LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

BUILD A SCROLLABLE LIST

Oleh:

Adrian Bintang Saputera NIM. 2310817110006

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

LEMBAR PENGESAHAN

LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 3

Laporan Praktikum Pemrograman Mobile Modul 3: Build a Scrollable List ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Prakitkum ini dikerjakan oleh:

Nama Praktikan : Adrian Bintang Saputera
NIM : 2310817110006

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Raka Azwar NIM. 2210817210012	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 1993070320190301011

DAFTAR ISI

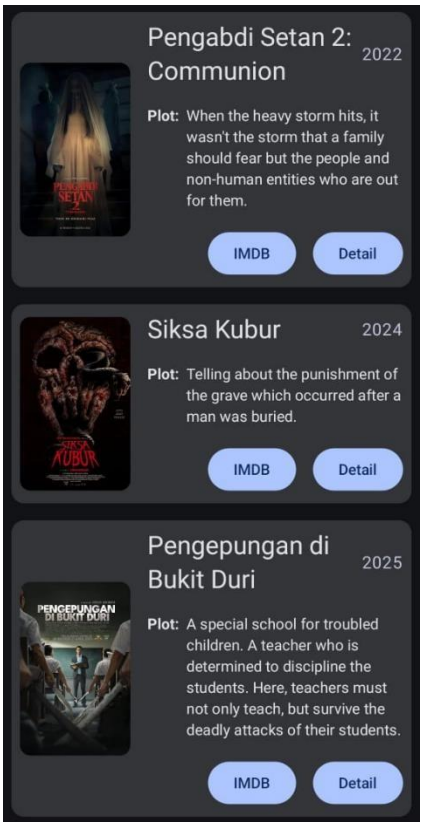
DAFTAR GAMBAR

DAFTAR TABEL

SOAL

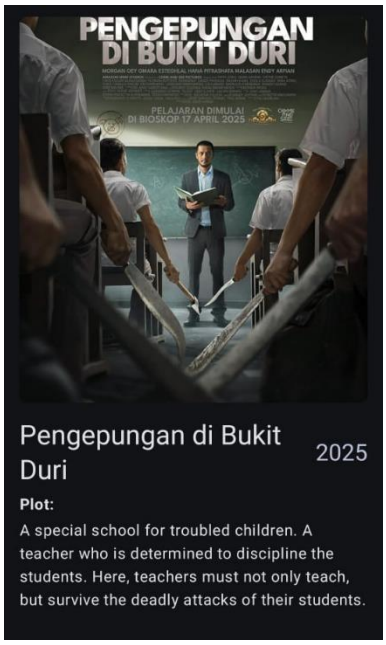
1. Buatlah sebuah aplikasi Android menggunakan XML dan Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:
    1. List menggunakan fungsi RecyclerView (XML) dan LazyColumn (Compose)
    2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas 3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah 4. Terdapat 2 button dalam list, dengan fungsi berikut:
        a Button pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain
        b Button kedua menggunakan Navigation component untuk membuka laman detail item
    3. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
    4. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang
    5. Aplikasi menggunakan arsitektur *single activity* (satu activity memiliki beberapa fragment)
    6. Aplikasi berbasis XML harus menggunakan ViewBinding
2. Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

UI item list harus berisi 1 gambar, 2 button (intent eksplisit dan navigasi), dan 2 baris teks dan setiap baris memiliki 2 teks yang berbeda. Diusahakan agar desain UI item list menyerupai UI berikut:



Gambar 1. Contoh UI List

Desain UI laman detail bebas, tetapi diusahakan untuk mengikuti kaidah desain Material Design dan data item ditampilkan penuh di laman detail seperti contoh berikut:



Gambar 2. Contoh UI Detail

2. RecyclerView masih digunakan secara luas meskipun memiliki kode yang lebih panjang dan bersifat boiler-plate dibandingkan LazyColumn, karena beberapa alasan penting. Pertama, banyak aplikasi Android yang masih dibangun menggunakan sistem UI berbasis View, sehingga RecyclerView tetap menjadi pilihan utama untuk menampilkan daftar data secara efisien. Migrasi ke Jetpack Compose dan penggunaan LazyColumn membutuhkan perubahan arsitektur yang cukup besar, yang tidak selalu memungkinkan dalam proyek yang sudah berjalan. Selain itu, RecyclerView memberikan kontrol yang sangat fleksibel terhadap tampilan item, animasi, dan tata letak kustom melalui penggunaan LayoutManager, yang belum sepenuhnya dapat disamai oleh LazyColumn. Oleh karena itu, meskipun LazyColumn lebih modern dan praktis dalam penulisan kode, RecyclerView tetap relevan dalam konteks kompatibilitas dan fleksibilitas.

A. Source Code

XML:

MainActivity.kt

Table 1. Source Code Jawaban Soal 1 XML

1 2	`package com.example.listxml`

```
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val fragmentManager = supportFragmentManager
        val homeFragment = HomeFragment()
        val fragment =
fragmentManager.findFragmentByTag(HomeFragment::class
.java.simpleName)
        if (fragment !is HomeFragment) {
            fragmentManager
                .beginTransaction()
                .add(R.id.frame_container,
homeFragment, HomeFragment::class.java.simpleName)
                .commit()
        }
    }
}
```

HomeFragment.kt

Table 2. Source Code Jawaban Soal 1 XML

```
package com.example.listxml

import android.content.Intent
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import
androidx.recyclerview.widget.LinearLayoutManager
import
com.example.listxml.databinding.FragmentHomeBinding
import android.net.Uri

class HomeFragment : Fragment() {
    private var _binding: FragmentHomeBinding? =
null
    private val binding get() = _binding!!
```

```
    private lateinit var agentAdapter: AgentAdapter
    private val list = ArrayList<Agent>()

    override fun onCreateView(
        inflater: LayoutInflater, container:
ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding =
FragmentHomeBinding.inflate(inflater, container,
false)

        list.clear()
        list.addAll(getListAgent())
        setupRecyclerView()
        return binding.root
    }

    private fun setupRecyclerView() {
        agentAdapter = AgentAdapter(
            list,
            object : OnAgentClickListener {
                override fun onDetailClicked(agent:
Agent) {
                    val detailFragment =
DetailFragment().apply {
                        arguments = Bundle().apply {
                            putString("EXTRA_NAME",
agent.name)
                            putString("EXTRA_DESC",
agent.desc)
                            putString("EXTRA_ROLE",
agent.role)
                            putInt("EXTRA_IMAGE",
agent.image)
                        }
                    }

parentFragmentManager.beginTransaction()

.replace(R.id.frame_container, detailFragment)
                        .addToBackStack(null)
                        .commit()
                }

```

```
                override fun onLinkClicked(linkUrl:
String) {
                    val intent =
Intent(Intent.ACTION_VIEW, Uri.parse(linkUrl))
                    startActivity(intent)
                }
            },
        )
        binding.rvAgent.apply {
            layoutManager =
LinearLayoutManager(context)
            adapter = agentAdapter
        }
    }

    private fun getListAgent(): ArrayList<Agent> {
        val dataName =
resources.getStringArray(R.array.data_agentName)
        val dataDesc =
resources.getStringArray(R.array.data_agentDesc)
        val dataRole =
resources.getStringArray(R.array.data_agentRole)
        val dataImage =
resources.obtainTypedArray(R.array.data_agentImage)
        val dataDetail =
resources.getStringArray(R.array.data_agentLink)
        val listAgent = ArrayList<Agent>()
        for (i in dataName.indices) {
            val agent = Agent(
                dataName[i],
                dataRole[i],
                dataDesc[i],
                dataImage.getResourceId(i, -1),
                dataDetail[i]
            )
            listAgent.add(agent)
        }
        dataImage.recycle()
        return listAgent
    }
}
```

DetailFragment.kt

Table 3. Source Code Jawaban Soal 1 XML

```
1  package com.example.listxml
2
3  import android.os.Bundle
4  import androidx.fragment.app.Fragment
5  import android.view.LayoutInflater
6  import android.view.View
7  import android.view.ViewGroup
8  import
9  com.example.listxml.databinding.FragmentDetailBinding
10
11 class DetailFragment : Fragment() {
12
13     private var _binding: FragmentDetailBinding? =
14 null
15     private val binding get() = _binding!!
16
17     private var name: String? = null
18     private var role: String? = null
19     private var desc: String? = null
20     private var image: Int = -1
21
22     override fun onCreate(savedInstanceState:
23 Bundle?) {
24         super.onCreate(savedInstanceState)
25
26         arguments?.let {
27             name = it.getString("EXTRA_NAME")
28             desc = it.getString("EXTRA_DESC")
29             role = it.getString("EXTRA_ROLE")
30             image = it.getInt("EXTRA_IMAGE")
31         }
32     }
33
34     override fun onCreateView(
35         inflater: LayoutInflater, container:
36 ViewGroup?,
37         savedInstanceState: Bundle?
38     ): View {
39         _binding =
40 FragmentDetailBinding.inflate(inflater, container,
41 false)
42         return binding.root
43     }
```

```
    override fun onViewCreated(view: View,
savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        binding.detailName.text = name
        binding.detailRole.text = role
        binding.detailDesc.text = desc
        binding.detailImage.setImageResource(image)
    }

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

AgentAdapter.kt

Table 4. Source Code Jawaban Soal 1 XML

```
package com.example.listxml

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import
com.example.listxml.databinding.ItemAgentBinding

interface OnAgentClickListener {
    fun onDetailClicked(agent: Agent)
    fun onLinkClicked(linkUrl: String)
}

class AgentAdapter(
    private val agentList: ArrayList<Agent>,
    private val listener: OnAgentClickListener
) :
RecyclerView.Adapter<AgentAdapter.AgentViewHolder>()
{

    inner class AgentViewHolder(val binding:
ItemAgentBinding) :
        RecyclerView.ViewHolder(binding.root)

    override fun onCreateViewHolder(parent:
ViewGroup, viewType: Int): AgentViewHolder {
```

```
        val binding =
ItemAgentBinding.inflate(LayoutInflater.from
(parent.context), parent, false)
        return AgentViewHolder(binding)
    }

    override fun onBindViewHolder(holder:
AgentViewHolder, position: Int) {
        val agent = agentList[position]
        holder.binding.apply {
            agentName.text = agent.name
            agentRole.text = agent.role
            agentDesc.text = agent.desc
            agentImage.setImageResource(agent.image)
            btnLink.setOnClickListener {
                listener.onLinkClicked(agent.detail)
            }
            btnDetail.setOnClickListener {
                listener.onDetailClicked(agent)
            }
        }
    }

    override fun getItemCount(): Int {
        return agentList.size
    }
}
```

Agent.kt

Table 5. Source Code Jawaban Soal 1 XML

```
package com.example.listxml

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

@Parcelize
data class Agent (
    val name : String,
    val role : String,
    val desc : String,
    val image : Int,
    val detail : String = "",
): Parcelable
```

13	

activity_main.xml

Table 6. Source Code Jawaban Soal 1 XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <FrameLayout
   xmlns:android="http://schemas.android.com/apk/res/and
   roid"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      tools:context=".MainActivity"
       android:id="@+id/frame_container">
7  </FrameLayout>
8
```

fragment_detail.xml

Table 7. Source Code Jawaban Soal 1 XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3  xmlns:android="http://schemas.android.com/apk/res/
4  android"
5      xmlns:app="http://schemas.android.com/apk/res-
6  auto"
7      xmlns:tools="http://schemas.android.com/tools"
8      android:layout_width="match_parent"
9      android:layout_height="match_parent"
10     android:padding="8dp"
11     tools:context=".DetailFragment">
12
13     <android.widget.ScrollView
14         android:layout_width="match_parent"
15         android:layout_height="match_parent">
16
17
18 <androidx.constraintlayout.widget.ConstraintLayout
19             android:layout_width="match_parent"
20             android:layout_height="wrap_content"
21             android:padding="8dp">
22
```

```
            <ImageView
                android:id="@+id/detail_image"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:scaleType="centerCrop"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent"
                tools:src="@tools:sample/avatars" />

            <TextView
                android:id="@+id/detail_name"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="16dp"
                android:textSize="20sp"
                android:textStyle="bold"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/detail_image"
                tools:text="Brimstone" />

            <TextView
                android:id="@+id/detail_role"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="#666"
                android:textSize="16sp"
                android:textStyle="italic"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/detail_name"

app:layout_constraintVertical_bias="0.5"
                tools:text="Controller" />

```

```
67
68              <TextView
69                  android:id="@+id/detail_desc"
70                  android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="8dp"
                    android:textSize="14sp"

    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintHorizontal_bias="0.0"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toBottomOf="@id/detail_role"
                    tools:text="Joining from the U.S.A..."
    />


    </androidx.constraintlayout.widget.ConstraintLayout>

        </android.widget.ScrollView>
    </androidx.constraintlayout.widget.ConstraintLayout>
```

fragment_home.xml

Table 8. Source Code Jawaban Soal 1 XML

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout
3   xmlns:android="http://schemas.android.com/apk/res/
4   android"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       tools:context=".HomeFragment">
9
10      <!-- TODO: Update blank fragment layout -->
11      <androidx.recyclerview.widget.RecyclerView
12          android:id="@+id/rvAgent"
13          android:layout_width="match_parent"
14          android:layout_height="match_parent"
15          android:text="@string/hello_blank_fragment"
    />
    </FrameLayout>
```

item_agent.xml

Table 9. Source Code Jawaban Soal 1 XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.cardview.widget.CardView
3  android:layout_width="match_parent"
4      android:id="@+id/card_view"
5      android:layout_height="wrap_content"
6      xmlns:tools="http://schemas.android.com/tools"
7      android:layout_margin="8dp"
8      app:cardCornerRadius="22dp"
9      app:cardElevation="4dp"
10     android:layout_gravity="center"
11     xmlns:app="http://schemas.android.com/apk/res-
12 auto"
13
14 xmlns:android="http://schemas.android.com/apk/res/
15 android">
16
17
18 <androidx.constraintlayout.widget.ConstraintLayout
19         android:layout_width="match_parent"
20         android:layout_height="wrap_content"
21         android:padding="8dp">
22
23         <ImageView
24             android:id="@+id/agent_image"
25             android:layout_width="150dp"
26             android:layout_height="200dp"
27             android:scaleType="centerCrop"
28
29 app:layout_constraintBottom_toBottomOf="parent"
30
31 app:layout_constraintEnd_toEndOf="parent"
32
33 app:layout_constraintHorizontal_bias="0.0"
34
35 app:layout_constraintStart_toStartOf="parent"
36
37 app:layout_constraintTop_toTopOf="parent"
38             app:layout_constraintVertical_bias="0.0"
39             tools:src="@tools:sample/avatars" />
40
41         <TextView
42             android:id="@+id/agent_name"
```

```
            android:layout_width="120dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="12dp"
            android:textSize="13sp"
            android:textStyle="bold"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="0.0"

app:layout_constraintStart_toEndOf="@+id/agent_image
"

app:layout_constraintTop_toTopOf="parent"

app:layout_constraintVertical_bias="0.082"
            tools:text="judul film" />

        <TextView
            android:id="@+id/agent_role"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="13sp"
            android:textStyle="bold"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintTop_toTopOf="@+id/agent_name"

app:layout_constraintBottom_toTopOf="@+id/agent_desc
"

app:layout_constraintStart_toEndOf="@+id/agent_name"
            tools:ignore="MissingConstraints"
            tools:text="year" />

        <TextView
            android:id="@+id/agent_desc"
            android:layout_marginStart="10dp"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:textSize="10sp"

app:layout_constraintBottom_toTopOf="@+id/btn_link"

```

```
app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="0.266"

app:layout_constraintStart_toEndOf="@+id/agent_image
"

app:layout_constraintTop_toBottomOf="@+id/agent_name
"
            app:layout_constraintVertical_bias="0.2"
            tools:text="hab" />

        <Button
            android:id="@+id/btn_link"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/btn_link"
            android:textSize="10sp"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toStartOf="@+id/btn_detail"

app:layout_constraintHorizontal_bias="0.609"

app:layout_constraintStart_toEndOf="@+id/agent_image
"

app:layout_constraintTop_toTopOf="parent"

app:layout_constraintVertical_bias="0.947" />

        <Button
            android:id="@+id/btn_detail"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="1.0"

app:layout_constraintStart_toEndOf="@+id/agent_image
"
```

```
    app:layout_constraintTop_toTopOf="parent"

    app:layout_constraintVertical_bias="0.947"
                android:textSize="10sp"
                android:text="@string/btn_detail" />


    </androidx.constraintlayout.widget.ConstraintLayout>
    </androidx.cardview.widget.CardView>
```

strings.xml

Table 10. Source Code Jawaban Soal 1 XML

```
1  <resources>
2      <string name="app_name">XML</string>
3      <string name="btn_detail">Detail</string>
4      <string name="btn_link">Link</string>
5
6      <string-array name="data_agentName">
7          <item>Brimstone</item>
8          <item>Phoenix</item>
9          <item>Chamber</item>
10         <item>Jett</item>
11         <item>Kay/o</item>
12         <item>Viper</item>
13     </string-array>
14
15     <string-array name="data_agentRole">
16         <item>Controller</item>
17         <item>Duelist</item>
18         <item>Sentinel</item>
19         <item>Duelist</item>
20         <item>Initiator</item>
21         <item>Controller</item>
22     </string-array>
23
24     <string-array name="data_agentDesc">
25         <item>Joining from the U.S.A., Brimstone`s
26 orbital arsenal ensures his squad always has the
27 advantage. His ability to deliver utility precisely
28 and safely make him the unmatched boots-on-the-
29 ground commander.</item>
30         <item>Representing her home country of South
   Korea, Jett`s agile and evasive fighting style lets
```

```
her take risks no one else can. She runs circles
around every skirmish, cutting enemies before they
even know what hit them.</item>
        <item>The American Chemist, Viper deploys an
array of poisonous chemical devices to control the
battlefield and choke the enemy`s vision. If the
toxins don`t kill her prey, her mindgames surely
will.</item>
        <item>Hailing from the U.K., Phoenix`s star
power shines through in his fighting style, igniting
the battlefield with flash and flare. Whether he`s
got backup or not, he`ll rush into a fight on his
own terms.</item>
        <item>KAY/O is a machine of war built for a
single purpose: neutralizing radiants. His power to
Suppress enemy abilities dismantles his opponents
capacity to fight back, securing him and his allies
the ultimate edge.</item>
        <item>Well-dressed and well-armed, French
weapons designer Chamber expels aggressors with
deadly precision. He leverages his custom arsenal to
hold the line and pick off enemies from afar, with a
contingency built for every plan.</item>
    </string-array>

    <string-array name="data_agentImage">
        <item>@drawable/brimstone</item>
        <item>@drawable/phoenix</item>
        <item>@drawable/chamber</item>
        <item>@drawable/jett</item>
        <item>@drawable/kayo</item>
        <item>@drawable/viper</item>
    </string-array>

    <string-array name="data_agentLink">
        <item>https://playvalorant.com/en-
us/agents/brimstone/</item>
        <item>https://playvalorant.com/en-
us/agents/phoenix/</item>
        <item>https://playvalorant.com/en-
us/agents/chamber/</item>
        <item>https://playvalorant.com/en-
us/agents/jett/</item>
        <item>https://playvalorant.com/en-
us/agents/kay-o/</item>
```

```
        <item>https://playvalorant.com/en-
us/agents/viper/</item>
    </string-array>

    <string name="hello_blank_fragment">Hello blank
fragment</string>
</resources>
```

Jetpack Compose:

MainActivity.kt

Table 11. Source Code Jawaban Soal 1 Jetpack Compose

```
1  package com.example.listcompose
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
6  import androidx.activity.enableEdgeToEdge
7  import androidx.compose.foundation.Image
8  import androidx.compose.foundation.layout.*
9  import androidx.compose.foundation.lazy.LazyColumn
10 import
11 androidx.compose.foundation.shape.RoundedCornerShape
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.platform.LocalContext
17 import androidx.compose.ui.res.painterResource
18 import androidx.compose.ui.text.font.FontWeight
19 import androidx.compose.ui.text.style.TextOverflow
20 import androidx.compose.ui.tooling.preview.Preview
21 import androidx.compose.ui.unit.dp
22 import androidx.compose.ui.unit.sp
23 import com.example.listcompose.ui.theme.ComposeTheme
24 import com.example.listcompose.agentList.agentList
25 import android.content.Intent
26 import android.net.Uri
27 import androidx.compose.foundation.rememberScrollState
28 import androidx.compose.foundation.verticalScroll
29 import androidx.core.net.toUri
30 import androidx.navigation.NavHostController
```

```
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import
androidx.navigation.compose.rememberNavController
import androidx.navigation.navArgument

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?)
{
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            ComposeTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color =
MaterialTheme.colorScheme.background
                ) {
                    val navController =
rememberNavController()
                    NavHost(
                        navController = navController,
                        startDestination = "agentList"
                    ) {
                        composable("agentList") {
                            AgentList(navController)
                        }
                        composable(

"deskripsi/{description}/{Image}",
                            arguments = listOf(

navArgument("description") { type = NavType.StringType
},
                                navArgument("Image") {
type = NavType.IntType }
                            )
                        ) { backStackEntry ->
                            val description =

backStackEntry.arguments?.getString("description") ?:
""
                            val image =
backStackEntry.arguments?.getInt("Image") ?: 0
```

```
75
76  DeskripsiScreen(description, image)
77                              }
78                          }
79                      }
80                  }
81              }
82          }
83
84          @Composable
85          fun AgentList(navController: NavHostController) {
86              LazyColumn(
87                  modifier = Modifier
88                      .fillMaxSize()
89
90  .padding(WindowInsets.statusBars.asPaddingValues())
91                      .padding(7.dp)
92              ) {
93                  items(agentList.size) { dataAgent ->
94                      val agent = agentList[dataAgent]
95                      AgentItem(
96                          name = agent.name,
97                          image = agent.image,
98                          url = agent.url,
99                          description = agent.description,
100                         navController = navController
101                     )
102                 }
103             }
104         }
105
106         @Composable
107         fun AgentItem(
108             name: String,
109             image: Int,
110             url: String,
111             description: String,
112             navController: NavHostController
113         ) {
114             val context = LocalContext.current
115             Card(
116                 modifier = Modifier
117                     .fillMaxSize()
118                     .padding(8.dp),
                    shape = RoundedCornerShape(16.dp),
```

```
            elevation =
CardDefaults.cardElevation(defaultElevation = 4.dp)
        ) {
            Row(
                modifier = Modifier
                    .padding(16.dp)
                    .fillMaxSize(),
                verticalAlignment =
Alignment.CenterVertically
            ) {
                Image(
                    painter = painterResource(id =
image),
                    contentDescription = null,
                    modifier = Modifier
                        .size(width = 100.dp, height =
120.dp)
                )
                Spacer(modifier =
Modifier.width(16.dp))
                Column(
                    modifier = Modifier
                        .weight(1f)
                ) {
                    Text(text = name, fontWeight =
FontWeight.Bold, fontSize = 20.sp)
                    Spacer(modifier =
Modifier.height(4.dp))
                    Text(
                        text = description,
                        fontSize = 14.sp,
                        maxLines = 3,
                        overflow =
TextOverflow.Ellipsis
                    )
                    Spacer(modifier =
Modifier.height(8.dp))
                    Row(
                        horizontalArrangement =
Arrangement.SpaceBetween,
                        modifier = Modifier
                            .fillMaxSize()

.wrapContentWidth(Alignment.Start),
                    ) {
```

```
                            Button(
                                onClick = {
                                    val intent =
Intent(Intent.ACTION_VIEW, url.toUri())

context.startActivity(intent)
                                },
                                contentPadding =
PaddingValues(horizontal = 16.dp, vertical = 8.dp),
                                shape =
RoundedCornerShape(50),
                                modifier =
Modifier.defaultMinSize(minWidth = 1.dp)
                            ) {
                                Text("Detail", fontSize =
13.sp)
                            }
                            Spacer(modifier =
Modifier.width(8.dp))
                            Button(
                                onClick = {
                                    val encodedDesc =
Uri.encode(description)

navController.navigate("deskripsi/$encodedDesc/$image")
                                },
                                contentPadding =
PaddingValues(horizontal = 16.dp, vertical = 8.dp),
                                shape =
RoundedCornerShape(50),
                                modifier =
Modifier.defaultMinSize(minWidth = 1.dp)
                            ) {
                                Text("Deskripsi", fontSize
= 13.sp)
                            }
                        }
                    }
                }
            }
        }
    }

    @Composable
    fun DeskripsiScreen(description: String, image:
Int) {
```

```
        Column(
            modifier = Modifier
                .verticalScroll(rememberScrollState())
                .fillMaxSize()

.padding(WindowInsets.statusBars.asPaddingValues())
                .padding(18.dp),
            horizontalAlignment =
Alignment.CenterHorizontally
        ) {
            Image(
                painter = painterResource(id = image),
                contentDescription = null,
                modifier = Modifier
                    .fillMaxWidth()
                    .aspectRatio(3f / 4f)
                    .height(500.dp)
            )
            Spacer(modifier = Modifier.height(16.dp))
            Text(
                text = description,
                fontSize = 16.sp
            )
        }
    }

    @Preview(showBackground = true)
    @Composable
    fun GreetingPreview() {
        ComposeTheme {
            Text("Preview List Hero")
        }
    }
}
```

dataList.kt

Table 12. Source Code Jawaban Soal 1 Jetpack Compose

```
1  package com.example.listcompose.valorantData
2
3  data class DataAgent (
4      val name : String,
5      val image : Int,
6      val url : String,
```

```
    val description : String
)
```

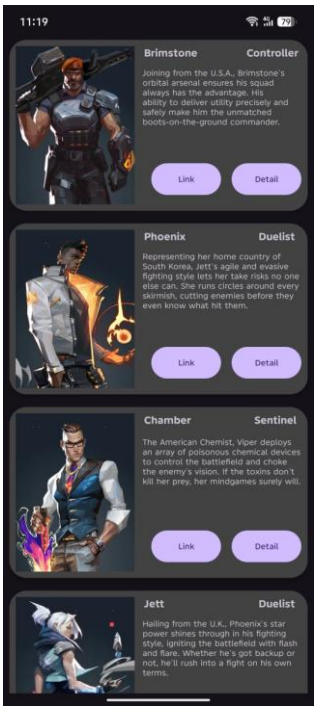
agentList.kt

Table 13. Source Code Jawaban Soal 1 Jetpack Compose

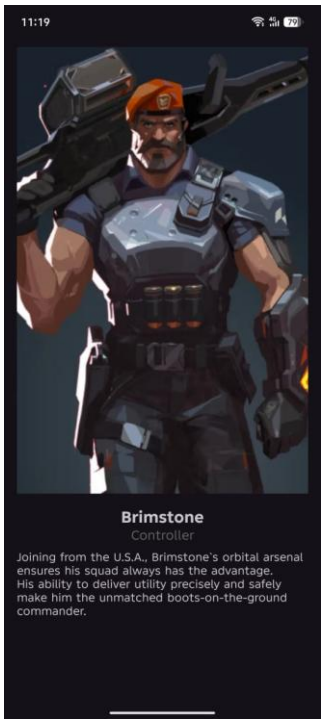
```
package com.example.listcompose.agentList


import com.example.listcompose.R
import
com.example.listcompose.valorantData.DataAgent

val agentList = listOf(
    DataAgent(
        name = "Jett",
        R.drawable.jett,
        url = "https://playvalorant.com/en-
us/agents/jett/",
        description = "Role: Duelist\n" +
                "Representing her home country of
South Korea, Jett's agile and evasive fighting style
lets her take risks no one else can. She runs
circles around every skirmish, cutting enemies
before they even know what hit them."
    ),
    DataAgent(
        name = "Brimstone",
        R.drawable.brimstone,
        url = "https://playvalorant.com/en-
us/agents/brimstone/",
        description = "Role: Controller\n" +
                "Joining from the U.S.A.,
Brimstone's orbital arsenal ensures his squad always
has the advantage. His ability to deliver utility
precisely and safely make him the unmatched boots-
on-the-ground commander."
    ),
    DataAgent(
        name = "Chamber",
        R.drawable.chamber,
        url = "https://playvalorant.com/en-
us/agents/chamber/",
        description = "Role: Sentinel\n" +
                "Well-dressed and well-armed, French
```

```
weapons designer Chamber expels aggressors with
deadly precision. He leverages his custom arsenal to
hold the line and pick off enemies from afar, with a
contingency built for every plan."
    ),
    DataAgent(
        name = "Viper",
        R.drawable.viper,
        url = "https://playvalorant.com/en-
us/agents/viper/",
        description = "Role: Controller\n" +
                "The American Chemist, Viper deploys
an array of poisonous chemical devices to control
the battlefield and choke the enemy's vision. If the
toxins don't kill her prey, her mindgames surely
will."
    ),
    DataAgent(
        name = "Phoenix",
        R.drawable.phoenix,
        url = "https://playvalorant.com/en-
us/agents/phoenix/",
        description = "Role: Duelist\n" +
                "Hailing from the U.K., Phoenix's
star power shines through in his fighting style,
igniting the battlefield with flash and flare.
Whether he's got backup or not, he'll rush into a
fight on his own terms."
    ),
    DataAgent(
        name = "Kay/o",
        R.drawable.kayo,
        url = "https://playvalorant.com/en-
us/agents/kay-o/",
        description = "Role: Initiator\n" +
                "KAY/O is a machine of war built for
a single purpose: neutralizing radiants. His power
to Suppress enemy abilities dismantles his
opponents' capacity to fight back, securing him and
his allies the ultimate edge."
    ),
)
```

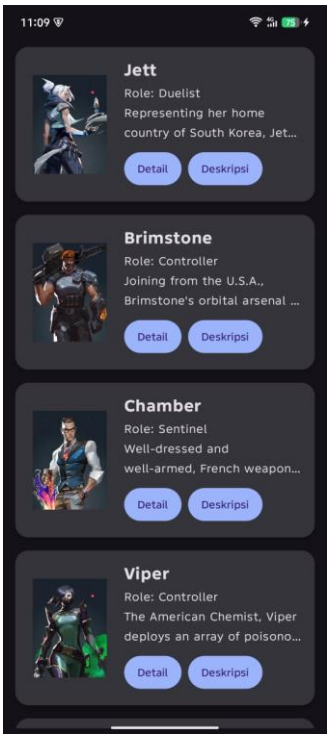
B. Output Program



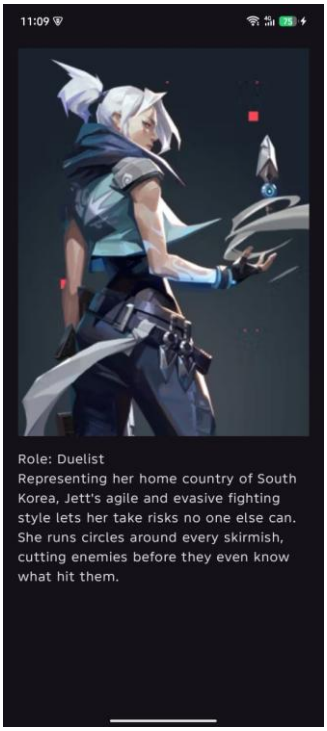
Gambar 3. Tampilan Awal Aplikasi XML



Gambar 4. Tampilan Detail List XML



Gambar 5. Tampilan Awal Aplikasi Jetpack Compose



Gambar 6. Tampilan Detail List XML

C. Pembahasan

XML:

MainActivity.kt:

Kode di atas merupakan implementasi dari sebuah aktivitas utama (MainActivity) dalam aplikasi Android yang menggunakan Fragment untuk menampilkan antarmuka pengguna. Pada metode onCreate(), pertama-tama aktivitas mengatur tampilan menggunakan setContentView() dengan layout activity_main. Selanjutnya, program mengambil FragmentManager yang digunakan untuk mengelola fragment-fragment dalam aktivitas ini. Kemudian, dibuat instance dari HomeFragment yang akan ditampilkan ke dalam container bernama frame_container yang terdapat dalam layout. Untuk mencegah fragment yang sama ditambahkan lebih dari sekali, program terlebih dahulu memeriksa apakah fragment dengan tag yang sesuai (nama kelas HomeFragment) sudah ada. Jika belum, maka HomeFragment akan ditambahkan ke dalam container tersebut melalui transaksi fragment dan langsung dikomit (commit) agar perubahan diterapkan.

HomeFragment.kt:

Kode di atas merupakan implementasi dari sebuah Fragment di aplikasi Android bernama HomeFragment, yang bertanggung jawab menampilkan daftar agent (karakter atau entitas tertentu) dalam bentuk RecyclerView. Di dalam metode onCreateView(), layout dari fragment diatur menggunakan view binding melalui FragmentHomeBinding. Setelah tampilan di-inflate, daftar agent diinisialisasi ulang dan diisi dengan data yang diperoleh dari metode getListAgent(). Setelah itu, RecyclerView diatur melalui fungsi setupRecyclerView(). Pada setupRecyclerView(), sebuah adapter khusus bernama AgentAdapter dibuat dan diberikan dua argumen utama: daftar data agent dan objek listener yang menangani dua jenis interaksi pengguna. Pertama, saat pengguna mengklik detail agent, maka akan dibuat dan ditampilkan fragment baru (DetailFragment) yang berisi informasi lengkap agent tersebut dengan menggunakan Bundle untuk mengirim data. Fragment baru ini menggantikan fragment yang sedang aktif dan transaksi disimpan ke dalam back stack

agar bisa kembali. Kedua, jika pengguna mengklik tautan link, maka intent dengan aksi ACTION_VIEW dijalankan untuk membuka URL tersebut menggunakan aplikasi browser. Data agent diambil dari resource XML melalui metode getListAgent(). Metode ini memuat array string dari nama, deskripsi, peran, dan link agent, serta array gambar bertipe TypedArray. Data-data tersebut kemudian digabungkan ke dalam objek Agent dan ditambahkan ke dalam list yang kemudian digunakan oleh adapter. TypedArray untuk gambar kemudian di-recycle untuk membebaskan sumber daya.

DetailFragment.kt:

Kode di atas adalah implementasi dari DetailFragment, yaitu salah satu Fragment dalam aplikasi Android yang digunakan untuk menampilkan detail dari sebuah objek Agent yang dipilih sebelumnya. Fragment ini menggunakan View Binding untuk mengakses elemen-elemen UI yang ada di layout-nya, yaitu FragmentDetailBinding. Binding diinisialisasi dalam metode onCreateView(), dan dilepaskan di onDestroyView() untuk menghindari memory leak. Di dalam metode onCreate(), data dikirim dari fragment sebelumnya melalui arguments berupa Bundle. Bundle tersebut membawa informasi seperti nama agent, deskripsi, peran, dan ID gambar (image resource) yang disimpan dalam variabel lokal (name, desc, role, image). Nilai-nilai tersebut kemudian digunakan di dalam metode onViewCreated(), di mana elemen UI seperti TextView dan ImageView diisi dengan data yang sesuai: nama agent ditampilkan di detailName, peran di detailRole, deskripsi di detailDesc, dan gambar agent diatur melalui setImageResource(image). Fragment ini bekerja sebagai bagian akhir dari navigasi pengguna yang dimulai saat memilih item di RecyclerView dalam HomeFragment. Dengan demikian, DetailFragment menyajikan tampilan informasi yang lebih lengkap dan spesifik berdasarkan item yang dipilih.

AgentAdapter.kt:

Kode ini adalah implementasi dari sebuah adapter untuk RecyclerView yang bertugas menampilkan daftar agent dalam bentuk item. Adapter ini menggunakan **AgentAdapter** yang menerima dua parameter: sebuah daftar agent dalam bentuk

ArrayList<Agent> dan listener (OnAgentClickListener) yang menangani interaksi pengguna dengan item yang ada. Dalam adapter ini, terdapat **AgentViewHolder** yang merupakan kelas di dalam adapter dan berfungsi untuk mengikat tampilan elemen-elemen di dalam item. ViewHolder ini menggunakan ItemAgentBinding yang merupakan hasil dari View Binding untuk mengakses elemen-elemen UI yang ada di dalam layout item (seperti agentName, agentRole, agentDesc, agentImage, dan tombol btnLink dan btnDetail). Pada metode **onCreateViewHolder()**, layout untuk setiap item di-inflate menggunakan ItemAgentBinding.inflate(). Di dalam **onBindViewHolder()**, data untuk setiap agent ditetapkan ke dalam elemen UI di dalam item, misalnya nama, peran, deskripsi, dan gambar agent. Dua tombol di setiap item (btnLink dan btnDetail) juga diberikan listener yang akan memanggil fungsi yang ada di OnAgentClickListener ketika tombol tersebut ditekan. onLinkClicked() dipanggil ketika tombol link diklik, sedangkan onDetailClicked() dipanggil ketika tombol detail diklik. Akhirnya, metode **getItemCount()** mengembalikan jumlah item dalam daftar, yang digunakan oleh RecyclerView untuk menampilkan item yang sesuai. Secara keseluruhan, adapter ini memungkinkan RecyclerView untuk menampilkan daftar agent dan menangani interaksi pengguna dengan setiap item di dalam daftar tersebut, mengarah ke detail agent atau membuka link terkait.

Agent.kt:

Kode di atas mendefinisikan sebuah kelas **Agent** yang merupakan data model untuk menyimpan informasi tentang agent, seperti nama, peran, deskripsi, gambar, dan detail tambahan yang berupa URL. Kelas ini menggunakan anotasi @Parcelize dari Kotlin yang memungkinkan objek **Agent** untuk diparceling (dikemas dalam format yang dapat dipindahkan antar komponen, seperti fragment atau aktivitas) menggunakan Parcelable. Dengan Parcelable, objek dapat dipindahkan antar komponen Android (seperti Activity atau Fragment) dengan lebih efisien dibandingkan menggunakan Serializable.

Dalam kelas **Agent**, terdapat lima properti:

1. **name**: String yang menyimpan nama agent.

2. **role**: String yang menyimpan peran atau jabatan agent.
3. **desc**: String yang menyimpan deskripsi tentang agent.
4. **image**: Integer yang menyimpan ID sumber daya gambar yang terkait dengan agent.
5. **detail**: String yang menyimpan URL atau informasi tambahan yang biasanya terkait dengan agent (misalnya link ke detail lebih lanjut tentang agent). Properti ini memiliki nilai default berupa string kosong.

Dengan menggunakan @Parcelize, setiap objek **Agent** dapat dipassing antar fragment dengan cara yang lebih sederhana, tanpa perlu menulis implementasi manual untuk proses parceling dan unparceling. Secara keseluruhan, kelas ini digunakan untuk menyimpan data agent yang akan digunakan dalam tampilan daftar dan ditampilkan dalam fragment-detail atau diklik untuk membuka tautan terkait.

activity_main.xml:

Kode XML di atas mendefinisikan sebuah **FrameLayout** yang digunakan dalam layout activity_main.xml pada aplikasi Android. **FrameLayout** berfungsi sebagai wadah atau container untuk menampilkan elemen UI, biasanya digunakan untuk menampilkan satu elemen pada satu waktu, meskipun dapat menampung beberapa elemen, hanya satu elemen yang akan terlihat di depan. Layout ini memiliki atribut layout_width dan layout_height yang diatur ke match_parent, yang artinya **FrameLayout** akan mengisi seluruh ruang yang tersedia sesuai dengan ukuran parent container-nya. Atribut tools:context digunakan untuk menghubungkan layout ini dengan **MainActivity** selama pengembangan, sehingga Android Studio bisa menampilkan tampilan yang lebih akurat di editor layout. ID frame_container diberikan pada **FrameLayout** ini agar bisa diakses dalam kode, misalnya untuk mengganti fragment secara dinamis menggunakan FragmentTransaction. Secara keseluruhan, layout ini berfungsi sebagai tempat untuk menampilkan fragment-fragment secara bergantian dalam aplikasi.

fragment_detail.xml:

Kode XML di atas mendefinisikan layout untuk **DetailFragment** dengan menggunakan **ConstraintLayout** yang dibungkus dalam **ScrollView**. Layout ini dirancang untuk menampilkan informasi detail tentang agent, seperti gambar, nama, peran, dan deskripsi agent tersebut. Di dalam **ScrollView**, seluruh konten dapat digulirkan, memastikan tampilan tetap dapat diakses meskipun kontennya panjang. **ImageView** digunakan untuk menampilkan gambar agent, yang disesuaikan dengan ukuran kontainer menggunakan properti centerCrop. Tiga **TextView** digunakan untuk menampilkan nama agent, peran agent, dan deskripsi agent, masing-masing diatur dengan ukuran font, gaya teks, dan margin untuk memastikan tampilan yang rapi dan terstruktur. Dengan menggunakan **ConstraintLayout**, posisi setiap elemen UI diatur secara fleksibel dengan menghubungkan elemen-elemen tersebut menggunakan constraints. Layout ini memastikan tampilan detail agent responsif dan dapat menyesuaikan dengan ukuran layar perangkat, serta memberikan pengalaman pengguna yang baik dengan tampilan yang dinamis dan mudah digulirkan.

fragment_home.xml:

Kode XML di atas mendefinisikan layout untuk **HomeFragment** dengan menggunakan **FrameLayout** sebagai container utama. Di dalam **FrameLayout**, terdapat sebuah **RecyclerView** yang diatur untuk mengisi seluruh ruang yang tersedia, dengan ukuran lebar dan tinggi match_parent, yang artinya RecyclerView ini akan mengisi seluruh layar. **RecyclerView** ini memiliki ID rvAgent, yang digunakan untuk menghubungkannya dengan kode di dalam **HomeFragment** untuk menampilkan daftar agent. Meskipun ada atribut android:text yang diberikan nilai @string/hello_blank_fragment, atribut ini tidak berlaku untuk **RecyclerView** karena tidak memiliki elemen teks. Tujuan utama layout ini adalah menyediakan tempat untuk menampilkan daftar agent secara dinamis menggunakan **RecyclerView**, yang akan dikelola dan diatur dalam kode fragment. Layout ini akan ditingkatkan lebih lanjut dengan data yang berasal dari adapter dan listener di dalam kode fragment.

item_agent.xml:

Kode XML di atas mendefinisikan layout untuk sebuah **CardView** yang berisi informasi mengenai agent dalam bentuk kartu, dengan desain yang menggunakan **ConstraintLayout** di dalamnya untuk menata elemen-elemen UI. **CardView** memiliki beberapa atribut untuk memberikan tampilan yang lebih estetis, seperti cardCornerRadius untuk memberi sudut melengkung pada kartu, cardElevation untuk memberikan efek bayangan, dan margin yang diterapkan di sekelilingnya. Di dalam **CardView**, terdapat **ConstraintLayout** yang mengatur posisi elemen-elemen seperti gambar, teks, dan tombol. Gambar agent ditampilkan menggunakan **ImageView**, yang disesuaikan dengan ukuran tetap 150dp x 200dp dan disetel untuk menggunakan centerCrop agar gambar mengisi seluruh ruang yang ada. Di samping gambar, terdapat tiga **TextView** yang menampilkan nama agent, peran agent, dan deskripsi singkat. Dua **Button** digunakan untuk memberikan interaksi, masing-masing dengan teks untuk membuka link terkait dan melihat detail lebih lanjut tentang agent tersebut. Semua elemen diatur secara dinamis menggunakan constraints untuk memastikan tampilan responsif dan terstruktur dengan baik di layar. Layout ini memberi pengalaman pengguna yang menarik dan fungsional dengan elemen-elemen yang tertata rapi dalam sebuah kartu yang dapat digunakan untuk menampilkan daftar agent dalam aplikasi.

strings.xml:

Kode XML di atas merupakan file **strings.xml** yang digunakan untuk mendefinisikan berbagai sumber daya string dalam aplikasi Android. Di dalamnya terdapat beberapa elemen **string** dan **string-array** yang menyediakan data yang digunakan oleh aplikasi. Di bagian pertama, terdapat beberapa **string** seperti nama aplikasi (app_name), teks untuk tombol detail (btn_detail), dan teks untuk tombol link (btn_link). Selanjutnya, ada beberapa **string-array** yang menyimpan informasi tentang agent, seperti nama agent (data_agentName), peran agent (data_agentRole), deskripsi agent (data_agentDesc), gambar agent (data_agentImage), dan link untuk detail agent (data_agentLink). Setiap item dalam array ini berisi data yang relevan untuk menampilkan informasi tentang karakter-karakter dari permainan, misalnya dalam sebuah aplikasi yang menampilkan informasi karakter dari permainan *Valorant*.

Dengan menggunakan **string-array**, aplikasi dapat mengakses dan menampilkan informasi agent secara dinamis, seperti gambar, deskripsi, dan link ke halaman detail masing-masing agent di situs resmi permainan. Kode ini sangat berguna dalam memisahkan data teks dari layout dan logika aplikasi, membuat pengelolaan dan pemeliharaan aplikasi menjadi lebih mudah.

Jetpack Compose:

MainActivity.kt:

Kode di atas merupakan implementasi aplikasi Android menggunakan **Jetpack Compose** untuk menampilkan daftar karakter (disebut "Agent") dengan gambar, deskripsi, dan tombol detail yang mengarah ke halaman web serta halaman deskripsi terperinci. Aplikasi ini dimulai dengan kelas MainActivity, yang mengatur alur navigasi menggunakan **Navigation Component** dari Jetpack Compose. Pada bagian onCreate, aplikasi menyiapkan antarmuka pengguna dengan menggunakan NavHost dan menentukan dua destinasi: "agentList" yang menampilkan daftar agen, dan "deskripsi/{description}/{Image}" yang menunjukkan detail lebih lanjut tentang agen ketika dipilih. Di dalam AgentList yang menggunakan **LazyColumn**, daftar agen ditampilkan secara vertikal menggunakan data dari agentList, yang diwakili oleh nama, gambar, deskripsi, dan URL agen. Setiap item agen dalam daftar ditampilkan dengan menggunakan komponen AgentItem, yang mengemas informasi agen dalam sebuah **Card** dengan gambar di sisi kiri dan teks di sebelah kanan. Ada dua tombol untuk setiap agen: tombol "Detail" yang membuka halaman web agen, serta tombol "Deskripsi" yang mengarahkan pengguna ke halaman detail deskripsi agen dengan gambar besar dan deskripsi panjang. Untuk navigasi antar layar, ketika tombol "Deskripsi" ditekan, aplikasi menavigasi ke DeskripsiScreen, yang menampilkan gambar agen dalam ukuran besar di bagian atas, diikuti dengan deskripsi lengkap agen. Data yang ditampilkan di halaman deskripsi dipass melalui argumen navigasi dan dikodekan untuk menangani karakter khusus di URL.

dataList.kt:

Kode di atas mendefinisikan sebuah kelas data Kotlin bernama DataAgent, yang digunakan untuk menyimpan informasi tentang agen dalam permainan *Valorant*. Kelas ini memiliki empat properti: name (yang menyimpan nama agen dalam bentuk String), image (yang menyimpan ID sumber daya gambar agen sebagai integer), url (yang berisi tautan atau URL ke halaman web agen dalam bentuk String), dan description (yang menyimpan deskripsi agen dalam bentuk String). Kelas ini bertujuan untuk menyederhanakan penyimpanan dan pengelolaan data agen, sehingga bisa digunakan di dalam aplikasi untuk menampilkan informasi terkait karakter agen secara terstruktur dan terorganisir.

agentList.kt:

Kode di atas mendefinisikan sebuah daftar bernama agentList, yang berisi beberapa objek DataAgent. Setiap objek DataAgent mewakili agen dalam permainan *Valorant* dan menyimpan informasi terkait agen tersebut, seperti nama, gambar (diambil dari sumber daya drawable), URL yang mengarah ke halaman detail agen, serta deskripsi mengenai peran dan keahlian masing-masing agen. Misalnya, agen "Jett" memiliki peran sebagai Duelist dan dikenal dengan gaya bertarung yang gesit dan evasif, sementara "Brimstone" adalah seorang Controller yang mengandalkan kemampuan untuk memberikan dukungan taktis kepada tim. Daftar ini digunakan untuk menampilkan informasi agen-agen tersebut dalam aplikasi dengan tujuan untuk memberikan pengenalan lebih dalam tentang karakter-karakter yang ada dalam *Valorant*.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/MadeByBintang/PraktikumMobile