LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

BUILD A SCROLLABLE LIST

Oleh:

Raudatul Sholehah NIM. 2310817220002

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2024

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 3

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

Nama Praktikan : 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

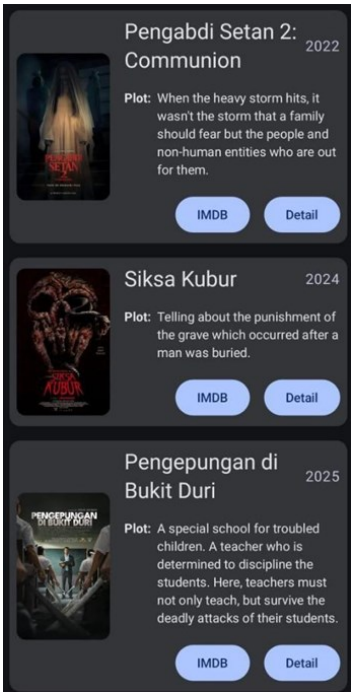
DAFTAR TABEL

SOAL 1

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

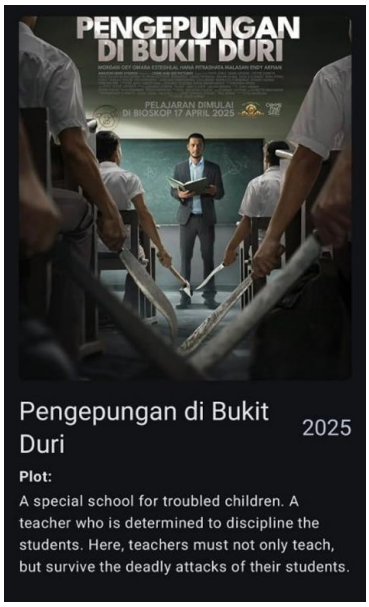
1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas
3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah
4. Terdapat 2 button dalam list, dengan fungsi berikut:
    a. Button pertama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

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



Gambar 1. Contoh UI List

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



Gambar 2. Contoh UI Detail

A. Source Code

1. MainActivity.kt

```
1  package com.example.modul3
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
6  import androidx.activity.enableEdgeToEdge
7  import androidx.compose.foundation.layout.fillMaxSize
8  import androidx.compose.foundation.layout.padding
9  import androidx.compose.material3.MaterialTheme
10 import androidx.compose.material3.Scaffold
11 import androidx.compose.runtime.Composable
12 import androidx.compose.ui.Modifier
13 import androidx.navigation.NavHostController
14 import androidx.navigation.compose.NavHost
15 import androidx.navigation.compose.composable
16 import androidx.navigation.compose.rememberNavController
17 import com.example.modul3.ui.screen.MakeupListScreen
18 import com.example.modul3.ui.screen.MakeupDetailScreen
19 import com.example.modul3.ui.theme.Modul3Theme
20
21 class MainActivity : ComponentActivity() {
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24         enableEdgeToEdge()
25         setContent {
26             Modul3Theme {
27                 MainApp()
28             }
29         }
30     }
31 }
32
33 @Composable
34 fun MainApp() {
35     val navController: NavHostController =
   rememberNavController()
36
37     Scaffold(
38         modifier = Modifier.fillMaxSize(),
39         containerColor = MaterialTheme.colorScheme.background
40     ) { padding ->
41         NavHost(
42             navController = navController,
43             startDestination = Navigation.ROUTE_LIST,
44             modifier = Modifier.padding(padding)
45         ) {
46             composable(Navigation.ROUTE_LIST) {
47                 MakeupListScreen(navController = navController)
48             }
```

```
49              composable(Navigation.ROUTE_DETAIL) {
    backStackEntry ->
50                  val makeupId =
    backStackEntry.arguments?.getString(Navigation.ARG_MAKEUP_ID)
    ?: ""
51                  MakeupDetailScreen(navController =
    navController, makeupId = makeupId)
52              }
53          }
54      }
55  }
```

Tabel 1. Source Code Jawaban Soal 1

2. Makeup.kt

```
1   package com.example.modul3.model
2
3   import com.example.modul3.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. Tidak 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(
```

```
                id = "2",
                name = "MakeOver Powerstay Glazed Lock Lip Pigment",
                type = "Lipstik",
                imageResId = R.drawable.makeover_lipstik,
                webUrl = "https://www.sociolla.com/lip-gloss/81512-powerstay-glazed-lock-lip-pigment?shade=d09_skye_glaze_%7C_3_g",
                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. Plump 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.",
                year = "2024"
            ),
            Makeup(
                id = "3",
                name = "Maskara Tahan Air MAKE OVER Lash Impulse",
                type = "Mata",
                imageResId = R.drawable.makeover_maskara,
                webUrl = "https://www.sociolla.com/mascara/56820-lash-impulse-waterproof-mascara",
                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. Maskara 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.",
                year = "2022"
            ),
            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. Teksturnya 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.",
50                  year = "2023"
51              ),
52              Makeup(
53                  id = "5",
54                  name = "MakeOver Hyperblack Superstay Liner 1
    g",
55                  type = "Mata",
56                  imageResId = R.drawable.makeover_liner,
57                  webUrl =
    "https://www.sociolla.com/eyeliner/9116-hyperblack-superstay-
    liner",
58                  description = "Pena eye liner warna intens
    dengan aplikator ujung kuas yang lembut, tipis, dan fleksibel
    untuk aplikasi yang tepat guna menciptakan tampilan berani dan
    kuat yang tahan lama.",
59                  year = "2023"
60              ),
61              Makeup(
62                  id = "6",
63                  name = "MakeOver Powerstay 24H Matte Powder
    Foundation",
64                  type = "Powder",
65                  imageResId = R.drawable.makeover_powder,
66                  webUrl = "https://www.sociolla.com/cake-
    foundation/80666-powerstay-24h-matte-powder-foundation",
67                  description = "Make Over Powerstay 24H Matte
    Powder Foundation merupakan bedak padat dengan kandungan
    foundation untuk mendapatkan hasil Airbrushed Smooth Cover
    hingga 24 jam. Partikel 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 = "Melembabkan dan mempersiapkan
    kulit untuk riasan, rahasia riasan tahan lama para penata rias
    profesional. Mengandung: - Aloe Vera sebagai agen penenang,
    pelembab dan anti iritasi - Pro Vitamin B5 sebagai pelembab -
```

```
    Vitamin E sebagai antioksidan",
77                  year = "2022"
78              )
79          )
80      }
81  }
```

Tabel 2. Source Code Jawaban Soal 1

3. Navigation.kt

```
1   package com.example.modul3
2
3   object Navigation {
4       const val ROUTE_LIST = "makeup_list"
5       const val ROUTE_DETAIL = "makeup_detail/{makeupId}"
6
7       fun createDetailRoute(makeupId: String): String {
8           return "makeup_detail/$makeupId"
9       }
10
11      const val ARG_MAKEUP_ID = "makeupId"
12  }
```

Tabel 3. Source Code Jawaban Soal 1

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

```
1   plugins {
2       alias(libs.plugins.android.application)
3       alias(libs.plugins.kotlin.android)
4       alias(libs.plugins.kotlin.compose)
5       id("org.jetbrains.kotlin.kapt")
6   }
7
8   android {
9       namespace = "com.example.modul3"
10      compileSdk = 35
11
12      defaultConfig {
13          applicationId = "com.example.modul3"
14          minSdk = 30
15          targetSdk = 35
16          versionCode = 1
17          versionName = "1.0"
18
19          testInstrumentationRunner =
    "androidx.test.runner.AndroidJUnitRunner"
20      }
21
22      buildTypes {
23          release {
24              isMinifyEnabled = false
25              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"
    }
}

dependencies {
    implementation("com.github.bumptech.glide:glide:4.15.1")
    kapt("com.github.bumptech.glide:compiler:4.15.1")
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(platform(libs.androidx.compose.bom))
    implementation("androidx.compose.ui:ui")
    implementation("androidx.compose.ui:ui-graphics")
    implementation("androidx.compose.ui:ui-tooling-preview")
    implementation("androidx.compose.material3:material3")
    implementation("androidx.compose.material:material-icons-extended")
    implementation("androidx.navigation:navigation-compose:2.7.7")
    implementation("androidx.lifecycle:lifecycle-viewmodel-compose:2.6.2")
    implementation("androidx.compose.runtime:runtime-saveable")
    implementation("androidx.fragment:fragment-ktx:1.6.2")
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)

androidTestImplementation(platform(libs.androidx.compose.bom))
    androidTestImplementation("androidx.compose.ui:ui-test-junit4")
    debugImplementation("androidx.compose.ui:ui-tooling")
```

```
72      debugImplementation("androidx.compose.ui:ui-test-manifest")
73  }
```

Tabel 4. Source Code Jawaban Soal 1

5. GlideImage.kt

```
1   package com.example.modul3.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 Jawaban Soal 1

6. MakeupDetailScreen.kt

```
1   package com.example.modul3.ui.screen
2
3   import androidx.compose.foundation.layout.*
4   import androidx.compose.foundation.rememberScrollState
5   import androidx.compose.foundation.verticalScroll
6   import androidx.compose.foundation.shape.RoundedCornerShape
7   import androidx.compose.material3.*
8   import androidx.compose.runtime.Composable
9   import androidx.compose.ui.Modifier
10  import androidx.compose.ui.draw.clip
11  import androidx.compose.ui.unit.dp
12  import androidx.navigation.NavController
13  import com.example.modul3.model.Makeup
14  import com.example.modul3.ui.components.GlideImageFit
15
16  @Composable
17  fun MakeupDetailScreen(
18      navController: NavController,
19      makeupId: String
20  ) {
21      val item = Makeup.makeupList.find { it.id == makeupId } ?:
    return
22      val scrollState = rememberScrollState()
23
24      Scaffold { padding ->
25          Column(
26              modifier = Modifier
27                  .padding(padding)
28                  .padding(16.dp)
29                  .verticalScroll(scrollState)
30          ) {
31              GlideImageFit(
32                  resId = item.imageResId,
33                  contentDescription = item.name,
34                  modifier = Modifier
35                      .fillMaxWidth()
36                      .height(220.dp)
37                      .clip(RoundedCornerShape(16.dp))
38              )
39
```

```
40              Spacer(modifier = Modifier.height(16.dp))
41              Text(text = "Nama: ${item.name}", style =
    MaterialTheme.typography.titleLarge)
42              Text(text = "Jenis: ${item.type}", style =
    MaterialTheme.typography.bodyLarge)
43              Spacer(modifier = Modifier.height(8.dp))
44              Text(text = "Deskripsi:\n${item.description}",
    style = MaterialTheme.typography.bodyMedium)
45              Spacer(modifier = Modifier.height(16.dp))
46              Button(onClick = {
47                  navController.popBackStack()
48              }) {
49                  Text("Kembali", style =
    MaterialTheme.typography.bodyMedium)
50              }
51          }
52      }
```

Tabel 6. Source Code Jawaban Soal 1

7. MakeupListScreen.kt

```
1   package com.example.modul3.ui.screen
2
3   import android.content.Intent
4   import android.net.Uri
5   import androidx.compose.foundation.layout.*
6   import androidx.compose.foundation.lazy.LazyColumn
7   import androidx.compose.foundation.lazy.items
8   import androidx.compose.foundation.shape.RoundedCornerShape
9   import androidx.compose.material3.*
10  import androidx.compose.runtime.Composable
11  import androidx.compose.ui.Modifier
12  import androidx.compose.ui.draw.clip
13  import androidx.compose.ui.platform.LocalContext
14  import androidx.compose.ui.text.style.TextOverflow
15  import androidx.compose.ui.unit.dp
16  import androidx.navigation.NavController
17  import com.example.modul3.Navigation
18  import com.example.modul3.model.Makeup
19  import com.example.modul3.ui.components.GlideImageCrop
20
21  @Composable
22  fun MakeupListScreen(navController: NavController) {
23      val makeupList = Makeup.makeupList
24      LazyColumn(
25          contentPadding = PaddingValues(16.dp),
26          verticalArrangement = Arrangement.spacedBy(16.dp)
27      ) {
28          items(makeupList) { item ->
29              MakeupItem(makeup = item, onDetailClick = {

30  navController.navigate(Navigation.createDetailRoute(item.id))
```

```
31              })
32          }
33      }
34  }
35
36  @Composable
37  fun MakeupItem(
38      makeup: Makeup,
39      onDetailClick: () -> Unit
40  ) {
41      val context = LocalContext.current
42      Card(
43          shape = RoundedCornerShape(20.dp),
44          modifier = Modifier.fillMaxWidth(),
45          colors = CardDefaults.cardColors(
46              containerColor = MaterialTheme.colorScheme.surface
47          ),
48          elevation =
    CardDefaults.cardElevation(defaultElevation = 8.dp)
49      ) {
50          Row(
51              modifier = Modifier
52                  .padding(16.dp)
53                  .fillMaxWidth()
54          ) {
55              GlideImageCrop(
56                  resId = makeup.imageResId,
57                  contentDescription = makeup.name,
58                  modifier = Modifier
59                      .size(100.dp, 150.dp)
60                      .clip(RoundedCornerShape(16.dp))
61              )
62
63              Spacer(modifier = Modifier.width(16.dp))
64
65              Column(
66                  modifier = Modifier.weight(1f)
67              ) {
68                  Row(
69                      modifier = Modifier.fillMaxWidth(),
70                      horizontalArrangement =
    Arrangement.SpaceBetween
71                  ) {
72                      Text(
73                          text = makeup.name,
74                          style =
    MaterialTheme.typography.titleMedium,
75                          color =
    MaterialTheme.colorScheme.primary
76                      )
77                      Text(
78                          text = makeup.year,
```

```
79                          style =
    MaterialTheme.typography.bodySmall,
80                          color =
    MaterialTheme.colorScheme.tertiary
81                      )
82                  }
83
84                  Spacer(modifier = Modifier.height(8.dp))
85
86                  Text(
87                      text = "Deskripsi: ${makeup.description}",
88                      style =
    MaterialTheme.typography.bodySmall,
89                      color =
    MaterialTheme.colorScheme.onSurface,
90                      maxLines = 4,
91                      overflow = TextOverflow.Ellipsis
92                  )
93
94                  Spacer(modifier = Modifier.height(12.dp))
95
96                  Row(
97                      modifier = Modifier.fillMaxWidth(),
98                      horizontalArrangement =
    Arrangement.SpaceEvenly
99                  ) {
100                     Button(
101                         onClick = {
102                             val intent =
    Intent(Intent.ACTION_VIEW, Uri.parse(makeup.webUrl))
103                             context.startActivity(intent)
104                         },
105                         colors = ButtonDefaults.buttonColors(
106                             containerColor =
    MaterialTheme.colorScheme.primary,
107                             contentColor =
    MaterialTheme.colorScheme.onPrimary
108                         ),
109                         shape = RoundedCornerShape(12.dp),
110                         elevation =
    ButtonDefaults.buttonElevation(6.dp)
111                     ) {
112                         Text("Kunjungi", style =
    MaterialTheme.typography.bodyMedium)
113                     }
114                     Button(
115                         onClick = onDetailClick,
116                         colors = ButtonDefaults.buttonColors(
117                             containerColor =
    MaterialTheme.colorScheme.secondary,
118                             contentColor =
    MaterialTheme.colorScheme.onSecondary
```

```
                        ),
                        shape = RoundedCornerShape(12.dp),
                        elevation =
ButtonDefaults.buttonElevation(6.dp)
                    ) {
                        Text("Detail", style =
MaterialTheme.typography.bodyMedium)
                    }
                }
            }
        }
    }
}
```

Tabel 7. Source Code 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 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
)

```

```
21 private val LightColorScheme = lightColorScheme(
22     primary = Rose,
23     secondary = SoftPink,
24     tertiary = LightLilac,
25     background = Color(0xFFFFF0F5),
26     surface = Color(0xFFFFF5F7),
27     onPrimary = Color.White,
28     onSecondary = Color.Black,
29     onTertiary = Color.Black,
30     onBackground = Color(0xFF333333),
31     onSurface = Color(0xFF333333)
32 )
33
34 @Composable
35 fun Modul3Theme(
36     darkTheme: Boolean = isSystemInDarkTheme(),
37     dynamicColor: Boolean = true,
38     content: @Composable () -> Unit
39 ) {
40     val colorScheme = when {
41         dynamicColor && Build.VERSION.SDK_INT >=
    Build.VERSION_CODES.S -> {
42             val context = LocalContext.current
43             if (darkTheme) dynamicDarkColorScheme(context) else
    dynamicLightColorScheme(context)
44         }
45         darkTheme -> DarkColorScheme
46         else -> LightColorScheme
47     }
48
49     MaterialTheme(
50         colorScheme = colorScheme,
51         typography = Typography,
52         content = content
53     )
54 }
```

Tabel 9. Source Code Jawaban Soal 1

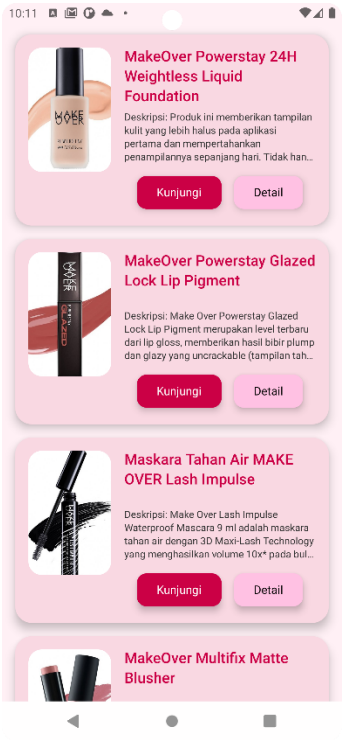
10. Type.kt

```
1  package com.example.modul3.ui.theme
2
3  import androidx.compose.material3.Typography
4  import androidx.compose.ui.text.TextStyle
5  import androidx.compose.ui.text.font.FontFamily
6  import androidx.compose.ui.text.font.FontWeight
7  import androidx.compose.ui.unit.sp
8
9  val Typography = Typography(
10     bodyLarge = TextStyle(
11         fontFamily = FontFamily.SansSerif,
12         fontWeight = FontWeight.Normal,
```

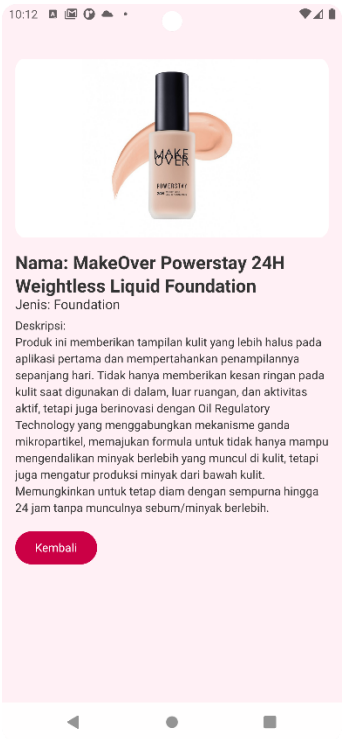
```
13          fontSize = 16.sp,
14          lineHeight = 24.sp
15      ),
16      titleLarge = TextStyle(
17          fontFamily = FontFamily.SansSerif,
18          fontWeight = FontWeight.Bold,
19          fontSize = 22.sp,
20          lineHeight = 28.sp
21      ),
22      titleMedium = TextStyle(
23          fontFamily = FontFamily.SansSerif,
24          fontWeight = FontWeight.Medium,
25          fontSize = 18.sp,
26          lineHeight = 24.sp
27      ),
28      bodyMedium = TextStyle(
29          fontFamily = FontFamily.SansSerif,
30          fontWeight = FontWeight.Normal,
31          fontSize = 14.sp,
32          lineHeight = 20.sp
33      ),
34      bodySmall = TextStyle(
35          fontFamily = FontFamily.SansSerif,
36          fontWeight = FontWeight.Normal,
37          fontSize = 12.sp,
38          lineHeight = 16.sp
39      )
40  )
```

Tabel 10. Source Code Jawaban Soal 1

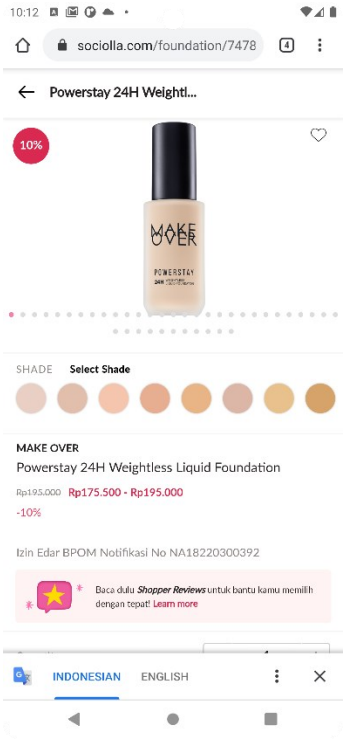
B. Output Program



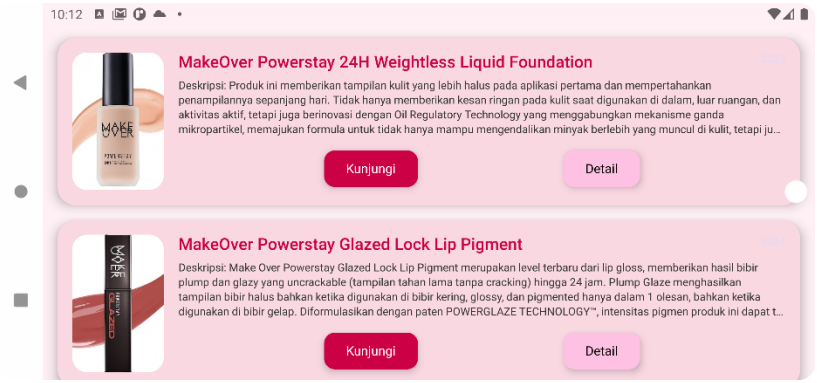
Gambar 3. Screenshot Hasil Jawaban Soal 1



Gambar 4. Screenshot Hasil Jawaban Soal 1

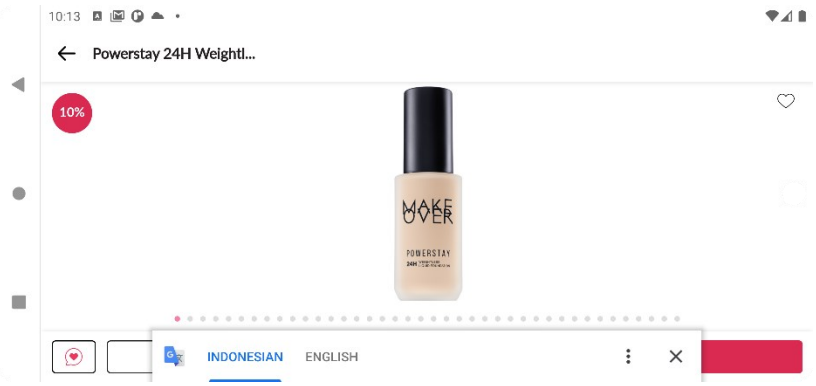


Gambar 5. Screenshot Hasil Jawaban Soal 1



Gambar 6. Screenshot Hasil Jawaban Soal 1

Gambar 7. Screenshot Hasil Jawaban Soal 1



Gambar 8. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt

- Baris 1–19:
  Deklarasi package dan import library yang dibutuhkan. `package com.example.modul3` menunjukkan bahwa file ini berada dalam struktur paket `com.example.modul3`. Library yang diimpor meliputi `Bundle` dan `ComponentActivity` dari AndroidX untuk mendukung lifecycle activity. `setContent` untuk mengisi UI dengan Composable. `enableEdgeToEdge` mengaktifkan tampilan layar penuh. Komponen Jetpack Compose seperti `Scaffold`, `Modifier`, `MaterialTheme`, dan layout control (`padding`, `fillMaxSize`). Komponen Navigation Compose seperti `NavHost`, `composable`, dan `rememberNavController` untuk sistem navigasi antar layar. Import dua screen `MakeupListScreen` dan `MakeupDetailScreen` dari package `ui.screen`. Import `Modul3Theme` untuk menerapkan tema aplikasi.
- Baris 21–27:
  Deklarasi kelas `MainActivity` sebagai turunan dari `ComponentActivity`. Pada method `onCreate`, terdapat Pemanggilan `enableEdgeToEdge()` untuk mengoptimalkan UI agar tampil penuh tanpa batas (edge-to-edge). `setContent { ... }` digunakan untuk menetapkan isi UI dengan menggunakan Composable. Di dalam `setContent`, fungsi `Modul3Theme` digunakan untuk membungkus tampilan UI dengan tema kustom, dan kemudian memanggil fungsi `MainApp()` sebagai root dari composable aplikasi ini.
- Baris 33–39:
  Fungsi `MainApp()` adalah composable utama yang mengatur navigasi aplikasi. `rememberNavController()` digunakan untuk membuat instance controller navigasi yang akan mengatur pergerakan antar layar. `Scaffold()` adalah komponen layout utama dari Material3 Compose yang menyediakan struktur dasar layar (seperti topBar, bottomBar, dsb, meskipun tidak digunakan di sini). Modifier `fillMaxSize()` membuat tampilan memenuhi seluruh layar. `containerColor` menggunakan warna latar belakang dari tema aplikasi.
- Baris 40–45:
  Isi dari `Scaffold` terdiri dari `NavHost`, yang mengatur navigasi antar composable (layar) yaitu `navController = navController` agar navigasi dikontrol oleh `navController` yang sudah dibuat. `startDestination = Navigation.ROUTE_LIST` artinya layar pertama yang ditampilkan adalah daftar makeup (`MakeupListScreen`). `modifier = Modifier.padding(padding)` menerapkan padding dari `Scaffold` ke konten agar tidak tertutupi UI sistem (misalnya status bar).
- Baris 46–51:
  Menambahkan dua `composable` sebagai rute navigasi yaitu diantaranya `ROUTE_LIST` untuk menampilkan layar daftar makeup. Fungsi `MakeupListScreen(navController)` dipanggil agar ketika item dipilih,

dapat melakukan navigasi ke detail. `ROUTE_DETAIL` untuk menampilkan layar detail makeup. Mengambil `makeupId` dari argument navigasi (`backStackEntry.arguments`) dan meneruskannya ke `MakeupDetailScreen`. Jika tidak ada argumen, nilai defaultnya adalah string kosong `""`.

2. Makeup.kt

- Baris 1–3:
  Deklarasi package dan import yaitu `package com.example.modul3.model` menunjukkan bahwa file ini berada dalam package `model` dari aplikasi `modul3`. Mengimpor `R` dari `com.example.modul3` 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 1-2

Deklarasi package dan pembuatan object yaitu `package com.example.modul3` menunjukkan bahwa file ini berada dalam package utama aplikasi `modul3`. `object Navigation` adalah objek singleton di Kotlin yang berisi konstanta dan fungsi navigasi, digunakan untuk mengelola rute navigasi antar layar (screen) di aplikasi Compose.

- Baris 4-5:
  Deklarasi konstanta `ROUTE_LIST` dan `ROUTE_DETAIL` yaitu `const val ROUTE_LIST = "makeup_list"` yang dimana rute ini digunakan untuk menampilkan layar daftar makeup (`MakeupListScreen`). `const val ROUTE_DETAIL = "makeup_detail/{makeupId}"` yang dimana rute ini menunjukkan layar detail makeup (`MakeupDetailScreen`) dan menggunakan parameter `makeupId` sebagai argumen dinamis dalam navigasi.
- Baris 7–9:
  Fungsi `createDetailRoute(makeupId: String): String` yang diaman fungsi ini membuat rute string lengkap ke layar detail makeup berdasarkan `makeupId`. Misalnya, jika `makeupId = "2"`, maka hasilnya adalah `"makeup_detail/2"`. Fungsi ini digunakan saat memanggil navigasi untuk berpindah ke detail dari item tertentu.
- Baris 11:
  Deklarasi konstanta `ARG_MAKEUP_ID` yaitu `const val ARG_MAKEUP_ID = "makeupId"` yang dimana konstanta ini digunakan untuk mengambil parameter `makeupId` dari `NavBackStackEntry` saat berada di layar detail. Ini berguna untuk menyamakan nama parameter yang dikirim dan yang diambil dari rute dinamis.

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 1-19:
  `package com.example.modul3.ui.screen` menunjukkan lokasi file dalam struktur proyek. Deklarasi `package` dan import library yang diperlukan yaitu `Intent` dan `Uri` digunakan untuk membuka link ke browser. Library dari `androidx.compose.*` digunakan untuk membangun UI seperti `Row`, `Column`, `LazyColumn`, dan elemen material seperti `Card`, `Text`, dan `Button`. `LocalContext` digunakan untuk mengakses `Context` Android dari dalam composable. `Navigation` adalah objek yang berisi rute navigasi.

`Makeup` adalah model data makeup. `GlideImageCrop` adalah composable khusus untuk menampilkan gambar dengan efek crop menggunakan Glide.

- Baris 21-34:
  Fungsi `MakeupListScreen` merupakan composable utama untuk menampilkan daftar produk makeup dalam bentuk scrollable list (menggunakan `LazyColumn`). `makeupList` mengambil data statis dari `Makeup.makeupList`. `LazyColumn` menampilkan setiap item makeup menggunakan fungsi `MakeupItem`. Setiap item memiliki aksi tombol "Detail" yang akan menavigasi ke halaman detail dengan membawa `makeup.id` menggunakan `Navigation.createDetailRoute()`.
- Baris 36-40:
  Fungsi `MakeupItem` adalah composable untuk menampilkan 1 kartu produk makeup. Parameter `makeup` adalah data makeup yang akan ditampilkan. Parameter `onDetailClick` adalah lambda function yang dipanggil saat tombol "Detail" ditekan.
- Baris 41:
  `context` diambil dari `LocalContext.current` agar bisa dipakai membuka link web via `Intent`.
- Baris 42-49:
  Membuat `Card` sebagai wadah utama dari satu item produk. `RoundedCornerShape(20.dp)` memberikan efek sudut membulat. `fillMaxWidth()` agar lebar sesuai layar. `cardColors` dan `cardElevation` mengatur warna dan bayangan.
- Baris 50-54:
  Menambahkan `Row` horizontal di dalam `Card` untuk menampilkan gambar dan detail di sampingnya.
- Baris 55-61:
  Menampilkan gambar makeup menggunakan `GlideImageCrop` dari resource ID. Ukuran ditentukan 100x150 dp dan dipotong dengan bentuk rounded (clip).
- Baris 63:
  Menambahkan spasi horizontal antar gambar dan teks.
- Baris 65-94:
  Baris 67-71 menampilkan teks detail produk dalam bentuk kolom menggunakan `Row(`. Baris 73-82 menampilkan nama produk dan tahun rilis menggunakan `Text(`, diatur dalam satu `Row` agar sejajar kanan-kiri. Baris 86-92 menampilkan deskripsi produk dengan maksimal 4 baris dan menggunakan `TextOverflow.Ellipsis` agar terpotong jika lebih panjang. Baris 94 berfungsi memberi jarak antara deskripsi dan tombol.
- Baris 96-124:
  Menampilkan dua tombol (`Kunjungi` dan `Detail`) secara horizontal dan merata (`Arrangement.SpaceEvenly`). Tombol "Kunjungi" saat ditekan,

membuka link `makeup.webUrl` menggunakan `Intent` ke browser. Warna tombol menggunakan `MaterialTheme.colorScheme.primary`. Tombol "Detail" saat ditekan, menjalankan `onDetailClick` untuk navigasi ke layar detail makeup. Warna tombol menggunakan warna sekunder.

7. MakeupDetailScreen.kt

- Baris 1-14:
  `package com.example.modul3.ui.screen` menunjukkan lokasi file dalam struktur proyek. Deklarasi `package` dan import library yang diperlukan untuk membuat layar detail produk makeup. `foundation.layout.*`, `rememberScrollState`, `verticalScroll` digunakan untuk mengatur layout vertikal dan memberikan kemampuan scroll pada layar. `shape.RoundedCornerShape` digunakan untuk memberikan bentuk sudut membulat pada gambar. `material3.*` dipakai untuk elemen UI modern seperti `Text`, `Button`, `Scaffold`, dan tema. `Composable` dari Jetpack Compose menandai fungsi UI yang dapat dirender. `Modifier`, `clip`, dan `dp` digunakan untuk mengatur ukuran dan bentuk elemen UI. `NavController` dari Navigation Compose digunakan untuk menangani perpindahan layar. `Makeup` adalah model data yang berisi informasi produk makeup. `GlideImageFit` adalah fungsi composable kustom yang menampilkan gambar dengan skala `FIT_CENTER`.
- Baris 16-20:
  Deklarasi fungsi composable utama `MakeupDetailScreen`, menerima dua parameter yaitu `navController` untuk mengontrol navigasi antar layar. `makeupId` ID produk makeup yang akan ditampilkan detailnya.
- Baris 21:
  Berfungsi mencari data makeup dari list statis `Makeup.makeupList` berdasarkan `makeupId`. Jika tidak ditemukan, langsung `return` untuk mencegah error.
- Baris 22:
  Menyimpan posisi scroll agar UI dapat digulir ke atas/bawah secara vertikal.
- Baris 24:
  `Scaffold` menyediakan struktur dasar UI seperti top bar, floating action button, dan content area dengan padding otomatis. Parameter `padding` akan diteruskan ke `Column` di dalamnya.
- Baris 31-38:
  Menampilkan gambar produk dengan `resId` dari model `Makeup`. Ukuran gambar memenuhi lebar layar `(fillMaxWidth())`, tinggi 220dp, dan sudut membulat 16dp. Fungsi `GlideImageFit` memanfaatkan library Glide untuk memuat gambar berbasis `resource ID`.

- Baris 41-42:
  Menampilkan informasi produk diantaranya `Text(text = "Nama: ...")` untuk menampilkan nama produk. `Text(text = "Jenis: ...")` untuk menampilkan jenis/merek dari makeup tersebut. Keduanya menggunakan `MaterialTheme.typography` untuk konsistensi gaya teks.
- Baris 43-45:
  Menampilkan deskripsi lengkap dari produk dalam teks multiline. `Spacer(8.dp)` dan `Spacer(16.dp)` digunakan untuk memberi jarak antar elemen teks dan tombol.
- Baris 46-50:
  Menampilkan tombol "Kembali" di bagian bawah layar. Fungsi `navController.popBackStack()` digunakan untuk menavigasi ke layar sebelumnya, yaitu `MakeupListScreen`. Gaya tombol mengikuti `MaterialTheme.typography.bodyMedium`.

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.

D. Tautan Git

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

SOAL 2

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

A. Pembahasan

Karena RecyclerView memfasilitasi penyajian kumpulan data besar dengan cara yang efisien. Pembuat menyuplai data dan mengatur tampilan untuk setiap item, serta library RecyclerView secara dinamis menghasilkan elemen saat dibutuhkan. Sesuai sebutannya, RecyclerView mengolah kembali elemen-elemen individual itu. XML mungkin tampak kuno dan agak membosankan, tetapi kemampuannya dalam mengorganisir data yang rumit dan mentransfernya antara sistem tetap menjadikannya banyak diterapkan di berbagai proyek, khususnya di perusahaan besar yang dimana tidak bisa langsung migrasi ke Compose.

RecyclerView merupakan salah satu tampilan yang umum dan banyak sekali digunakan untuk menampilkan informasi yang ingin disampaikan lewat aplikasi Android, khususnya saat informasi yang ingin disampaikan tersebut banyak dan relatif seragam, jadi bisa dikatakan sebagai jenis view yang fleksibel, stabil dan konsisten. Meski lebih verbose, RecyclerView memberikan fleksibilitas yang lebih luas dibandingkan LazyColumn dan tetap menjadi pilihan yang kuat dan layak, terutama untuk proyek lama atau aplikasi yang memerlukan kontrol kinerja yang tepat.

Untuk kurva pembelajaran RecyclerView adalah komponen matang yang telah banyak digunakan dalam pengembangan Android, sehingga memudahkan pencarian sumber daya dan dukungan komunitas, sedangkan LazyRow/LazyColumn, yang relatif baru di Jetpack Compose, mungkin mengharuskan pengembang untuk membiasakan diri dengan kerangka kerja Compose. Selain itu karena Jetpack Compose membawa runtime Compose ke dalam aplikasi, developer dapat menggunakan RecyclerView untuk mengurangi ukuran aplikasi mereka yang kecil atau ringan.

B. Referensi:

- https://developer.android.com/develop/ui/views/layout/recyclerview?hl=id
- Mengenal XML: Sejarah, Fungsi, dan Manfaatnya untuk Programmer dan Mahasiswa IT - Java Community
- Android RecyclerView: Basic dan Templating | by Dion Saputra | Ristex | Medium
- https://www.linkedin.com/pulse/jetpack-compose-vs-android-view-system-which-ui-toolkit-rahul-pahuja-ufjtf/
- https://medium.com/@humzakhalid94/recyclerview-vs-lazyrow-lazycolumn-choosing-the-right-layout-for-dynamic-lists-in-android-ec9912049375