

**LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3**



BUILD A SCROLLABLE LIST

Oleh:

Noviana Nur Aisyah

NIM. 2310817120005

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025**

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE
MODUL 3

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

Nama Praktikan : Noviana Nur Aisyah
NIM : 2310817120005

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

LEMBAR PENGESAHAN	2
DAFTAR ISI	3
DAFTAR GAMBAR.....	4
DAFTAR TABEL	5
SOAL 1	6
A. Source Code.....	6
B. Output Program	26
C. Pembahasan	29
SOAL 2.....	39
A. Penjelasan	39
Tautan Git	40

DAFTAR GAMBAR

Gambar 1. Screenshot Hasil Jawaban Soal 1	26
Gambar 2. Screenshot Hasil Jawaban Soal 1	27
Gambar 3. Screenshot Hasil Jawaban Soal 1	28
Gambar 4. Screenshot Hasil Jawaban Soal 1	29

DAFTAR TABEL

Tabel 1. Source Code Jawaban Soal 1.....	6
Tabel 2. Source Code Jawaban Soal 1.....	8
Tabel 3. Source Code Jawaban Soal 1.....	8
Tabel 4. Source Code Jawaban Soal 1.....	10
Tabel 5. Source Code Jawaban Soal 1.....	14
Tabel 6. Source Code Jawaban Soal 1.....	15
Tabel 7. Source Code Jawaban Soal 1.....	16
Tabel 8. Source Code Jawaban Soal 1.....	20
Tabel 9. Source Code Jawaban Soal 1.....	22

SOAL 1

Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:

1. List menggunakan fungsi RecyclerView (XML) atau 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/intent 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.

A. Source Code

1. MainActivity.kt

Tabel 1. Source Code Jawaban Soal 1

1	<code>package com.example.scrollablelist</code>
2	
3	<code>import android.os.Bundle</code>
4	<code>import android.util.Log</code>
5	<code>import androidx.activity.enableEdgeToEdge</code>
6	<code>import androidx.appcompat.app.AppCompatActivity</code>

```

7 import androidx.core.view.ViewCompat
8 import androidx.core.view.WindowInsetsCompat
9 import
10 com.example.scrollablelist.databinding.ActivityMainBindin
11 g
12
13 class MainActivity : AppCompatActivity() {
14     private lateinit var binding: ActivityMainBinding
15
16     override fun onCreate(savedInstanceState: Bundle?) {
17         super.onCreate(savedInstanceState)
18
19         binding =
20 ActivityMainBinding.inflate(layoutInflater)
21         setContentView(binding.root)
22
23         val fragmentManager = supportFragmentManager
24         val homeFragment = HomeFragment()
25         val fragment =
26 fragmentManager.findFragmentByTag(HomeFragment::class.java
27 a.simpleName)
28         if (fragment !is HomeFragment) {
29             Log.d("MyFlexibleFragment", "Fragment Name : "
30 + HomeFragment::class.java.simpleName)
31             fragmentManager
32                 .beginTransaction()
33                 .add(R.id.main, homeFragment,
34 HomeFragment::class.java.simpleName)
35                 .commit()
36         }
37

```

38	}
	}

2. Song.kt

Tabel 2. Source Code Jawaban Soal 1

1	package com.example.scrollablelist
2	
3	import android.icu.text.CaseMap.Title
4	import android.os.Parcel
5	import kotlinx.parcelize.Parcelize
6	
7	@Parcelize
8	data class Song(
9	val title: String,
10	val link: String,
11	val photo: String,
12	val singer: String,
13	val album: String,
14	val lyrics: String,
15	val desc: String
16	
17):Parcel

3. DetailFragment.kt

Tabel 3. Source Code Jawaban Soal 1

1	package com.example.scrollablelist
2	
3	import android.os.Bundle
4	import android.view.LayoutInflater
5	import android.view.View
6	import android.view.ViewGroup


```

7 import androidx.fragment.app.Fragment
8 import com.bumptech.glide.Glide
9 import
10 com.example.scrollablelist.databinding.FragmentDetailBinding
11
12 class DetailFragment : Fragment() {
13
14     private var _binding: FragmentDetailBinding? = null
15     private val binding get() = _binding!!
16
17     override fun onCreateView(
18         inflater: LayoutInflater,
19         container: ViewGroup?,
20         savedInstanceState: Bundle?
21     ): View? {
22         _binding = FragmentDetailBinding.inflate(inflater,
23 container, false)
24
25         val title = arguments?.getString("TITLE")
26         val photo = arguments?.getString("PHOTO")
27         val singer = arguments?.getString("SINGER")
28         val album = arguments?.getString("ALBUM")
29         val lyrics = arguments?.getString("LYRICS")
30
31         binding.songTitle.text = title
32         photo?.let {
33             Glide.with(requireContext())
34                 .load(it)
35                 .into(binding.songCover)
36         }
37         binding.songSinger.text = singer

```

38	binding.songAlbum.text = album
39	binding.songLyrics.text = lyrics
40	
41	binding.toolbar.setNavigationOnClickListener {
42	parentFragmentManager.popBackStack()
43	}
44	
45	return binding.root
46	}
47	
48	override fun onDestroyView() {
49	super.onDestroyView()
50	_binding = null
51	}
52	}

4. HomeFragment.kt

Tabel 4. Source Code Jawaban Soal 1

1	package com.example.scrollablelist
2	
3	import android.os.Bundle
4	import android.view.LayoutInflater
5	import android.view.View
6	import android.view.ViewGroup
7	import
8	androidx.constraintlayout.helper.widget.Carousel.Adapter
9	import androidx.fragment.app.Fragment
10	import
11	com.example.scrollablelist.databinding.FragmentHomeBindi
12	ng
13	import android.content.Intent

```

14 import android.net.Uri
15 import androidx.core.os.bundleOf
16 import androidx.recyclerview.widget.LinearLayoutManager
17
18 class HomeFragment : Fragment() {
19     private var _binding: FragmentHomeBinding? = null
20     private val binding get() = _binding!!
21
22     private lateinit var songAdapter: ListSongAdapter
23     private val list = ArrayList<Song>()
24
25     override fun onCreateView(
26         inflater: LayoutInflater,
27         container: ViewGroup?,
28         savedInstanceState: Bundle?
29     ): View? {
30         _binding = FragmentHomeBinding.inflate(inflater,
31 container, false)
32
33         list.clear()
34         list.addAll(getListSong())
35         setupRecyclerView()
36
37         return binding.root
38     }
39
40     private fun setupRecyclerView() {
41         songAdapter = ListSongAdapter(
42             list,
43             onSpotifyClick = { link ->
44

```

```

45         val intent = Intent(Intent.ACTION_VIEW,
46 Uri.parse(link))
47         startActivity(intent)},
48         onClick = { title, photo, singer,
49 album, lyrics ->
50             val detailFragment =
51 DetailFragment().apply {
52                 arguments = Bundle().apply {
53                     putString("TITLE", title)
54                     putString("PHOTO", photo)
55                     putString("SINGER", singer)
56                     putString("ALBUM", album)
57                     putString("LYRICS", lyrics)
58                 }
59             }
60
61         parentFragmentManager.beginTransaction()
62             .replace(R.id.main, detailFragment)
63             .addToBackStack(null)
64             .commit()
65     }
66 )
67
68     binding.rvSong.apply {
69         layoutManager = LinearLayoutManager(context)
70         adapter = songAdapter
71         setHasFixedSize(true)
72     }
73 }
74
75     private fun getListSong() : ArrayList<Song> {

```

76	val dataTitle =
77	resources.getStringArray(R.array.data_title)
78	val dataLink =
79	resources.getStringArray(R.array.data_link)
80	val dataPhoto =
81	resources.getStringArray(R.array.data_photo)
82	val dataSinger =
83	resources.getStringArray(R.array.data_singer)
84	val dataAlbum =
85	resources.getStringArray(R.array.data_album)
86	val dataLyrics =
87	resources.getStringArray(R.array.data_lyrics)
88	val dataDesc =
89	resources.getStringArray(R.array.data_desc)
90	val listSong = ArrayList<Song>()
91	
92	for (i in dataTitle.indices) {
93	val song = Song(title = dataTitle[i], link =
94	dataLink[i], photo = dataPhoto[i], singer =
95	dataSinger[i], album = dataAlbum[i], lyrics =
96	dataLyrics[i], desc = dataDesc[i])
97	listSong.add(song)
98	}
99	
100	return listSong
101	}
102	
103	override fun onDestroyView() {
104	super.onDestroyView()
105	_binding = null
106	}

107	}
-----	---

5. ListSongAdapter.kt

Tabel 5. Source Code Jawaban Soal 1

1	package com.example.scrollablelist
2	
3	import android.view.LayoutInflater
4	import android.view.ViewGroup
5	import androidx.recyclerview.widget.RecyclerView
6	import com.bumptech.glide.Glide
7	import
8	com.example.scrollablelist.databinding.ItemSongBinding
9	
10	class ListSongAdapter(
11	private val listSong: ArrayList<Song>,
12	private val onSpotifyClick: (String) -> Unit,
13	private val onDetailClick: (String, String, String,
14	String, String) -> Unit)
15	: RecyclerView.Adapter<ListSongAdapter.ListViewHolder>()
16	{
17	
18	inner class ListViewHolder(val binding: ItemSongBinding)
19	: RecyclerView.ViewHolder(binding.root) {
20	fun bind(song: Song) {
21	binding.tvTitle.text = song.title
22	binding.tvDesc.text = song.desc
23	Glide.with(binding.root.context)
24	.load(song.photo)
25	.into(binding.imgCover)
26	
27	

28	binding.buttonSpotify.setOnClickListener {
29	onSpotifyClick(song.link)
30	}
31	
32	binding.buttonDetail.setOnClickListener {
33	onDetailClick(song.title, song.photo,
34	song.singer, song.album, song.lyrics)
35	}
36	}
37	}
38	
39	override fun onCreateViewHolder(parent: ViewGroup,
40	viewType: Int): ViewHolder {
41	val binding =
42	ItemSongBinding.inflate(LayoutInflater.from(parent.context),
43	parent, false)
44	return ViewHolder(binding)
45	}
46	
47	override fun getItemCount(): Int {
48	return listSong.size
49	}
50	
51	override fun onBindViewHolder(holder: ViewHolder,
52	position: Int) {
53	holder.bind(listSong[position])
54	}
55	}

6. activity_main.xml

Tabel 6. Source Code Jawaban Soal 1

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

7. fragment_detail.xml

Tabel 7. Source Code Jawaban Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	android:layout_width="match_parent"
5	android:layout_height="match_parent"
6	xmlns:app="http://schemas.android.com/apk/res-auto"
7	xmlns:tools="http://schemas.android.com/tools"
8	tools:context=".DetailFragment">
9	
10	<androidx.appcompat.widget.Toolbar
11	android:id="@+id/toolbar"
12	android:layout_width="match_parent"
13	android:layout_height="wrap_content"
14	android:background="@drawable/linear_gradient"
15	android:theme="?attr/actionBarTheme"
16	app:title="About Song"
17	app:titleTextColor="@color/white"
18	app:navigationIcon="@drawable/arrow_back"

19	app:layout_constraintTop_toTopOf="parent"
20	app:layout_constraintStart_toStartOf="parent"
21	app:layout_constraintEnd_toEndOf="parent"
22	/>
23	
24	<androidx.core.widget.NestedScrollView
25	android:id="@+id/scroll_view"
26	android:layout_width="0dp"
27	android:layout_height="0dp"
28	app:layout_constraintTop_toBottomOf="@id/toolbar"
29	app:layout_constraintBottom_toBottomOf="parent"
30	app:layout_constraintStart_toStartOf="parent"
31	app:layout_constraintEnd_toEndOf="parent"
32	android:scrollbars="vertical"
33	android:padding="16dp"
34	android:layout_margin="8dp"
35	>
36	
37	<LinearLayout
38	android:layout_width="match_parent"
39	android:layout_height="wrap_content"
40	android:orientation="vertical"
41	android:padding="8dp">
42	
43	<ImageView
44	android:id="@+id/song_cover"
45	android:layout_width="300dp"
46	android:layout_height="300dp"
47	android:layout_marginTop="8dp"
48	android:src="@drawable/place_holder"
49	android:layout_gravity="center_horizontal"

50	<code></></code>
51	
52	<code><TextView</code>
53	<code> android:id="@+id/song_title"</code>
54	<code> android:layout_width="wrap_content"</code>
55	<code> android:layout_height="wrap_content"</code>
56	
57	<code>app:layout_constraintStart_toStartOf="parent"</code>
58	<code> app:layout_constraintEnd_toEndOf="parent"</code>
59	
60	<code>app:layout_constraintTop_toBottomOf="@id/song_cover"</code>
61	<code> android:layout_marginTop="16dp"</code>
62	<code> android:text="TEDDY PICKER"</code>
63	<code> android:textStyle="bold"</code>
64	<code> android:textSize="20sp"</code>
65	<code> android:padding="2dp"</code>
66	<code> android:layout_gravity="center_horizontal"</code>
67	<code></></code>
68	
69	<code><TextView</code>
70	<code> android:id="@+id/song_singer"</code>
71	<code> android:layout_width="wrap_content"</code>
72	<code> android:layout_height="wrap_content"</code>
73	
74	<code>app:layout_constraintStart_toStartOf="parent"</code>
75	<code> app:layout_constraintEnd_toEndOf="parent"</code>
76	
77	<code>app:layout_constraintTop_toBottomOf="@id/song_title"</code>
78	<code> android:layout_marginTop="16dp"</code>
79	<code> android:padding="2dp"</code>
80	<code> android:text="Singer: Arctic Monkeys"</code>

81	android:textSize="14sp"
82	android:layout_gravity="center_horizontal"
83	/>
84	
85	<TextView
86	android:id="@+id/song_album"
87	android:layout_width="wrap_content"
88	android:layout_height="wrap_content"
89	
90	app:layout_constraintStart_toStartOf="parent"
91	app:layout_constraintEnd_toEndOf="parent"
92	android:layout_marginTop="4dp"
93	android:padding="2dp"
94	
95	app:layout_constraintTop_toBottomOf="@id/song_singer"
96	android:text="Album: Favourite Worst
97	Nightmare"
98	android:textSize="14sp"
99	android:layout_gravity="center_horizontal"
100	/>
101	
102	<TextView
103	android:id="@+id/song_content"
104	android:layout_width="wrap_content"
105	android:layout_height="wrap_content"
106	
107	app:layout_constraintStart_toStartOf="parent"
108	app:layout_constraintEnd_toEndOf="parent"
109	android:layout_marginTop="16dp"
110	android:padding="2dp"
111	

112	
113	app:layout_constraintTop_toBottomOf="@id/song_album"
114	android:text="Lyrics"
115	android:textStyle="bold"
116	android:textSize="14sp"
117	android:layout_gravity="center_horizontal"
118	/>
119	
120	<TextView
121	android:id="@+id/song_lyrics"
122	android:layout_width="300dp"
123	android:layout_height="wrap_content"
124	
125	app:layout_constraintStart_toStartOf="parent"
126	app:layout_constraintEnd_toEndOf="parent"
127	android:layout_marginTop="8dp"
128	
129	app:layout_constraintTop_toBottomOf="@id/song_content"
130	android:layout_gravity="center_horizontal"
131	android:lineSpacingMultiplier="1.5"
132	android:text="llorem ipsum dolor ist
133	amenfdusbgjrabjfbshbfrabjrft jfasjfkufheragjrgjjbgj
134	garhgiargjkafjkdgjdfjgdjkbgnajnngjd gaigirigoreio
135	igiarigreiogjraei garjggjewoiwetieri "
136	/>
137	</LinearLayout>
138	</androidx.core.widget.NestedScrollView>
139	</androidx.constraintlayout.widget.ConstraintLayout>

8. fragment_home.xml

Tabel 8. Source Code Jawaban Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	android:layout_width="match_parent"
5	android:layout_height="match_parent"
6	xmlns:app="http://schemas.android.com/apk/res-auto"
7	xmlns:tools="http://schemas.android.com/tools"
8	tools:context=".HomeFragment">
9	
10	<androidx.appcompat.widget.Toolbar
11	android:id="@+id/toolbar"
12	android:layout_width="match_parent"
13	android:layout_height="wrap_content"
14	android:background="@drawable/linear_gradient"
15	android:theme="?attr/actionBarTheme"
16	app:title="My Favourite Songs"
17	app:titleTextColor="@color/white"
18	app:layout_constraintTop_toTopOf="parent"
19	app:layout_constraintStart_toStartOf="parent"
20	app:layout_constraintEnd_toEndOf="parent"
21	/>
22	
23	<androidx.recyclerview.widget.RecyclerView
24	android:id="@+id/rv_song"
25	android:layout_width="0dp"
26	android:layout_height="0dp"
27	android:layout_marginTop="85dp"
28	android:layout_marginLeft="20dp"
29	android:layout_marginRight="20dp"
30	android:layout_marginBottom="20dp"
31	app:layout_constraintBottom_toBottomOf="parent"

32	app:layout_constraintEnd_toEndOf="parent"
33	app:layout_constraintStart_toStartOf="parent"
34	app:layout_constraintTop_toTopOf="parent"
35	android:scrollbars="vertical"
36	/>
37	
38	</androidx.constraintlayout.widget.ConstraintLayout>

9. item_song.xml

Tabel 9. Source Code Jawaban Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.cardview.widget.CardView
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	d"
5	xmlns:card_view="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="wrap_content"
10	android:id="@+id/card_view"
11	android:layout_gravity="center"
12	android:layout_marginStart="8dp"
13	android:layout_marginTop="4dp"
14	android:layout_marginEnd="8dp"
15	android:layout_marginBottom="8dp"
16	card_view:cardCornerRadius="20dp"
17	>
18	
19	<androidx.constraintlayout.widget.ConstraintLayout
20	android:layout_width="match_parent"
21	android:layout_height="wrap_content"

22	android:padding="15dp">
23	
24	<ImageView
25	android:id="@+id/img_cover"
26	android:layout_width="120dp"
27	android:layout_height="120dp"
28	android:scaleType="centerCrop"
29	
30	card_view:layout_constraintBottom_toBottomOf="parent"
31	
32	card_view:layout_constraintStart_toStartOf="parent"
33	
34	card_view:layout_constraintTop_toTopOf="parent"
35	android:src="@drawable/place_holder" />
36	
37	<TextView
38	android:id="@+id/tv_title"
39	android:layout_width="180dp"
40	android:layout_height="wrap_content"
41	android:layout_marginTop="8dp"
42	android:layout_marginLeft="6dp"
43	android:textSize="16sp"
44	android:textStyle="bold"
45	
46	card_view:layout_constraintEnd_toEndOf="parent"
47	
48	card_view:layout_constraintHorizontal_bias="0.1"
49	
50	card_view:layout_constraintStart_toEndOf="@id/img_cover"
51	
52	card_view:layout_constraintTop_toTopOf="parent"

53	tools:text="Jakarta Hari Ini"
54	android:padding="2dp"
55	/>
56	
57	<TextView
58	android:id="@+id/tv_desc"
59	android:layout_width="200dp"
60	android:layout_height="wrap_content"
61	android:textSize="12sp"
62	android:layout_marginTop="8dp"
63	android:layout_marginLeft="6.67dp"
64	android:padding="2dp"
65	
66	card_view:layout_constraintEnd_toEndOf="parent"
67	
68	card_view:layout_constraintHorizontal_bias="0.1"
69	
70	card_view:layout_constraintStart_toEndOf="@id/img_cover"
71	
72	card_view:layout_constraintTop_toBottomOf="@id/tv_title"
73	tools:text="a song by LANY, talk about a
74	person who lose his home, his comfort person....." />
75	
76	<androidx.appcompat.widget.AppCompatButton
77	android:id="@+id/button_detail"
78	android:layout_width="wrap_content"
79	android:layout_height="wrap_content"
80	android:layout_marginTop="16dp"
81	android:text="Detail"
82	android:textSize="12dp"
83	android:textColor="@color/white"

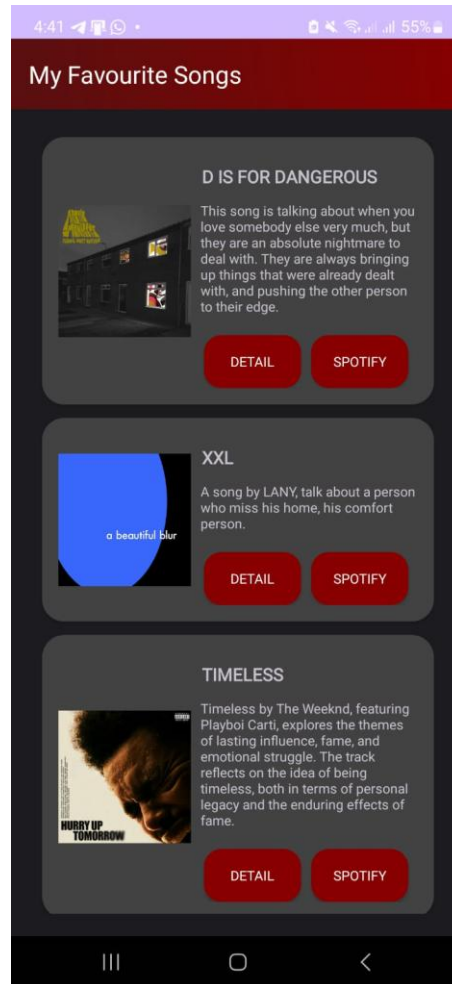
84	
85	android:background="@drawable/button_background"
86	
87	card_view:layout_constraintBottom_toBottomOf="parent"
88	
89	card_view:layout_constraintEnd_toEndOf="parent"
90	
91	card_view:layout_constraintHorizontal_bias="0.1"
92	
93	card_view:layout_constraintStart_toEndOf="@id/img_cover"
94	
95	card_view:layout_constraintTop_toBottomOf="@id/tv_desc"
96	
97	card_view:layout_constraintVertical_bias="0.0" />
98	
99	<androidx.appcompat.widget.AppCompatButton
100	android:id="@+id/button_spotify"
101	android:layout_width="wrap_content"
102	android:layout_height="wrap_content"
103	android:layout_marginTop="16dp"
104	android:layout_marginLeft="100dp"
105	android:text="Spotify"
106	android:textSize="12dp"
107	android:textColor="@color/white"
108	
109	android:background="@drawable/button_background"
110	
111	card_view:layout_constraintBottom_toBottomOf="parent"
112	
113	card_view:layout_constraintEnd_toEndOf="parent"
114	

```

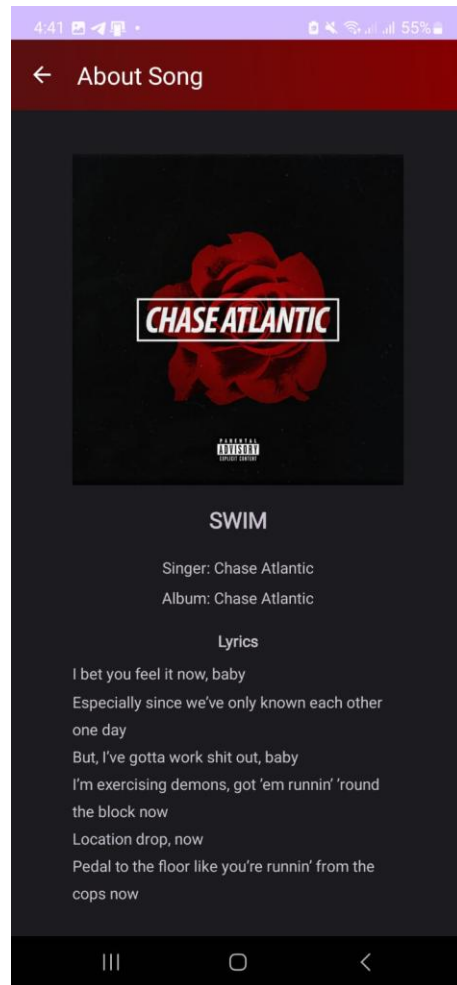
115
116 card_view:layout_constraintStart_toEndOf="@id/img_cover"
117
118 card_view:layout_constraintTop_toBottomOf="@id/tv_desc"
119
120 card_view:layout_constraintVertical_bias="0.0" />
121     </androidx.constraintlayout.widget.ConstraintLayout>
122
123 </androidx.cardview.widget.CardView>

```

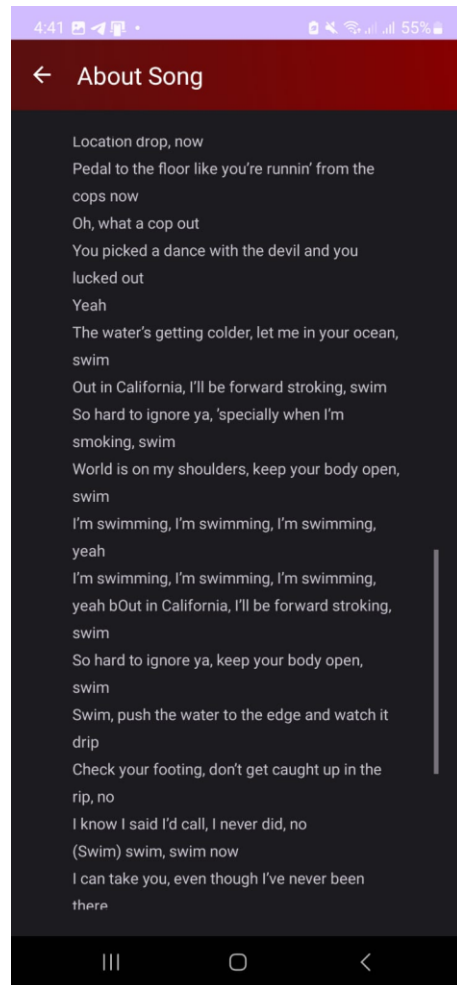
B. Output Program



Gambar 1. Screenshot Hasil Jawaban Soal 1



Gambar 2. Screenshot Hasil Jawaban Soal 1



Gambar 3. Screenshot Hasil Jawaban Soal 1



Gambar 4. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt

File ini merupakan logika utama dari aplikasi untuk menampilkan UI dan mengatur interaksi pengguna. Terdapat beberapa bagian penting, yaitu menambahkan HomeFragment sebagai tampilan awal saat aplikasi dijalankan. Terdapat beberapa bagian penting, di antaranya yaitu:

a) View Binding

```
private lateinit var binding: ActivityMainBinding
binding = ActivityMainBinding.inflate(layoutInflater)
setContentView(binding.root)
```

Digunakan untuk menghubungkan file layout `activity_main.xml` ke file `MainActivity.kt` tanpa perlu menggunakan `findViewById`. Sementara itu, `binding.root` akan menjadi tampilan utama yang ditampilkan di layar.

b) **Fragment Manager**

```
val fragmentManager = supportFragmentManager
```

Digunakan untuk mengelola fragment dalam activity. Lalu, `supportFragmentManager` merupakan salah satu bagian AndroidX yang mendukung fragment secara kompatibel di berbagai versi Android.

c) **Memeriksa Fragment**

```
val homeFragment = HomeFragment()  
val fragment =  
fragmentManager.findFragmentByTag(HomeFragment::class  
    .java.simpleName)
```

Digunakan untuk membuat instance `HomeFragment`. Kemudian, melakukan pemeriksaan apakah fragment dengan tag `HomeFragment` sudah ditambahkan sebelumnya, hal ini untuk mencegah fragment saat activity dipanggil ulang.

d) **Menambahkan Fragment ke Activity**

```
if (fragment !is HomeFragment) {  
    Log.d("MyFlexibleFragment", "Fragment Name : " +  
        HomeFragment::class.java.simpleName)  
    fragmentManager  
        .beginTransaction()  
        .add(R.id.main, homeFragment,  
            HomeFragment::class.java.simpleName)  
        .commit()  
}
```

Jika fragment belum ada, maka akan ditambahkan ke dalam container `R.id.main`. Proses ini dilakukan menggunakan transaksi fragment. `Log.d(...)` digunakan untuk mencatat nama fragment yang sedang ditambahkan ke logcat untuk keperluan debugging.

2. Song.kt

File ini mendefinisikan struktur data yang bernama Song. File ini digunakan untuk merepresentasikan informasi song/lagu secara lengkap dan dapat dikirim antar komponen Android. Terdapat beberapa bagian, yaitu:

a) Anotasi @Parcelize

```
@Parcelize
```

Anotasi ini digunakan untuk membuat objek Song dapat di-serialize secara otomatis menjadi bentuk yang dikirim melalui Intent atau Bundle. Digunakan bersama dengan interface Parcelable.

b) Deklarasi

```
data class Song(  
    val title: String,  
    val link: String,  
    val photo: String,  
    val singer: String,  
    val album: String,  
    val lyrics: String,  
    val desc: String  
) : Parcelable
```

Bagian ini mendeklarasikan data class.

c) Parcelable

```
) : Parcelable
```

Bagian ini membuat Song dapat dikirim melalui Intent dan Bundle.

3. activity_main.xml

File ini merupakan Fragment yang digunakan untuk menampilkan detail dari sebuah lagu yang dipilih. Fragment ini menerima data dari argument Bundle dan menampilkannya di layout fragment_detail.xml. Terdapat beberapa bagian, yaitu:

a) View Binding pada Fragment

```
private var _binding: FragmentDetailBinding? = null  
private val binding get() = _binding!!
```

Digunakan untuk mencegah memory leak, di mana `_binding` hanya aktif selama `onCreateView` hingga `OnDestroyView` dan `binding.get()` digunakan untuk mengakses binding secara aman selama fragment aktif.

b) `onCreateView()`

```
_binding = FragmentDetailBinding.inflate(inflater,
container, false)
```

Digunakan untuk meng-inflate layout fragment menggunakan view binding. Lalu, `binding.root` akan dikembalikan sebagai tampilan utama fragment.

c) Menerima Argument dari Fragment Lain

```
val title = arguments?.getString("TITLE")
val photo = arguments?.getString("PHOTO")
val singer = arguments?.getString("SINGER")
val album = arguments?.getString("ALBUM")
val lyrics = arguments?.getString("LYRICS")
```

Digunakan untuk menerima data berupa `String` dari Fragment sebelumnya menggunakan `Bundle`.

d) Menampilkan Data ke UI

```
binding.songTitle.text = title
binding.songSinger.text = singer
binding.songAlbum.text = album
binding.songLyrics.text = lyrics
```

Menampilkan data song ke dalam `TextView` di layout.

e) Menampilkan Gambar dengan Glide

```
photo?.let {
    Glide.with(requireContext())
        .load(it)
        .into(binding.songCover)
}
```

Digunakan untuk menampilkan gambar dari URL ke dalam `Image View` `songCover` menggunakan library `Glide`.

f) Tombol Back di Toolbar


```
binding.toolbar.setNavigationOnClickListener {
    parentFragmentManager.popBackStack()
}
```

Digunakan untuk mengatur aksi tombol back di toolbar agar Kembali ke Fragment sebelumnya.

g) `onDestroyView()`

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

Digunakan untuk membersihkan objek binding saat fragment dihancurkan agar tidak terjadi memory leak.

4. HomeFragment.kt

File ini merupakan tampilan utama dari aplikasi yang menampilkan daftar song dalam bentuk RecyclerView. Terdapat beberapa bagian, yaitu:

a) View Binding pada Fragment

```
private var _binding: FragmentHomeBinding? = null
private val binding get() = _binding!!
```

Digunakan untuk mengakses elemen XML (`fragment_home.xml`) secara langsung dan aman dari `NullPointerException`.

b) Deklarasi Variabel

```
private lateinit var songAdapter: ListSongAdapter
private val list = ArrayList<Song>()
```

Di sini terdapat adapter khusus untuk menampilkan data song dalam RecyclerView dan list digunakan untuk menyimpan data lagu dari resource (array).

c) `onCreateView()`

```
list.clear()
list.addAll(getListSong())
setupRecyclerView()
```

Digunakan untuk mengambil data dari array dan menyimpannya dala list. Kemudian, memanggil fungsi `setupRecyclerView()` untuk menampilkan daftar lagu ke layar.

d) `setupRecyclerView()`

```
songAdapter = ListSongAdapter(  
    list,  
    onSpotifyClick = { link -> ... },  
    onDetailClick = { title, photo, ... -> ... }  
)
```

`onSpotifyClick`: ketika tombol Spotify di item lagu ditekan, membuka link ke aplikasi/browser Spotify.

`onDetailClick`: ketika item lagu ditekan, pindah ke `DetailFragment` dan mengirim data lagu lewat arguments.

e) `getListSong()`

```
val dataTitle =  
resources.getStringArray(R.array.data_title)  
...
```

Digunakan untuk mengambil semua data lagu dari file `strings.xml` (berupa array resource). Kemudian, data dikonversi menjadi objek `Song` dan dimasukkan ke list.

f) Navigasi ke `DetailFragment`

```
val detailFragment = DetailFragment().apply {  
    arguments = Bundle().apply {  
        putString("TITLE", title)  
        ...  
    }  
}
```

Digunakan untuk membuat instance `detailFragment`, lalu megirim data melalui arguments. Kemudian, tampilan diubah dengan `DetailFragment` menggunakan `FragmentManager`.

g) `onDestroyView()`

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

Digunakan untuk membersihkan binding saat tampilan dihancurkan untuk mencegah memory leak

5. ListSongAdapter.kt

File ini adalah adapter untuk RecyclerView yang bertanggung jawab menampilkan daftar lagu dalam tampilan list. Adapter ini juga menangani event klik tombol Spotify dan tombol Detail untuk setiap item lagu. Terdapat beberapa bagian, yaitu:

a) Deklarasi Class dan Konstruktor

```
class ListSongAdapter(  
    private val listSong: ArrayList<Song>,  
    private val onSpotifyClick: (String) -> Unit,  
    private val onDetailClick: (String, String,  
String, String, String) -> Unit)  
:  
RecyclerView.Adapter<ListSongAdapter.ListViewHolder>(
```

`listSong` merupakan data berisi objek `Song`.

`onSpotifyClick` merupakan function yang dipanggil saat tombol Spotify diklik.

`onDetailClick` merupakan function saat tombol Detail diklik dan data lagu dikirim ke fragment detail.

b) ViewHolder

```
inner class ListViewHolder(val binding:  
ItemSongBinding) :  
RecyclerView.ViewHolder(binding.root)
```

ViewHolder menyimpan referensi ke layout item lagu (item_song.xml) menggunakan View Binding.

c) `bind()`

```
fun bind(song: Song) {  
    binding.tvTitle.text = song.title  
    binding.tvDesc.text = song.desc  
    Glide.with(binding.root.context)  
        .load(song.photo)  
        .into(binding.imgCover)  
  
    binding.buttonSpotify.setOnClickListener {  
        onSpotifyClick(song.link)  
    }  
  
    binding.buttonDetail.setOnClickListener {  
        onDetailClick(song.title, song.photo,  
song.singer, song.album, song.lyrics)  
    }  
}
```

`tvTitle` dan `tvDesc` digunakan untuk menampilkan judul dan deskripsi song.
`Glide` digunakan untuk memuat dan menampilkan gambar cover song dari URL.

`buttonSpotify` digunakan untuk membuka link Spotify.

`buttonDetail` digunakan untuk mengirim data lagu ke `DetailFragment`.

d) `onCreateViewHolder()`

```
val binding =  
ItemSongBinding.inflate(LayoutInflater.from(parent.co  
ntext), parent, false)  
return ViewHolder(binding)
```

Bagian ini berfungsi membuat tampilan satu item daftar lagu menggunakan `item_song.xml` dan mengembalikan `ViewHolder`.

e) getItemCount()

```
override fun getItemCount(): Int = listSong.size
```

Bagian ini berfungsi untuk menentukan jumlah item dalam list, sesuai dengan jumlah data song.

f) onBindViewHolder

```
holder.bind(listSong[position])
```

Digunakan untuk memanggil fungsi bind() untuk menampilkan data song pada posisi yang sesuai di RecyclerView.

6. activity_main.xml

activity_main.xml adalah tata letak kosong dengan FrameLayout, dirancang agar fleksibel menampilkan konten fragment. Ini umum digunakan dalam aplikasi Android berbasis fragment agar UI dapat berganti tanpa membuat banyak activity.

7. Fragment_detail.xml

Terdapat beberapa bagian penting, yaitu:

a) Root Layout

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

ConstraintLayout digunakan sebagai root agar kita bisa menempatkan elemen UI secara fleksibel dengan constraint.

b) Toolbar

```
<androidx.appcompat.widget.Toolbar ... />
```

Digunakan untuk membuat custom toolbar.

c) NestedScrollView

```
<androidx.core.widget.NestedScrollView ... >
```

Digunakan untuk membungkus konten agar bisa di-scroll secara vertikal.

d) LinearLayout

Di dalam NestedScrollView, digunakan LinearLayout vertikal untuk menampilkan isi konten.

8. Fragment_home.kt

a) Root

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

Memungkinkan penempatan elemen UI secara fleksibel dengan constraint antar elemen.

b) Toolbar

```
<androidx.appcompat.widget.Toolbar ... />
```

Bagian ini dapat menampilkan judul halaman, yaitu “My Favourite Song”.

c) RecyclerView

```
<androidx.recyclerview.widget.RecyclerView ... />
```

Menampilkan list song secara vertikal (akan diatur melalui adapter di kode Kotlin).

9. item_song.xml

a) Root: CardView

```
<androidx.cardview.widget.CardView ...>
```

Merupakan pembungkus utama yang membentuk tampilan item seperti kartu dengan sudut melengkung (`cardCornerRadius="20dp"`).

b) ConstraintLayout

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

Digunakan untuk mengatur komponen di dalam CardView secara fleksibel.

SOAL 2

Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

A. Penjelasan

RecyclerView masih banyak digunakan karena memiliki API yang matang. Tidak semua pengembang aplikasi dapat menggunakan dan mau beralih Jetpack Compose walaupun lebih mudah, hal ini karena mereka sudah terbiasa. Kemudian, jika pengembang ingin membangun sebuah aplikasi yang mendukung perangkat minSDK di bawah 21, maka RecyclerView masih sangat relevan (Compose membutuhkan API 21 ke atas).

Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat
<https://github.com/Noviana21/Pemrograman-Mobile/tree/main/modul%203>.