LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

VIEWMODEL & DEBUGGING

Oleh:

Noviana Nur Aisyah NIM. 2310817120005

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 : 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

DAFTAR GAMBAR

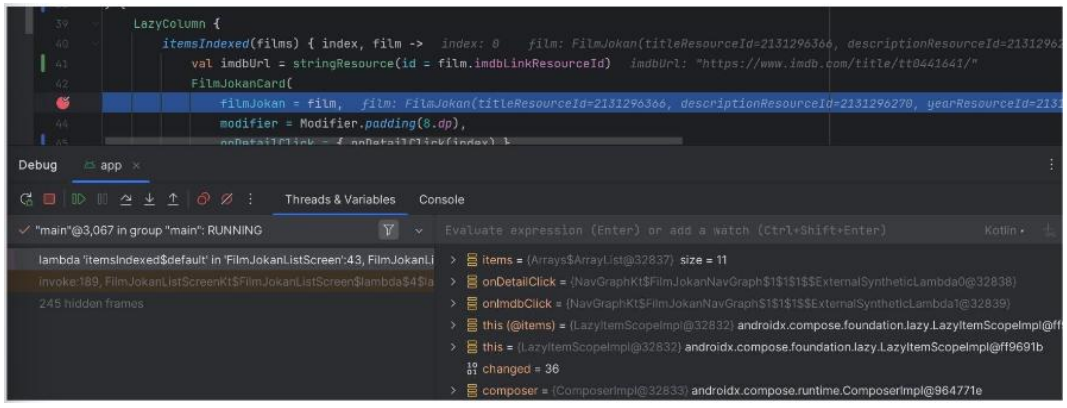
DAFTAR TABEL

SOAL 1

1. Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:
   a. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
   b. Gunakan ViewModelFactory dalam pembuatan ViewModel.
   c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment.
   d. 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
   e. 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
2. Jelaskan Apllication class dalam arsitektur aplikasi Android dan fungsinya

Aplikasi harus dapat mempertahankan fitur-fitur yang sudah dibuat pada modul sebelumnya. Berikut adalah contoh debugging dalam Android Studio.



A. Source Code

1. MainActivity.kt

Tabel 1. Source Code Jawaban Soal 1

```
1  package com.example.scrollablelist
2
3  import android.os.Bundle
4  import android.util.Log
5  import androidx.activity.enableEdgeToEdge
6  import androidx.appcompat.app.AppCompatActivity
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.jav
27 a.simpleName)
28         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()
        }
    }
}
```

2. Song.kt

Tabel 2. Source Code Jawaban Soal 1

```
package com.example.scrollablelist

import android.icu.text.CaseMap.Title
import android.os.Parcelable
import kotlinx.parcelize.Parcelize

@Parcelize
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
```

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.home
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.view.LayoutInflater
```

```
7  import android.view.View
8  import android.view.ViewGroup
9  import androidx.fragment.app.Fragment
10 import androidx.fragment.app.viewModels
11 import androidx.lifecycle.lifecycleScope
12 import androidx.recyclerview.widget.LinearLayoutManager
13 import com.example.scrollablelist.R
14 import
15 com.example.scrollablelist.adapter.ListSongAdapter
16 import
17 com.example.scrollablelist.databinding.FragmentHomeBindi
18 ng
19 import com.example.scrollablelist.detail.DetailFragment
20 import com.example.scrollablelist.model.Song
21 import
22 com.example.scrollablelist.utils.HomeViewModelFactory
23 import kotlinx.coroutines.flow.collectLatest
24 import kotlinx.coroutines.launch
25 import android.util.Log
26
27 class HomeFragment : Fragment() {
28     private var _binding: FragmentHomeBinding? = null
29     private val binding get() = _binding!!
30
31     private lateinit var songAdapter: ListSongAdapter
32
33     private val viewModel: HomeViewModel by viewModels {
34         HomeViewModelFactory(resources)
35     }
36
37     override fun onCreateView(
```

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

        return binding.root
    }

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

        setupRecyclerView()
        observeSongList()

        viewModel.loadSongs()
    }

    private fun setupRecyclerView() {
        songAdapter = ListSongAdapter(
        ArrayList(),
            onSpotifyClick = { link ->
                Log.d("HomeFragment", "Spotify button
clicked for link: $link")
                Log.e("Intent to Spotify", "Going to
$link")
                val intent = Intent(Intent.ACTION_VIEW,
Uri.parse(link))
                startActivity(intent)
```

```
            },
            onDetailClick = { title, photo, singer,
album, lyrics ->
                Log.d(
                    "HomeFragment",
                    "Detail button clicked - Title:
$title, Singer: $singer, Album: $album"
                )
                val detailFragment =
DetailFragment().apply {
                    arguments = Bundle().apply {
                        putString("TITLE", title)
                        putString("PHOTO", photo)
                        putString("SINGER", singer)
                        putString("ALBUM", album)
                        putString("LYRICS", lyrics)
                    }
                }

                parentFragmentManager.beginTransaction()
                    .replace(R.id.main, detailFragment)
                    .addToBackStack(null)
                    .commit()
            }
        )

        binding.rvSong.apply {
            layoutManager = LinearLayoutManager(context)
            adapter = songAdapter
            setHasFixedSize(true)
        }
```

```
100     }
101
102     private fun observeSongList() {
103         viewLifecycleOwner.lifecycleScope.launch {
104             viewModel.songList.collectLatest { list ->
105                 songAdapter.setData(list)
106             }
107         }
108     }
109
110     override fun onDestroyView() {
111         super.onDestroyView()
112         _binding = null
113     }
114 }
```

5. HomeViewModel.kt

Tabel 5. Source Code Jawaban Soal 1

```
1   package com.example.scrollablelist.home
2
3   import android.content.res.Resources
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.viewModelScope
6   import com.example.scrollablelist.R
7   import com.example.scrollablelist.model.Song
8   import kotlinx.coroutines.flow.Flow
9   import kotlinx.coroutines.flow.MutableStateFlow
10  import kotlinx.coroutines.flow.StateFlow
11  import kotlinx.coroutines.flow.flow
12  import kotlinx.coroutines.flow.onStart
13  import kotlinx.coroutines.launch
```

```
14  import android.util.Log
15
16  class HomeViewModel(private val resources: Resources) :
17  ViewModel() {
18      //MutableStateFlow untuk menyimpan list
19  item(private)
20      private val _songList =
21  MutableStateFlow<List<Song>>(emptyList())
22
23      //public read-only access untuk mengobservasi list
24  item
25      val songList: StateFlow<List<Song>> get() =
26  _songList
27
28      //function yang mengembalikan data list chara
29      private fun getSongFlow(): Flow<List<Song>> = flow {
30          //mengambil data dari Resources
31          val dataTitle =
32  resources.getStringArray(R.array.data_title)
33          val dataLink =
34  resources.getStringArray(R.array.data_link)
35          val dataPhoto =
36  resources.getStringArray(R.array.data_photo)
37          val dataSinger =
38  resources.getStringArray(R.array.data_singer)
39          val dataAlbum =
40  resources.getStringArray(R.array.data_album)
41          val dataLyrics =
42  resources.getStringArray(R.array.data_lyrics)
43          val dataDesc =
44  resources.getStringArray(R.array.data_desc)
```

```
45
46          //membuat list
47          val listSong = ArrayList<Song>()
48          for (i in dataTitle.indices) {
49              val song = Song(
50                  title = dataTitle[i],
51                  link = dataLink[i],
52                  photo = dataPhoto[i],
53                  singer = dataSinger[i],
54                  album = dataAlbum[i],
55                  lyrics = dataLyrics[i],
56                  desc = dataDesc[i]
57              )
58              listSong.add(song)
59          }
60          //mengirim item ke collector
61          emit(listSong)
62      }
63      //function load item
64      fun loadSongs() {
65          viewModelScope.launch {
66              collectSongs()
67          }
68      }
69
70      private suspend fun collectSongs() {
71          getSongFlow()
72              .onStart {
73                  _songList.value = emptyList()
74              }
75              .collect { songs ->
```

```
76                  Log.d("HomeViewModel", "Songs loaded:
77  ${songs.size} items")
78                  _songList.value = songs
79              }
80      }
81
82  }
```

6. ListSongAdapter.kt

Tabel 6. 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
```

```
            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)
            }
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup,
viewType: Int): ListViewHolder {
        val binding =
ItemSongBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
        return ListViewHolder(binding)
    }

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

    override fun onBindViewHolder(holder: ListViewHolder,
position: Int) {
```

```
53          holder.bind(listSong[position])
54      }
55      }
```

7. activity_main.xml

Tabel 7. 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>
```

8. fragment_detail.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=".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
```

```
                android:id="@+id/song_cover"
                android:layout_width="300dp"
                android:layout_height="300dp"
                android:layout_marginTop="8dp"
                android:src="@drawable/place_holder"
                android:layout_gravity="center_horizontal"
                />

            <TextView
                android:id="@+id/song_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"

app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintTop_toBottomOf="@id/song_cover"
                android:layout_marginTop="16dp"
                android:text="TEDDY PICKER"
                android:textStyle="bold"
                android:textSize="20sp"
                android:padding="2dp"
                android:layout_gravity="center_horizontal"
                />

            <TextView
                android:id="@+id/song_singer"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"

app:layout_constraintStart_toStartOf="parent"
```

```
75                  app:layout_constraintEnd_toEndOf="parent"
76
77  app:layout_constraintTop_toBottomOf="@id/song_title"
78                  android:layout_marginTop="16dp"
79                  android:padding="2dp"
80                  android:text="Singer: Arctic Monkeys"
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 app:layout_constraintTop_toBottomOf="@id/song_album"
113                 android:text="Lyrics"
114                 android:textStyle="bold"
115                 android:textSize="14sp"
116                 android:layout_gravity="center_horizontal"
117                 />
118
119             <TextView
120                 android:id="@+id/song_lyrics"
121                 android:layout_width="300dp"
122                 android:layout_height="wrap_content"
123
124 app:layout_constraintStart_toStartOf="parent"
125                 app:layout_constraintEnd_toEndOf="parent"
126                 android:layout_marginTop="8dp"
127
128 app:layout_constraintTop_toBottomOf="@id/song_content"
129                 android:layout_gravity="center_horizontal"
130                 android:lineSpacingMultiplier="1.5"
131                 android:text="llorem ipsum dolor ist
132 amenfdusbgjrabjfbsjhbfrabjrft jfasjfksufheragjrgjjbgj
133 garhgiargjkafjkdgjdfjgdjkbggnajngjd gaigirigoreio
134 igiarigreiogjraei garjgjewoiiwetieri "
135                 />
136         </LinearLayout>
```

```
137      </androidx.core.widget.NestedScrollView>
138
139  </androidx.constraintlayout.widget.ConstraintLayout>
```

9. fragment_home.xml

Tabel 9. 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"
```

```
        android:layout_height="0dp"
        android:layout_marginTop="85dp"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:layout_marginBottom="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:scrollbars="vertical"
        />

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

10. item_song.xml

Tabel 10. Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/androi
d"
    xmlns:card_view="http://schemas.android.com/apk/res-
auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/card_view"
    android:layout_gravity="center"
    android:layout_marginStart="8dp"
    android:layout_marginTop="4dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
```

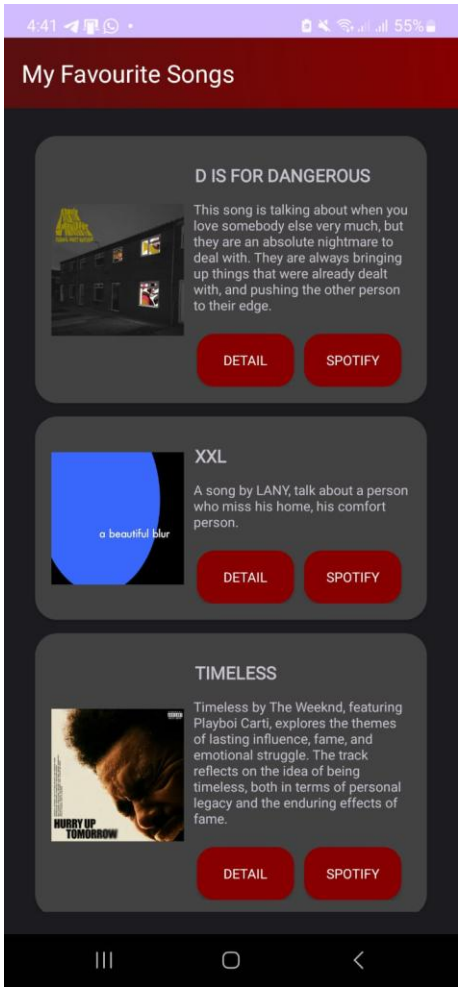
```
16      card_view:cardCornerRadius="20dp"
17      >
18
19      <androidx.constraintlayout.widget.ConstraintLayout
20          android:layout_width="match_parent"
21          android:layout_height="wrap_content"
22          android:padding="15dp">
23
24          <ImageView
25              android:id="@+id/img_cover"
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"
```

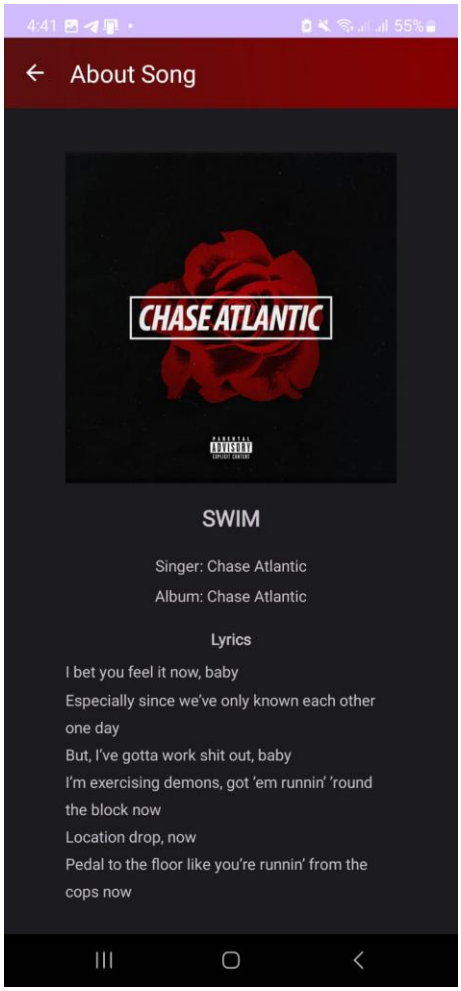
```
78              android:layout_width="wrap_content"
79              android:layout_height="wrap_content"
80              android:layout_marginTop="16dp"
81              android:text="Detail"
82              android:textSize="12dp"
83              android:textColor="@color/white"
84
85  android:background="@drawable/button_background"
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
110 android:background="@drawable/button_background"
111
112 card_view:layout_constraintBottom_toBottomOf="parent"
113
114 card_view:layout_constraintEnd_toEndOf="parent"
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>
```

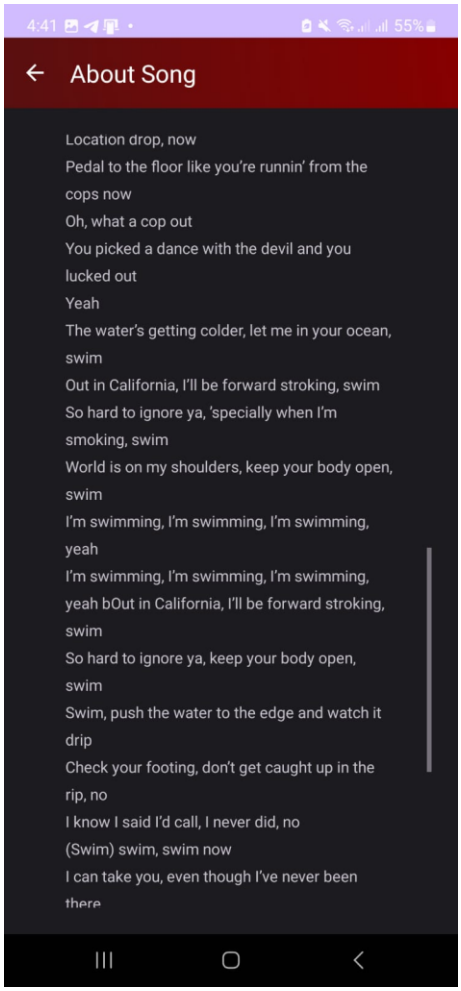
B. Output Program



Gambar 1. Screenshot Hasil Jawaban Soal 1

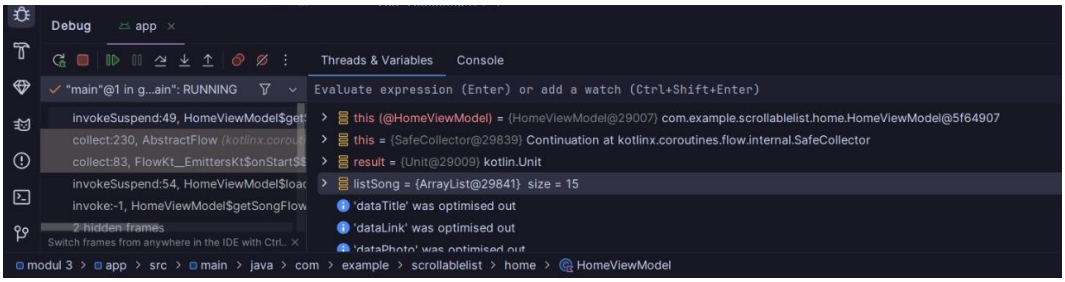


Gambar 2. Screenshot Hasil Jawaban Soal 1



Gambar 3. Screenshot Hasil Jawaban Soal 1

Gambar 4. Screenshot Hasil Jawaban Soal 1



Gambar 5. *Timber Button*

C. Pembahasan

1. MainActivity.kt

File ini merupakan logika utama dari aplikasi untuk menampilkan UI dan mengatur interaksi pengguna. Terdapat beberapa bagian penting, yaitumenambahkan

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 pad 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 mengkases binding secara aman selama fragment aktif.

b) onCreateView()

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

Digunakan untuk meng-inflate layout fragment menggunakan view binding. Lau, `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 utnuk menampilkan gambar dari URL ke dalam `Image View songCover` menggunakan library Glide.

f) Tombol Back di Toolbar

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

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

g) onDestroyView()

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

Digunakan untuk membersigkan 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 `FragmentTransaction`.

g) onDestroyView()

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

Digunakan untuk membersihkan `binding` saat tampilan dihancurkan untuk mencegah memory leak

h) Fungsi Log

- `Log.d("HomeFragment", "Spotify button clicked for link: $link")`

  Berfungsi untuk menampilkan log ketika pengguna menekan tombol `Spotify` dan memastikan bahwa klik pada tombol `Spotify` terdeteksi dan link-nya benar.
- `Log.e("Intent to Spotify", "Going to $link")`

  Berfungsi untuk menampilkan log error.
- `Log.d("HomeFragment", "Detail button clicked - Title: $title, Singer: $singer, Album: $album")`

  Berfungsi unntuk menampilkan informasi ketika pengguna menekan tombol `Detail` dan memastikan data yang dikirim ke `DetailFragment` sudah tepat.

5. HomeViewModel.kt

File ini merupakan kelas `ViewModel` yang berfungsi untuk mengelola dan menyediakan data `Song` ke `HomeFragment`. Di mana data diambil dari resources dan dikirimkan secara asynchronous menggunakan `StateFlow`.

a) `StateFlow`

```
private val _songList =
MutableStateFlow<List<Song>>(emptyList())
val songList: StateFlow<List<Song>> get() = _songList
```

`_songList` sebagai penyimpanan data lagu yang dapat berubah (mutable).

`songList` sebagai versi readd-only yang digunakan fragment untuk mengobservasi perubahan data.

b) Function `getSongFlow()`

```
private fun getSongFlow(): Flow<List<Song>>
```

Digunakan untuk mengambil data dari strings.xml yang berupa array. Kemudian, data dikonversi menjadi list `Song` dan dikirim menggunakan `emit()` ke flow.

c) Function `loadSongs()`

```
fun loadSongs() {
    viewModelScope.launch {
        collectSongs()
    }
}
```

Digunakan untuk memulai proses pemuatan lagu. Bagian ini menggunakan coroutine `viewModelScope` untuk menjalankan secara lifecycle-aware.

d) Function `collectSongs()`

```
private suspend fun collectSongs()
```

Digunakan untuk emngosongkan data terlebih dahulu (`onStart`) dan lalu mengisi ulang. Kemudian, hasil dari flow ke `_songList` disimpan.

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

e) Fungsi Log

```
Log.d("HomeViewModel", "Songs loaded: ${songs.size}
items")
```

Digunakan untuk menampilkan jumlah lagu yang dimuat ke log, membantu debugging saat data diambil dari resource.

6. ListSonngAdapter.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 dat 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` diguunakan 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 ListViewHolder(binding)
```

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

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

7. 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.

8. 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.

9. 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).

10. 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

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya!

A. Penjelasan

Kelas `android.app.Application` adalah kelas dasar yang berfungsi menjaga status aplikasi secara global. Objek Application diinisialisasi pertama kali ketika proses aplikasi dijalankan, sehingga bersifat singleton sepanjang siklus hidup aplikasi. Dengan demikian kelas ini menyediakan application context yang konstan dan dapat diakses dari komponen mana pun (misal Activity, Service, Receiver) selama aplikasi aktif. Fungsi utamanya adalah bertindak sebagai entry point dan tempat sentral untuk mengelola sumber daya atau data bersama yang dipakai lintas komponen aplikasi, seperti menyimpan variabel global, konfigurasi, atau referensi objek yang dipakai bersama.

Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/Noviana21/Pemrograman-Mobile