LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

VIEWMODEL AND DEBUGGING

Oleh:

Muhammad Fauzan Ahsani NIM. 2310817310009

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE
MODUL 4

Laporan Praktikum Pemrograman Mobile Modul 4: ViewModel and Debugging 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 berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:

1. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
2. Gunakan ViewModelFactory dalam pembuatan ViewModel
3. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
4. Gunakan logging untuk event berikut:
   a. Log saat data item masuk ke dalam list
   b. Log saat tombol Detail dan tombol Explicit Intent ditekan
   c. Log data dari list yang dipilih ketika berpindah ke halaman Detail
5. Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi. Cari setidaknya satu breakpoint yang relevan dengan aplikasi. Lalu, gunakan fitur Step Into, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out

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 androidx.activity.enableEdgeToEdge
 7 import com.example.treblewinner.ui.theme.TrebleWinnerTheme
 8 import androidx.navigation.compose.rememberNavController
 9 import
   com.bumptech.glide.integration.compose.ExperimentalGlideCompo
   seApi
10 import com.example.treblewinner.navigation.AppNavHost
11
12 class MainActivity : ComponentActivity() {
13     @OptIn(ExperimentalGlideComposeApi::class)
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         enableEdgeToEdge()
17         setContent {
18             TrebleWinnerTheme {
19                 val navController = rememberNavController()
20                 AppNavHost(navController = navController)
```

```
            }
        }
    }
}
```

2. Club.kt

Tabel 2. Source Code Club.kt

```
package com.example.treblewinner.model

import android.os.Parcelable
import kotlinx.parcelize.Parcelize

@Parcelize
data class Club (
    val name: String,
    val country: String,
    val confederation: String,
    val trebleYears: List<String>,
    val competitions: List<Competition>,
    val logoUrl: String,
    val webUrl: String,
    val description: String
): Parcelable
```

3. Competition.kt

Tabel 3. Source Code Competitions.kt

```
package com.example.treblewinner.model

import android.os.Parcelable
import kotlinx.parcelize.Parcelize

@Parcelize
data class Competition (
    val name: String,
    val country: String,
    val confederation: String,
    val logoUrl: String,
    val logoUrlDark: String
): Parcelable
```

4. ClubConstant.kt

Tabel 4. Source Code ClubConstant.kt

```
package com.example.treblewinner.constants
```

```
 2
 3 import com.example.treblewinner.model.Club
 4 import kotlin.reflect.full.memberProperties
 5
 6 object ClubConstant {
 7     val BARCELONA = Club(
 8         name = "FC Barcelona",
 9         country = "Spain",
10         confederation = "UEFA",
11         trebleYears = listOf("2009", "2015"),
12         competitions = listOf(
13             CompetitionsConstant.LA_LIGA,
14             CompetitionsConstant.COPA_DEL_REY,
15             CompetitionsConstant.UEFA_CHAMPIONS_LEAGUE
16         ),
17         logoUrl =
   "https://blogger.googleusercontent.com/img/b/R29vZ2xl/A
   VvXsEh4xprb5TfHqTe6hCvl4hiV6pdlgPfiG_722ZGkfNOPbK1K7bWr
   klpdZ2wMR_qvSuCSpXuLKMSGAH7IhB9PY61vG5ctNQ4-R-Je18Uq5-
   oYEN8pfP0z7c7-EtQE_gjr-
   iDR2D3t6F26mr8/s16000/FC+Barcelona.png",
18         webUrl = "https://www.fcbarcelona.com",
19         description = """
               In 2009, FC Barcelona, under the management
   of Pep Guardiola, achieved a historic treble, becoming
   the first Spanish club to do so. The team secured the
   La Liga title with a blend of attacking flair and
   defensive solidity. In the Copa del Rey final,
   Barcelona dominated Athletic Bilbao with a 4-1 victory,
   showcasing their superiority in domestic competitions.
   The pinnacle of their season was the UEFA Champions
   League final, where they faced Manchester United.
   Barcelona triumphed 2-0, with goals from Samuel Eto'o
   and Lionel Messi, completing a season that redefined
   modern football with their tiki-taka style and cohesive
   team play.
20
21             Barcelona replicated their treble success
   in 2015 under manager Luis Enrique, becoming the first
   European club to achieve this feat twice. They clinched
   the La Liga title, demonstrating consistency and
   resilience throughout the season. In the Copa del Rey
   final, Barcelona defeated Athletic Bilbao 3-1, with
   Lionel Messi delivering a standout performance. The
   UEFA Champions League final saw Barcelona overcome
   Juventus 3-1, with goals from Ivan Rakitić, Luis
   Suárez, and Neymar. This treble was marked by the
```

```
    formidable attacking trio of Messi, Suárez, and Neymar,
    who were instrumental in Barcelona's dominance across
    all competitions.
22          """.trimIndent()
23      )
24  }
```

5. CompetitionsConstant.kt

Tabel 5. Source Code CompetitionsConstant.kt

```
 1  package com.example.treblewinner.constants
 2
 3  import com.example.treblewinner.model.Competition
 4
 5  object CompetitionsConstant{
 6      val UEFA_CHAMPIONS_LEAGUE = Competition(
 7          name = "UEFA Champions League",
 8          country = "Europe",
 9          confederation = "UEFA",
10          logoUrl =
    "https://blogger.googleusercontent.com/img/b/R29vZ2xl/A
    VvXsEiWR16FIxnI46HJeFwoEVVS7S8D_3XXpHo0LYEPEM-
    6DFks_dqRDrixc1LC065bDaKrLo9Rbh-
    AlY67dr7kQrH_zdzetMZ_bGDW686hZJXo1RBsS-
    X_xOjauC6QyXkKI09euk88wrxphFg/s16000/UEFACL.png",
11          logoUrlDark =
    "https://blogger.googleusercontent.com/img/b/R29vZ2xl/A
    VvXsEg7QlZm7AO2tnSmuc3FkayP98OGUbZ2ly2t5idIn9NtmX6r3Iph
    S1ifIASwkMkEJUYfuRofPHC8c3RPQ9xxpcYUnAE8-
    5dhTPCVOhfn8p6gKG1GHQrHJ59BAUQJYw7uKAZPXt6jpREPDfY/s512
    /UCL.png"
12      )
13  }
```

6. AppNavHost.kt

Tabel 6. Source Code AppNavHost.kt

```
 1  package com.example.treblewinner.navigation
 2
 3  import androidx.compose.foundation.layout.*
 4  import androidx.compose.material3.Text
 5  import androidx.compose.runtime.*
 6  import androidx.compose.ui.Alignment
 7  import androidx.compose.ui.Modifier
 8  import androidx.lifecycle.viewmodel.compose.viewModel
```

```
 9 import androidx.navigation.NavHostController
10 import androidx.navigation.compose.NavHost
11 import androidx.navigation.compose.composable
12 import com.example.treblewinner.constants.ClubConstant
13 import com.example.treblewinner.screen.ClubViewModel
14 import
   com.example.treblewinner.screen.clubdetails.ClubDetailS
   creen
15 import
   com.example.treblewinner.screen.clublist.ClubListScreen
16 import
   com.example.treblewinner.utils.ClubViewModelFactory
17
18 @Composable
19 fun AppNavHost(navController: NavHostController) {
20     val factory = remember {
   ClubViewModelFactory(ClubConstant.ALL) }
21     val clubVM: ClubViewModel = viewModel(factory =
   factory)
22
23     NavHost(
24         navController = navController,
25         startDestination = "club_list"
26     ) {
27         composable("club_list") {
28             ClubListScreen(navController =
   navController, clubVM = clubVM)
29         }
30
31         composable("club_detail") {
32             if (clubVM.selectedClub != null) {
33                 ClubDetailScreen(navController =
   navController, clubVM = clubVM)
34             } else {
35                 Box(modifier = Modifier.fillMaxSize(),
   contentAlignment = Alignment.Center) {
36                     Text("No club selected")
37                 }
38             }
39         }
40     }
41 }
```

7. ClubViewModelFactory.kt

Tabel 7. Source Code ClubViewModelFactory.kt

```
 1 package com.example.treblewinner.utils
 2
 3 import androidx.lifecycle.ViewModel
 4 import androidx.lifecycle.ViewModelProvider
 5 import com.example.treblewinner.model.Club
 6 import com.example.treblewinner.screen.ClubViewModel
 7
 8 class ClubViewModelFactory(
 9     private val clubList: List<Club>
10 ) : ViewModelProvider.Factory {
11     @Suppress("UNCHECKED_CAST")
12     override fun <T: ViewModel> create(modelClass:
   Class<T>): T {
13         if
   (modelClass.isAssignableFrom(ClubViewModel::class.java)
   ){
14             return ClubViewModel(clubList) as T
15         }
16         throw IllegalArgumentException("Unknown
   ViewModel class")
17     }
18 }
```

8. ClubViewModel.kt

Tabel 8. Source Code ClubViewModel.kt

```
 1 package com.example.treblewinner.screen
 2
 3 import android.util.Log
 4 import androidx.compose.runtime.getValue
 5 import androidx.compose.runtime.mutableStateOf
 6 import androidx.compose.runtime.setValue
 7 import androidx.lifecycle.ViewModel
 8 import androidx.lifecycle.viewModelScope
 9 import com.example.treblewinner.model.Club
10 import kotlinx.coroutines.flow.MutableStateFlow
11 import kotlinx.coroutines.flow.StateFlow
12 import kotlinx.coroutines.flow.asStateFlow
13 import kotlinx.coroutines.launch
14
15 class ClubViewModel(private val clubList: List<Club>) :
   ViewModel() {
```

```
    private val _clubs = MutableStateFlow<List<Club>>(emptyList())
    val clubs: StateFlow<List<Club>> get() = _clubs.asStateFlow()

    var selectedClub by mutableStateOf<Club?>(null)
        private set

    fun selectClub(club: Club){
        selectedClub = club
        Log.i("ClubVM", "Successfully load ${selectedClub?.name} data")
    }

    private var hasLoaded = false

    fun loadData() {
        if (!hasLoaded) {
            Log.i("ClubVM", "First load data")
            viewModelScope.launch {
                _clubs.value = clubList
                hasLoaded = true
            }
        }
        else {
            Log.i("ClubVM", "Data already loaded!")
        }
    }
}
```

9. ClubListScreen.kt

Tabel 9. Source Code ClubListScreen.kt

```
package com.example.treblewinner.screen.clublist

import android.content.ActivityNotFoundException
import android.content.Intent
import android.util.Log
import android.widget.Toast
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
```

```
13 import androidx.compose.ui.Modifier
14 import androidx.compose.ui.platform.LocalContext
15 import androidx.compose.ui.tooling.preview.Preview
16 import androidx.compose.ui.unit.dp
17 import androidx.core.net.toUri
18 import androidx.navigation.NavController
19 import
   androidx.navigation.compose.rememberNavController
20 import
   com.bumptech.glide.integration.compose.ExperimentalGli
   deComposeApi
21 import
   com.bumptech.glide.integration.compose.GlideImage
22 import com.example.treblewinner.model.Club
23 import com.example.treblewinner.screen.ClubViewModel
24
25 @OptIn(ExperimentalMaterial3Api::class,
   ExperimentalGlideComposeApi::class)
26 @Composable
27 fun ClubListScreen(
28     navController: NavController,
29     clubVM: ClubViewModel
30 ) {
31     val context = LocalContext.current
32
33     LaunchedEffect(Unit) {
34         clubVM.loadData()
35         Log.d("ClubListScreen", "Club data loaded")
36     }
37
38     val clubs by
   clubVM.clubs.collectAsState(emptyList())
39
40     Scaffold(
41         topBar = {
42             TopAppBar(
43                 title = { Text(text = "Treble Winner
   Clubs List") }
44             )
45         }
46     ) { paddingValues ->
47         LazyColumn(
48             contentPadding = paddingValues,
49             verticalArrangement =
   Arrangement.spacedBy(4.dp)
50         ) {
51             items(clubs) { club ->
```

```
52                  ClubCard(
53                      club = club,
54                      onDetailClick = {
55                          Log.i("ClubListScreen", "User
    click the detail button for ${club.name}")
56                          clubVM.selectClub(club)
57
    navController.navigate("club_detail")
58                      },
59                      onVisitClick = {
60                          try {
61                              val intent =
    Intent(Intent.ACTION_VIEW, club.webUrl.toUri())
62
    context.startActivity(intent)
63                              Log.d("ClubListScreen",
    "User going to the web ${club.webUrl}")
64                          } catch (e:
    ActivityNotFoundException) {
65                              Log.e("ClubListScreen",
    "Browser not found", e)
66                              Toast.makeText(context,
    "No browser available", Toast.LENGTH_SHORT).show()
67                          }
68                      }
69                  )
70              }
71          }
72      }
73  }
74
75  @OptIn(ExperimentalGlideComposeApi::class)
76  @Composable
77  fun ClubCard(
78      club: Club,
79      onDetailClick: () -> Unit,
80      onVisitClick: () -> Unit
81  ) {
82      Card(
83          modifier = Modifier
84              .fillMaxWidth()
85              .padding(6.dp)
86      ) {
87          Row(
88              modifier = Modifier
89                  .fillMaxWidth()
90                  .padding(8.dp),
```

```
            verticalAlignment =
Alignment.CenterVertically
        ) {
            GlideImage(
                model = club.logoUrl,
                contentDescription = club.name,
                modifier = Modifier
                    .size(150.dp)
                    .aspectRatio(1f)
                    .padding(end = 8.dp)
            )

            Column(modifier = Modifier.weight(1f)) {
                Text(
                    text = club.name,
                    style =
MaterialTheme.typography.headlineSmall
                )
                Text(
                    text =
club.trebleYears.joinToString(", "),
                    style =
MaterialTheme.typography.bodyMedium
                )
                Spacer(modifier =
Modifier.height(8.dp))
                Row(horizontalArrangement =
Arrangement.spacedBy(8.dp)) {
                    Button(onClick = onDetailClick) {
                        Text("Details")
                    }
                    Button(onClick = onVisitClick) {
                        Text("Visit Site")
                    }
                }
            }
        }
    }
}

@Preview(showBackground = true, showSystemUi = true)
@Composable
fun ClubListPreview() {
    val dummyViewModel = ClubViewModel(
        clubList =
com.example.treblewinner.constants.ClubConstant.ALL
    )
```

```
130     ClubListScreen(
131         navController = rememberNavController(),
132         clubVM = dummyViewModel
133     )
134 }
135
```

10. ClubDetailsScreen.kt

Tabel 10. Source Code ClubDetailsScreen.kt

```
  1 package com.example.treblewinner.screen.clubdetails
  2
  3 import android.content.Context
  4 import android.content.res.Configuration
  5 import android.util.Log
  6 import androidx.compose.foundation.BorderStroke
  7 import androidx.compose.foundation.layout.*
  8 import androidx.compose.foundation.rememberScrollState
  9 import androidx.compose.foundation.verticalScroll
 10 import androidx.compose.material.icons.Icons
 11 import androidx.compose.material3.*
 12 import androidx.compose.runtime.Composable
 13 import androidx.compose.ui.Alignment
 14 import androidx.compose.ui.Modifier
 15 import androidx.compose.ui.layout.ContentScale
 16 import androidx.compose.ui.text.font.FontWeight
 17 import androidx.compose.ui.tooling.preview.Preview
 18 import androidx.compose.ui.unit.dp
 19 import androidx.navigation.NavController
 20 import
    com.bumptech.glide.integration.compose.ExperimentalGli
    deComposeApi
 21 import
    com.bumptech.glide.integration.compose.GlideImage
 22 import com.example.treblewinner.constants.ClubConstant
 23 import com.example.treblewinner.model.Club
 24 import com.example.treblewinner.screen.ClubViewModel
 25 import androidx.compose.foundation.layout.Arrangement
 26 import
    androidx.compose.foundation.shape.RoundedCornerShape
 27 import
    androidx.compose.material.icons.automirrored.filled.Ar
    rowBack
 28 import androidx.compose.ui.graphics.Color
 29 import androidx.compose.ui.text.style.TextAlign
 30 import androidx.compose.ui.platform.LocalContext
```

```
31 import androidx.compose.ui.unit.sp
32 import com.example.treblewinner.model.Competition
33
34 @Composable
35 fun ClubDetailScreen(
36     navController: NavController,
37     clubVM: ClubViewModel
38 ) {
39     val club = clubVM.selectedClub
40
41     if (club == null) {
42         Text("No club selected")
43         Log.w("ClubDetrailScreen", "No club is
   selected")
44     } else {
45         ClubDetailContent(club = club) {
46             navController.popBackStack()
47         }
48     }
49 }
50
51 fun Context.isDarkTheme(): Boolean {
52     return (resources.configuration.uiMode and
   Configuration.UI_MODE_NIGHT_MASK) ==
   Configuration.UI_MODE_NIGHT_YES
53 }
54
55 @OptIn(ExperimentalMaterial3Api::class,
   ExperimentalGlideComposeApi::class)
56 @Composable
57 fun ClubDetailContent(
58     club: Club,
59     onBack: () -> Unit = {}
60 ) {
61     Log.i("ClubDetrailScreen", "Club is loaded...")
62     Scaffold(
63         topBar = {
64             TopAppBar(
65                 title = { Text(text = club.name) },
66                 navigationIcon = {
67                     IconButton(onClick = {
68                         Log.i("ClubDetailScreen",
   "User click back button to the ClubListScreen")
69                         onBack()
70                     }) {
71
```

```
                            Icon(imageVector =
Icons.AutoMirrored.Filled.ArrowBack,
contentDescription = "Back")
                        }
                    }
                )
            }
        ) { padding ->
            Column(
                modifier = Modifier
                    .verticalScroll(rememberScrollState())
                    .padding(16.dp)
                    .padding(padding)
            ) {
                GlideImage(
                    model = club.logoUrl,
                    contentDescription = "${club.name}
logo",
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(200.dp),
                    contentScale = ContentScale.Fit
                )
                Log.i("ClubDetailScreen", "Club logo is
loaded")

                Spacer(modifier = Modifier.height(24.dp))

                Text(
                    text = club.name,
                    style =
MaterialTheme.typography.headlineMedium,
                    fontWeight = FontWeight.Bold
                )
                Log.i("ClubDetailScreen", "Club name is
loaded")

                Spacer(modifier = Modifier.height(8.dp))

                Row(
                    horizontalArrangement =
Arrangement.spacedBy(8.dp),
                    verticalAlignment =
Alignment.CenterVertically
                ) {
                    Text(text = club.country, style =
MaterialTheme.typography.bodyLarge)
```

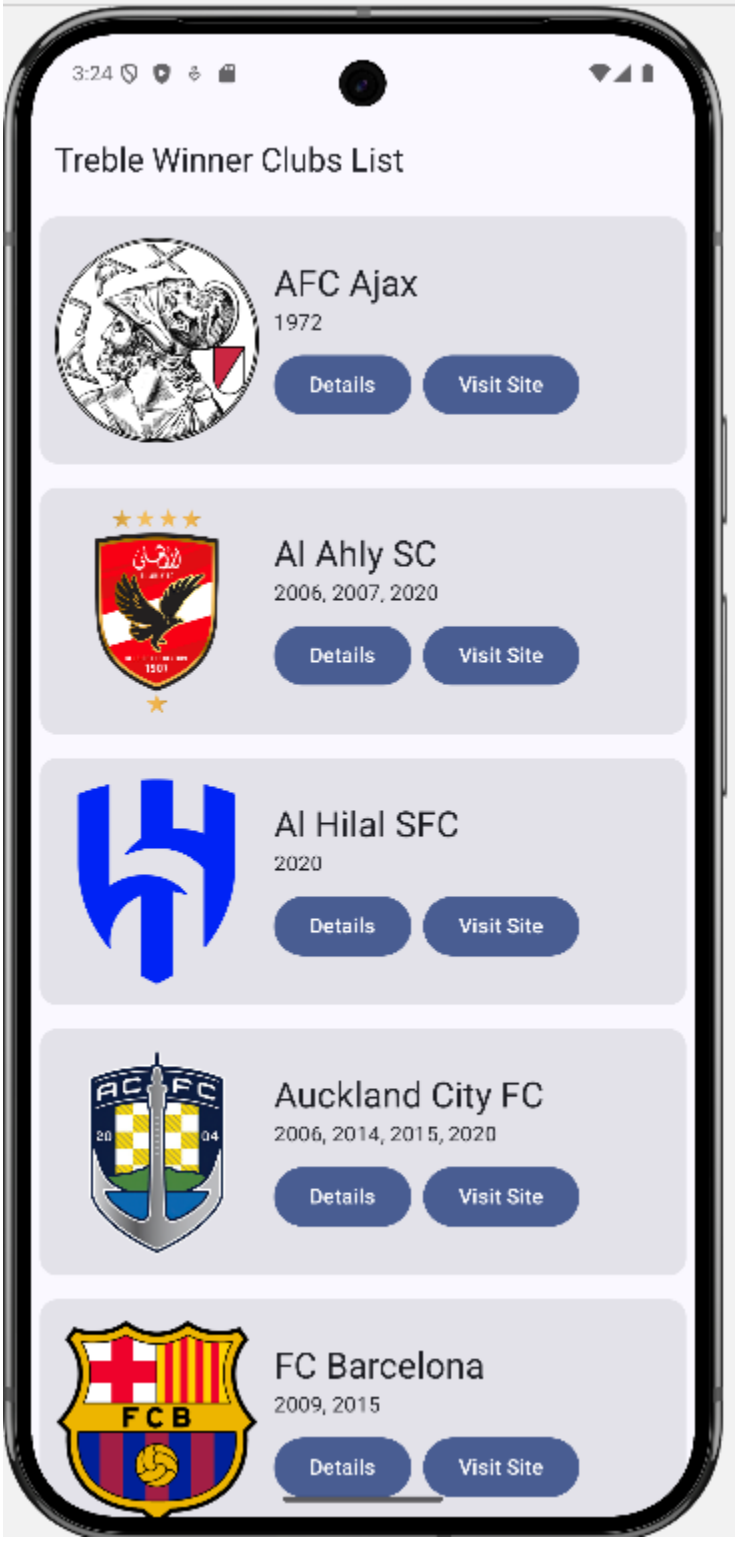
```
                Log.i("ClubDetailScreen", "Club country is loaded")
                Text(text = "•", style = MaterialTheme.typography.bodyLarge)
                Text(text = club.confederation, style = MaterialTheme.typography.bodyLarge)
                Log.i("ClubDetailScreen", "Club confederation is loaded")
            }

            Spacer(modifier = Modifier.height(16.dp))

            Text(
                text = "Treble Years:",
                style = MaterialTheme.typography.titleMedium,
                fontWeight = FontWeight.Bold
            )

            Spacer(modifier = Modifier.height(4.dp))

            Row(
                modifier = Modifier
                    .fillMaxWidth(),
                horizontalArrangement = Arrangement.spacedBy(16.dp, Alignment.CenterHorizontally)
            ){
                club.trebleYears.forEach { year ->
                    YearTagOutlined(year)
                }
            }


            Spacer(modifier = Modifier.height(16.dp))

            Text(
                text = "Competitions Won:",
                style = MaterialTheme.typography.titleMedium,
                fontWeight = FontWeight.Bold
            )
            Spacer(modifier = Modifier.height(4.dp))

            Row(
                modifier = Modifier.fillMaxWidth(),
```

```
                    horizontalArrangement =
148 Arrangement.spacedBy(16.dp,
149 Alignment.CenterHorizontally)
                ) {
150                 club.competitions.forEach {
    competition ->
151                     CompetitionCard(competition =
152 competition)
153                 }
154             }
155
156
157             Spacer(modifier = Modifier.height(16.dp))
158
159             Text(
                    text = "About:",
160                 style =
161 MaterialTheme.typography.titleMedium,
162                 fontWeight = FontWeight.Bold
163             )
164             Spacer(modifier = Modifier.height(4.dp))
165             Text(
166                 text = club.description,
                    textAlign = TextAlign.Justify,
167                 style =
168 MaterialTheme.typography.bodyMedium
                )
169             Log.i("ClubDetailScreen", "Club
170 description is loaded")
171         }
172     }
173 }
174
175 @Composable
176 fun YearTagOutlined(year: String) {
177     val tagColor = Color(0xFFD2B571)
178     Surface(
179         shape = RoundedCornerShape(50),
180         border = BorderStroke(2.dp, tagColor),
181         color = Color.Transparent,
182         tonalElevation = 1.dp // Optional
183     ) {
184         Text(
                text = year,
                style =
185 MaterialTheme.typography.titleMedium,
```

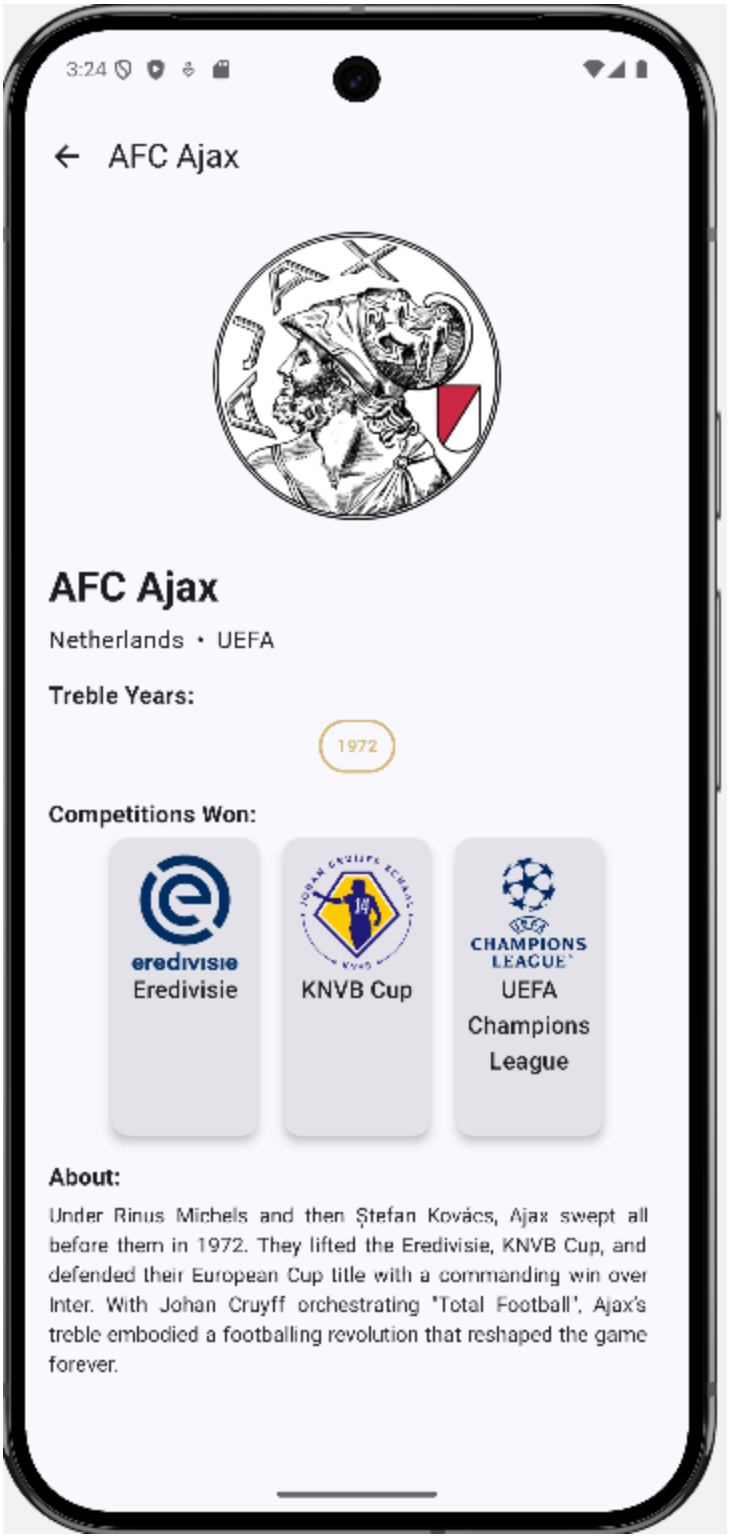
```
            modifier = Modifier.padding(horizontal =
12.dp, vertical = 6.dp),
            fontSize = 12.sp,
            color = tagColor
        )
    }
    Log.i("ClubDetailScreen", "Club treble year is
loaded")
}

@OptIn(ExperimentalGlideComposeApi::class)
@Composable
fun CompetitionCard(competition: Competition) {
    val context = LocalContext.current
    val logo = if (context.isDarkTheme() &&
competition.logoUrlDark.isNotBlank()) {
        Log.i("ClubDetailScreen", "Competition
${competition.name} is loaded with dark logo version")
        competition.logoUrlDark
    } else {
        Log.i("ClubDetailScreen", "Competition
${competition.name} is loaded with normal version")
        competition.logoUrl
    }

    Card(
        modifier = Modifier
            .width(100.dp)
            .height(200.dp),
        shape = MaterialTheme.shapes.medium,
        elevation =
CardDefaults.cardElevation(defaultElevation = 4.dp)
    ) {
        Column(
            modifier = Modifier.fillMaxSize(),
            verticalArrangement = Arrangement.Top,
            horizontalAlignment =
Alignment.CenterHorizontally
        ) {
            Spacer(modifier = Modifier.height(10.dp))

            GlideImage(
                model = logo,
                contentDescription = competition.name,
                modifier = Modifier.size(80.dp),
                contentScale = ContentScale.Fit
            )
```

```
227
228             Text(
                    text = competition.name,
229                 style =
230 MaterialTheme.typography.titleMedium,
231                 textAlign = TextAlign.Center
                )
232             Log.i("ClubDetailScreen", "Competition
233 name is loaded")
234         }
235     }
236 }
237
238
239 @Preview(showBackground = true)
240 @Composable
241 fun ClubDetailContentPreview() {
242     val mockClub = ClubConstant.BARCELONA
    ClubDetailContent(club = mockClub)
}
```

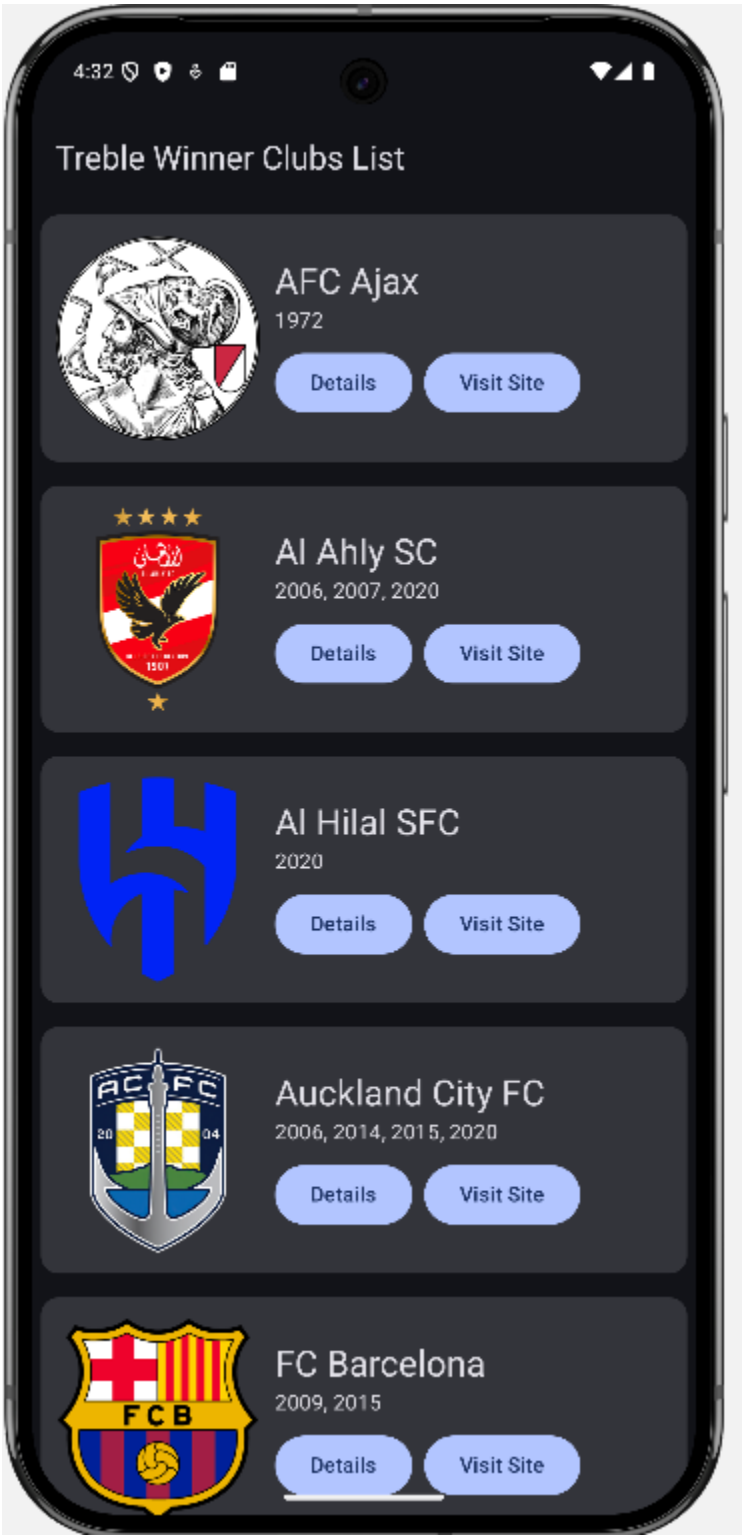
B. Output Program



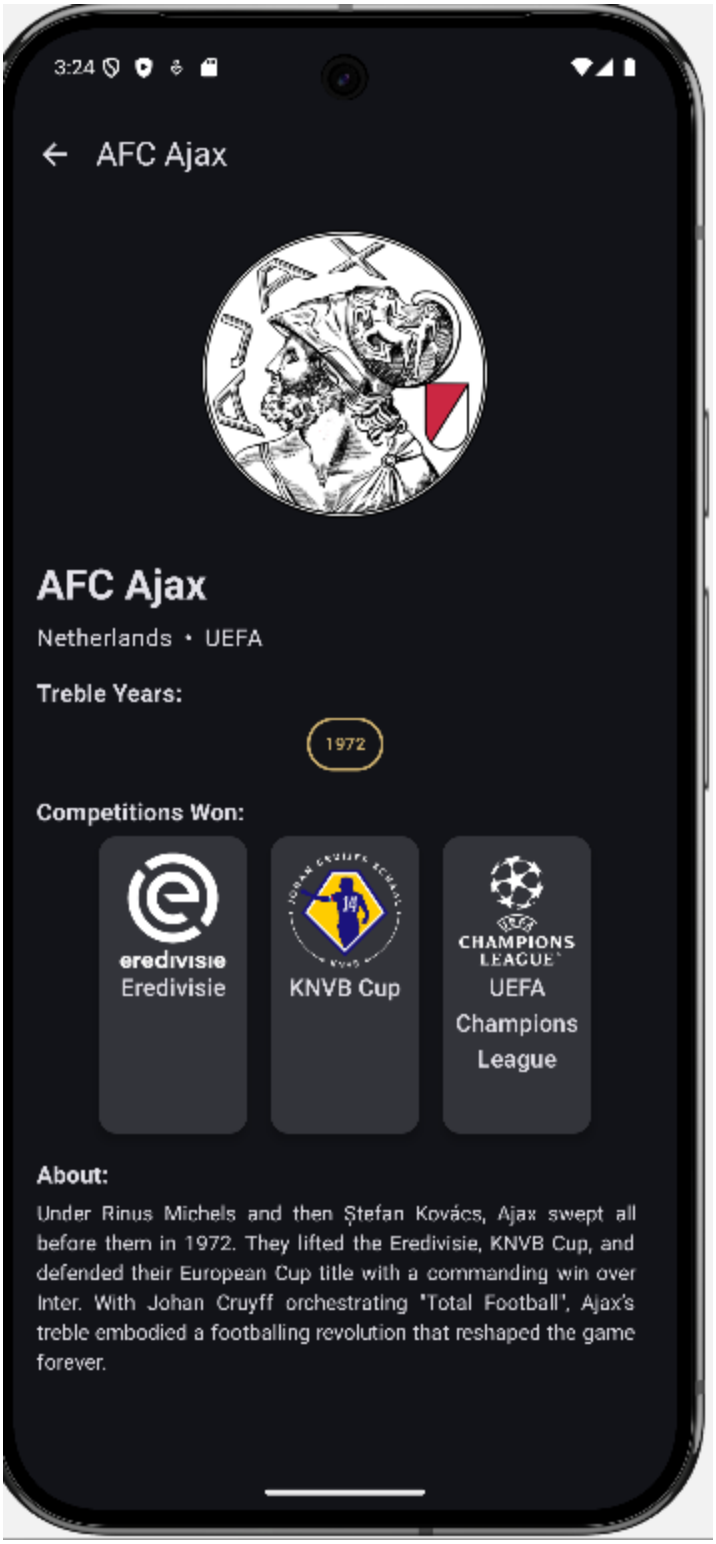
Gambar 1. Tampilan Halaman Utama dengan Light Mode



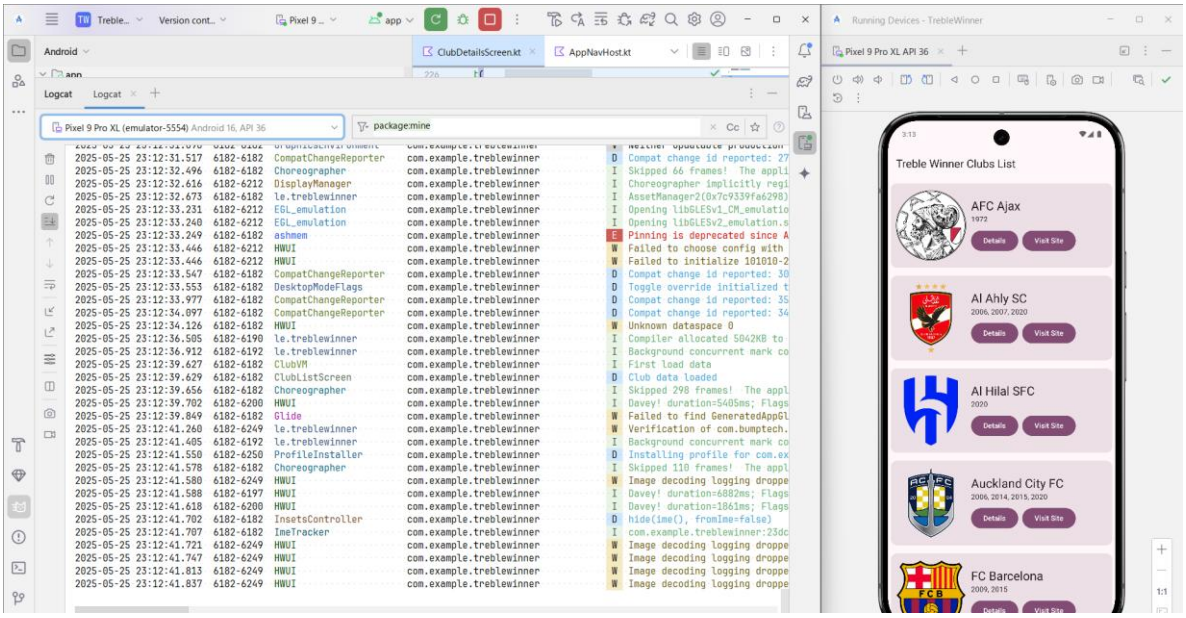
Gambar 2. Tampilan Halaman Detail dengan Light Mode



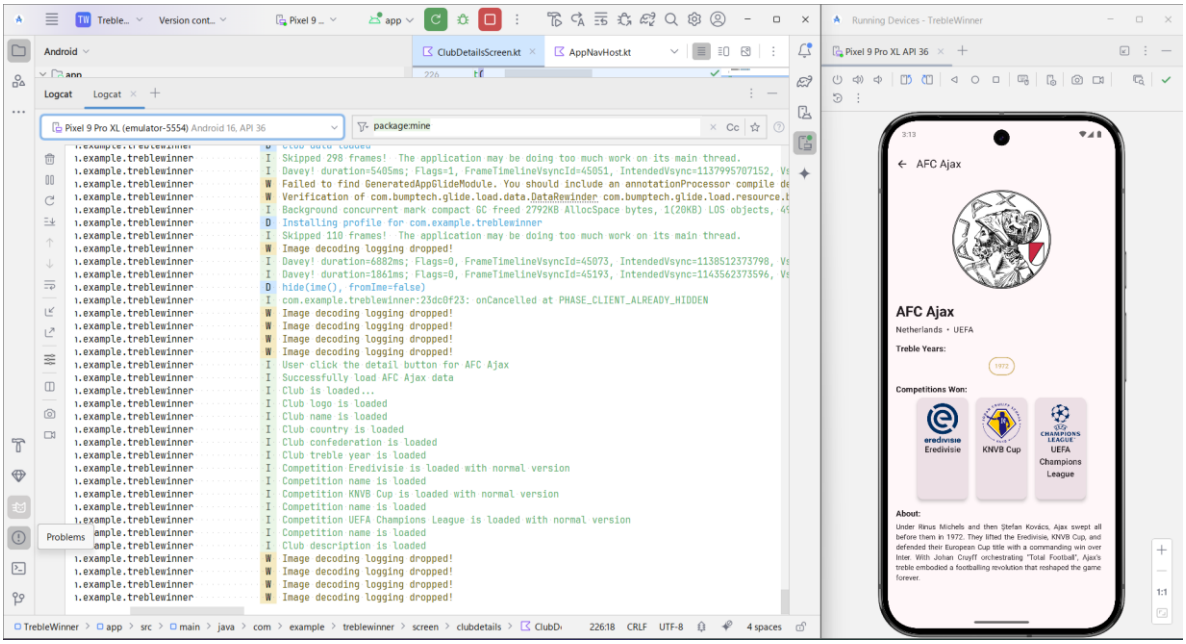
Gambar 3. Tampilan Halaman Utama dengan Dark Mode



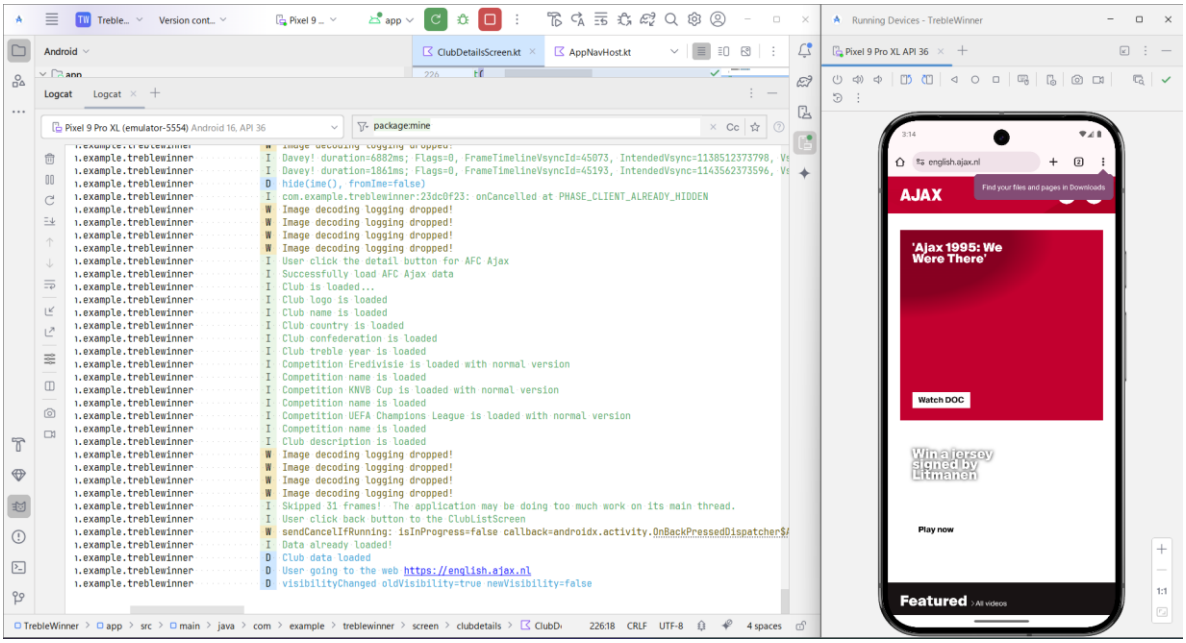
Gambar 4. Tampilan Halaman Detail dengan Dark Mode



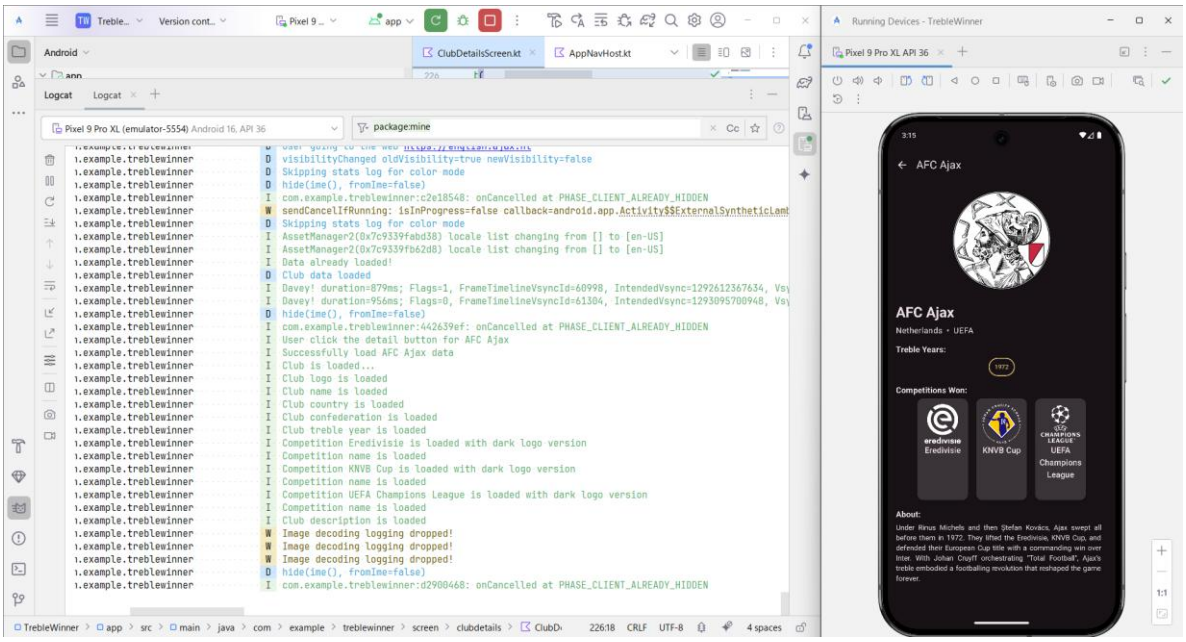
Gambar 5. Tampilan LogCat pada Android Studio saat Aplikasi Dijalankan



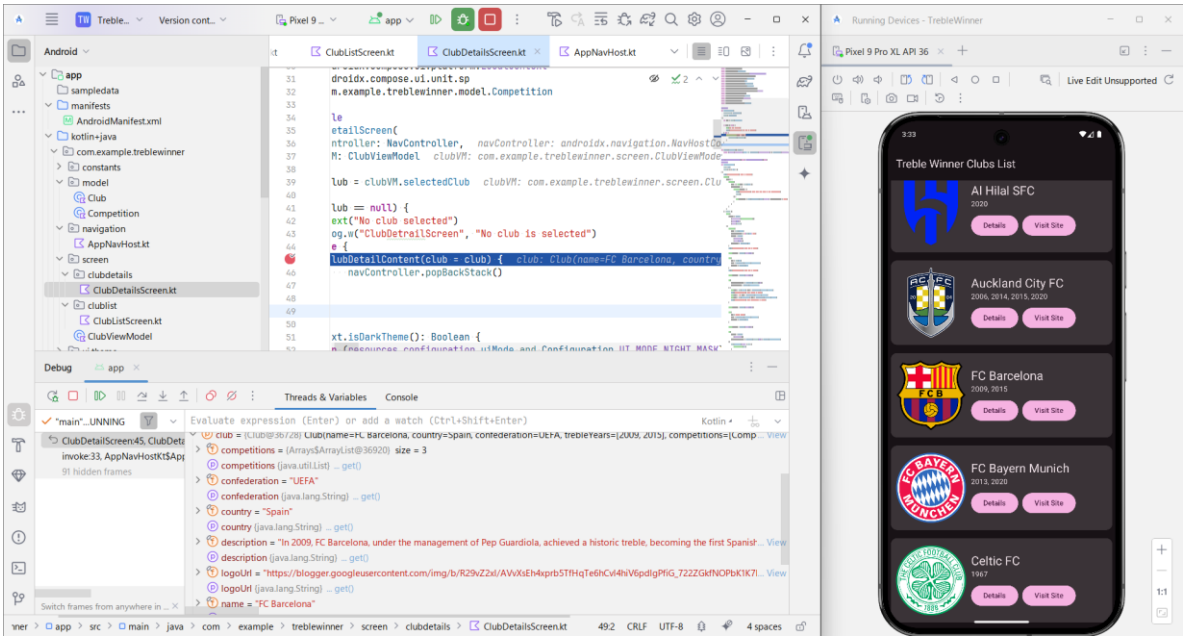
Gambar 6. Tampilan LogCat saat Membuka Halaman Detail



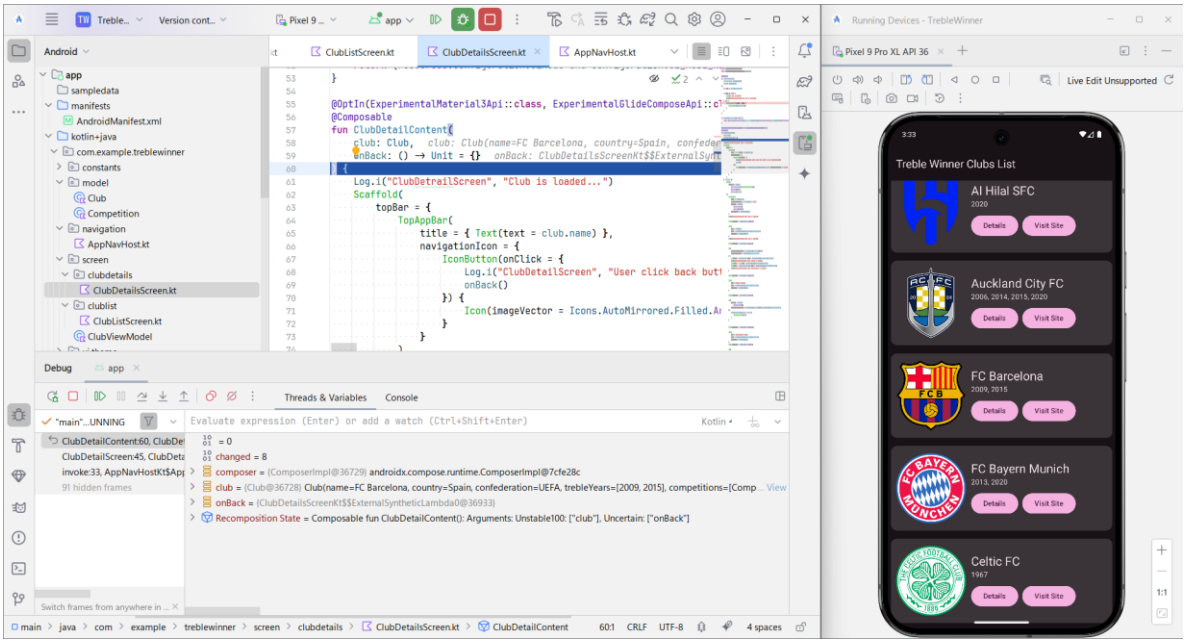
Gambar 7. Tampilan LogCat saat Membuka Klik Website Resmi Klub



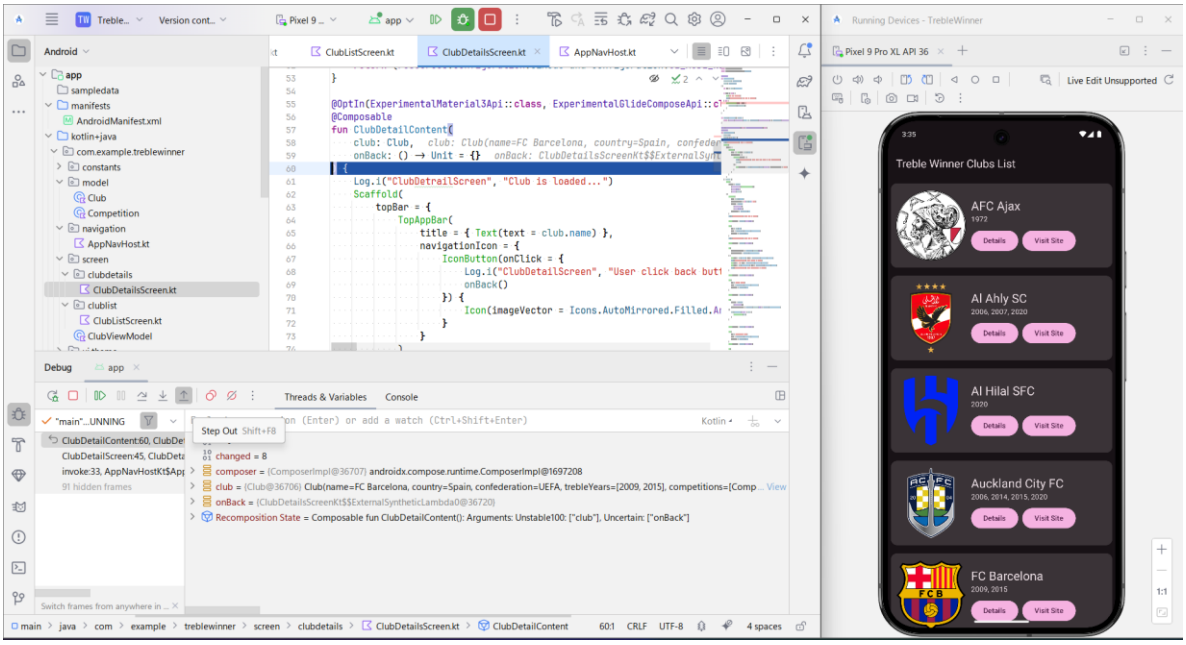
Gambar 8. Tampilan LogCat saat Memuat Logo saat Dark Mode



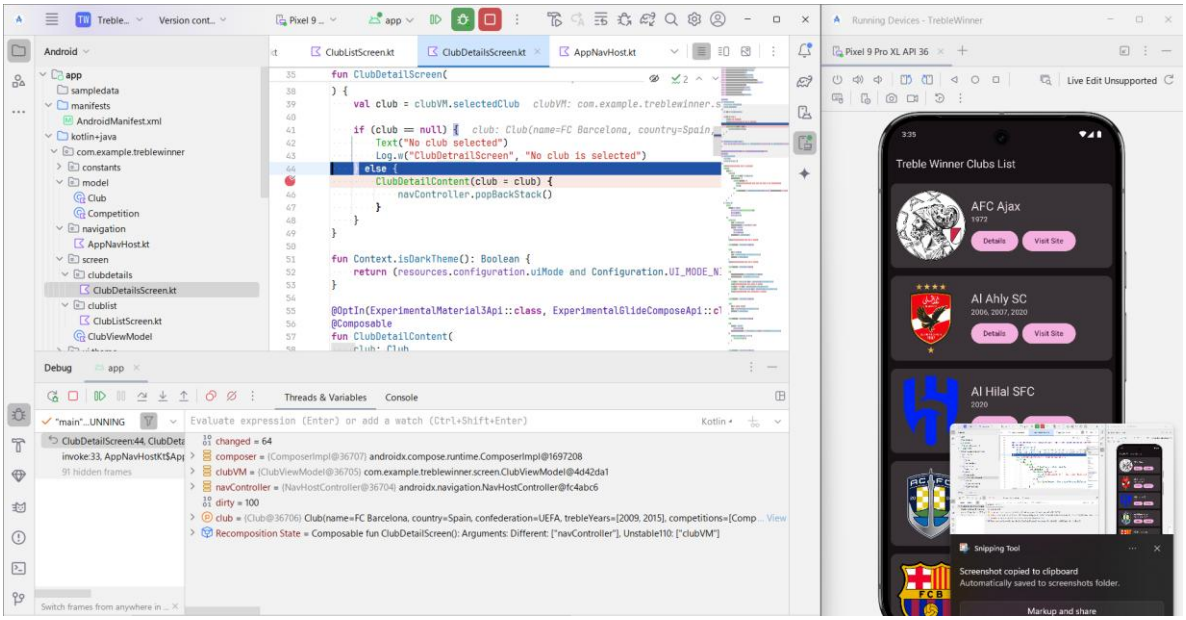
Gambar 9. Contoh Debugging saat Menavigasi ke Halaman Detail Klub



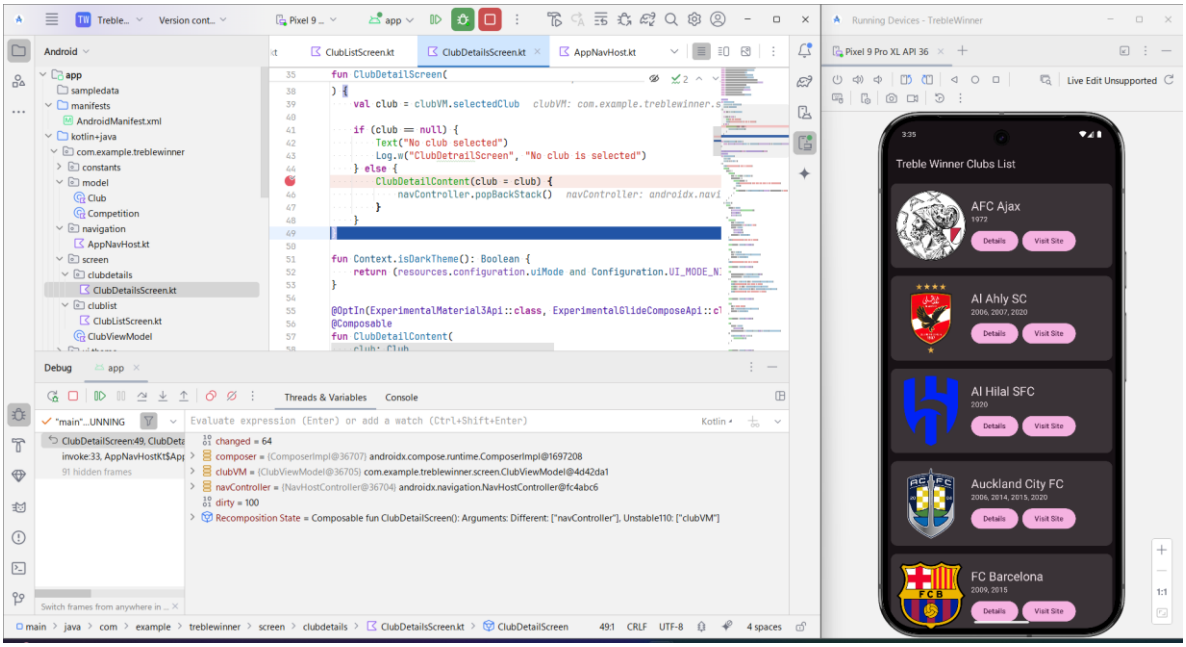
Gambar 10. Tampilan Debugging setelah Step Into



Gambar 11. Tampilan Debugger saat Ingin Step Out



Gambar 12. Tampilan Debugger setelah Step Out



Gambar 13. Tampilan Debugger saat Step Over

C. Pembahasan

Saya membuat scrollable list yang berisikan klub-klub sepakbola yang pernah Treble Continental. Treble Continental adalah pencapaian yang sangat langka dalam dunia sepakbola, di mana sebuah klub berhasil memenangkan tiga gelar utama dalam satu musim: liga domestik, piala domestik utama, dan kompetisi antarklub benua.

Logo-logo klub tersebut saya dapatkan melalui situs komunitas modder game PES yang menyediakan logo-logo kluub dengan kualitas tinggi. Beberapa pengecualian karena terdapat beberapa klub atau kompetisi yang kurang popular sehingga perlu menyelam ke DeepWeb demi mendapatkan aset grafis yang layak. Beberepa logo kompetisi memiliki versi resmi khusus untuk mode gelap (dark mode), dan saya pun menambahkan atribut tersebut dalam model agar tampil optimal dalam berbagai tema aplikasi. Meskipun demikian, klub-klub yang berhasil meraih Treble kebetulan tidak memiliki versi logo khusus untuk dark mode, sehingga tidak ditambahkan atribut untuk kasus logo mode gelap.

1. **MainActivity.kt:**

Pertama, mendeklarasikan nama package `com.example.treblewinner`, lalu mengimpor berbagai library yang dibutuhkan, termasuk untuk komponen Activity, Jetpack Compose, Navigation, dan Glide Compose. Kelas `MainActivity` meng-extend `ComponentActivity`, yang merupakan turunan dari Activity.

Di dalam metode `onCreate()`, fungsi `enableEdgeToEdge()` dipanggil terlebih dahulu untuk mengaktifkan tampilan layar penuh hingga ke tepi layar, termasuk area status bar dan navigasi. Kemudian, UI aplikasi diatur melalui `setContent`. Di dalamnya, seluruh konten UI dibungkus oleh `TrebleWinnerTheme`.

Selanjutnya, sebuah instance `NavController` dibuat menggunakan `rememberNavController()` yang berfungsi untuk mengelola navigasi antar layar (screen) dalam Compose. Objek ini kemudian diteruskan ke fungsi `AppNavHost`, sebuah composable khusus yang mendefinisikan struktur navigasi aplikasi, termasuk daftar screen yang tersedia dan bagaimana perpindahan antar screen dilakukan. Anotasi `@OptIn(ExperimentalGlideComposeApi::class)` ditambahkan untuk mengizinkan penggunaan API eksperimental dari Glide Compose (karena Glide masih beta di Compose), yang digunakan untuk menampilkan gambar logo klub dan kompetisi sepakbola dalam daftar yang ditampilkan aplikasi.

2. **Club.kt**

Pertama, mendeklarasikan nama package `com.example.treblewinner.model`, lalu mengimpor library yang diperlukan seperti `Parcelable` dari Android dan anotasi `@Parcelize` dari Kotlin untuk mempermudah proses pengiriman (passing) data antar komponen Android. Di dalam file ini, dibuat sebuah `data class` bernama `Club` yang merepresentasikan informasi lengkap mengenai klub sepakbola yang pernah meraih Treble Continental. Anotasi `@Parcelize` digunakan agar objek `Club` dapat diubah menjadi `parcel` dan ditransfer antar `Activity` atau `Composable` dengan mudah tanpa harus menuliskan implementasi `Parcelable` secara manual.

Kelas `Club` memiliki beberapa properti penting yang mendukung penyajian data secara menyeluruh. Properti `name` menyimpan nama klub, `country` menunjukkan asal negara klub, dan `confederation` menjelaskan konfederasi sepakbola tempat klub tersebut bernaung seperti UEFA (Eropa) atau CONMEBOL (Amerika Selatan). Informasi tahun-tahun di mana klub meraih Treble Continental disimpan dalam bentuk list string pada properti `trebleYears`. Selain itu, properti

`competitions` menyimpan daftar kompetisi yang dimenangkan klub dalam musim treble, yang ditulis dalam bentuk list dari objek `Competition`. Properti `logoUrl` yang menunjuk ke gambar logo klub, serta `webUrl` sebagai tautan ke situs resmi klub. Terakhir, properti `description` digunakan untuk menyimpan penjelasan singkat atau latar belakang dari klub tersebut.

3. Competition.kt

Pertama, mendeklarasikan nama package `com.example.treblewinner.model`, kemudian mengimpor library yang dibutuhkan, yaitu `Parcelable` dari Android dan anotasi `@Parcelize` dari Kotlin Extensions. Di dalam file ini didefinisikan sebuah `data class` bernama `Competition` yang digunakan untuk merepresentasikan informasi dari kompetisi sepakbola yang menjadi bagian dari Treble Continental yang diraih oleh sebuah klub. Dengan menggunakan anotasi `@Parcelize`, objek dari kelas ini dapat secara otomatis dikonversi menjadi parcel, sehingga memudahkan proses pengiriman data antar komponen dalam aplikasi Android, seperti saat berpindah dari satu layar ke layar lainnya.

Kelas `Competition` terdiri dari beberapa properti yang menyimpan data penting tentang kompetisi tersebut. Properti `name` menyimpan nama kompetisi, `country` menyimpan nama negara penyelenggara atau lokasi, dan `confederation` menunjukkan konfederasi sepakbola yang mengatur kompetisi tersebut. Properti `logoUrl` dan `logoUrlDark` masing-masing menyimpan URL untuk logo kompetisi dalam versi standar dan versi gelap, yang bisa digunakan sesuai tema terang atau gelap yang sedang aktif di aplikasi.

4. ClubConstant.kt

Untuk file ini, jumlah semua objek sebenarnya sangatlah banyak dan memakan hampir 500 baris. Maka diambilah salah satu contoh representasi dari objek untuk mempermudah penulisan.

Pertama, mendeklarasikan nama package `com.example.treblewinner.constants`, lalu mengimpor kelas `Club` dari `package model`. Di dalam file ini dibuat sebuah object bernama `ClubConstant`, yang berfungsi sebagai tempat penyimpanan data konstan atau data statis terkait klub sepakbola. Salah satu entri penting di dalamnya adalah `BARCELONA`, yaitu sebuah instance dari kelas `Club` yang merepresentasikan klub FC Barcelona beserta informasi lengkap terkait raihan Treble Continental mereka.

Objek `BARCELONA` menyimpan berbagai properti yang menjelaskan pencapaian klub. Nama klub diisi dengan "FC Barcelona", negara asalnya adalah Spanyol, dan konfederasinya adalah UEFA. Daftar tahun ketika mereka meraih Treble Continental adalah 2009 dan 2015. Informasi tentang kompetisi yang dimenangkan pada musim-musim tersebut direpresentasikan dalam bentuk list berisi konstanta dari `CompetitionsConstant`, yakni `LA_LIGA`, `COPA_DEL_REY`, dan `UEFA_CHAMPIONS_LEAGUE`. Properti `logoUrl` berisi tautan ke gambar logo

klub, sementara `webUrl` menyimpan alamat situs resmi FC Barcelona. `description`, berisi narasi historis lengkap tentang dua musim luar biasa ketika Barcelona meraih treble. Deskripsi ini menjelaskan bagaimana klub mencapai keberhasilan mereka pada tahun 2009 di bawah Pep Guardiola dan tahun 2015 di bawah Luis Enrique, termasuk rincian pertandingan final, gaya bermain, serta pemain-pemain kunci yang berkontribusi pada kejayaan mereka.

5. CompetitionsConstant.kt

Untuk file ini, jumlah semua objek sebenarnya sangatlah banyak dan memakan hampir 500 baris. Maka diambilah salah satu contoh representasi dari objek untuk mempermudah penulisan.

Pertama, mendeklarasikan nama package `com.example.treblewinner.constants`, kemudian mengimpor kelas `Competition` dari package `model`. File ini berfungsi sebagai tempat penyimpanan data konstan terkait kompetisi sepakbola yang menjadi bagian dari Treble Continental. Di dalamnya didefinisikan sebuah `object` bernama `CompetitionsConstant`, yang menyimpan data kompetisi dalam bentuk properti `immutable` agar dapat digunakan secara global di berbagai bagian aplikasi.

Salah satu data yang dideklarasikan dalam objek ini adalah `UEFA_CHAMPIONS_LEAGUE`, yaitu sebuah instance dari kelas `Competition` yang merepresentasikan Liga Champions UEFA, kompetisi antarklub paling prestisius di Eropa. Properti `name` diisi dengan "UEFA Champions League", sementara `country` menunjukkan wilayah cakupannya yaitu Eropa, dan `confederation` adalah UEFA sebagai badan pengatur kompetisi. Untuk mendukung tampilan visual yang sesuai dengan tema aplikasi, disediakan dua URL logo, yakni `logoUrl` untuk tampilan normal dan `logoUrlDark` untuk tampilan mode gelap (dark mode).

6. AppNavHost.kt

Pertama, mendeklarasikan nama package `com.example.treblewinner.navigation`, lalu mengimpor berbagai komponen dari Jetpack Compose dan Android Navigation untuk mendukung navigasi antar tampilan dalam aplikasi. Di dalam file ini terdapat sebuah fungsi `@Composable` bernama `AppNavHost`, yang berfungsi sebagai pusat navigasi (navigation host) aplikasi. Fungsi ini menerima parameter `navController` bertipe `NavHostController`, yang digunakan untuk mengatur perpindahan antar halaman UI (screen).

Di dalam `AppNavHost`, sebuah instance dari `ClubViewModel` diinisialisasi menggunakan fungsi `viewModel()` dengan menyuplai `ClubViewModelFactory`, yang menyertakan data awal daftar klub `(ClubConstant.ALL)`. Navigasi didefinisikan melalui `NavHost`, dengan

`startDestination` diatur ke `"club_list"` sebagai layar pertama yang ditampilkan saat aplikasi dijalankan. Kemudian, dua rute composable didefinisikan: `"club_list"` dan `"club_detail"`.

Rute `"club_list"` akan menampilkan tampilan `ClubListScreen`, yaitu layar utama berisi daftar klub yang pernah meraih Treble Continental, di mana `navController` dan `viewModel` diteruskan sebagai parameter. Sementara itu, rute `"club_detail"` digunakan untuk menampilkan detail klub. Sebelum menampilkan `ClubDetailScreen`, dilakukan pengecekan apakah properti `selectedClub` di dalam `clubVM` sudah tidak `null`, artinya pengguna telah memilih klub dari daftar. Jika belum, maka akan ditampilkan teks "No club selected" yang diposisikan di tengah layar menggunakan `Box` dengan `Modifier.fillMaxSize()` dan `contentAlignment = Alignment.Center`.

7. ClubViewModelFactory.kt

File ini merupakan implementasi dari `ViewModelProvider.Factory` yang digunakan untuk membuat instance `ClubViewModel` secara manual dengan parameter tertentu. Di dalam metode `create`, kelas ini memeriksa apakah `modelClass` yang diminta adalah turunan dari `ClubViewModel`. Jika iya, maka `ClubViewModel` akan dibuat dengan menyuplai `clubList`, dan hasilnya dikembalikan setelah dikonversi secara aman ke tipe generik `T` menggunakan cast. Jika `modelClass` bukan `ClubViewModel`, maka akan dilemparkan `IllegalArgumentException` yang menandakan bahwa kelas ViewModel yang diminta tidak dikenal.

8. ClubViewModel.kt

File ini mendefinisikan `ClubViewModel`, sebuah `class` yang meng-extend `ViewModel`, yang merupakan bagian dari arsitektur MVVM pada Android dan digunakan untuk menyimpan serta mengelola data UI yang bersifat tahan terhadap perubahan siklus hidup. Pertama, file ini mendeklarasikan package `com.example.treblewinner.screen`, lalu mengimpor berbagai komponen Compose dan coroutine yang diperlukan untuk manajemen state dan data asynchronous.

Di dalam `ClubViewModel`, terdapat properti `private _clubs` yang bertipe `MutableStateFlow<List<Club>>`, digunakan untuk menyimpan daftar klub secara reaktif. Properti `public clubs` mengekspos versi read-only dari `_clubs` menggunakan `asStateFlow()`, sehingga hanya `ViewModel` yang dapat memodifikasi nilainya, sementara UI hanya dapat mengobservasinya.

Selain itu, terdapat properti `selectedClub` yang bertipe `mutableStateOf<Club?>`, digunakan untuk menyimpan klub yang sedang dipilih oleh pengguna. Properti ini dapat berubah dan otomatis memicu

recomposition pada composable yang mengamatinya. Metode `selectClub(club: Club)` disediakan agar UI dapat menetapkan klub yang dipilih secara eksplisit, namun tidak dapat mengubah langsung nilai `selectedClub` dari luar `ViewModel` karena properti ini memiliki `private set`. Metode ini juga mencatat log menggunakan `Log.i` untuk menandai bahwa pemilihan klub berhasil dilakukan, termasuk mencetak nama klub yang dipilih, agar memudahkan proses debugging.

Selain pemilihan klub, `ClubViewModel` juga memiliki mekanisme pemuatan data satu kali melalui fungsi `loadData()`. Fungsi ini akan memuat daftar klub (`clubList`) ke dalam `_clubs` hanya jika data belum pernah dimuat sebelumnya, yang dilacak menggunakan variabel `hasLoaded`. Jika belum dimuat, `loadData()` akan menjalankan `coroutine` di dalam `viewModelScope` agar tidak memblokir thread UI saat menetapkan `_clubs.value`. Jika `loadData()` dipanggil kembali setelah data pernah dimuat, maka log akan mencetak bahwa data sudah dimuat sebelumnya dan tidak akan dilakukan pemuatan ulang.

9. ClubListScreen.kt

File ini mendefinisikan tampilan utama dari daftar klub sepakbola yang pernah meraih Treble Continental, yaitu `ClubListScreen`. Fungsi ini merupakan composable yang mengambil `ClubViewModel` dan `NavController` sebagai parameter. `ClubViewModel` berperan menyediakan data daftar klub melalui `StateFlow`, yang kemudian dikonversi ke dalam bentuk state Compose menggunakan `collectAsState`. Data ini bersifat reaktif sehingga UI akan diperbarui secara otomatis saat data berubah.

Di awal fungsi, terdapat blok `LaunchedEffect(Unit)` yang digunakan untuk mengeksekusi `clubVM.loadData()` saat `ClubListScreen` pertama kali dikomposisi. `LaunchedEffect` ini dijalankan dalam scope `coroutine` milik Compose dan disarankan. Pemanggilan `Log.d` ditambahkan untuk membantu proses debugging dengan pesan yang menunjukkan bahwa data telah dimuat dari ViewModel.

Selanjutnya, data klub diamati menggunakan `val clubs by clubVM.clubs.collectAsState(emptyList())`. Di sini, `clubs` adalah hasil observasi dari `StateFlow` milik ViewModel yang dikonversi menjadi `State` Compose. Dengan menggunakan `collectAsState`, tampilan akan otomatis melakukan recomposition setiap kali isi daftar klub berubah. Parameter `emptyList()` digunakan sebagai nilai awal sebelum data benar-benar dimuat.

Struktur UI dibangun menggunakan `Scaffold`, dengan `TopAppBar` yang menampilkan judul aplikasi. Konten utama berupa `LazyColumn`, yang digunakan untuk menampilkan daftar klub secara efisien dalam bentuk scrollable list. Setiap klub ditampilkan dalam sebuah `Card` yang di dalamnya memuat gambar logo klub (melalui `GlideImage` dari Glide Compose), nama klub, dan daftar tahun saat klub

meraih treble. Layout `card` dibagi menjadi dua kolom utama, yaitu logo di kiri dan teks beserta tombol di kanan.

Dua tombol disediakan untuk setiap klub: tombol "Details" dan "Visit Site". Tombol "Details" akan memanggil fungsi `selectClub()` pada `ViewModel` untuk menyimpan klub yang dipilih, kemudian melakukan navigasi ke halaman detail melalui `NavController`. Sedangkan tombol "Visit Site" akan membuka halaman web resmi klub menggunakan `Intent.ACTION_VIEW` beserta log-nya. Jika tidak ada browser yang tersedia, aplikasi akan menampilkan `Toast` untuk memberi tahu pengguna.

Di bagian bawah terdapat fungsi `ClubListPreview`, sebuah composable yang dianotasi dengan `@Preview` untuk menampilkan pratinjau tampilan langsung di Android Studio. Fungsi ini membuat instance sementara `ClubViewModel` dan `NavController`.

10. ClubDetailsScreen.kt

File `ClubDetailScreen.kt` merupakan bagian dari aplikasi untuk menampilkan halaman detail sebuah klub sepak bola yang meraih treble winner. Halaman ini bekerja dengan mengambil data dari `ClubViewModel`, khususnya properti `selectedClub` yang menyimpan klub yang dipilih oleh pengguna dari layar sebelumnya. Bila `selectedClub` bernilai `null`, tampilan hanya menampilkan teks sederhana “No club selected”, tetapi jika klub telah dipilih, maka komponen `ClubDetailContent` akan digunakan untuk menyusun antarmuka detail klub tersebut secara lengkap dan informatif.

Fungsi `ClubDetailContent` dibuat menggunakan struktur `Scaffold` yang menyediakan `TopAppBar` dengan judul berupa nama klub dan ikon panah kembali yang memungkinkan pengguna kembali ke layar sebelumnya. Di dalam tubuh `Scaffold`, terdapat kolom utama yang dapat discroll secara vertikal, berisi elemen-elemen visual dan teks yang disusun secara proporsional dan estetis. Logo klub ditampilkan di bagian atas dalam ukuran besar menggunakan `GlideImage`, diikuti dengan nama klub dalam ukuran font besar dan gaya bold. Informasi asal negara dan konfederasi klub ditampilkan dalam satu baris menggunakan `Row`, diikuti oleh daftar tahun-tahun di mana klub tersebut berhasil meraih treble. Tahun-tahun ini ditampilkan dalam bentuk tag kecil berbentuk oval yang memiliki border berwarna emas, dibuat melalui fungsi composable `YearTagOutlined`.

Selanjutnya, aplikasi menampilkan bagian kompetisi yang dimenangkan oleh klub tersebut pada musim treble melalui fungsi `CompetitionCard()`. Masing-masing kompetisi divisualisasikan dalam kartu vertikal kecil dengan logo kompetisi dan nama kompetisinya. Logo yang digunakan dapat berubah sesuai dengan mode terang atau gelap dari sistem, yang ditentukan menggunakan fungsi ekstensi `Context.isDarkTheme()`. Setelah bagian kompetisi, terdapat juga paragraf deskripsi klub yang ditampilkan dalam gaya teks justify, memberikan latar belakang atau cerita singkat mengenai sejarah klub tersebut. Seluruh desain didasarkan pada prinsip-prinsip Material 3 dengan penggunaan komponen modern seperti `Surface`, `Card`, dan `Text` dengan pengaturan padding dan jarak antar elemen yang konsisten.

Selain itu, terdapat juga fungsi pratinjau `ClubDetailContentPreview` yang memungkinkan pengembang melihat tampilan layar ini di Android Studio menggunakan data dari `ClubConstant.BARCELONA`.

Terakhir, terdapat juga penambahan log menggunakan `Log.i` saat data berhasil dimuat dan kemabli ke halaman awal, serta `Log.w` jika terjadi eror.

Tambahan: Debugging

Pada Gambar 9, dapat dilihat bahwa breakpoint ditempatkan pada baris ke-45 di file `ClubDetailScreen.kt`. Di editor, muncul teks berwarna abu-abu yang merupakan inline hints atau teks khayal yang menunjukkan isi dari variabel-variabel saat program dijalankan dalam mode debugging. Teks ini muncul ketika tombol Details ditekan pada kartu tiap klub di `ClubListScreen.kt`. Pada panel Debugger di Thread & Variables, terlihat bahwa variabel `club` telah terisi dengan data klub yang dipilih, yaitu FC Barcelona.

Pada Gambar 10, ketika tombol Step Into ditekan, debugger berpindah ke isi dari fungsi `ClubDetailConten`t, yaitu fungsi yang dipanggil tepat di bawah baris tempat breakpoint berada. Tombol Step Into berfungsi untuk masuk ke dalam fungsi yang sedang dipanggil, sehingga bisa melacak proses eksekusi secara lebih detail di dalam fungsi tersebut.

Selanjutnya, pada Gambar 11 dan Gambar 12, ketika tombol Step Out ditekan, debugger keluar dari fungsi `ClubDetailContent` dan kembali ke baris setelah pemanggilan fungsi tersebut. Dengan kata lain, Step Out berfungsi untuk menyelesaikan eksekusi fungsi yang sedang ditelusuri lalu kembali ke fungsi pemanggilnya.

Terakhir, pada Gambar 13, ketika tombol Step Over ditekan, debugger tidak masuk ke dalam fungsi yang dipanggil, melainkan langsung melanjutkan ke baris berikutnya, sambil tetap melakukan evaluasi fungsi secara internal. Tombol Step Over berguna ketika kita ingin melompati fungsi tanpa menelusuri isinya secara mendalam, namun tetap ingin melihat hasil akhirnya pada baris berikutnya.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/MinamotoYuki46/MeineStudienArbeit/tree/main/MobileDevelopment/Codex-Practicus/Modul%204/TrebleWinner

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

A. Pembahasan

Dalam arsitektur aplikasi Android, Application class merupakan titik awal global yang mewakili keseluruhan siklus hidup aplikasi. Kelas merupakan turunan dari kelas `android.app.Application` dan hanya dibuat sekali saat proses aplikasi pertama kali diluncurkan. Application berbeda dari Activity atau Service, karena tidak terkait langsung dengan antarmuka pengguna atau komponen spesifik, melainkan menyediakan konteks global dan manajemen status aplikasi.

Fungsinya adalah sebagai

1. Inisialisasi global, menjalankan setup satu kali saat aplikasi dimulai.
2. Memanajemen state global, menyimpan data atau status yang dibuthkan di selurh bagain aplikasi.
3. Merespons perubahan konfigurasi tingkat aplikasi, menangani perubahan seperti bahasa atau oritentasi secara global.
4. Merespons situasi memori rendah, mengosongkan cache atau membebaskan sumber daya saat sistem kekurangan memori.
5. Mengakses konteks aplikasi, memberikan akses ke Context yang tidak bergantung kepada Activity.
6. Lifecycle awal, menyediakan titik masuk sebelum Activity atau Service dibuat.