LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

VIEWMODEL AND DEBUGGING

Oleh:

Raudatul Sholehah NIM. 2310817220002

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2024

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
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 : Raudatul Sholehah
NIM : 2310817220002

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

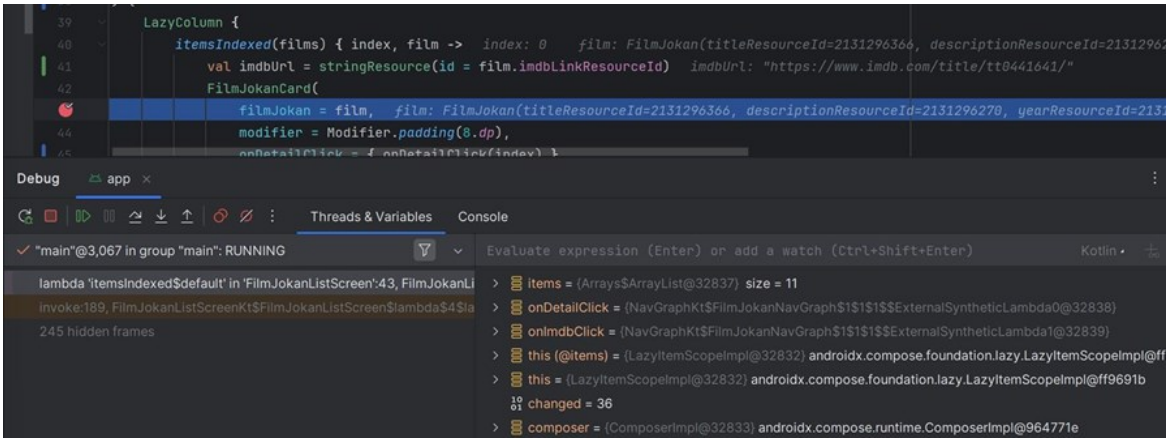
DAFTAR TABEL

SOAL 1

Soal Praktikum:
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

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



Gambar 1. Contoh Penggunaan Debugger

A. Source Code

1. MainActivity.kt

```
1   package com.example.modul4
2
3   import android.os.Bundle
4   import android.util.Log
5   import androidx.activity.ComponentActivity
6   import androidx.activity.compose.setContent
7   import androidx.activity.enableEdgeToEdge
8   import androidx.compose.foundation.layout.fillMaxSize
9   import androidx.compose.foundation.layout.padding
10  import androidx.compose.material3.MaterialTheme
11  import androidx.compose.material3.Scaffold
12  import androidx.compose.runtime.Composable
13  import androidx.compose.ui.Modifier
14  import androidx.navigation.NavHostController
15  import androidx.navigation.compose.NavHost
16  import androidx.navigation.compose.composable
17  import androidx.navigation.compose.rememberNavController
18  import com.example.modul4.ui.screen.MakeupDetailScreen
19  import com.example.modul4.ui.screen.MakeupListScreen
20  import com.example.modul4.ui.theme.Modul4Theme
21
22  class MainActivity : ComponentActivity() {
23      override fun onCreate(savedInstanceState: Bundle?) {
24          super.onCreate(savedInstanceState)
25          enableEdgeToEdge()
26
27          Log.d("MainActivity", "onCreate called - App
    started")
28
29          setContent {
30              Modul4Theme {
31                  MainApp()
32              }
33          }
34      }
35  }
36
37  @Composable
38  fun MainApp() {
39      val navController: NavHostController =
    rememberNavController()
40
41      Scaffold(
42          modifier = Modifier.fillMaxSize(),
43          containerColor = MaterialTheme.colorScheme.background
44      ) { padding ->
45          NavHost(
46              navController = navController,
47              startDestination = Navigation.ROUTE_LIST,
```

```
48              modifier = Modifier.padding(padding)
49          ) {
50              composable(Navigation.ROUTE_LIST) {
51                  Log.d("Navigation", "Navigated to
    MakeupListScreen")
52                  MakeupListScreen(navController =
    navController)
53              }
54              composable(Navigation.ROUTE_DETAIL) {
    backStackEntry ->
55                  val makeupId =
    backStackEntry.arguments?.getString(Navigation.ARG_MAKEUP_ID)
    ?: ""
56                  Log.d("Navigation", "Navigated to
    MakeupDetailScreen with ID: $makeupId")
57                  MakeupDetailScreen(navController =
    navController, makeupId = makeupId)
58              }
59          }
60      }
61  }
```

Tabel 1. Source Code MainActivity.kt Jawaban Soal 1

2. Makeup.kt

```
1   package com.example.modul4.model
2
3   import com.example.modul4.R
4
5   data class Makeup(
6       val id: String,
7       val name: String,
8       val type: String,
9       val imageResId: Int,
10      val webUrl: String,
11      val description: String,
12      val year: String
13  ) {
14      companion object {
15          val makeupList = listOf(
16              Makeup(
17                  id = "1",
18                  name = "MakeOver Powerstay 24H Weightless
    Liquid Foundation",
19                  type = "Foundation",
20                  imageResId = R.drawable.makeover_foundation,
21                  webUrl =
    "https://www.sociolla.com/foundation/74784-powerstay-24h-
    weightless-liquid-foundation",
22                  description = "Produk ini memberikan tampilan
    kulit yang lebih halus pada aplikasi pertama dan
    mempertahankan penampilannya sepanjang hari.\n\nTidak hanya
```

```
    memberikan kesan ringan pada kulit saat digunakan di dalam,
    luar ruangan, dan aktivitas aktif, tetapi juga berinovasi
    dengan Oil Regulatory Technology yang menggabungkan mekanisme
    ganda mikropartikel, memajukan formula untuk tidak hanya
    mampu mengendalikan minyak berlebih yang muncul di kulit,
    tetapi juga mengatur produksi minyak dari bawah kulit.
    Memungkinkan untuk tetap diam dengan sempurna hingga 24 jam
    tanpa munculnya sebum/minyak berlebih.",
23                  year = "2023"
24              ),
25              Makeup(
26                  id = "2",
27                  name = "MakeOver Powerstay Glazed Lock Lip
    Pigment",
28                  type = "Lipstik",
29                  imageResId = R.drawable.makeover_lipstik,
30                  webUrl = "https://www.sociolla.com/lip-
    gloss/81512-powerstay-glazed-lock-lip-
    pigment?shade=d09_skye_glaze_%7C_3_g",
31                  description = "Make Over Powerstay Glazed
    Lock Lip Pigment merupakan level terbaru dari lip gloss,
    memberikan hasil bibir plump dan glazy yang uncrackable
    (tampilan tahan lama tanpa cracking) hingga 24 jam.\n\nPlump
    Glaze menghasilkan tampilan bibir halus bahkan ketika
    digunakan di bibir kering, glossy, dan pigmented hanya dalam
    1 olesan, bahkan ketika digunakan di bibir gelap.
    Diformulasikan dengan paten POWERGLAZE TECHNOLOGY™,
    intensitas pigmen produk ini dapat tahan sepanjang hari tanpa
    retak, tahan transfer, dan tahan noda.",
32                  year = "2024"
33              ),
34              Makeup(
35                  id = "3",
36                  name = "Maskara Tahan Air MAKE OVER Lash
    Impulse",
37                  type = "Mata",
38                  imageResId = R.drawable.makeover_maskara,
39                  webUrl =
    "https://www.sociolla.com/mascara/56820-lash-impulse-
    waterproof-mascara",
40                  description = "Make Over Lash Impulse
    Waterproof Mascara 9 ml adalah maskara tahan air dengan 3D
    Maxi-Lash Technology yang menghasilkan volume 10x* pada bulu
    mata anda menjadikannya lebih tebal, lentik, dan
    lebat.\n\nMaskara ini diformulasikan dengan formula zero-
    smudge yang menjaga kinerja maskara ini tidak luntur hingga
    12 jam. Diinovasikan dengan customized dual sided flat-curve
    brush yang secara presisi didesain untuk memberikan hasil
    maksimal tanpa menggumpal.",
41                  year = "2022"
42              ),
43              Makeup(
```

```
                id = "4",
                name = "MakeOver Multifix Matte Blusher",
                type = "Pipi",
                imageResId = R.drawable.makeover_blush,
                webUrl =
"https://www.sociolla.com/blush/10174-multifix-matte-
blusher",
                description = "Semarakkan penampilan Anda
dengan blusher stick transformasi unik ini. Dengan Powder
Shifter Technology, tekstur krimnya langsung berubah menjadi
bedak lembut yang meleleh di kulit.\n\nTeksturnya yang lembut
dan creamy membuat blush ini menyatu sempurna di kulit, namun
hasil akhir bedak lembut memberikan tampilan matte yang
menggoda dan aplikasi yang ringan. Dilengkapi formula Hi-
Impact Pigmented dan Color Diffused Tone Up yang memberikan
rona alami yang intens namun halus yang menghadirkan kesan
modern pada wajah yang lebih cerah.",
                year = "2023"
            ),
            Makeup(
                id = "5",
                name = "MakeOver Hyperblack Superstay Liner 1
g",
                type = "Mata",
                imageResId = R.drawable.makeover_liner,
                webUrl =
"https://www.sociolla.com/eyeliner/9116-hyperblack-superstay-
liner",
                description = "Pena eyeliner dengan warna
intens yang dirancang untuk memberikan hasil akhir yang tegas
dan dramatis.\n\nDilengkapi dengan aplikator berujung kuas
yang lembut, tipis, dan sangat fleksibel, produk ini
memungkinkan pengaplikasian yang presisi tinggi — mulai dari
garis halus natural hingga winged liner yang berani. Formula
cepat kering dan tahan lama memastikan riasan mata tetap
tajam dan tidak luntur sepanjang hari, bahkan di kondisi
lembap atau berminyak. Cocok untuk digunakan sehari-hari
maupun tampilan glamor di acara khusus, eyeliner ini
memberikan sentuhan akhir yang profesional tanpa repot.",
                year = "2023"
            ),
            Makeup(
                id = "6",
                name = "MakeOver Powerstay 24H Matte Powder
Foundation",
                type = "Powder",
                imageResId = R.drawable.makeover_powder,
                webUrl = "https://www.sociolla.com/cake-
foundation/80666-powerstay-24h-matte-powder-foundation",
                description = "Make Over Powerstay 24H Matte
Powder Foundation merupakan bedak padat dengan kandungan
foundation untuk mendapatkan hasil Airbrushed Smooth Cover
```

```
    hingga 24 jam.\n\nPartikel mikro yang sangat halus dapat
    menempel secara sempurna dan menutupi segala permasalahan
    kulit wajah, bahkan pada kulit mudah berjerawat, kulit kering
    dan penggunaan diatas tabir surya. Dapatkan hasil yang tahan
    lama dan nyaman bahkan di kulit yang sangat berminyak dengan
    24H Strong-wear Triple Oil Control.",
68                  year = "2024"
69              ),
70              Makeup(
71                  id = "7",
72                  name = "MakeOver Hydration Serum",
73                  type = "Kulit",
74                  imageResId = R.drawable.makeover_serum,
75                  webUrl = "https://www.sociolla.com/face-
    serum/2314-hydration-serum-33-ml",
76                  description = "Primer wajah ini melembapkan
    dan mempersiapkan kulit sebelum makeup, menjadi rahasia
    tampilan riasan yang tahan lama dan flawless.\n\nDengan
    tekstur ringan yang mudah meresap, produk ini membantu
    menghaluskan kulit dan membuat makeup menempel lebih baik.
    Diperkaya dengan Aloe Vera untuk menenangkan dan melembapkan,
    Pro-Vitamin B5 untuk menjaga kelembapan kulit, serta Vitamin
    E sebagai antioksidan pelindung, primer ini menjadikan kulit
    terasa lebih sehat, kenyal, dan siap untuk riasan yang awet
    sepanjang hari.",
77                  year = "2022"
78              )
79          )
80      }
81  }
```

Tabel 2. Source Code Makeup.kt Jawaban Soal 1

3. Navigation.kt

```
1   package com.example.modul4
2
3   import android.util.Log
4
5   object Navigation {
6       const val ROUTE_LIST = "makeup_list"
7       const val ROUTE_DETAIL = "makeup_detail/{makeupId}"
8
9       const val ARG_MAKEUP_ID = "makeupId"
10
11      fun createDetailRoute(makeupId: String): String {
12          val route = "makeup_detail/$makeupId"
13          Log.d("Navigation", "Created route: $route")
14          return route
15      }
16  }
```

Tabel 3. Source Code Navigation.kt Jawaban Soal 1

4. build.gradle.kts (Module :app)

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose)
    id("org.jetbrains.kotlin.kapt")
}

android {
    namespace = "com.example.modul4"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.modul4"
        minSdk = 30
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner =
"androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-
optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }

    kotlinOptions {
        jvmTarget = "11"
    }

    buildFeatures {
        compose = true
    }

    composeOptions {
        kotlinCompilerExtensionVersion = "1.5.10"
    }
}
```

```
49
50 dependencies {
51     implementation("com.github.bumptech.glide:glide:4.15.1")
52     kapt("com.github.bumptech.glide:compiler:4.15.1")
53     implementation(libs.androidx.core.ktx)
54     implementation(libs.androidx.lifecycle.runtime.ktx)
55     implementation(libs.androidx.activity.compose)
56     implementation(platform(libs.androidx.compose.bom))
57     implementation("androidx.compose.ui:ui")
58     implementation("androidx.compose.ui:ui-graphics")
59     implementation("androidx.compose.ui:ui-tooling-preview")
60     implementation("androidx.compose.material3:material3")
61     implementation("androidx.compose.material:material-icons-
   extended")
62     implementation("androidx.navigation:navigation-
   compose:2.7.7")
63     implementation("androidx.lifecycle:lifecycle-viewmodel-
   compose:2.6.2")
64     implementation("androidx.compose.runtime:runtime-
   saveable")
65     implementation("androidx.fragment:fragment-ktx:1.6.2")
66     testImplementation(libs.junit)
67     androidTestImplementation(libs.androidx.junit)
68     androidTestImplementation(libs.androidx.espresso.core)

69 androidTestImplementation(platform(libs.androidx.compose.bom)
   )
70     androidTestImplementation("androidx.compose.ui:ui-test-
   junit4")
71     debugImplementation("androidx.compose.ui:ui-tooling")
72     debugImplementation("androidx.compose.ui:ui-test-
   manifest")
73 }
```

Tabel 4. Source Code build.gradle.kts (Module :app) Jawaban Soal 1

5. GlideImage.kt

```
1  package com.example.modul4.ui.components
2
3  import android.widget.ImageView
4  import androidx.annotation.DrawableRes
5  import androidx.compose.runtime.Composable
6  import androidx.compose.ui.Modifier
7  import androidx.compose.ui.viewinterop.AndroidView
8  import com.bumptech.glide.Glide
9
10 @Composable
11 fun GlideImageCrop(
12     @DrawableRes resId: Int,
13     contentDescription: String?,
14     modifier: Modifier = Modifier
15 ) {
```

```
16      AndroidView(
17          factory = { context ->
18              ImageView(context).apply {
19                  scaleType = ImageView.ScaleType.CENTER_CROP
20                  contentDescription?.let {
    this.contentDescription = it }
21              }
22          },
23          update = { imageView ->
24              Glide.with(imageView.context)
25                  .load(resId)
26                  .into(imageView)
27          },
28          modifier = modifier
29      )
30  }
31
32  @Composable
33  fun GlideImageFit(
34      @DrawableRes resId: Int,
35      contentDescription: String?,
36      modifier: Modifier = Modifier
37  ) {
38      AndroidView(
39          factory = { context ->
40              ImageView(context).apply {
41                  scaleType = ImageView.ScaleType.FIT_CENTER
42                  contentDescription?.let {
    this.contentDescription = it }
43              }
44          },
45          update = { imageView ->
46              Glide.with(imageView.context)
47                  .load(resId)
48                  .into(imageView)
49          },
50          modifier = modifier
51      )
52  }
```

Tabel 5. Source Code GlideImage.kt Jawaban Soal 1

6. MakeupDetailScreen.kt

```
1   package com.example.modul4.ui.screen
2
3   import android.util.Log
4   import androidx.compose.foundation.layout.*
5   import androidx.compose.foundation.rememberScrollState
6   import androidx.compose.foundation.verticalScroll
7   import androidx.compose.foundation.shape.RoundedCornerShape
8   import androidx.compose.material3.*
9   import androidx.compose.runtime.*
```

```
10  import androidx.compose.ui.Modifier
11  import androidx.compose.ui.draw.clip
12  import androidx.compose.ui.text.style.TextAlign
13  import androidx.compose.ui.unit.dp
14  import
    androidx.lifecycle.compose.collectAsStateWithLifecycle
15  import androidx.lifecycle.viewmodel.compose.viewModel
16  import androidx.navigation.NavController
17  import com.example.modul4.ui.components.GlideImageFit
18  import com.example.modul4.viewmodel.MakeupViewModel
19  import com.example.modul4.viewmodel.MakeupViewModelFactory
20
21  private const val TAG = "MakeupDetailScreen"
22
23  @Composable
24  fun MakeupDetailScreen(
25      navController: NavController,
26      makeupId: String,
27      viewModel: MakeupViewModel = viewModel(factory =
    MakeupViewModelFactory())
28  ) {
29      val makeupList by
    viewModel.makeupList.collectAsStateWithLifecycle()
30      val item = makeupList.find { it.id == makeupId }
31
32      Log.d(TAG, "Opening detail for ID: $makeupId")
33
34      if (item == null) {
35          Log.w(TAG, "Item with ID $makeupId not found")
36          return
37      } else {
38          Log.i(TAG, "Item found: ${item.name}")
39          Log.d(
40              TAG, """
41              Selected item details:
42              ID: ${item.id}
43              Name: ${item.name}
44              Type: ${item.type}
45              Year: ${item.year}
46              URL: ${item.webUrl}
47              Description (short):
    ${item.description.take(30)}...
48          """.trimIndent()
49          )
50      }
51
52      val scrollState = rememberScrollState()
53
54      Scaffold { padding ->
55          Column(
56              modifier = Modifier
57                  .padding(padding)
```

```
                .padding(16.dp)
                .verticalScroll(scrollState)
        ) {
            GlideImageFit(
                resId = item.imageResId,
                contentDescription = item.name,
                modifier = Modifier
                    .fillMaxWidth()
                    .height(220.dp)
                    .clip(RoundedCornerShape(16.dp))
            )

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

            Text(
                text = item.name,
                style =
MaterialTheme.typography.headlineMedium,
                modifier = Modifier.fillMaxWidth(),
                color = MaterialTheme.colorScheme.primary,
                textAlign = TextAlign.Center
            )

            Spacer(modifier = Modifier.height(6.dp))

            Text(
                text = "${item.type} | ${item.year}",
                style = MaterialTheme.typography.bodyMedium,
                modifier = Modifier.fillMaxWidth(),
                textAlign = TextAlign.Center
            )

            Spacer(modifier = Modifier.height(20.dp))

            Text(
                text = "Deskripsi",
                style = MaterialTheme.typography.titleSmall,
                color = MaterialTheme.colorScheme.primary
            )

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

            Text(
                text = item.description,
                style = MaterialTheme.typography.bodyMedium,
                textAlign = TextAlign.Justify
            )

            Spacer(modifier = Modifier.height(32.dp))

            Button(
                onClick = {
```

```
109                     Log.d(TAG, "Back button clicked")
110                     navController.popBackStack()
111                 },
112                 modifier = Modifier.fillMaxWidth()
113             ) {
114                 Text("Kembali", style =
    MaterialTheme.typography.bodyMedium)
115             }
116         }
117     }
118 }
```

Tabel 6. Source Code MakeupDetailScreen.kt Jawaban Soal 1

7. MakeupListScreen.kt

```
1   package com.example.modul4.ui.screen
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.util.Log
6   import androidx.compose.foundation.layout.*
7   import androidx.compose.foundation.lazy.LazyColumn
8   import androidx.compose.foundation.lazy.items
9   import androidx.compose.foundation.shape.RoundedCornerShape
10  import androidx.compose.material3.*
11  import androidx.compose.runtime.*
12  import androidx.compose.ui.Modifier
13  import androidx.compose.ui.draw.clip
14  import androidx.compose.ui.platform.LocalContext
15  import androidx.compose.ui.text.style.TextOverflow
16  import androidx.compose.ui.unit.dp
17  import
    androidx.lifecycle.compose.collectAsStateWithLifecycle
18  import androidx.lifecycle.viewmodel.compose.viewModel
19  import androidx.navigation.NavController
20  import com.example.modul4.Navigation
21  import com.example.modul4.model.Makeup
22  import com.example.modul4.ui.components.GlideImageCrop
23  import com.example.modul4.viewmodel.MakeupViewModel
24  import com.example.modul4.viewmodel.MakeupViewModel.UiEvent
25  import com.example.modul4.viewmodel.MakeupViewModelFactory
26  import kotlinx.coroutines.flow.collectLatest
27
28  private const val TAG = "MakeupListScreen"
29
30  @Composable
31  fun MakeupListScreen(
32      navController: NavController,
33      viewModel: MakeupViewModel = viewModel(factory =
    MakeupViewModelFactory())
34  ) {
35      val makeupList by
```

```
    viewModel.makeupList.collectAsStateWithLifecycle()
36      val context = LocalContext.current
37
38      LaunchedEffect(Unit) {
39          viewModel.eventFlow.collectLatest { event ->
40              when (event) {
41                  is UiEvent.NavigateToDetail -> {
42                      Log.d(TAG, "Navigating to detail screen:
    ${event.makeupId}")

43  navController.navigate(Navigation.createDetailRoute(event.ma
    keupId))
44                  }
45
46                  is UiEvent.OpenWebUrl -> {
47                      Log.d(TAG, "Opening URL: ${event.url}")
48                      val intent = Intent(Intent.ACTION_VIEW,
    Uri.parse(event.url))
49                      context.startActivity(intent)
50                  }
51              }
52          }
53      }
54
55      LazyColumn(
56          contentPadding = PaddingValues(16.dp),
57          verticalArrangement = Arrangement.spacedBy(16.dp)
58      ) {
59          items(makeupList) { item ->
60              MakeupItem(
61                  makeup = item,
62                  onVisitClick = {
    viewModel.onVisitClicked(item.webUrl) },
63                  onDetailClick = {
    viewModel.onDetailClicked(item.id) }
64              )
65          }
66      }
67  }
68
69  @Composable
70  fun MakeupItem(
71      makeup: Makeup,
72      onVisitClick: () -> Unit,
73      onDetailClick: () -> Unit
74  ) {
75      Card(
76          shape = RoundedCornerShape(20.dp),
77          modifier = Modifier.fillMaxWidth(),
78          colors = CardDefaults.cardColors(
79              containerColor =
    MaterialTheme.colorScheme.surface
```

```
80          ),
81          elevation =
    CardDefaults.cardElevation(defaultElevation = 8.dp)
82      ) {
83          Row(
84              modifier = Modifier
85                  .padding(16.dp)
86                  .fillMaxWidth()
87          ) {
88              GlideImageCrop(
89                  resId = makeup.imageResId,
90                  contentDescription = makeup.name,
91                  modifier = Modifier
92                      .size(100.dp, 150.dp)
93                      .clip(RoundedCornerShape(16.dp))
94              )
95
96              Spacer(modifier = Modifier.width(16.dp))
97
98              Column(modifier = Modifier.weight(1f)) {
99                  Text(
100                     text = makeup.name,
101                     style =
    MaterialTheme.typography.titleMedium,
102                     color =
    MaterialTheme.colorScheme.primary,
103                     maxLines = 2
104                 )
105
106                 Spacer(modifier = Modifier.height(4.dp))
107
108                 Text(
109                     text = "${makeup.type} |
    ${makeup.year}",
110                     style =
    MaterialTheme.typography.bodySmall,
111                     color =
    MaterialTheme.colorScheme.primary
112                 )
113
114                 Spacer(modifier = Modifier.height(8.dp))
115
116                 Text(
117                     text = "Deskripsi:
    ${makeup.description}",
118                     style =
    MaterialTheme.typography.bodySmall,
119                     color =
    MaterialTheme.colorScheme.onSurface,
120                     maxLines = 4,
121                     overflow = TextOverflow.Ellipsis
122                 )
```

```
123
124                 Spacer(modifier = Modifier.height(12.dp))
125
126                 Row(
127                     modifier = Modifier.fillMaxWidth(),
128                     horizontalArrangement =
    Arrangement.SpaceEvenly
129                 ) {
130                     Button(
131                         onClick = {
132                             Log.d(TAG, "Tombol Visit
    ditekan: ${makeup.webUrl}")
133                             onVisitClick()
134                         },
135                         colors =
    ButtonDefaults.buttonColors(
136                             containerColor =
    MaterialTheme.colorScheme.primary,
137                             contentColor =
    MaterialTheme.colorScheme.onPrimary
138                         ),
139                         shape = RoundedCornerShape(12.dp),
140                         elevation =
    ButtonDefaults.buttonElevation(6.dp)
141                     ) {
142                         Text("Kunjungi", style =
    MaterialTheme.typography.bodyMedium)
143                     }
144
145                     Button(
146                         onClick = {
147                             Log.d(TAG, "Tombol Detail
    ditekan: ${makeup.id}")
148                             onDetailClick()
149                         },
150                         colors =
    ButtonDefaults.buttonColors(
151                             containerColor =
    MaterialTheme.colorScheme.secondary,
152                             contentColor =
    MaterialTheme.colorScheme.onSecondary
153                         ),
154                         shape = RoundedCornerShape(12.dp),
155                         elevation =
    ButtonDefaults.buttonElevation(6.dp)
156                     ) {
157                         Text("Detail", style =
    MaterialTheme.typography.bodyMedium)
158                     }
159                 }
160             }
161         }
```

```
    }
}
```

Tabel 7. Source Code MakeupListScreen.kt Jawaban Soal 1

8. Color.kt

```
package com.example.modul3.ui.theme

import androidx.compose.ui.graphics.Color

val SoftPink = Color(0xFFFFC1E3)
val LightLilac = Color(0xFFE6DAF5)
val BlushPink = Color(0xFFFFD6E8)

val Rose = Color(0xFFCB0045)
val Plum = Color(0xFF8E24AA)
val Mauve = Color(0xFFCE93D8)
```

Tabel 8. Source Code Color.kt Jawaban Soal 1

9. Theme.kt

```
package com.example.modul3.ui.theme

import android.os.Build
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.graphics.Color

private val DarkColorScheme = darkColorScheme(
    primary = Mauve,
    secondary = LightLilac,
    tertiary = BlushPink,
    background = Color(0xFF2A2A2E),
    surface = Color(0xFF2A2A2E),
    onPrimary = Color.White,
    onSecondary = Color.White,
    onTertiary = Color.White
)

private val LightColorScheme = lightColorScheme(
    primary = Rose,
    secondary = SoftPink,
    tertiary = LightLilac,
    background = Color(0xFFFFF0F5),
    surface = Color(0xFFFFF5F7),
    onPrimary = Color.White,
    onSecondary = Color.Black,
    onTertiary = Color.Black,
    onBackground = Color(0xFF333333),
    onSurface = Color(0xFF333333)
```

```
)

@Composable
fun Modul3Theme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        dynamicColor && Build.VERSION.SDK_INT >=
Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme) dynamicDarkColorScheme(context)
else dynamicLightColorScheme(context)
        }
        darkTheme -> DarkColorScheme
        else -> LightColorScheme
    }

    MaterialTheme(
        colorScheme = colorScheme,
        typography = Typography,
        content = content
    )
}
```

Tabel 9. Source Code Theme.kt Jawaban Soal 1

10. Type.kt

```
package com.example.modul3.ui.theme

import androidx.compose.material3.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

val Typography = Typography(
    bodyLarge = TextStyle(
        fontFamily = FontFamily.SansSerif,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp,
        lineHeight = 24.sp
    ),
    titleLarge = TextStyle(
        fontFamily = FontFamily.SansSerif,
        fontWeight = FontWeight.Bold,
        fontSize = 22.sp,
        lineHeight = 28.sp
    ),
    titleMedium = TextStyle(
        fontFamily = FontFamily.SansSerif,
```

```
        fontWeight = FontWeight.Medium,
        fontSize = 18.sp,
        lineHeight = 24.sp
    ),
    bodyMedium = TextStyle(
        fontFamily = FontFamily.SansSerif,
        fontWeight = FontWeight.Normal,
        fontSize = 14.sp,
        lineHeight = 20.sp
    ),
    bodySmall = TextStyle(
        fontFamily = FontFamily.SansSerif,
        fontWeight = FontWeight.Normal,
        fontSize = 12.sp,
        lineHeight = 16.sp
    )
)
```

Tabel 10. Source Code Type.kt Jawaban Soal 1

11. MakeupRepository.kt

```
package com.example.modul4.repository

import com.example.modul4.model.Makeup

class MakeupRepository {
    fun getMakeupList(): List<Makeup> = Makeup.makeupList
}
```

Tabel 11. Source Code MakeupRepository.kt Jawaban Soal 1

12. MakeupViewModel.kt

```
package com.example.modul4.viewmodel

import android.util.Log
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.modul4.model.Makeup
import com.example.modul4.repository.MakeupRepository
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.SharedFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.launch

private const val TAG = "MakeupViewModel"

class MakeupViewModel(
    private val repository: MakeupRepository
) : ViewModel() {

```

```
21      private val _makeupList =
    MutableStateFlow<List<Makeup>>(emptyList())
22      val makeupList: StateFlow<List<Makeup>> = _makeupList
23
24      private val _eventFlow = MutableSharedFlow<UiEvent>()
25      val eventFlow: SharedFlow<UiEvent> =
    _eventFlow.asSharedFlow()
26
27      init {
28          loadMakeup()
29      }
30
31      private fun loadMakeup() {
32          val list = repository.getMakeupList()
33          list.forEach { makeup ->
34              Log.d(TAG, "Data masuk: ${makeup.name} (ID:
    ${makeup.id})")
35          }
36          _makeupList.value = list
37      }
38
39      fun onDetailClicked(id: String) {
40          Log.d(TAG, "Navigasi ke detail: ID = $id")
41          viewModelScope.launch {
42              _eventFlow.emit(UiEvent.NavigateToDetail(id))
43          }
44      }
45
46      fun onVisitClicked(url: String) {
47          Log.d(TAG, "Tombol Kunjungi ditekan, URL = $url")
48          viewModelScope.launch {
49              _eventFlow.emit(UiEvent.OpenWebUrl(url))
50          }
51      }
52
53      sealed class UiEvent {
54          data class NavigateToDetail(val makeupId: String) :
    UiEvent()
55          data class OpenWebUrl(val url: String) : UiEvent()
56      }
57  }
```

Tabel 12. Source Code MakeupViewModel.kt Jawaban Soal 1

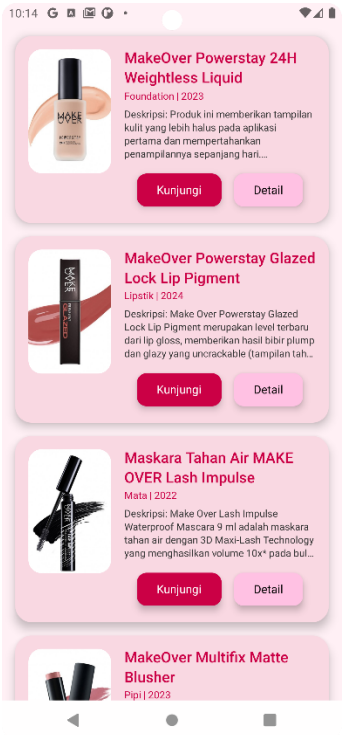
13. MakeupViewModelFactory.kt

```
1   package com.example.modul4.viewmodel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5   import com.example.modul4.repository.MakeupRepository
6
7   class MakeupViewModelFactory : ViewModelProvider.Factory {
```

```
8       override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
9           if
    (modelClass.isAssignableFrom(MakeupViewModel::class.java)) {
10              return MakeupViewModel(MakeupRepository()) as T
11          }
12          throw IllegalArgumentException("Unknown ViewModel
    class")
13      }
14  }
```

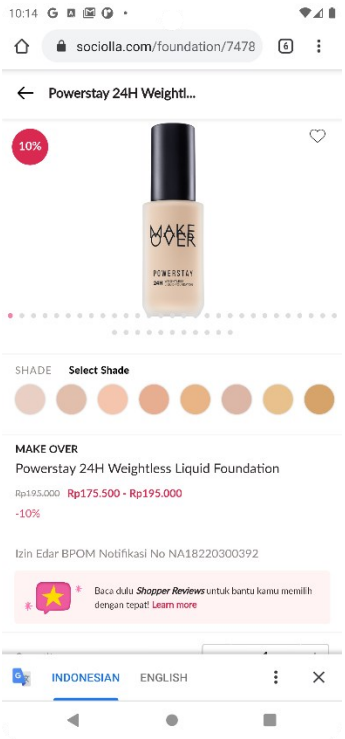
Tabel 13. Source Code MakeupViewModelFactory.kt Jawaban Soal 1

B. Output Program

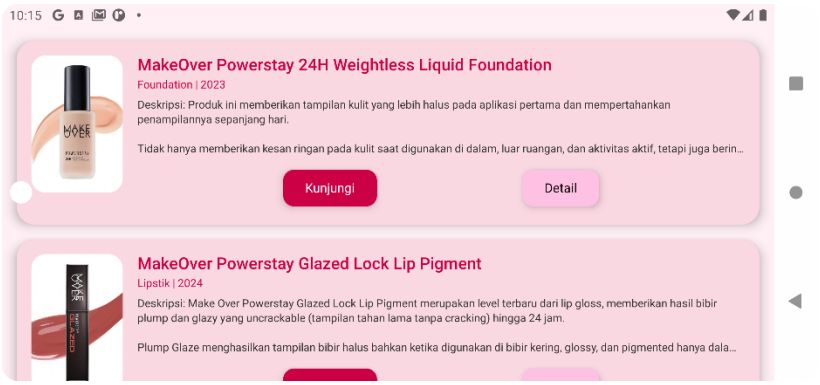


Gambar 2. Screenshot Hasil Tampilan List Jawaban Soal 1

Gambar 3. Screenshot Hasil Tampilan Detail Jawaban Soal 1

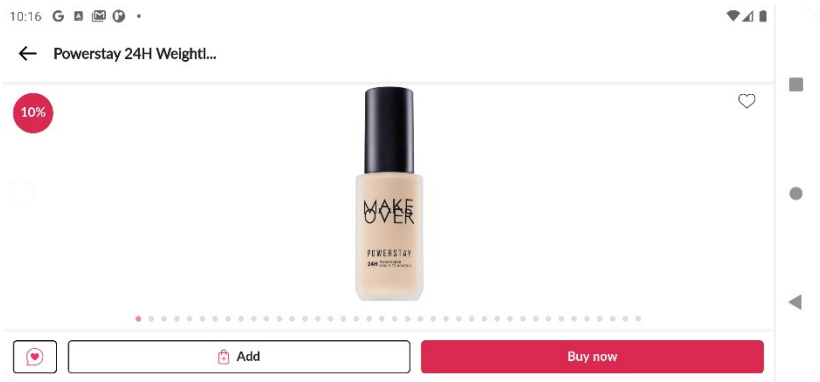


Gambar 4. Screenshot Hasil Tampilan Kunjungi Jawaban Soal 1

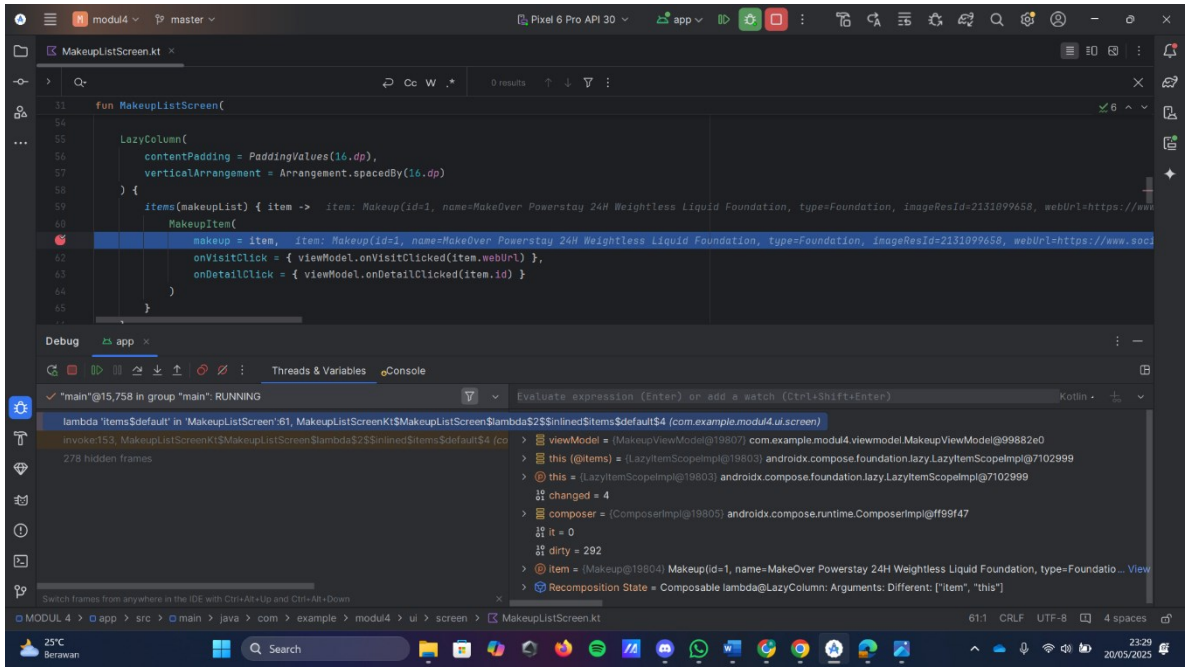


Gambar 5. Screenshot Hasil Tampilan List (landscape) Jawaban Soal 1

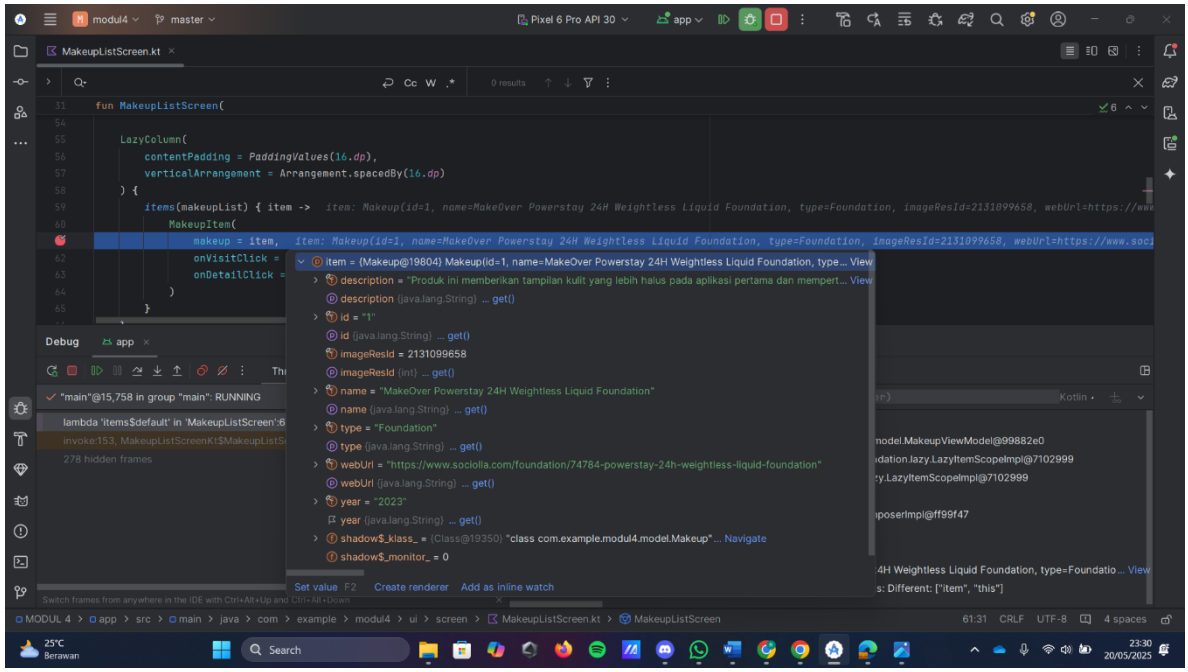
Gambar 6. Screenshot Hasil Tampilan Detail (landscape) Jawaban Soal 1



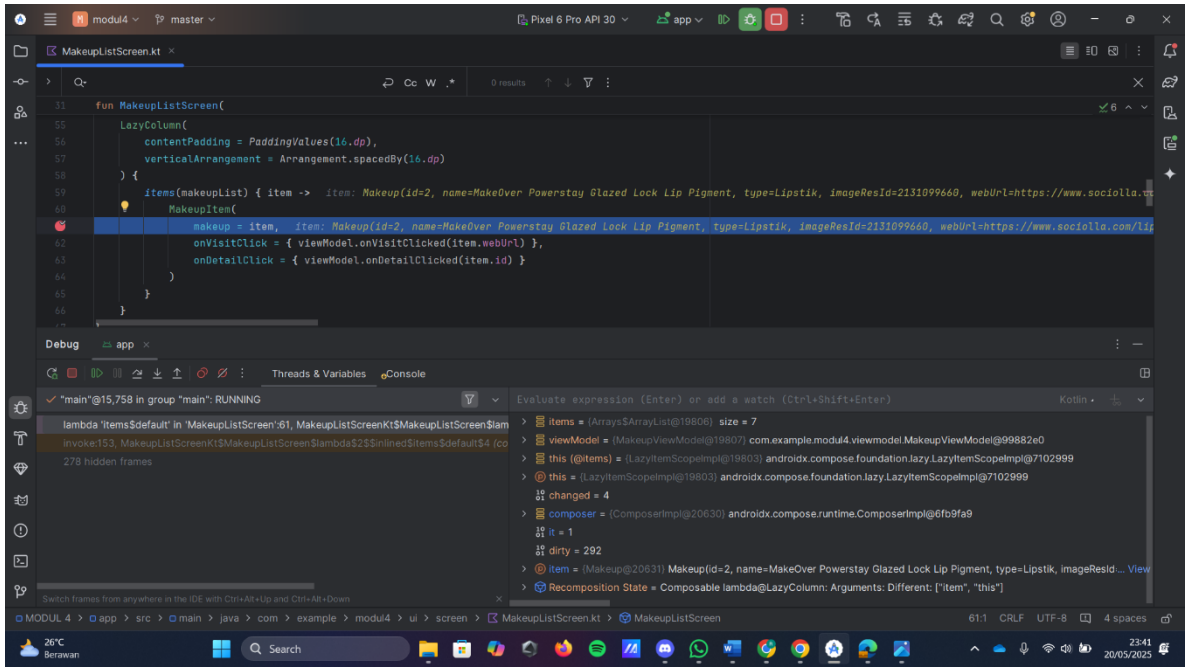
Gambar 7. Screenshot Hasil Tampilan Kunjungi (landscape) Jawaban Soal 1



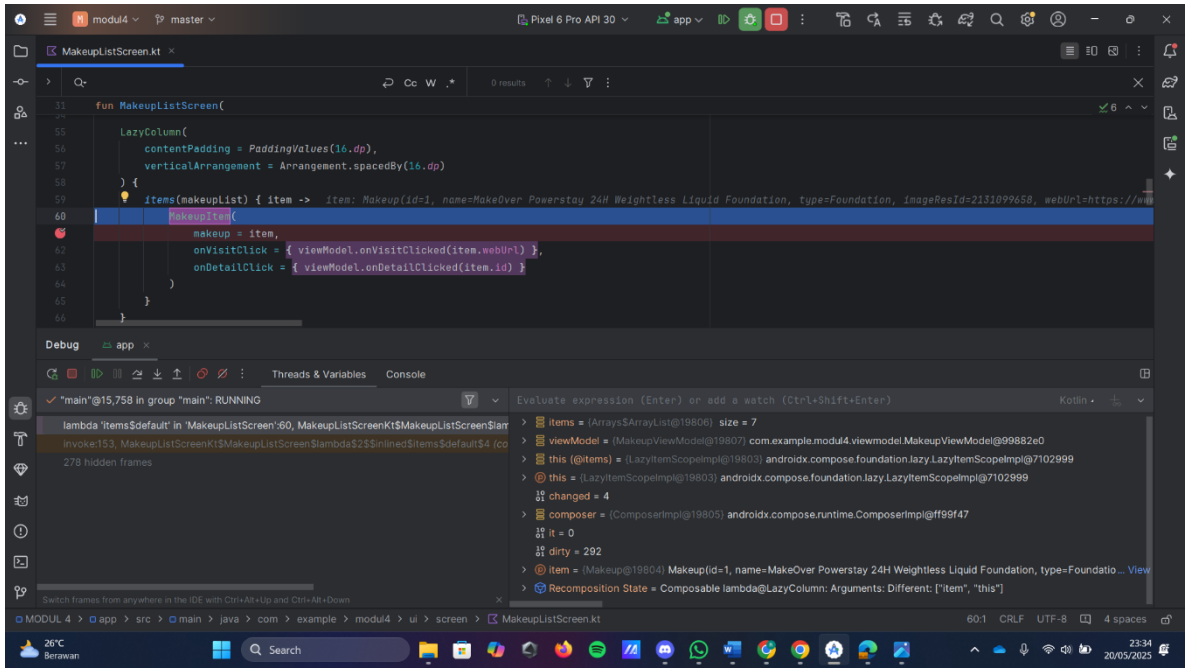
Gambar 8. Screenshot Hasil Tampilan debugging Jawaban Soal 1



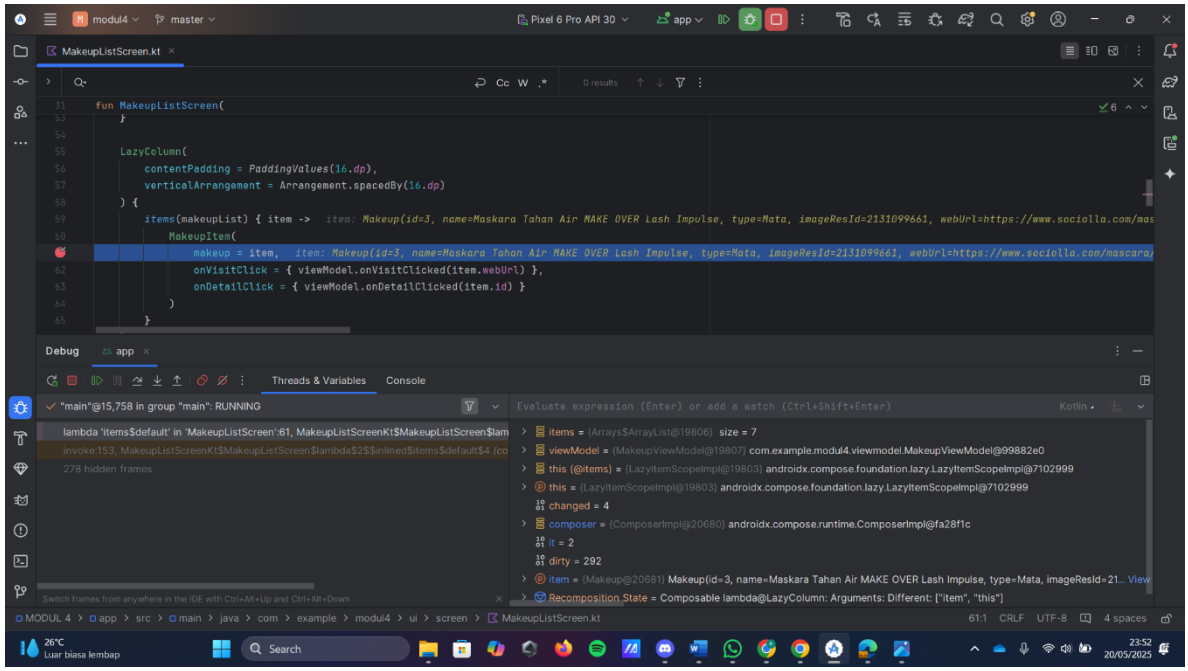
Gambar 9. Screenshot Hasil Tampilan Isi debugging Jawaban Soal 1



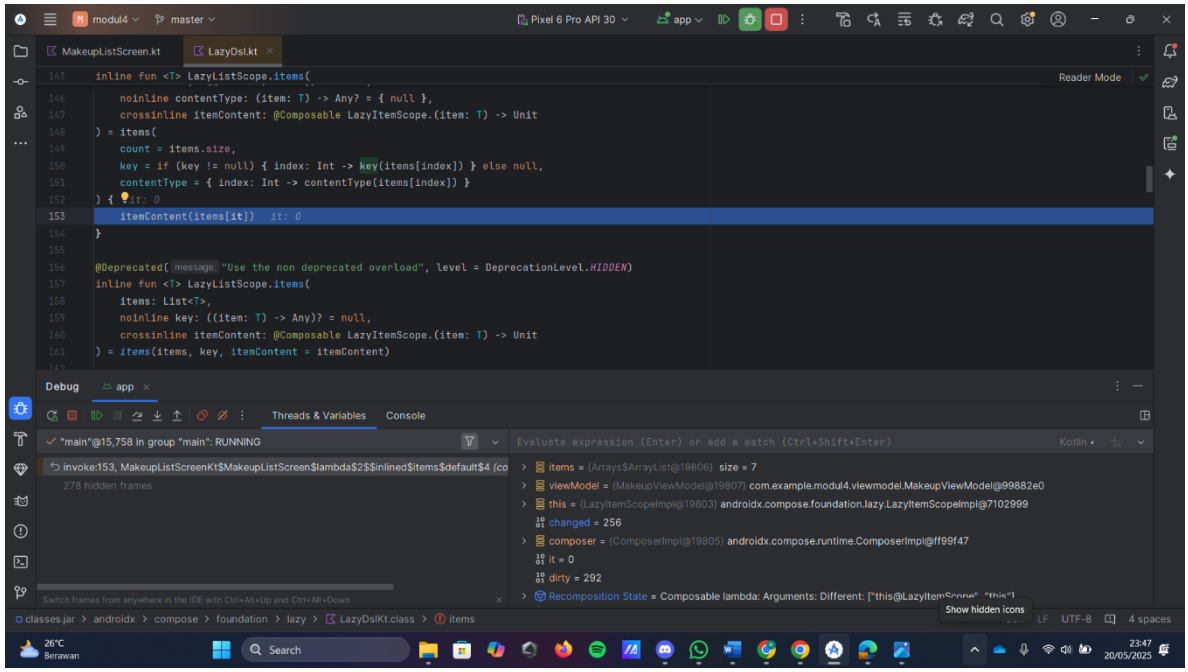
Gambar 10. Screenshot Hasil Tampilan Resume Program Jawaban Soal 1



Gambar 11. Screenshot Hasil Tampilan Step Into Jawaban Soal 1



Gambar 12. Screenshot Hasil Tampilan Step Over Jawaban Soal 1



Gambar 13. Screenshot Hasil Tampilan Step Out Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt

- Baris 1–20:
  Bagian awal kode ini berisi deklarasi package `com.example.modul4`, yang menunjukkan bahwa file ini merupakan bagian dari struktur paket `modul4`. Selanjutnya, terdapat sejumlah import library yang digunakan dalam aplikasi. Library yang diimpor mencakup `Bundle` dan `ComponentActivity` dari AndroidX untuk menangani siklus hidup activity, serta `setContent` untuk menetapkan UI berbasis Jetpack Compose. Fungsi `enableEdgeToEdge` diimpor untuk memungkinkan tampilan layar penuh tanpa batas sisi. Komponen dari Jetpack Compose seperti `Scaffold`, `Modifier`, `MaterialTheme`, `fillMaxSize`, dan `padding` digunakan untuk menyusun dan mengatur tampilan UI. Selain itu, terdapat juga import dari Navigation Compose, seperti `NavHost`, `composable`, dan `rememberNavController`, yang memungkinkan implementasi sistem navigasi antar layar. Dua screen utama aplikasi, yaitu `MakeupListScreen` dan `MakeupDetailScreen`, diimpor dari package `ui.screen`, serta `Modul4Theme` digunakan untuk menerapkan tema aplikasi. Selain itu, `Log` diimpor untuk mencatat log aktivitas selama aplikasi berjalan.
- Baris 22–35:
  Di bagian ini, dideklarasikan kelas `MainActivity` yang merupakan turunan dari `ComponentActivity`. Pada metode `onCreate`, pertama-tama dipanggil `enableEdgeToEdge()` untuk memastikan tampilan aplikasi dapat menjangkau tepi layar secara penuh. Kemudian, `setContent` digunakan untuk menetapkan tampilan UI dengan menggunakan komponen berbasis Compose. Di dalam blok `setContent`, fungsi `Modul4Theme` digunakan untuk membungkus UI dengan tema yang telah ditentukan. Selanjutnya, `MainApp()` dipanggil sebagai composable utama aplikasi yang akan mengelola struktur dan navigasi antar layar. Pada bagian ini juga terdapat log dengan tag "`MainActivity`" yang mencatat pesan "`onCreate called - App started`" ke Logcat sebagai indikator bahwa aplikasi telah dimulai, yang berguna untuk keperluan debugging.
- Baris 37–43:
  Fungsi `MainApp` merupakan composable utama yang menangani sistem navigasi antar layar dalam aplikasi. Pertama-tama, `rememberNavController()` digunakan untuk membuat dan mengingat instance `NavHostController` yang mengelola navigasi. Komponen `Scaffold` digunakan sebagai kerangka tata letak utama, meskipun tidak menggunakan elemen tambahan seperti `topBar` atau `bottomBar`. Modifier `fillMaxSize()` memastikan seluruh layar digunakan sepenuhnya oleh konten, sementara `containerColor` disesuaikan dengan warna latar belakang dari tema aplikasi (`MaterialTheme.colorScheme.background`), menjaga konsistensi tampilan UI.

- Baris 45–49:
  Isi dari `Scaffold` adalah komponen `NavHost`, yang bertugas sebagai pengatur navigasi antar composable. `navController` yang sudah dibuat sebelumnya disisipkan agar `NavHost` dapat mengatur perpindahan antar layar. `startDestination` diatur ke `Navigation.ROUTE_LIST`, yang menandakan bahwa layar pertama yang akan ditampilkan adalah daftar makeup (`MakeupListScreen`). Modifier `Modifier.padding(padding)` digunakan untuk memberikan padding dari `Scaffold`, agar konten tidak bertabrakan dengan elemen sistem seperti status bar atau navigation bar pada perangkat.
- Baris 50–59:
  Pada bagian ini, ditambahkan dua buah `composable` yang merepresentasikan rute navigasi: `ROUTE_LIST` dan `ROUTE_DETAIL`. Untuk `ROUTE_LIST`, saat navigasi diarahkan ke rute ini, akan ditampilkan `MakeupListScreen` dengan parameter `navController`. Di sini juga dicatat log dengan pesan `"Navigated to MakeupListScreen"` untuk menandai bahwa layar daftar telah diakses. Untuk `ROUTE_DETAIL`, layar detail makeup akan ditampilkan. Nilai `makeupId` diambil dari argument yang dibawa saat navigasi, melalui `backStackEntry.arguments`. Jika tidak ada argument yang dikirim, maka digunakan string kosong sebagai nilai default. Setelah itu, `MakeupDetailScreen` ditampilkan dengan makeupId yang diteruskan, dan log dengan pesan `"Navigated to MakeupDetailScreen with ID: $makeupId"` dicetak untuk membantu proses debugging.

2. Makeup.kt

- Baris 1–3:
  Deklarasi package dan import yaitu `package com.example.modul4.model` menunjukkan bahwa file ini berada dalam package `model` dari aplikasi `modul4`. Mengimpor `R` dari `com.example.modul4` untuk mengakses *resource drawable* (gambar) yang digunakan dalam daftar makeup seperti `R.drawable.makeover_foundation`, dll.
- Baris 5–13:
  Pendefinisian data class `Makeup`, yang mewakili satu item produk makeup.
  Properti dalam class ini meliputi:

`id`	:	ID unik dari item makeup (tipe `String`)
`name`	:	Nama produk makeup
`type`	:	Jenis produk (misalnya Foundation, Lipstik, dll)
`imageResId`	:	ID dari resource gambar (tipe `Int`, mengarah ke `R.drawable.*`)
`webUrl`	:	Link ke halaman web marketplace produk
`description`		Deskripsi lengkap produk, digunakan untuk tampilan detail
`year`	:	Tahun rilis produk

Data class ini sangat berguna dalam Compose untuk menampilkan daftar dan detail produk karena mendukung destrukturisasi dan immutability.

- Baris 14–81:
  Deklarasi companion object yang berisi properti `makeupList`, yaitu list statis berisi instance `Makeup`. Ini berfungsi sebagai data dummy (mock data) untuk ditampilkan di aplikasi. Isi `makeupList` berupa 7 objek `Makeup`, masing-masing dengan nilai yaitu ID unik (`id = "1"` sampai `"7"`), Nama produk dan jenisnya (misalnya "Foundation", "Lipstik", dll), Resource gambar (`imageResId`) merujuk ke file drawable, misalnya `makeover_foundation`, `makeover_lipstik`, dll, `webUrl` menuju ke halaman produk di Sociolla, `description` panjang menjelaskan keunggulan dan fitur tiap produk, `year` rilis dari produk tersebut. Daftar ini kemungkinan digunakan dalam `MakeupListScreen` untuk menampilkan daftar produk dan mengirimkan `id` ke `MakeupDetailScreen`.

3. Navigation.kt

- Baris 1–3
  Deklarasi `package com.example.modul4` menunjukkan bahwa file ini berada dalam package utama dari aplikasi `modul4`. File ini mengatur navigasi antar layar di aplikasi menggunakan pendekatan deklaratif. Impor `android.util.Log` digunakan untuk mencetak log pada proses navigasi, terutama saat membentuk rute detail.
- Baris 5:
  Deklarasi `object Navigation` sebagai objek singleton. Objek ini menyimpan semua rute navigasi dan parameter penting yang digunakan di seluruh aplikasi. Pendekatan ini membantu menjaga konsistensi nama rute dan parameter dalam aplikasi Jetpack Compose.
- Baris 6–7:
  `const val ROUTE_LIST = "makeup_list"` merupakan konstanta untuk rute layar daftar makeup `(MakeupListScreen)`. `const val ROUTE_DETAIL = "makeup_detail/{makeupId}"` adalah rute dinamis menuju halaman detail produk, di mana `{makeupId}` merupakan placeholder untuk ID produk tertentu.
- Baris 9:
  `const val ARG_MAKEUP_ID = "makeupId"` menyimpan nama parameter yang digunakan dalam rute dinamis. Ini berguna untuk mengambil data dari `NavBackStackEntry` secara konsisten saat berada di layar detail.
- Baris 11–16:
  Fungsi `createDetailRoute(makeupId: String):` String membentuk dan mengembalikan string rute yang lengkap menuju halaman detail berdasarkan ID produk. Misalnya, jika `makeupId = "3"`, maka akan menghasilkan `makeup_detail/3`. Baris `Log.d("Navigation", "Created route: $route")` mencetak rute yang dibuat ke dalam logcat, dengan tag "Navigation". Ini berguna untuk proses debugging saat pengembangan.

4. build.gradle (Module :app)

- Baris 1–5 (Plugins):
  `alias(libs.plugins.android.application)` adalah plugin untuk aplikasi Android. `alias(libs.plugins.kotlin.android)` adalah plugin untuk menulis kode Android menggunakan Kotlin. `alias(libs.plugins.kotlin.compose)` adalah plugin khusus untuk Jetpack Compose. `id("org.jetbrains.kotlin.kapt")` untuk mengaktifkan `kapt` (Kotlin Annotation Processing Tool), digunakan untuk library Glide.
- Baris 8–48 (Blok android):
  Bagian ini adalah bagian utama konfigurasi Android.
- Baris 9–10 (Namespace dan compile SDK):
  `namespace` dan `applicationId` menentukan identitas unik aplikasi yaitu `"com.example.modul3"`, sedangkan `compileSdk = 35` Proyek yang akan dikompilasi menggunakan Android API level 35.
- Baris 12–20 (defaultConfig):
  Pada bagian ini berfungsi mengkonfigurasi dasar aplikasi diantaranya `applicationId` yaitu nama unik aplikasi (digunakan saat install di perangkat). `minSdk = 30` adalah minimum versi Android yang didukung. `targetSdk = 35` adalah target versi Android saat aplikasi dijalankan. `versionCode` dan `versionName` adalah informasi versi aplikasi. `testInstrumentationRunner` adalah runner default untuk instrumented test.
- Baris 22–30 (buildTypes):
  Bagian ini mengatur tipe build menggunakan `release` yaitu konfigurasi untuk versi rilis aplikasi. `isMinifyEnabled = false` yang artinya tidak mengaktifkan shrinking/proguard. `proguardFiles` untuk menentukan aturan ProGuard yang digunakan jika shrinking diaktifkan.
- Baris 32–35 (compileOptions):
  Bagian ini berfungsi menentukan versi Java yang digunakan untuk kompilasi yaitu `Java 11` agar sesuai dengan Compose dan library modern lainnya.
- Baris 37–39 (kotlinOptions):
  Bagian ini berfungsi untuk mengatur Kotlin agar menggunakan target JVM versi 11, agar kompatibel dengan fitur Java 11.
- Baris 41–43 (buildFeatures):
  `buildFeatures.compose = true` yaitu berfungsi untuk mngaktifkan Jetpack Compose.
- Baris 45–48 (buildFeatures):
  `composeOptions.kotlinCompilerExtensionVersion = "1.5.10"` adalah versi ekstensi compiler untuk Compose yang kompatibel.
- Baris 50–73 (Dependencies):

Adalah daftar library yang digunakan dalam proyek yaitu gambar dan prosesor dengan menggunakan `Glide` untuk memuat gambar, dan `kapt` digunakan untuk proses anotasi Glide. Jetpack dan Compose menggunakan library AndroidX dasar seperti `core.ktx`, `lifecycle.runtime.ktx`, dan `activity.compose`. `platform(libs.androidx.compose.bom)` untuk mengelola versi Compose yang konsisten. `compose.ui, ui-tooling, material3, material-icons-extended`, dll. untuk membuat antarmuka Jetpack Compose. Navigasi dan ViewModel dengan menggunakan `navigation-compose` untuk navigasi antar composable. `lifecycle-viewmodel-compose` untuk ViewModel yang terintegrasi dengan Compose. `runtime-saveable` untuk menyimpan state saat konfigurasi berubah. Fragment KTX untuk interoperabilitas dengan fragment jika dibutuhkan. Testing dengan menggunakan `junit, espresso,` dan `compose.ui:ui-test-junit4` untuk pengujian unit dan UI. `ui-tooling` dan `ui-test-manifest` membantu saat debugging dan pengujian Compose UI.

5. GlideImage.kt

- Baris 1–8:
  `package com.example.modul3.ui.components` menunjukkan lokasi file dalam struktur proyek. Library yang diimpor diantaranya `ImageView` dari Android untuk membuat view gambar. `@DrawableRes` sebagai anotasi agar parameter resId hanya menerima ID dari drawable resource. `@Composable` dari Jetpack Compose untuk membuat fungsi UI. `Modifier` dari Compose UI untuk mengatur layout atau style. `AndroidView` untuk menyisipkan view tradisional Android ke dalam UI berbasis Compose. `Glide` dari library Glide untuk memuat dan menampilkan gambar dengan efisien.
- Baris 10–22:
  Fungsi `GlideImageCrop` adalah fungsi composable yang digunakan untuk menampilkan gambar dengan mode pemotongan (crop). Parameternya yaitu `resId`adalah ID dari resource drawable (misal: `R.drawable.makeover_foundation`). `contentDescription` adalah deskripsi konten untuk keperluan aksesibilitas. `Modifier` adalah parameter opsional untuk pengaturan tampilan tambahan (seperti padding, ukuran, dll). Fungsi menggunakan `AndroidView` untuk menyisipkan `ImageView`. Di dalam `factory`, dibuat objek `ImageView` yang `scaleType`-nya diatur ke `CENTER_CROP`, artinya gambar akan dipotong untuk memenuhi area tampilan. Jika ada `contentDescription`, maka disetel juga.
- Baris 23–30:
  Blok `update` di dalam `AndroidView` memuat gambar menggunakan Glide. Fungsi `Glide.with().load().into()` digunakan untuk memuat gambar

dari resource ID ke dalam `ImageView`. Modifier yang diberikan juga diterapkan untuk menyesuaikan tampilan.

- Baris 32–52:
  Fungsi `GlideImageFit` hampir sama dengan `GlideImageCrop`, tetapi dengan perbedaan utama pada `scaleType`. Di sini `scaleType` disetel ke `FIT_CENTER`, yang artinya gambar akan ditampilkan secara utuh dan dipaskan ke tengah, tanpa dipotong. Sisanya sama dari fungsi `GlideImageCrop` yaitu menerima ID drawable, deskripsi konten, dan modifier opsional, serta menggunakan Glide untuk memuat gambar.

6. MakeupListScreen.kt

- Baris 1–28:
  `package com.example.modul4.ui.screen` menunjukkan lokasi file dalam struktur proyek. Baris import mencakup berbagai komponen penting. `Intent` dan `Uri` digunakan untuk membuka tautan ke browser. `Log` digunakan untuk mencetak log aktivitas ke Logcat. Library dari `androidx.compose.*` digunakan untuk membangun UI seperti `LazyColumn`, `Row`, `Column`, `Card`, `Text`, `Spacer`, dan `Button`. `LocalContext` digunakan untuk mendapatkan context Android dari dalam composable. `Navigation` menyimpan rute navigasi aplikasi. Makeup adalah data model untuk produk. `GlideImageCrop` adalah composable kustom untuk menampilkan gambar. `MakeupViewModel`, `UiEvent`, dan `MakeupViewModelFactory` digunakan dalam arsitektur MVVM. `collectLatest` digunakan untuk menangani aliran event dari ViewModel. Konstanta `TAG` digunakan untuk menandai log.
- Baris 30–36:
  Fungsi `MakeupListScreen` adalah composable utama yang menampilkan daftar produk makeup. `navController` digunakan untuk berpindah antar layar, dan `viewModel` diambil menggunakan `MakeupViewModelFactory`. Data produk diambil dari ViewModel melalui `makeupList` yang diamati dengan `collectAsStateWithLifecycle`. `context` diambil dari `LocalContext` untuk menjalankan `Intent`.
- Baris 28–53:
  Blok `LaunchedEffect` digunakan untuk mengamati aliran event dari ViewModel secara asinkron. Jika event `UiEvent.NavigateToDetail` diterima, maka navigasi ke layar detail makeup dijalankan menggunakan `navController.navigate()` dan route dibuat dengan `Navigation.createDetailRoute()`. Jika event `UiEvent.OpenWebUrl` diterima, maka aplikasi akan membuka browser dengan link dari `makeup.webUrl`.
- Baris 55–66:

Menggunakan `LazyColumn` untuk menampilkan daftar produk makeup dalam tampilan scrollable list. Padding konten diatur sebesar 16 dp dan jarak antar item menggunakan `Arrangement.spacedBy(16.dp)`. Setiap item ditampilkan menggunakan composable `MakeupItem`.

- Baris 69–74:
  Fungsi `MakeupItem` menerima parameter `makeup` sebagai data, `onVisitClick` untuk aksi ketika tombol "Kunjungi" ditekan, dan `onDetailClick` untuk tombol "Detail". Ini memisahkan logika UI dari logika bisnis di ViewModel.
- Baris 75–62:
  Membuat `Card` dengan bentuk sudut membulat `RoundedCornerShape(20.dp)`, lebar penuh `(fillMaxWidth)`, warna latar dari tema, dan bayangan sebesar 8 dp agar tampak timbul.
- Baris 83–87:
  Menggunakan `Row` horizontal untuk menyusun gambar dan informasi makeup secara berdampingan. `Modifier.padding` memberi jarak dalam kartu, dan `fillMaxWidth` membuat Row memenuhi lebar kartu.
- Baris 88–94:
  Menampilkan gambar makeup menggunakan `GlideImageCrop`. Gambar diambil dari resource ID `(makeup.imageResId)`, ukuran diatur 100x150 dp, dan sudut dibulatkan menggunakan `clip`.
- Baris 96:
  `Spacer` digunakan untuk memberi jarak horizontal antara gambar dan teks dengan lebar 16 dp.
- Baris 98–129:
  Menampilkan informasi makeup dalam `Column`. Baris 84–87 menampilkan nama produk dalam dua baris maksimal, dengan warna utama aplikasi. Baris 89–91 menampilkan jenis produk dan tahun rilis. Baris 93–94 menampilkan deskripsi produk maksimal 4 baris, dengan overflow berupa ellipsis jika terlalu panjang.
- Baris 130–159:
  Menambahkan dua tombol aksi di dalam `Row` dengan lebar penuh dan `Arrangement.SpaceEvenly` untuk menyebarkan tombol secara merata. Tombol "Kunjungi" menggunakan warna `primary`, dan saat ditekan akan mencetak log ke Logcat serta memanggil `onVisitClick` untuk membuka tautan web. Tombol "Detail" menggunakan warna `secondary`, dan saat ditekan mencetak log ke Logcat dan menjalankan `onDetailClick` untuk navigasi ke layar detail makeup. Kedua tombol memiliki bentuk membulat `RoundedCornerShape(12.dp)` dan efek bayangan `elevation` sebesar 6 dp.

7. MakeupDetailScreen.kt

- Baris 1–21:
  package `com.example.modul4.ui.screen` menunjukkan lokasi file dalam struktur proyek Android. Import `foundation.layout.*`, `rememberScrollState`, `verticalScroll`, dan `shape.RoundedCornerShape` digunakan untuk mengatur tata letak UI dengan layout kolom yang bisa discroll secara vertikal dan memiliki sudut elemen yang membulat. `material3.*` digunakan untuk membuat elemen UI modern seperti `Text`, `Button`, dan `Scaffold`. `Modifier`, `clip`, dan `dp` digunakan untuk mengatur tampilan elemen seperti ukuran, bentuk, padding, dan layout. `NavController` digunakan untuk navigasi antar layar. `collectAsStateWithLifecycle` dan `viewModel` digunakan untuk mengambil state data dari `MakeupViewModel` secara reaktif dan terikat siklus hidup. `GlideImageFit` adalah fungsi composable kustom untuk menampilkan gambar dari resource ID menggunakan pendekatan `fit center`. Konstanta `TAG` digunakan untuk menandai log
- Baris 23–28:
  Deklarasi fungsi composable `MakeupDetailScreen` yang menerima diantaranya `navController` digunakan untuk berpindah antar layar. `makeupId` ID dari produk makeup yang akan ditampilkan. `viewModel` mengambil data produk dari ViewModel menggunakan `MakeupViewModelFactory`.
- Baris 29–30:
  Mengambil list produk dari `viewModel` menggunakan `collectAsStateWithLifecycle`, lalu mencari produk yang ID-nya sesuai dengan `makeupId`. Jika tidak ditemukan, maka tidak akan ditampilkan.
- Baris 32:
  Mencetak log untuk keperluan debugging saat layar detail dibuka dan ID produk diproses.
- Baris 34–50:
  Jika produk tidak ditemukan, log akan menampilkan peringatan dan fungsi `return` akan menghentikan rendering layar. Jika ditemukan, akan ditampilkan log detail produk seperti nama, jenis, tahun, dan sebagian deskripsi.
- Baris 52:
  `rememberScrollState()` digunakan untuk menyimpan dan mengontrol posisi scroll sehingga konten bisa digulir secara vertikal.
- Baris 54–60:
  Menggunakan `Scaffold` sebagai struktur UI utama, menyediakan padding dan pengaturan layout yang konsisten.
- Baris 61–68:

Menampilkan gambar produk menggunakan `GlideImageFit` dengan Lebar penuh `(fillMaxWidth())`, Tinggi 220 dp Sudut membulat menggunakan `clip(RoundedCornerShape(16.dp))`.

- Baris 72–78:
  Menampilkan nama produk menggunakan `Text` dengan style `headlineMedium`, warna utama dari tema, dan teks diratakan di tengah (`TextAlign.Center`).
- Baris 82–87:
  Menampilkan jenis dan tahun produk makeup, dengan style `bodyMedium`, teks rata tengah.
- Baris 91–103:
  Menampilkan judul "Deskripsi" dan isi deskripsi lengkap dari produk diantaranya Judul menggunakan style `titleSmall` dengan warna tema utama. Deskripsi menggunakan `bodyMedium`, dan teks diratakan kiri-kanan (`TextAlign.Justify`) agar lebih rapi.
- Baris 107–116:
  Menampilkan tombol "Kembali" di bagian bawah layar diantaranya Tombol menggunakan `fillMaxWidth()` agar memenuhi lebar. Saat ditekan, `navController.popBackStack()` dipanggil untuk kembali ke layar sebelumnya. Log dicetak saat tombol ditekan untuk keperluan debugging.

8. Color.kt

File ini berada dalam `package com.example.modul3.ui.theme`, yang digunakan untuk mendefinisikan warna-warna khusus dalam aplikasi berbasis Jetpack Compose. Di dalamnya terdapat beberapa variabel warna yang ditentukan menggunakan objek `Color` dari `androidx.compose.ui.graphics.Color`, dengan format hexadecimal RGB. Warna-warna tersebut diberi nama yang sesuai dengan nuansa atau kesan visual yang ditampilkan, seperti `SoftPink`, `LightLilac`, dan `BlushPink` untuk warna-warna pastel atau lembut, serta `Rose`, `Plum`, dan `Mauve` untuk warna-warna yang lebih kuat dan mencolok. Variabel warna ini biasanya digunakan dalam tema aplikasi untuk menjaga konsistensi tampilan UI, dan memudahkan pengaturan ulang skema warna di seluruh aplikasi hanya dengan mengubah nilainya di satu tempat. File ini berperan penting dalam menciptakan identitas visual yang menarik dan seragam dalam proyek Compose.

9. Theme.kt

Kode ini berada dalam package `com.example.modul3.ui.theme` dan bertugas untuk mengatur tema visual aplikasi dengan Material 3 dan Jetpack Compose. Baris-baris awal (baris 1–6) adalah deklarasi package serta import library Compose seperti `MaterialTheme`, `Color`, dan `isSystemInDarkTheme`,

yang digunakan untuk mendukung pengaturan tema gelap dan terang serta pengambilan konteks sistem.

Selanjutnya, pada baris 8–17, didefinisikan `DarkColorScheme` menggunakan `darkColorScheme()`. Skema ini menetapkan kombinasi warna untuk tema gelap, seperti `primary` yang diisi dengan warna `Mauve`, `secondary` dengan `LightLilac`, dan `tertiary` dengan `BlushPink`. Selain itu, latar belakang (`background`) dan permukaan (`surface`) diberi warna abu gelap, serta warna teks (`onPrimary`, `onSecondary`, dan `onTertiary`) diatur menjadi putih agar kontras dengan latar belakang gelap.

Pada baris 19–30, didefinisikan `LightColorScheme` menggunakan `lightColorScheme()` untuk tema terang. Di sini, warna `primary` menggunakan `Rose`, `secondary` menggunakan `SoftPink`, dan `tertiary` memakai `LightLilac`. Warna latar belakang terang dan permukaan ditetapkan menggunakan nuansa pink lembut, sementara warna teks (`onPrimary`, `onSecondary`, dan seterusnya) diatur agar tetap terbaca di atas latar yang terang.

Kemudian, pada baris 32–43, terdapat fungsi `Modul3Theme()` yang merupakan fungsi `@Composable`. Fungsi ini menentukan tema mana yang akan digunakan, apakah `DarkColorScheme`, `LightColorScheme`, atau tema dinamis dari sistem Android (yang tersedia mulai Android 12 ke atas). Pemilihan ini dilakukan dengan logika kondisi `when`, di mana `LocalContext.current` digunakan untuk mendapatkan konteks sistem saat ini ketika tema dinamis diperlukan.

Terakhir, fungsi `MaterialTheme` digunakan untuk menerapkan `colorScheme` yang telah dipilih, bersama dengan `Typography` yang mewakili gaya teks, serta `content` yang merupakan isi dari tampilan UI. Dengan struktur seperti ini, seluruh aplikasi akan mengikuti tema yang telah ditentukan secara konsisten berdasarkan preferensi pengguna atau sistem.

10. Type.kt

Kode ini berada dalam package `com.example.modul3.ui.theme` dan bertujuan untuk mendefinisikan gaya teks (tipografi) yang digunakan di seluruh aplikasi berbasis Jetpack Compose. Pada baris 1–6, dilakukan import terhadap `Typography` dari Material3, `TextStyle`, `FontFamily`, `FontWeight`, dan satuan ukuran `sp` dari Jetpack Compose UI, yang semuanya diperlukan untuk membangun skema tipografi kustom.

Selanjutnya, pada baris 8 hingga akhir, didefinisikan sebuah objek `Typography` yang merupakan instance dari kelas `Typography`. Di dalam objek ini terdapat beberapa elemen gaya teks yang dideklarasikan secara eksplisit, seperti `bodyLarge`, `titleLarge`, `titleMedium`, `bodyMedium`, dan `bodySmall`. Masing-masing elemen menggunakan `TextStyle` yang berisi konfigurasi seperti jenis huruf (`fontFamily` yang diatur ke `SansSerif`),

ketebalan huruf (`fontWeight` yang dapat berupa `Normal, Medium,` atau `Bold`), ukuran huruf (`fontSize` dalam `sp`), serta tinggi baris (`lineHeight`). Sebagai contoh, `bodyLarge` menggunakan ukuran font 16.sp dan tinggi baris 24.sp dengan berat normal, cocok untuk teks utama. `titleLarge` lebih besar dan tebal, dengan ukuran font 22.sp dan tinggi baris 28.sp, sehingga cocok untuk judul atau heading. Sementara itu, `bodySmall` menggunakan ukuran terkecil, yaitu 12.sp, yang lebih sesuai untuk teks tambahan atau catatan kaki. Semua pengaturan ini memastikan bahwa teks dalam aplikasi memiliki konsistensi visual dan keterbacaan yang baik, mengikuti prinsip desain Material 3.

11. MakeupRepository.kt

- Baris 1–3:
  `package com.example.modul4.repository` menunjukkan bahwa file ini berada dalam package `repository`, yang biasanya digunakan untuk mengelola akses data di arsitektur aplikasi. Import `com.example.modul4.model.Makeup` digunakan untuk mengakses data model `Makeup`, yang berisi informasi produk makeup, seperti nama, jenis, tahun rilis, gambar, dan deskripsi.
- Baris 5:
  Deklarasi class `MakeupRepository` yang berfungsi sebagai lapisan repository dalam arsitektur aplikasi. Repository bertugas mengelola sumber data dan menyediakan data ke ViewModel.
- Baris 6:
  Fungsi `getMakeupList()` akan mengembalikan list statis dari objek `Makeup`. `Makeup.makeupList` diasumsikan sebagai properti statis yang berisi kumpulan data produk makeup yang telah didefinisikan sebelumnya. Fungsi ini tidak mengambil data dari jaringan (API) atau database, melainkan langsung dari list statis yang disimpan di dalam model, sehingga cocok untuk kebutuhan pengembangan awal, demo, atau data dummy.

12. MakeupViewModel.kt

- Baris 1–15:
  Baris ini mendeklarasikan lokasi file `MakeupViewModel.kt` di dalam struktur proyek, yaitu `com.example.modul4.viewmodel`. Baris-baris berikutnya mengimpor pustaka yang dibutuhkan, seperti `ViewModel, viewModelScope` untuk coroutine yang mengikuti siklus hidup ViewModel, serta `MutableStateFlow` dan `SharedFlow` dari `kotlinx.coroutines.flow` untuk pengelolaan data reaktif. `Log` dari Android juga diimpor untuk mencatat aktivitas log yang membantu saat debugging.
- Baris 17–25:

Baris ini mendeklarasikan `MakeupViewModel`, sebuah class yang mewarisi dari `ViewModel`. ViewModel ini menerima parameter `repository` bertipe `MakeupRepository`, yang berfungsi untuk menyediakan data makeup. Di dalamnya, terdapat `MutableStateFlow` bernama `_makeupList` yang menyimpan daftar makeup dan diubah secara internal. Nilai ini diekspos keluar sebagai `StateFlow` bernama `makeupList` agar dapat di-observasi namun tidak bisa dimodifikasi dari luar. Selain itu, `MutableSharedFlow` digunakan untuk mengelola event satu arah yang hanya terjadi sekali, seperti navigasi dan membuka link.

- Baris 27–37:
  Dalam blok `init, fungsi loadMakeup()` langsung dipanggil saat ViewModel diinisialisasi. Fungsi ini mengambil daftar makeup dari repository menggunakan `getMakeupList()` dan mencatat data yang berhasil dimuat dengan `Log.d`, termasuk nama dan ID produk. Setelah itu, data dimasukkan ke dalam `_makeupList`, yang akan diteruskan ke UI.
- Baris 39–51:
  Fungsi `onDetailClicked(id: String)` akan dijalankan saat pengguna memilih produk dari daftar. Fungsi ini mencatat ID yang diklik melalui `Log.d`, kemudian mengirim event `NavigateToDetail` ke `eventFlow` dalam coroutine. Selanjutnya, fungsi `onVisitClicked(url: String)` mencatat URL yang akan dibuka dan mengirim event `OpenWebUrl` ke UI. Kedua fungsi ini dijalankan di dalam coroutine dengan `viewModelScope.launch` agar tidak memblokir thread utama.
- Baris 51–56:
  Bagian ini berisi `sealed class` bernama `UiEvent`, yang mendefinisikan dua jenis event: `NavigateToDetail` dengan parameter `makeupId`, dan `OpenWebUrl` dengan parameter `url`. Struktur ini memungkinkan UI merespons berbagai jenis aksi pengguna dengan aman dan terorganisir.

13. MakeupViewModelFactory.kt

- Baris 1–5:
  Kode ini berada di dalam package `com.example.modul4.viewmodel`, yang menunjukkan bahwa fungsinya berhubungan dengan pengelolaan ViewModel di aplikasi modul 4. Pada bagian ini juga diimpor beberapa class penting seperti `ViewModel, ViewModelProvider,` serta `MakeupRepository` yang digunakan untuk menyuplai data ke ViewModel.
- Baris 7:
  Class `MakeupViewModelFactory` merupakan implementasi dari antarmuka `ViewModelProvider.Factory`. Kelas ini diperlukan untuk membuat instance dari `MakeupViewModel` dengan parameter khusus, yaitu `MakeupRepository`. Hal ini penting karena konstruktor

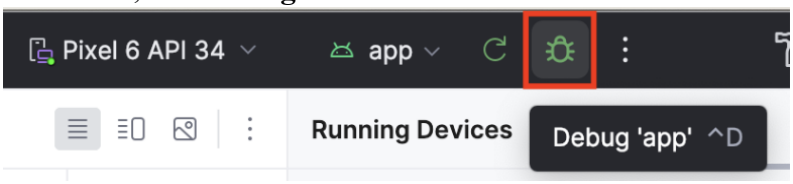
`MakeupViewModel` tidak kosong dan memerlukan repository sebagai dependensi.

- Baris 8–13:
  Fungsi `create()` akan dipanggil oleh sistem saat ViewModel dibutuhkan. Fungsi ini memeriksa apakah kelas ViewModel yang diminta adalah turunan dari `MakeupViewModel` menggunakan `isAssignableFrom()`. Jika ya, maka instance `MakeupViewModel` dibuat dengan menyuntikkan `MakeupRepository` secara langsung dan dikembalikan sebagai tipe `T`. Jika tidak, fungsi ini akan melempar pengecualian `IllegalArgumentException` dengan pesan bahwa ViewModel tidak dikenal.

14. Pembahasan Debugger

1) Fugsi Debugger
   Debugging di Android adalah proses mengidentifikasi, memperbaiki, dan mengatasi masalah dan bug yang terjadi pada perangkat Android. Proses ini melibatkan analisis dan penelusuran kode program untuk menemukan dan memperbaiki kesalahan yang menyebabkan masalah. Debugging di Android memungkinkan pengembang dan pengguna untuk melacak dan memperbaiki masalah seperti crash aplikasi, performa yang lambat, kesalahan logika, dan banyak lagi.
   Android Studio menyediakan debugger yang memungkinkan kita untuk dapat melakukan banyak hal, beberapa contohnya yaitu:
   - Memilih perangkat untuk men-debug aplikasi Anda.
   - Menetapkan titik henti sementara dalam kode Java, Kotlin, dan C/C++.
   - Memeriksa variabel dan mengevaluasi ekspresi pada saat runtime.
   - Mengidentifikasi masalah, Debugging memungkinkan pengguna untuk mengidentifikasi masalah yang terjadi pada perangkat Android mereka.
   - Memperbaiki bug, Debugging membantu pengguna untuk menemukan dan memperbaiki bug dalam kode program.
   - Meningkatkan pengembangan aplikasi, bagi pengembang aplikasi Android dengan melakukan debugging, pengembang dapat menemukan dan memperbaiki masalah sebelum aplikasi dirilis ke publik.
   - Menghemat waktu dan sumber daya, Debugging yang tepat dapat membantu mengidentifikasi masalah dengan cepat dan menemukan solusi yang efisien, menghindari pemborosan waktu dan upaya yang tidak perlu.
2) Cara menggunakan Debugger
   - Mengaktifkan proses debug
     Jika Anda menggunakan emulator, proses debug diaktifkan secara default. Namun, untuk perangkat terhubung, Anda perlu mengaktifkan proses debug dalam opsi developer perangkat. Gunakan varian build yang menyertakan `debuggable true` (`isDebuggable = true` dalam skrip Kotlin) dalam konfigurasi build.

- Memulai proses debug
    1. Tetapkan breakpoint atau titik henti sementara dalam kode aplikasi Anda.
    2. Pada toolbar, pilih perangkat untuk men-debug aplikasi Anda dari menu perangkat target.
    3. Di toolbar, klik **Debug**



    4. Saat breakpoint tercapai, Android Studio akan berhenti dan menampilkan kondisi program.

3) Fitur Step Into (F7)
   Tombol Step Into debugger adalah cara praktis untuk lebih memahami kode saat runtime. Jika sebuah petunjuk membuat panggilan ke metode atau potongan kode lainnya, tombol Step Into memungkinkan Anda memasukkan kode tanpa perlu menavigasi ke sana secara manual sebelum meluncurkan debugger untuk menetapkan titik henti sementara.
4) Fitur Step Over (F8)
   Tombol Step Over menyediakan cara lain untuk menelusuri kode aplikasi saat runtime. Kode ini akan memindahkan eksekusi ke baris kode berikutnya dan mempercepat debugger.
5) Fitur Step Out (Shift+F8)
   Fungsi dari tombol Step Out adalah kebalikan dari tombol Step Into. Tombol Step Out akan mengarahkan ke stack panggilan, bukan melihat rincian stack panggilan. Tombol Step Out adalah alat yang berguna ketika posisi Anda berada terlalu dalam di stack panggilan metode. Dengan bantuan tombol ini, Anda dapat meningkatkan stack panggilan tanpa perlu menelusuri semua kode untuk setiap metode yang Anda masuki.
6) Resume Program (F9)
   Lanjut ke kode selanjutnya atau sampai breakpoint berikutnya. Hal ini menunjukkan cara memeriksa variabel saat runtime dengan debugger.

Referensi:

- Men-debug aplikasi | Android Studio | Android Developers
- Apa Itu Debug di Android? Panduan Lengkap untuk Pemula
- Menggunakan debugger di Android Studio

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/rrdtlsh/Praktikum_Mobile

SOAL 2

Soal Praktikum:
Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

A. Pembahasan

Kelas Application di Android adalah kelas dasar dalam aplikasi Android yang berisi semua komponen lain seperti aktivitas dan layanan. Kelas Aplikasi, atau subkelas apa pun dari kelas Aplikasi, dibuat sebelum kelas lain saat proses untuk dibuat. Kelas ini terutama digunakan untuk inisialisasi status global sebelum status pertama Activity ditampilkan. Dengan membuat subkelas dan menentukan nama lengkap subkelas ini sebagai "android:name" atribut dalam tag AndroidManifest.xml <application>. Kelas Aplikasi, atau subkelas dari kelas Aplikasi, dibuat sebelum kelas lain saat proses untuk aplikasi/paket dibuat.

Fungsi Application class:

1. Manajemen Siklus Hidup Aplikasi
   Kelas Aplikasi memungkinkan mengelola siklus hidup aplikasi Anda. Ini adalah komponen pertama yang dibuat saat aplikasi Anda dimulai, dan ada sepanjang masa pakai aplikasi Anda. Anda dapat mengganti metode seperti `` `onCreate()` ``, `` `onTerminate()` ``, dan `` `onLowMemory()` `` untuk melakukan inisialisasi, pembersihan, dan menangani berbagai peristiwa siklus hidup.
2. Status Aplikasi Global
   Kelas Aplikasi menyediakan lokasi pusat untuk menyimpan dan mengelola status aplikasi global. Anda dapat mendeklarasikan dan mengelola variabel, objek, atau sumber daya yang perlu diakses di beberapa komponen aplikasi Anda, seperti aktivitas, layanan, dan penerima siaran. Ini dapat berguna untuk memelihara data sesi, pengaturan konfigurasi, atau sumber daya bersama.
3. Inisialisasi Seluruh Aplikasi
   Metode `` `onCreate()` `` dari kelas Aplikasi dipanggil saat aplikasi Anda pertama kali dibuat. Ini memberikan kesempatan untuk melakukan tugas inisialisasi yang perlu dijalankan sekali saat aplikasi dimulai, seperti menginisialisasi pustaka pihak ketiga, menyiapkan koneksi basis data, mendaftarkan pendengar peristiwa global, atau mengonfigurasi kerangka kerja pencatatan.
4. Konteks Aplikasi
   Kelas Aplikasi menyediakan referensi ke konteks aplikasi, yang merupakan konteks global yang tetap konstan selama siklus hidup aplikasi. Konteks aplikasi dapat diakses dari mana saja dalam aplikasi Anda, termasuk komponen yang tidak memiliki referensi langsung ke suatu aktivitas atau konteks, seperti layanan latar belakang atau penerima siaran. Ini dapat berguna untuk mengakses sumber daya, memperoleh layanan sistem, atau melakukan operasi yang memerlukan konteks.
5. Kustomisasi dan Ekstensi
   Dengan memperluas kelas Aplikasi, Anda dapat menyesuaikan dan memperluas perilaku kerangka kerja aplikasi Android. Misalnya, Anda dapat mengganti metode seperti `` `onConfigurationChanged()` `` untuk menangani perubahan

konfigurasi secara dinamis, `onTrimMemory()` untuk menangani peristiwa terkait memori, atau menerapkan metode dan logika khusus aplikasi Anda sendiri.

Secara keseluruhan, kelas Aplikasi memainkan peran penting dalam mengelola siklus hidup, status, dan perilaku aplikasi Android Anda. Kelas ini menyediakan titik masuk utama untuk inisialisasi dan kustomisasi, yang memungkinkan Anda untuk mengontrol dan mengelola berbagai aspek perilaku dan sumber daya aplikasi Anda.

B. Referensi:

- https://guides.codepath.com/android/Understanding-the-Android-Application-Class#custom-application-classes
- https://developer.android.com/reference/android/app/Application
- https://medium.com/@sdycode/application-class-in-android-474b55507f6f