import
   com.example.treblewinner.domain.repository.TrebleWinnerR
   epository
 5 import kotlinx.coroutines.flow.Flow
 6 import javax.inject.Inject
 7
 8 class GetBookmarkedClubsUseCase @Inject
   constructor(private val repository:
   TrebleWinnerRepository) {
```

```
 9      operator fun invoke(): Flow<List<Club>> =
   repository.getBookmarkedClubs()
10  }
```

30. GetClubDetailUseCase.kt

Tabel 30. Source Code GetClubDetailUseCase.kt

```
 1  package com.example.treblewinner.domain.usecase
 2
 3  import com.example.treblewinner.domain.model.Club
 4  import
    com.example.treblewinner.domain.repository.TrebleWinnerRe
    pository
 5  import javax.inject.Inject
 6
 7  class GetClubDetailUseCase @Inject constructor(private val
    repository: TrebleWinnerRepository) {
 8      suspend operator fun invoke(id: String): Club? =
    repository.getClubDetail(id)
 9  }
```

31. ToggleClubBookmarkUseCase.kt

Tabel 31. Source Code ToggleClubBookmarkUseCase.kt

```
 1  package com.example.treblewinner.domain.usecase
 2
 3  import
    com.example.treblewinner.domain.repository.TrebleWinnerRe
    pository
 4  import javax.inject.Inject
 5
 6  class          ToggleClubBookmarkUseCase          @Inject
    constructor(private            val            repository:
    TrebleWinnerRepository) {
 7      suspend   operator   fun   invoke(id:   String)   =
    repository.toggleClubBookmark(id)
 8  }
```

Package: Presentation

32. BookmarkedClubsScreen.kt

Tabel 32. Source Code BookmarkedClubsScreen.kt

```
 1  package com.example.treblewinner.presentation.bookmark
 2
```

```
 3 import android.content.ActivityNotFoundException
 4 import android.content.Intent
 5 import android.util.Log
 6 import android.widget.Toast
 7 import androidx.compose.foundation.layout.Arrangement
 8 import androidx.compose.foundation.layout.Box
 9 import androidx.compose.foundation.layout.fillMaxSize
10 import androidx.compose.foundation.lazy.LazyColumn
11 import androidx.compose.foundation.lazy.LazyListState
12 import androidx.compose.foundation.lazy.items
13 import
   androidx.compose.foundation.lazy.rememberLazyListState
14 import
   androidx.compose.material3.CircularProgressIndicator
15 import androidx.compose.material3.Text
16 import androidx.compose.material3.AlertDialog
17 import androidx.compose.material3.TextButton
18 import androidx.compose.runtime.Composable
19 import androidx.compose.runtime.LaunchedEffect
20 import androidx.compose.runtime.collectAsState
21 import androidx.compose.ui.Alignment
22 import androidx.compose.ui.Modifier
23 import androidx.compose.ui.unit.dp
24 import androidx.hilt.navigation.compose.hiltViewModel
25 import com.example.treblewinner.domain.model.Club
26 import androidx.compose.runtime.getValue
27 import androidx.compose.runtime.mutableStateOf
28 import androidx.compose.runtime.remember
29 import androidx.compose.runtime.setValue
30 import androidx.compose.ui.platform.LocalContext
31 import androidx.core.net.toUri
32 import
   com.example.treblewinner.presentation.clublist.ClubCard
33
34 @Composable
35 fun BookmarkedClubsScreen(
36     onNavigateToDetail: (String) -> Unit,
37     lazyListState: LazyListState =
   rememberLazyListState(),
38     viewModel: BookmarkedClubsViewModel =
   hiltViewModel()
39 ) {
40     val context = LocalContext.current
41     val uiState by viewModel.uiState.collectAsState()
42
43     var clubToUnbookmark by remember {
   mutableStateOf<Club?>(null) }
```

```
    var showConfirmDialog by remember { mutableStateOf(false) }

    LaunchedEffect(Unit) {
        viewModel.loadData()
    }

    when {
        uiState.isLoading -> {
            Box(
                modifier = Modifier.fillMaxSize(),
                contentAlignment = Alignment.Center
            ) {
                CircularProgressIndicator()
            }
        }
        uiState.error != null -> {
            Box(
                modifier = Modifier.fillMaxSize(),
                contentAlignment = Alignment.Center
            ) {
                Text(text = "Error: ${uiState.error}")
            }
        }
        else -> {
            LazyColumn(
                state = lazyListState,
                modifier = Modifier.fillMaxSize(),
                verticalArrangement = Arrangement.spacedBy(4.dp)
            ) {
                items(uiState.bookmarkedClubs) { club ->
                    ClubCard(
                        club = club,
                        onDetailClick = {

Log.i("BookmarkedClubScreen", "User click the detail button for ${club.name}")
                            onNavigateToDetail(club.id)
                        },
                        onVisitClick = {
                            try {
                                val intent = Intent(Intent.ACTION_VIEW, club.webUrl.toUri())

context.startActivity(intent)
```

```
 84
    Log.d("BookmarkedClubScreen", "User going to the web
    ${club.name}")
 85                             } catch (e:
    ActivityNotFoundException) {
 86
    Log.e("BookmarkedClubScreen", "Browser not found", e)
 87                                 Toast.makeText(context,
    "No browser available", Toast.LENGTH_SHORT)
 88                                         .show()
 89                             }
 90                         },
 91                         onBookmarkClick = {
 92                             if (club.isBookmarked) {
 93                                 clubToUnbookmark = club
 94                                 showConfirmDialog =
    true
 95                             } else {
 96
    viewModel.toggleBookmark(club.id)
 97                             }
 98                         }
 99                     )
100                 }
101             }
102         }
103     }
104
105     if (showConfirmDialog && clubToUnbookmark != null)
    {
106         AlertDialog(
107             onDismissRequest = { showConfirmDialog =
    false },
108             title = { Text("Remove Bookmark") },
109             text = {
110                 Text("Are you sure you want to remove
    '${clubToUnbookmark!!.name}' from your bookmarks?")
111             },
112             confirmButton = {
113                 TextButton(
114                     onClick = {
115
    viewModel.toggleBookmark(clubToUnbookmark!!.id)
116                         showConfirmDialog = false
117                     }
118                 ) {
119                     Text("Yes")
```

```
120                 }
121             },
122             dismissButton = {
123                 TextButton(
124                     onClick = {
125                         showConfirmDialog = false
126                     }
127                 ) {
128                     Text("No")
129                 }
130             }
131         )
132     }
133 }
```

33. BookmarkedClubsUIState.kt

Tabel 33. Source Code BookmarkedClubsUIState.kt

```
1 package com.example.treblewinner.presentation.bookmark
2
3 import com.example.treblewinner.domain.model.Club
4
5 data class BookmarkedClubsUIState(
6     val bookmarkedClubs: List<Club> = emptyList(),
7     val isLoading: Boolean = false,
8     val error: String? = null
9 )
```

34. BookmarkedClubsViewModel.kt

Tabel 34. Source Code BookmarkedClubsViewModel.kt

```
 1 package com.example.treblewinner.presentation.bookmark
 2
 3 import android.util.Log
 4 import
   com.example.treblewinner.domain.usecase.GetBookmarkedCl
   ubsUseCase
 5 import
   com.example.treblewinner.domain.usecase.ToggleClubBookm
   arkUseCase
 6 import dagger.hilt.android.lifecycle.HiltViewModel
 7 import kotlinx.coroutines.flow.MutableStateFlow
 8 import kotlinx.coroutines.flow.StateFlow
 9 import javax.inject.Inject
```

```
10 import androidx.lifecycle.viewModelScope
11 import androidx.lifecycle.ViewModel
12 import kotlinx.coroutines.launch
13 import kotlinx.coroutines.flow.catch
14 import kotlinx.coroutines.flow.update
15
16 @HiltViewModel
17 class BookmarkedClubsViewModel @Inject constructor(
18     private val getBookmarkedClubsUseCase:
   GetBookmarkedClubsUseCase,
19     private val toggleClubBookmarkUseCase:
   ToggleClubBookmarkUseCase
20 ): ViewModel() {
21     private val _uiState =
   MutableStateFlow(BookmarkedClubsUIState())
22     val uiState: StateFlow<BookmarkedClubsUIState> =
   _uiState
23
24     private var hasLoaded = false
25
26     fun toggleBookmark(clubId: String) {
27         viewModelScope.launch {
28             toggleClubBookmarkUseCase(clubId)
29             val updatedClubs =
   _uiState.value.bookmarkedClubs.map { club ->
30                 if (club.id == clubId)
   club.copy(isBookmarked = !club.isBookmarked)
31                 else club
32             }
33             _uiState.update { it.copy(bookmarkedClubs =
   updatedClubs) }
34         }
35         loadData(forceReload = true)
36     }
37
38     fun loadData(forceReload: Boolean = false) {
39         if (!hasLoaded || forceReload) {
40             hasLoaded = true
41             viewModelScope.launch {
42                 _uiState.update { it.copy(isLoading =
   true, error = null) }
43                 getBookmarkedClubsUseCase()
44                     .catch { e ->
45                         Log.e("Bookmark VM", "Error
   loading bookmarked clubs", e)
```

```
46                          _uiState.update {
    it.copy(isLoading = false, error = e.message ?:
    "Unknown error") }
47                      }
48                      .collect { clubs ->
49                          _uiState.update {
    it.copy(isLoading = false, bookmarkedClubs = clubs,
    error = null) }
50                      }
51              }
52          } else {
53              Log.i("Bookmark VM", "Data already loaded")
54          }
55      }
56  }
```

35. ClubDetailsScreen.kt

Tabel 35. Source Code ClubDetailsScreen.kt

```
 1  package
 2  com.example.treblewinner.presentation.clubdetails
 3
 4  import android.content.Context
 5  import android.content.res.Configuration
 6  import android.util.Log
 7  import androidx.compose.foundation.BorderStroke
 8  import androidx.compose.foundation.layout.*
 9  import androidx.compose.foundation.rememberScrollState
10  import androidx.compose.foundation.verticalScroll
11  import androidx.compose.material.icons.Icons
12  import androidx.compose.material3.*
13  import androidx.compose.runtime.Composable
14  import androidx.compose.ui.Alignment
15  import androidx.compose.ui.Modifier
16  import androidx.compose.ui.layout.ContentScale
17  import androidx.compose.ui.text.font.FontWeight
18  import androidx.compose.ui.unit.dp
19  import
    com.bumptech.glide.integration.compose.ExperimentalGli
    deComposeApi
20  import
    com.bumptech.glide.integration.compose.GlideImage
21  import com.example.treblewinner.domain.model.Club
22  import androidx.compose.foundation.layout.Arrangement
23  import
    androidx.compose.foundation.shape.RoundedCornerShape
```

```
24 import
   androidx.compose.material.icons.automirrored.filled.Ar
   rowBack
25 import androidx.compose.material.icons.filled.Star
26 import androidx.compose.material.icons.outlined.Star
27 import androidx.compose.runtime.LaunchedEffect
28 import androidx.compose.runtime.collectAsState
29 import androidx.compose.ui.graphics.Color
30 import androidx.compose.ui.text.style.TextAlign
31 import androidx.compose.ui.platform.LocalContext
32 import androidx.compose.ui.unit.sp
33 import androidx.hilt.navigation.compose.hiltViewModel
34 import
   com.example.treblewinner.domain.model.Competition
   import androidx.compose.runtime.getValue
35
36
37
38 @Composable
39 fun ClubDetailsScreen(
40     clubId: String,
41     onBack: () -> Unit,
42     viewModel: ClubDetailsViewModel = hiltViewModel()
43 ) {
44     val context = LocalContext.current
45     val uiState by viewModel.uiState.collectAsState()
46
47     LaunchedEffect(clubId) {
48         viewModel.loadClubDetail(clubId)
49     }
50
51     when {
52         uiState.isLoading -> {
53             Box(modifier = Modifier.fillMaxSize(),
   contentAlignment = Alignment.Center) {
54                 CircularProgressIndicator()
55             }
56         }
57         uiState.error != null -> {
58             Box(modifier = Modifier.fillMaxSize(),
   contentAlignment = Alignment.Center) {
59                 Text(text = "Error: ${uiState.error}")
60             }
61         }
62         uiState.club != null -> {
63             val club = uiState.club
64             val competitions = uiState.competitions
```

```
 65              ClubDetailContent(
 66                  club,
 67                  competitions,
 68                  onBack,
 69                  onToggleBookmark = {
    viewModel.toggleBookmark()}
 70              )
 71          }
 72      }
 73  }
 74
 75
 76
 77  fun Context.isDarkTheme(): Boolean {
 78      return (resources.configuration.uiMode and
    Configuration.UI_MODE_NIGHT_MASK) ==
    Configuration.UI_MODE_NIGHT_YES
 79  }
 80
 81  @OptIn(ExperimentalMaterial3Api::class,
    ExperimentalGlideComposeApi::class)
 82  @Composable
 83  fun ClubDetailContent(
 84      club: Club?,
 85      competitions: List<Competition>,
 86      onBack: () -> Unit,
 87      onToggleBookmark: () -> Unit
 88  ) {
 89      Log.i("ClubDetailScreen", "Club is loaded...")
 90      Scaffold(
 91          topBar = {
 92              TopAppBar(
 93                  title = { Text(text = club?.name ?:
    "") },
 94                  navigationIcon = {
 95                      IconButton(onClick = {
 96                          Log.i("ClubDetailScreen",
    "User click back button to the ClubListScreen")
 97                          onBack()
 98                      }) {
 99                          Icon(imageVector =
    Icons.AutoMirrored.Filled.ArrowBack,
    contentDescription = "Back")
100                      }
101                  },
102                  actions = {
```

```
103                     IconButton(onClick =
    onToggleBookmark) {
104                         Icon(
105                             imageVector = if
    (club?.isBookmarked == true) Icons.Filled.Star else
    Icons.Outlined.Star,
106                             contentDescription = if
    (club?.isBookmarked == true) "Unbookmark" else
    "Bookmark",
107                             tint = if
    (club?.isBookmarked == true) Color(0xFFFFD700) else
    LocalContentColor.current
108                         )
109                     }
110                 }
111             )
112         }
113     ) { padding ->
114         Column(
115             modifier = Modifier
116                 .verticalScroll(rememberScrollState())
117                 .padding(16.dp)
118                 .padding(padding)
119         ) {
120             GlideImage(
121                 model = club?.logoUrl,
122                 contentDescription = "${club?.name}
    logo",
123                 modifier = Modifier
124                     .fillMaxWidth()
125                     .height(200.dp),
126                 contentScale = ContentScale.Fit
127             )
128             Log.i("ClubDetailScreen", "Club logo is
    loaded")
129
130             Spacer(modifier = Modifier.height(24.dp))
131
132             Text(
133                 text = club?.name ?: "",
134                 style =
    MaterialTheme.typography.headlineMedium,
135                 fontWeight = FontWeight.Bold
136             )
137             Log.i("ClubDetailScreen", "Club name is
    loaded")
138
```

```
139             Spacer(modifier = Modifier.height(8.dp))
140
141             Row(
142                 horizontalArrangement =
    Arrangement.spacedBy(8.dp),
143                 verticalAlignment =
    Alignment.CenterVertically
144             ) {
145                 Text(text = club?.country ?:"", style
    = MaterialTheme.typography.bodyLarge)
146                 Log.i("ClubDetailScreen", "Club
    country is loaded")
147                 Text(text = "•", style =
    MaterialTheme.typography.bodyLarge)
148                 Text(text = club?.confederation ?:"",
    style = MaterialTheme.typography.bodyLarge)
149                 Log.i("ClubDetailScreen", "Club
    confederation is loaded")
150             }
151
152             Spacer(modifier = Modifier.height(16.dp))
153
154             Text(
155                 text = "Treble Years:",
156                 style =
    MaterialTheme.typography.titleMedium,
157                 fontWeight = FontWeight.Bold
158             )
159
160             Spacer(modifier = Modifier.height(4.dp))
161
162             Row(
163                 modifier = Modifier
164                     .fillMaxWidth(),
165                 horizontalArrangement =
    Arrangement.spacedBy(16.dp,
    Alignment.CenterHorizontally)
166             ){
167                 club?.trebleYears?.forEach { year ->
168                     YearTagOutlined(year)
169                 }
170             }
171
172
173             Spacer(modifier = Modifier.height(16.dp))
174
175             Text(
```

```
176                 text = "Competitions Won:",
177                 style =
    MaterialTheme.typography.titleMedium,
178                 fontWeight = FontWeight.Bold
179             )
180             Spacer(modifier = Modifier.height(4.dp))
181
182             Row(
183                 modifier = Modifier.fillMaxWidth(),
184                 horizontalArrangement =
    Arrangement.spacedBy(16.dp,
    Alignment.CenterHorizontally)
185             ) {
186                 competitions.forEach { competition ->
187                     CompetitionCard(competition =
    competition)
188                 }
189             }
190
191
192             Spacer(modifier = Modifier.height(16.dp))
193
194             Text(
195                 text = "About:",
196                 style =
    MaterialTheme.typography.titleMedium,
197                 fontWeight = FontWeight.Bold
198             )
199             Spacer(modifier = Modifier.height(4.dp))
200             Text(
201                 text = club?.description ?:"",
202                 textAlign = TextAlign.Justify,
203                 style =
    MaterialTheme.typography.bodyMedium
204             )
205             Log.i("ClubDetailScreen", "Club
    description is loaded")
206         }
207     }
208 }
209
210 @Composable
211 fun YearTagOutlined(year: String) {
212     val tagColor = Color(0xFFD2B571)
213     Surface(
214         shape = RoundedCornerShape(50),
215         border = BorderStroke(2.dp, tagColor),
```

```
216         color = Color.Transparent,
217         tonalElevation = 1.dp // Optional
218     ) {
219         Text(
220             text = year,
221             style =
    MaterialTheme.typography.titleMedium,
222             modifier = Modifier.padding(horizontal =
    12.dp, vertical = 6.dp),
223             fontSize = 12.sp,
224             color = tagColor
225         )
226     }
227     Log.i("ClubDetailScreen", "Club treble year is
    loaded")
228 }
229
230 @OptIn(ExperimentalGlideComposeApi::class)
231 @Composable
232 fun CompetitionCard(competition: Competition) {
233     val context = LocalContext.current
234     val logo = if (context.isDarkTheme() &&
    competition.logoUrlDark.isNotBlank()) {
235         competition.logoUrlDark
236     } else {
237         competition.logoUrl
238     }
239
240     LaunchedEffect(competition) {
241         Log.i("ClubDetailScreen", "Competition
    ${competition.name} loaded logo ${if
    (context.isDarkTheme()) "dark" else "normal"}")
242     }
243
244     Card(
245         modifier = Modifier
246             .width(100.dp)
247             .height(200.dp),
248         shape = MaterialTheme.shapes.medium,
249         elevation =
    CardDefaults.cardElevation(defaultElevation = 4.dp)
250     ) {
251         Column(
252             modifier = Modifier.fillMaxSize(),
253             verticalArrangement = Arrangement.Top,
254             horizontalAlignment =
    Alignment.CenterHorizontally
```

```
255         ) {
256             Spacer(modifier = Modifier.height(10.dp))
257
258             GlideImage(
259                 model = logo,
260                 contentDescription = competition.name,
261                 modifier = Modifier.size(80.dp),
262                 contentScale = ContentScale.Fit
263             )
264
265             Text(
266                 text = competition.name,
267                 style =
    MaterialTheme.typography.titleMedium,
268                 textAlign = TextAlign.Center
269             )
270             Log.i("ClubDetailScreen", "Competition
    name is loaded")
271         }
272     }
273 }
```

36. ClubDetailsUIState.kt

Tabel 36. Source Code ClubDetailsUIState.kt

```
 1 package
   com.example.treblewinner.presentation.clubdetails
 2
 3 import com.example.treblewinner.domain.model.Club
 4 import
   com.example.treblewinner.domain.model.Competition
 5
 6 data class ClubDetailsUIState(
 7     val isLoading: Boolean = true,
 8     val club: Club? = null,
 9     val competitions: List<Competition> = emptyList(),
10     val error: String? = null
11 )
```

37. ClubDetailsViewModel.kt

Tabel 37. Source Code ClubDetailsViewModel.kt

```
 1 package
   com.example.treblewinner.presentation.clubdetails
 2
```

```
 3 import android.util.Log
 4 import androidx.lifecycle.ViewModel
 5 import androidx.lifecycle.viewModelScope
 6 import
   com.example.treblewinner.domain.usecase.GetAllCompetiti
   onsUseCase
 7 import
   com.example.treblewinner.domain.usecase.GetClubDetailUs
   eCase
   import
 8 com.example.treblewinner.domain.usecase.ToggleClubBookm
   arkUseCase
 9 import dagger.hilt.android.lifecycle.HiltViewModel
10 import kotlinx.coroutines.flow.MutableStateFlow
11 import kotlinx.coroutines.flow.StateFlow
12 import kotlinx.coroutines.flow.first
13 import kotlinx.coroutines.flow.update
14 import kotlinx.coroutines.launch
15 import javax.inject.Inject
16
17 @HiltViewModel
18 class ClubDetailsViewModel @Inject constructor(
19     private val getClubDetailUseCase:
   GetClubDetailUseCase,
20     private val toggleClubBookmarkUseCase:
   ToggleClubBookmarkUseCase,
21     private val getAllCompetitionsUseCase:
   GetAllCompetitionsUseCase
22 ): ViewModel() {
23
24     private val _uiState =
   MutableStateFlow(ClubDetailsUIState())
25     val uiState: StateFlow<ClubDetailsUIState> =
   _uiState
26
27     fun loadClubDetail(clubId: String) {
28         viewModelScope.launch {
29             try {
30                 _uiState.update { it.copy(isLoading =
   true) }
31
32                 val club = getClubDetailUseCase(clubId)
33                 val allCompetitions =
   getAllCompetitionsUseCase().first()
34                 val filteredCompetitions =
   allCompetitions.filter { competition ->
```

```
35                      competition.id in
(club?.competitionIds ?: emptyList())
36                  }
37
38                  _uiState.update {
39                      it.copy(
40                          isLoading = false,
41                          club = club,
42                          competitions =
filteredCompetitions,
43                          error = null
44                      )
45                  }
46
47              } catch (e: Exception) {
48                  Log.e("ClubDetailViewModel", "Error
loading club detail", e)
49                  _uiState.update {
50                      it.copy(isLoading = false, error =
e.message ?: "Unknown error")
51                  }
52              }
53          }
54      }
55
56      fun toggleBookmark() {
57          val currentClub = _uiState.value.club ?: return
58          viewModelScope.launch {
59              toggleClubBookmarkUseCase(currentClub.id)
60              _uiState.update {
61                  it.copy(club =
currentClub.copy(isBookmarked =
!currentClub.isBookmarked))
62              }
63          }
64      }
65  }
```

38. ClubListScreen.kt

Tabel 38. Source Code ClubListScreen.kt

```
1  package com.example.treblewinner.presentation.clublist
2
3  import android.content.ActivityNotFoundException
4  import android.content.Intent
5  import android.util.Log
```

```
 6 import android.widget.Toast
 7 import androidx.compose.foundation.layout.*
 8 import androidx.compose.foundation.lazy.LazyColumn
 9 import androidx.compose.foundation.lazy.LazyListState
10 import androidx.compose.foundation.lazy.items
11 import
   androidx.compose.foundation.lazy.rememberLazyListState
12 import androidx.compose.material.icons.Icons
13 import androidx.compose.material.icons.filled.Bookmark
14 import
   androidx.compose.material.icons.outlined.BookmarkAdd
15 import androidx.compose.material3.*
16 import androidx.compose.runtime.Composable
17 import androidx.compose.runtime.LaunchedEffect
18 import androidx.compose.runtime.collectAsState
19 import androidx.compose.ui.Alignment
20 import androidx.compose.ui.Modifier
21 import androidx.compose.ui.platform.LocalContext
22 import androidx.compose.ui.unit.dp
23 import com.example.treblewinner.domain.model.Club
24 import androidx.hilt.navigation.compose.hiltViewModel
25 import
   com.bumptech.glide.integration.compose.ExperimentalGli
   deComposeApi
26 import
   com.bumptech.glide.integration.compose.GlideImage
27 import androidx.compose.runtime.getValue
28 import androidx.compose.runtime.remember
29 import androidx.compose.ui.tooling.preview.Preview
30 import androidx.core.net.toUri
31
32
33 @OptIn(ExperimentalMaterial3Api::class,
   ExperimentalGlideComposeApi::class)
34 @Composable
35 fun ClubListScreen(
36     onNavigateToDetail: (String) -> Unit,
37     lazyListState: LazyListState =
   rememberLazyListState(),
38     viewModel: ClubListViewModel = hiltViewModel()
39 ){
40     val context = LocalContext.current
41     val uiState by viewModel.uiState.collectAsState()
42
43     LaunchedEffect(Unit) {
44         viewModel.loadData()
45     }
```

```
46
47
48      when {
49          uiState.isLoading -> {
50              Box(
51                  modifier = Modifier.fillMaxSize(),
52                  contentAlignment = Alignment.Center
53              ) {
54                  CircularProgressIndicator()
55              }
56          }
57          uiState.error != null -> {
58              Box(
59                  modifier = Modifier.fillMaxSize(),
60                  contentAlignment = Alignment.Center
61              ) {
62                  Text(text = "Error: ${uiState.error}")
63              }
64          }
65          uiState.clubs.isEmpty() -> {
66              Text("No clubs found")
67          }
68          else -> {
69              LazyColumn(
70                  state = lazyListState,
71                  modifier = Modifier.fillMaxSize(),
72                  verticalArrangement =
    Arrangement.spacedBy(4.dp)
73              ) {
74                  items(uiState.clubs) { club ->
75                      ClubCard(
76                          club = club,
77                          onDetailClick = {
78                              Log.i("ClubListScreen",
    "User click the detail button for ${club.name}")
79
    onNavigateToDetail(club.id)
80                          },
81                          onVisitClick = {
82                              try {
83                                  val intent =
    Intent(Intent.ACTION_VIEW, club.webUrl.toUri())
84
    context.startActivity(intent)
85
    Log.d("ClubListScreen", "User going to the web
    ${club.name}")
```

```
                                } catch (e:
ActivityNotFoundException) {
                                    Log.e("ClubListScreen", "Browser not found", e)
                                    Toast.makeText(context, "No browser available",
Toast.LENGTH_SHORT).show()
                                }
                            },
                            onBookmarkClick = {
                                Log.i("ClubListScreen",
"User toggled bookmark for ${club.name}")
                                viewModel.toggleBookmark(club.id)
                            }
                        )
                    }
                }
            }
        }
    }
}

@OptIn(ExperimentalGlideComposeApi::class)
@Composable
fun ClubCard(
    club: Club,
    onDetailClick: () -> Unit,
    onVisitClick: () -> Unit,
    onBookmarkClick: () -> Unit
) {
    LaunchedEffect(club.id) {
        Log.d("ClubCard", "Rendering ${club.name},
isBookmarked = ${club.isBookmarked}")
    }


    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(6.dp)
    ) {
        Row(
            modifier = Modifier
                .fillMaxWidth()
                .padding(8.dp),
            verticalAlignment =
Alignment.CenterVertically
```

```
125         ) {
126             GlideImage(
127                 model = club.logoUrl,
128                 contentDescription = club.name,
129                 modifier = Modifier
130                     .size(100.dp)
131                     .aspectRatio(1f)
132                     .padding(end = 8.dp)
133             )
134 
135             Column(modifier = Modifier.weight(1f)) {
136                 Text(
137                     text = club.name,
138                     style =
    MaterialTheme.typography.headlineSmall
139                 )
140                 Text(
141                     text =
    club.trebleYears.joinToString(", "),
142                     style =
    MaterialTheme.typography.bodyMedium
143                 )
144                 Spacer(modifier =
    Modifier.height(8.dp))
145                 Row(
146                     modifier =
    Modifier.fillMaxWidth(),
147                     horizontalArrangement =
    Arrangement.SpaceEvenly
148                 ) {
149                     Button(onClick = onDetailClick) {
150                         Text("Details")
151                     }
152                     Button(onClick = onVisitClick) {
153                         Text("Visit Site")
154                     }
155                     IconButton(onClick = {
156                         Log.d("BookmarkToggle",
    "Toggle bookmark: ${club.name}, current:
    ${club.isBookmarked}")
157                         onBookmarkClick()
158                     }) {
159                         Icon(
160                             imageVector = if
    (club.isBookmarked == true) Icons.Filled.Bookmark else
    Icons.Outlined.BookmarkAdd,
161                             contentDescription = null,
```

```
                                modifier =
Modifier.size(32.dp)
                            )
                        }
                    }
                }
            }
        }
}


val sampleClubs = listOf(
    Club(
        id = "fc_barcelona",
        name = "FC Barcelona",
        country = "Spain",
        confederation = "UEFA",
        trebleYears = listOf("2009", "2015"),
        competitionIds = listOf("la_liga",
"copa_del_rey", "uefa_champions_league"),
        logoUrl =
"https://blogger.googleusercontent.com/img/b/R29vZ2xl/
AVvXsEh4xprb5TfHqTe6hCvl4hiV6pdlgPfiG_722ZGkfNOPbK1K7b
WrklpdZ2wMR_qvSuCSpXuLKMSGAH7IhB9PY61vG5ctNQ4-R-
Je18Uq5-oYEN8pfP0z7c7-EtQE_gjr-
iDR2D3t6F26mr8/s16000/FC+Barcelona.png",
        webUrl = "https://www.fcbarcelona.com",
        description = "One of the greatest football
clubs in the world.",
        isBookmarked = false
    ),
    Club(
        id = "manchester_united",
        name = "Manchester United",
        country = "England",
        confederation = "UEFA",
        trebleYears = listOf("1999"),
        competitionIds = listOf("premier_league",
"fa_cup", "uefa_champions_league"),
        logoUrl =
"https://upload.wikimedia.org/wikipedia/en/7/7a/Manche
ster_United_FC_crest.svg",
        webUrl = "https://www.manutd.com",
        description = "Legendary English football club
based in Manchester.",
        isBookmarked = true
    ),
```

```
198     Club(
199         id = "bayern_munchen",
200         name = "Bayern München",
201         country = "Germany",
202         confederation = "UEFA",
203         trebleYears = listOf("2013"),
204         competitionIds = listOf("bundesliga",
    "dfb_pokal", "uefa_champions_league"),
205         logoUrl =
    "https://upload.wikimedia.org/wikipedia/en/1/1f/FC_Bay
    ern_München_logo_%282017%29.svg",
206         webUrl = "https://fcbayern.com",
207         description = "The most successful club in
    German football history.",
208         isBookmarked = false
209     )
210 )
211
212
    @Preview(
213     name = "My Phone (1080x2400)",
214     widthDp = 360,
215     heightDp = 800,
216     showBackground = true,
217     showSystemUi = true
218 )
219 @Composable
220 fun ClubListPreview0() {
221     LazyColumn(
222         modifier = Modifier.fillMaxSize(),
223         verticalArrangement =
    Arrangement.spacedBy(4.dp)
224     ) {
225         items(sampleClubs) { club ->
226             ClubCard(
227                 club = club,
228                 onDetailClick = { Log.d("Preview",
    "Detail clicked for ${club.name}") },
229                 onVisitClick = { Log.d("Preview",
    "Visit clicked for ${club.name}") },
230                 onBookmarkClick = { Log.d("Preview",
    "Bookmark toggled for ${club.name}") }
231             )
232         }
233     }
234 }
235
```

```
@Preview(
    name = "Pixel 9 Pro XL (1344 x 2992)",
    widthDp = 448,
    heightDp = 998,
    showBackground = true,
    showSystemUi = true
)
@Composable
fun ClubListPreview1() {
    LazyColumn(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement =
Arrangement.spacedBy(4.dp)
    ) {
        items(sampleClubs) { club ->
            ClubCard(
                club = club,
                onDetailClick = { Log.d("Preview",
"Detail clicked for ${club.name}") },
                onVisitClick = { Log.d("Preview",
"Visit clicked for ${club.name}") },
                onBookmarkClick = { Log.d("Preview",
"Bookmark toggled for ${club.name}") }
            )
        }
    }
}
```

39. ClubListUIState.kt

Tabel 39. Source Code ClubListUIState.kt

```
package com.example.treblewinner.presentation.clublist

import com.example.treblewinner.domain.model.Club

data class ClubListUIState(
    val isLoading: Boolean = false,
    val clubs: List<Club> = emptyList(),
    val error: String? = null
)
```

40. ClubListViewModel.kt

Tabel 40. Source Code ClubListViewModel.kt

```
 1 package com.example.treblewinner.presentation.clublist
 2
 3 import android.util.Log
 4 import androidx.lifecycle.ViewModel
 5 import androidx.lifecycle.viewModelScope
 6 import
   com.example.treblewinner.domain.usecase.FetchAndCacheDa
   taUseCase
 7 import
   com.example.treblewinner.domain.usecase.GetAllClubsUseC
   ase
 8 import
   com.example.treblewinner.domain.usecase.ToggleClubBookm
   arkUseCase
 9 import dagger.hilt.android.lifecycle.HiltViewModel
10 import kotlinx.coroutines.flow.MutableStateFlow
11 import kotlinx.coroutines.flow.StateFlow
12 import kotlinx.coroutines.flow.asStateFlow
13 import kotlinx.coroutines.flow.catch
14 import kotlinx.coroutines.flow.update
15 import kotlinx.coroutines.launch
16 import javax.inject.Inject
17
18 @HiltViewModel
19 class ClubListViewModel @Inject constructor(
20     private val getAllClubsUseCase: GetAllClubsUseCase,
21     private val toggleClubBookmarkUseCase:
   ToggleClubBookmarkUseCase,
22     private val fetchAndCacheDataUseCase:
   FetchAndCacheDataUseCase
23 ) : ViewModel(){
24
25     private val _uiState =
   MutableStateFlow(ClubListUIState())
26     val uiState: StateFlow<ClubListUIState> =
   _uiState.asStateFlow()
27
28     fun toggleBookmark(clubId: String) {
29         viewModelScope.launch {
30             toggleClubBookmarkUseCase(clubId)
31             val updatedClubs = _uiState.value.clubs.map
   { club ->
32                 if (club.id == clubId)
   club.copy(isBookmarked = !club.isBookmarked)
```

```
33                  else club
34              }
35              _uiState.update { it.copy(clubs =
    updatedClubs) }
36          }
37      }
38
39
40      private var hasLoaded = false
41
42      fun loadData() {
43          if (!hasLoaded) {
44              Log.i("ClubVM", "First load data")
45              viewModelScope.launch {
46                  try {
47                      fetchAndCacheDataUseCase()
48                      getAllClubsUseCase()
49                          .catch { e ->
50                              Log.e("ClubVM", "Error
    loading clubs", e)
51                          }
52                          .collect { result ->
53                              Log.i("ClubVM", "Received
    clubs: ${result.size}")
54                              _uiState.value =
    ClubListUIState(
55                                  isLoading = false,
56                                  clubs = result,
57                                  error = null
58                              )
59                              hasLoaded = true
60                          }
61                  } catch (e: Exception) {
62                      Log.e("ClubVM", "Unexpected error",
    e)
63                      _uiState.value = ClubListUIState(
64                          isLoading = false,
65                          clubs = emptyList(),
66                          error = e.message ?: "Unknown
    error"
67                      )
68                  }
69              }
70          }
71          else {
72              Log.i("ClubVM", "Data already loaded!")
73          }
```

```
74      }
75  }
```

41. HomeScreen.kt

Tabel 41. Source Code HomeScreen.kt

```
 1  package com.example.treblewinner.presentation.home
 2
 3  import androidx.compose.material3.*
 4  import androidx.compose.runtime.*
 5  import androidx.compose.ui.Modifier
 6  import androidx.compose.foundation.layout.*
 7  import
    androidx.compose.foundation.pager.rememberPagerState
 8  import androidx.compose.ui.graphics.vector.ImageVector
 9  import androidx.compose.material.icons.Icons
10  import
    androidx.compose.material.icons.automirrored.filled.Lis
    t
11  import androidx.compose.material.icons.filled.Star
12  import
    androidx.compose.runtime.saveable.rememberSaveable
13  import
    com.example.treblewinner.presentation.clublist.ClubList
    Screen
14  import
    com.example.treblewinner.presentation.bookmark.Bookmark
    edClubsScreen
15  import androidx.compose.foundation.lazy.LazyListState
16  import
    androidx.compose.foundation.pager.HorizontalPager
17  import androidx.compose.runtime.LaunchedEffect
18  import kotlinx.coroutines.launch
19
20
21  @OptIn(ExperimentalMaterial3Api::class)
22  @Composable
23  fun HomeScreen(
24      onNavigateToDetail: (clubId: String) -> Unit,
25  ) {
26      val tabs = listOf(
27          NavTab("All Clubs",
    Icons.AutoMirrored.Filled.List),
28          NavTab("Bookmarks", Icons.Default.Star)
29      )
30
```

```
31      val pagerState = rememberPagerState(pageCount = {
    tabs.size })
32      var selectedTabIndex by rememberSaveable {
    mutableIntStateOf(0)}
33      val scope = rememberCoroutineScope()
34
35      val scrollStates = remember {
36          List(tabs.size) { LazyListState() }
37      }
38
39      Scaffold(
40          topBar = {
41              TopAppBar(
42                  title = { Text("Treble Winner Clubs") }
43              )
44          },
45          bottomBar = {
46              NavigationBar {
47                  tabs.forEachIndexed { index, tab ->
48                      NavigationBarItem(
49                          icon = { Icon(tab.icon,
    contentDescription = tab.title) },
50                          label = { Text(tab.title) },
51                          selected = selectedTabIndex ==
    index,
52                          onClick = {
53                              selectedTabIndex = index
54
55                              scope.launch {
56
    pagerState.animateScrollToPage(index)
57                              }
58                          }
59                      )
60                  }
61              }
62          }
63      ) { innerPadding ->
64
65          HorizontalPager(
66              state = pagerState,
67              modifier = Modifier.padding(innerPadding),
68              userScrollEnabled = true,
69          ) { page ->
70              when (page) {
71                  0 -> ClubListScreen(
```

```
                        onNavigateToDetail =
onNavigateToDetail,
                        lazyListState = scrollStates[0]
                    )
                    1 -> BookmarkedClubsScreen(
                        onNavigateToDetail =
onNavigateToDetail,
                        lazyListState = scrollStates[1]
                    )
                }
            }

            LaunchedEffect(pagerState.currentPage) {
                selectedTabIndex = pagerState.currentPage
            }
        }
}

data class NavTab(val title: String, val icon:
ImageVector)
```

42. HomeViewModel.kt

Tabel 42. Source Code HomeViewModel.kt

```
package com.example.treblewinner.presentation.home

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import
com.example.treblewinner.domain.repository.TrebleWinner
Repository
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class HomeViewModel @Inject constructor(
    private val repository: TrebleWinnerRepository
) : ViewModel() {

    init {
        viewModelScope.launch {
            repository.fetchAndCacheData()
        }
    }
}
```

43. Navigation.kt

Tabel 43. Source Code Navigotion.kt

```
 1  package
    com.example.treblewinner.presentation.navigation
 2
 3  import androidx.compose.runtime.Composable
 4  import androidx.navigation.compose.NavHost
 5  import
    androidx.navigation.compose.rememberNavController
 6  import androidx.navigation.compose.composable
 7  import
    com.example.treblewinner.presentation.home.HomeScreen
 8  import
    com.example.treblewinner.presentation.clubdetails.ClubD
    etailsScreen
 9  import
    com.example.treblewinner.presentation.clublist.ClubList
    Screen
10
11  sealed class Screen(val route: String) {
12      object Main : Screen("main")
13      object List : Screen("club_list")
14      object Detail : Screen("detail/{clubId}") {
15          fun createRoute(clubId: String) =
    "detail/$clubId"
16      }
17  }
18
19  @Composable
20  fun Navigation(){
21      val navController = rememberNavController()
22
23      NavHost(
24          navController = navController,
25          startDestination = Screen.Main.route
26      ){
27          composable(Screen.Main.route){
28              HomeScreen(
29                  onNavigateToDetail = { clubId ->
30
    navController.navigate(Screen.Detail.createRoute(clubId
    ))
31                  }
32              )
33          }
34
```

```
35          composable(Screen.List.route) {
36              ClubListScreen(
37                  onNavigateToDetail = { clubId ->
38
    navController.navigate(Screen.Detail.createRoute(clubId
    ))
39                  }
40              )
41          }
42
43          composable(Screen.Detail.route) {
    backStackEntry ->
44              val clubId =
    backStackEntry.arguments?.getString("clubId") ?: ""
45              ClubDetailsScreen(
46                  clubId = clubId,
47                  onBack = { navController.popBackStack()
    }
48              )
49          }
50      }
51  }
```

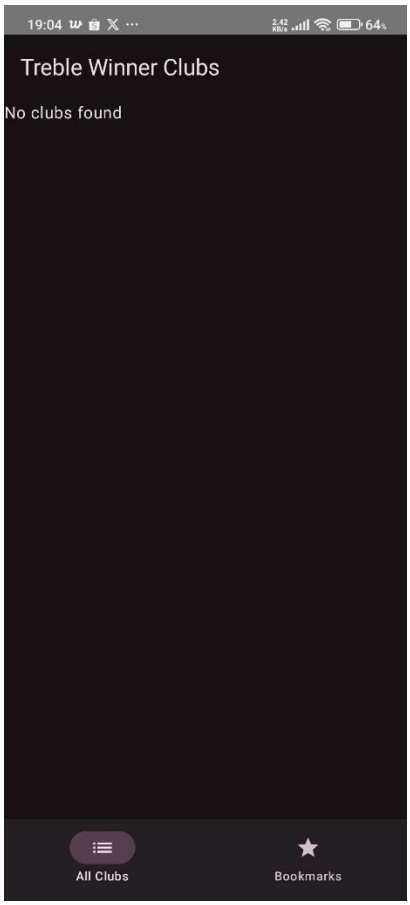
44. TrebleWinnerTheme.kt

Tabel 44. Source Code TrebleWinnerTheme.kt

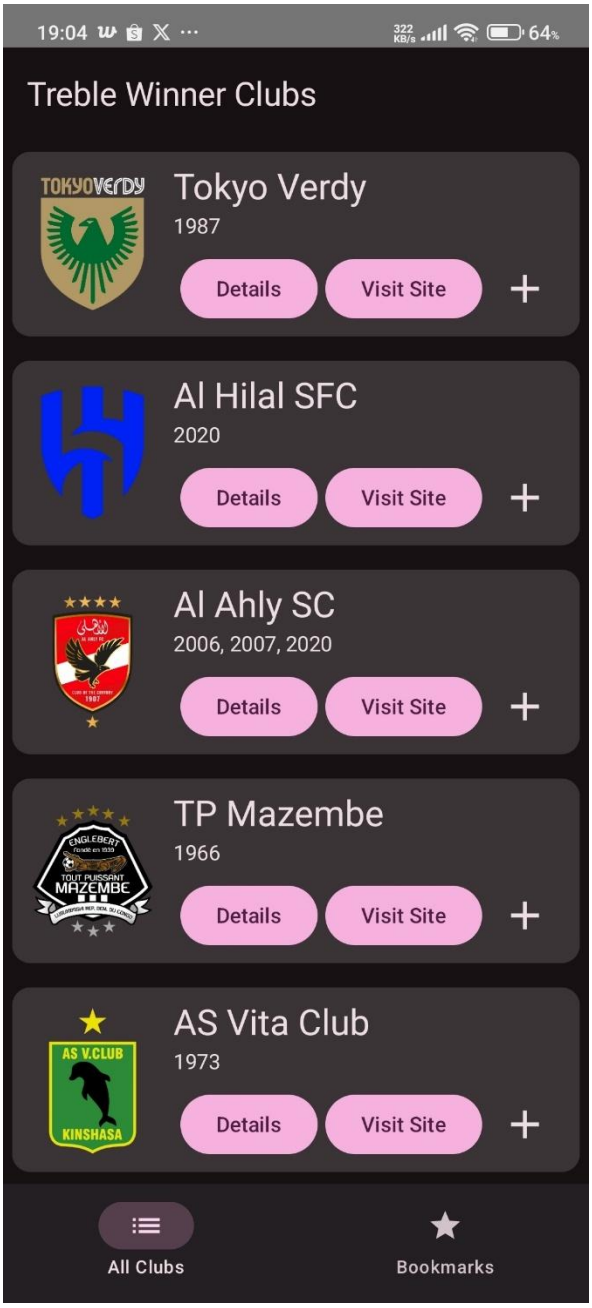
```
 1  package com.example.treblewinner.presentation.theme
 2
 3  import
    androidx.compose.material3.dynamicDarkColorScheme
 4  import
    androidx.compose.material3.dynamicLightColorScheme
 5  import androidx.compose.runtime.Composable
 6  import androidx.compose.ui.platform.LocalContext
 7  import androidx.compose.foundation.isSystemInDarkTheme
 8  import androidx.compose.material3.MaterialTheme
 9
10  @Composable
11  fun TrebleWinnerTheme(
12      darkTheme: Boolean = isSystemInDarkTheme(),
13      content: @Composable () -> Unit
14  ) {
15      val context = LocalContext.current
16
17      val colorScheme = if (darkTheme) {
18          dynamicDarkColorScheme(context)
```

```
19      } else {
20          dynamicLightColorScheme(context)
21      }
22
23      MaterialTheme(
24          colorScheme = colorScheme,
25          content = content
26      )
27  }
```

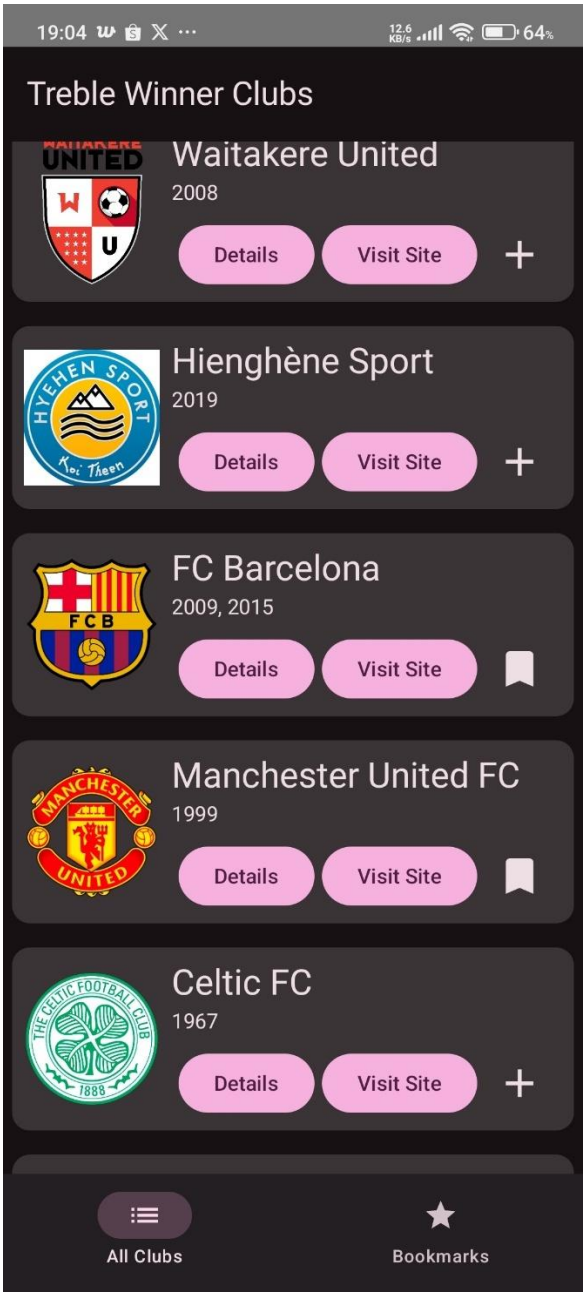
B. Output Program



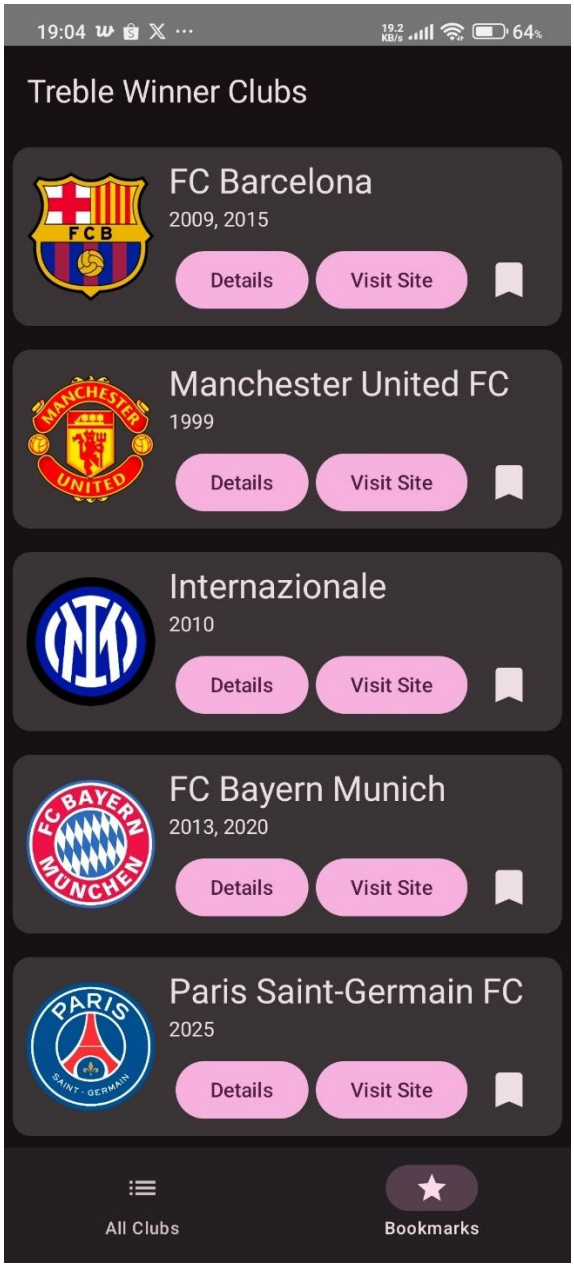
Gambar 1. Tampilan Apikasi Saat Pertama Kali Dibuka (Sedang Mengambil Data dari API)



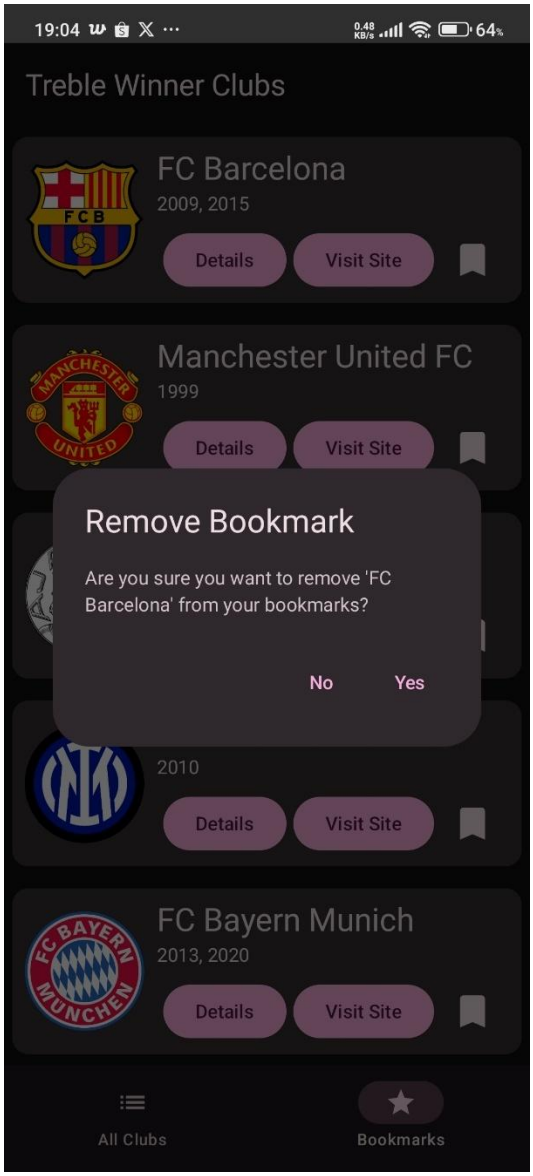
Gambar 2. Tampilan Aplikasi Setelah Data Dimuat



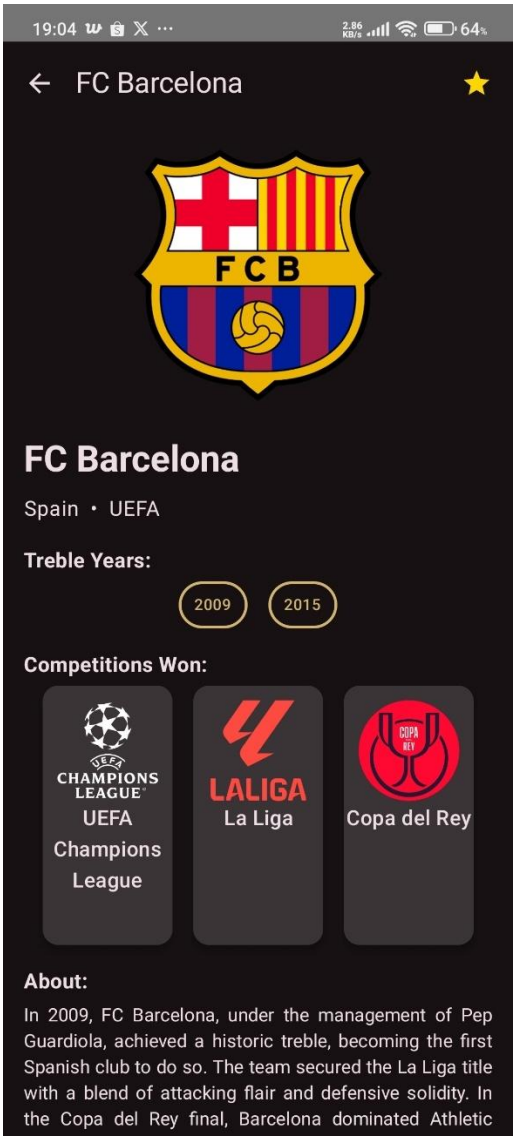
Gambar 3. Tampilan Beberapa Klub yang Ditandai "Favorit" di Halaman Daftar Klub



Gambar 4. Tampilan Halaman Daftar Klub Favorit



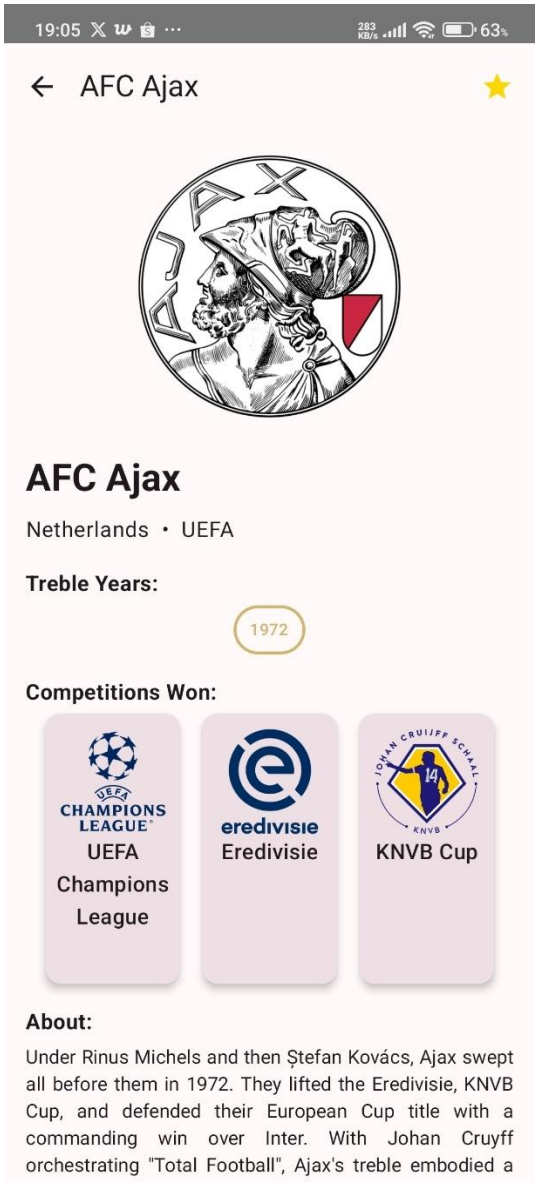
Gambar 5. Tampilan Pop-up Alert Saat Pengguna Menekan Tombol Bookmark untuk Menghapus Klub dari Daftar Favorit



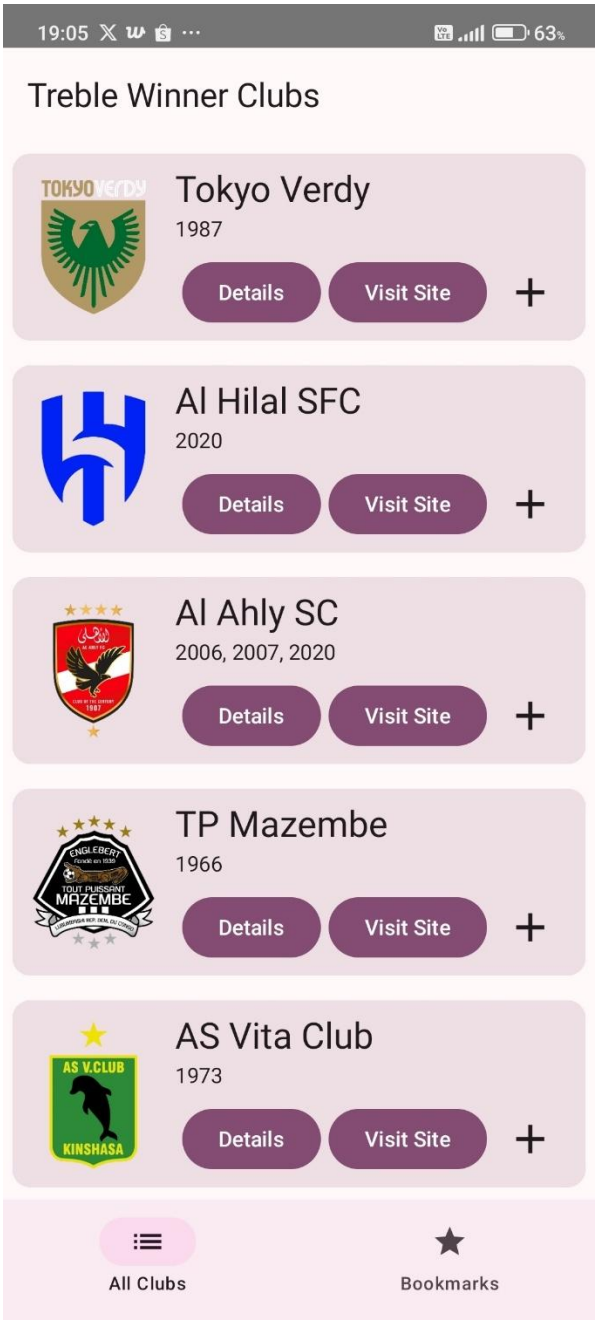
Gambar 6. Tampilan Halaman Detail Klub FC Barcelona saat Pengguna Mengeklik "Details"

Gambar 7. Tampilan Halaman Situs Resmi Klub FC Barcelona saat Pengguna Mengeklik "Visit Site"

Gambar 8. Tampilan Halaman Detail Klub Ajax saat Mode Gelap



Gambar 9. Tampilan Halaman Detail Klub Ajax saat Mode Terang



Gambar 10. Tampilan Aplikasi yang dapat Memuat Data Meski Tanpa Internet

C. Pembahasan

Pada modul 5 ini saya melakukan eksperimen menggunakan pendekatan clean architecture. Maka dari itu, file-filenya sangat banyak. Selain itu, karena tema atau data aplikasi yang saya buat sangatlah spesifik dan tidak ada sumber data dari API manapun, maka dibuatlah API custom yang saya buat sendiri menggunakan Fast API dari Python yang dapat dilihat di repositori github modul ini.

Umum

1. MainActivity.kt
   File ini merupakan file utama (main) yang akan dieksekusi oleh program sebagai titik awal

2. TrebleWinnerTheme.kt
   File ini merupakan tempat konfigurasi tema, seperti warna, tipografi, bentuk, dan lain-lain yang digunakan di aplikasi agar konsisten dan rapi.

3. AndroidManifest.xml
   File ini merupakan file konfigurasi dasar, seperti pengaturan izin akses yang diperlukan karena menggunakan internet untuk mengambil data dari API, versi SDK, dan lain-lain.

Package: data

Folder ini berperan sebagai pengelola sumber data yang digunakan di aplikasi, baik data dari database (server) maupun lokal.

4. RepositoryModule.kt
   File ini bertujuan untuk menyediakan dependency injection untuk implementasi aplikasi saya.

5. Converters.kt
   File ini bertujuan untuk melakukan konversi data yang kompleks. Dalam kasus saya, saya perlu menyimpan id kompetisi dari tiap klub dalam sebuah array JSON yang nantinya perlu dikonvesi menjadi array list string.

6. TrebleWinnerDao.kt
   File ini berisi DAO (Data Access Object) yang merupakan kueri-kueri untuk mengakses data dari database Room.

7. AppDatabase.kt
   File ini merupakan kelas utama database Room aplikasi saya. Kelas ini merupakan kelas abstrak yang di-extends dari RoomDatabase yang berisi entitas-entitas dan DAO yang akan digunakan.

8. DatabaseModule.kt
   File ini merupakan modul dependency injection untuk menyediakan instance dari AppDatabase dan TrebleWinnerDao agar database diinisialisasi dengan tepat.

9. ClubCompetitionCrossRef.kt
   File ini merupakan kelas untuk mengelola relasi dari entitas yang dibuat. File ini diperlukan karena hubungan antara klub dan kompetisi adalah many-to-many.

10. ClubEntity.kt

File ini berisikan representasi tabel dari data klub di dalam database lokal aplikasi. Berisikan nama tabel, kolom, tipe data, indeks kunci, dan lain-lain. Selain itu, terdapat kolom tambahan is_bookmarked untuk menyimpan apakah klub ditandai (sebagai favorit) oleh pengguna atau tidak dengan nilai default false. Karena data tersebut tidak disimpan di API/server.

11. CompetitionEntity.kt
    File ini berisikan representasi tabel dari data kompetisi di dalam database lokal aplikasi. Berisikan nama tabel, kolom, tipe data, indeks kunci, dan lain-lain.

12. ClubWithCompetitions.kt
    File ini berfungsi untuk mengambil data klub dan kompetisi sekaligus. Karena saat memuat detail nanti akan menampilkan data dari kedua entitas tersebut.

13. DtoMapper.kt
    File ini berfungsi sebagai mapper (memetakan) untuk mengubah data transfer object (DTO) dari API menjadi entitas atau tabel database.

14. RemoteDataSource.kt
    File ini berfungsi sebagai abstraksi untuk mengakses data dari API, validasi, dan penanganan eror.

15. TrebleWinnerAPI.kt
    File ini berfungsi sebagai interface yang berisikan HTTP request ke API untuk mengambil datanya ke lokal.

16. NetworkModule.kt
    File ini berfungsi untuk meyediakan instance Retrofit dan OkHttpClient untuk menghubungi API dan memastikan konfigurasi jaringan yang digunakan sudah benar.

17. ClubDto.kt
    File DTO ini merupakan representasi data dari API (JSON) untuk klub.

18. CompetitionDto.kt
    File DTO ini merupakan representasi data dari API (JSON) untuk kompetisi.

19. ResponseMapper.kt
    File ini berfungsi sebagai mapper (memetakan) untuk mengubah data response dari API menjadi model di domain.

20. ClubResponse.kt
    File ini merupakan representasi data respons API untuk data klub.

21. CompetitionResponse.kt
    File ini merupakan representasi data respons API untuk data kompetisi.

22. TrebleWinnerRepositoryImpl.kt
    File ini merupakan implementasi dari TrebleWinnerRepository yang mengatur bagaimana pengambilan data dari RemoteDataSource dilakukan dan penyimpanan data ke Room.

Package: domain
Folder ini berfungsi sebagai lapisan logika bisnis dari model data secara independen dari database dan UI.

23. Club.kt
    File ini merupakan model dari data (kelas) klub yang akan digunakan di aplikasi.

24. Competition.kt
    File ini merupakan model dari data (kelas) kompetisi yang akan digunakan di aplikasi.

25. TrebleWinnerRepository.kt
    File ini merupakan interface yang mendefinisikan syarat atau kontrak apa saja yang aplikasi harus kelakukan.

26. FetchAndCacheDataUseCase.kt
    File ini merupakan interactor use case yang berfungsi untuk mengambil data dari API lalu menyimpannya (cache) ke database lokal agar data tetap sinkron.

27. GetAllClubsUseCase.kt
    File ini merupakan interactor use case yang berfungsi untuk mengambil semua daftar klub yang ada.

28. GetAllCompetitionsUseCase.kt
    File ini merupakan interactor use case yang berfungsi untuk mengambil semua daftar kompetisi yang ada.

29. GetBookmarkedClubUseCase.kt
    File ini merupakan interactor use case yang berfungsi untuk mengambil semua daftar klub favorit yang ada.

30. GetClubDetailUseCase.kt
    File ini merupakan interactor use case yang berfungsi untuk mengambil semua daftar detail klub yang ada.

31. ToggleClubBookmarkUseCase.kt
    File ini merupakan interactor use case yang berfungsi untuk menandai status favorit pada daftar klub.

Package: presentation
Folder ini berisikan semua komponen yang berkaitan dengan UI.

32. BookmarkedClubsScreen.kt
    File ini merupakan fungsi Composable yang mendefinisikan layar tampilan untuk halaman daftar klub favorit (bookmark).

33. BookmarkedClubsUIState.kt
    File ini bersikan data apa saja yang diperlukan (daftar klub favorit), status UI (seperti loading atau eror) saat halaman daftar klub favorit dibuka (di-render).

34. BookmarkedClubsViewModel.kt
    File ini berfungsi untuk mengelola logika dan status halaman daftar klub favorit. Serta memanggil use case yang diperlukan, yaitu GetBookmarkedClubUseCase dan ToggleClubBookmarkUseCase.

35. ClubDetailsScreen.kt
    File ini merupakan fungsi Composable yang mendefinisikan layar tampilan untuk halaman detail klub.

36. ClubDetailsUIState.kt
    File ini bersikan data apa saja yang diperlukan (data rinci kompetisi, tahun, dan teks perjalanan klub mencapai treble), status UI (seperti loading atau eror) saat halaman detail klub dibuka (di-render).

37. ClubDetailsViewModel.kt
    File ini berfungsi untuk mengelola logika dan status halaman daftar klub favorit. Serta memanggil use case yang diperlukan, yaitu GetClubDetailUseCase, GetAllCompetitionsUseCase, dan ToggleClubBookmarkUseCase.

38. ClubListScreen.kt
    File ini merupakan fungsi Composable yang mendefinisikan layar tampilan untuk halaman daftar klub.

39. ClubListUIState.kt
    File ini bersikan data apa saja yang diperlukan (daftar klub beserta tahun treble-nya), status UI (seperti loading atau eror) saat halaman daftar klub dibuka (di-render).

40. ClubListViewModel.kt
    File ini berfungsi untuk mengelola logika dan status halaman daftar klub favorit. Serta memanggil use case yang diperlukan, yaitu GetAllClubsUseCase, FetchAndCacheDataUseCase dan ToggleClubBookmarkUseCase.

41. HomeScreen.kt
    File ini merupakan halaman utama aplikasi yang akan memuat dua halaman sekaligus, yaitu halaman daftar klub dan halaman daftar klub favorit, yang dipisah menggunakan horizontal pager serta agar tiap halaman tidak terender ulang saat membuka Kembali halamannya.

42. HomeViewModel.kt
    File ini berfungsi untuk mengelola logika halaman utama. Saat aplikasi dimulai, file ini akan memanggil FetchAndCacheDataUseCase untuk memuat data.

43. Navigation.kt
    File ini berfungsi untuk mendefinisikan rute atau navigasi antar halaman dan argument apa yang akan dioper/ditransfer dari layer sebelumnya ke layer yang baru.

44. TrebleWinnerTheme.kt
    File ini berfungsi untuk mengatur tema khusus untuk komponen UI yang ada di folder atau lapisan presentasi/presentation. Konfigurasi yang dilakukan adalah untuk menangani mode gelap.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/MinamotoYuki46/MeineStudienArbeit/tree/main/MobileDevelopment/Codex-Practicus/Modul%205