LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4

VIEWMODEL AND DEBUGGING

Oleh:

Jovan Gilbert Natamasindah NIM. 2310817310002

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

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 : Jovan Gilbert Natamasindah
NIM : 2310817310002

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

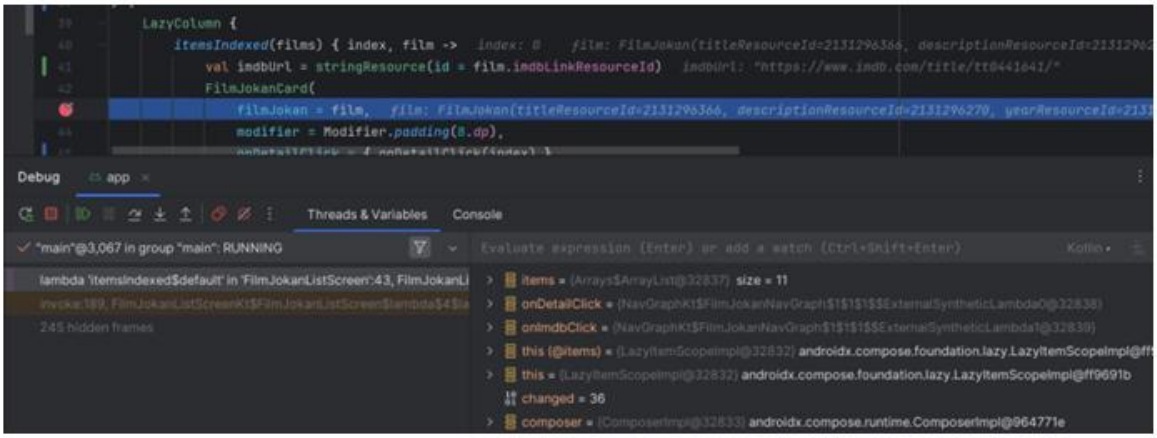
DAFTAR TABEL

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.android.modul4
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.padding
8   import androidx.compose.material3.Scaffold
9   import androidx.compose.runtime.Composable
10  import androidx.compose.ui.Modifier
11  import androidx.lifecycle.viewmodel.compose.viewModel
12  import androidx.navigation.compose.NavHost
13  import androidx.navigation.compose.composable
14  import androidx.navigation.compose.rememberNavController
15  import com.android.modul4.data.sourceData
16  import com.android.modul4.screens.CardList
17  import com.android.modul4.screens.DetailPage
18  import com.android.modul4.ui.theme.Modul4Theme
19  import com.android.modul4.viewmodel.CardViewModel
20  import com.android.modul4.viewmodel.CardViewModelFactory
21
22  class MainActivity : ComponentActivity() {
23      override fun onCreate(savedInstanceState: Bundle?) {
24          super.onCreate(savedInstanceState)
25          enableEdgeToEdge()
26          setContent {
27              Modul4Theme {
28                  Display()
29              }
30          }
31      }
32  }
33
34  @Composable
35  fun Display() {
36      val navController = rememberNavController()
37      val viewModelFactory = CardViewModelFactory(sourceData())
38      val viewModel: CardViewModel = viewModel(factory =
    viewModelFactory)
39
40      Scaffold() { innerPadding ->
41          NavHost(
42              navController = navController,
43              startDestination = "card_list",
44              modifier = Modifier.padding(innerPadding)
45          ) {
46              composable("card_list") {
47                  CardList(navController, viewModel)
48              }
49              composable("detail/{itemTitle}") { backStackEntry
    ->
50                  val itemTitle =
    backStackEntry.arguments?.getString("itemTitle")!!
51                  DetailPage(itemTitle, navController,
    viewModel)
52              }
53          }
54
```

```
55      }
}
```

Tabel 1. Source Code Jawaban Soal 1 MainActivity

2. data/sourceData.kt

```
1   package com.android.modul4.data
2
3   import com.android.modul4.models.CardProp
4
5   class sourceData {
6       fun loadCardProps(): List<CardProp> {
7           return listOf<CardProp>(
8               CardProp(
9                   title = "Garuda Indonesia",
10                  desc = "Garuda Indonesia adalah maskapai
    penerbangan nasional Indonesia yang didirikan pada tahun 1949.
    Berbasis di Jakarta, Garuda dikenal dengan layanan
    penerbangannya yang berkualitas tinggi dan keramahannya,
    mencerminkan budaya Indonesia. Maskapai ini mengoperasikan
    penerbangan domestik dan internasional ke berbagai tujuan di
    Asia, Australia, Eropa, dan Timur Tengah. Garuda Indonesia juga
    merupakan anggota dari aliansi penerbangan global SkyTeam dan
    telah beberapa kali meraih penghargaan dunia atas pelayanan
    kabinnya.",
11                  ImageURL = "https://cdn.plnspttrs.net/11964/pk-
    gib-garuda-indonesia-mcdonnell-douglas-dc-10-
    30_PlanespottersNet_714188_a594861cb0_o.jpg",
                    Wiki =
12  "https://id.wikipedia.org/wiki/Garuda_Indonesia",
                    tglBerdiri = "1 Agustus 1947",
13                  armada = "77",
14                  rute = "Belanda, Thailand, China, Hong Kong,
15  Arab Saudi, Malaysia, Arab Saudi, Australia, Korea Selatan,
    Singapura, Jepang, United Arab Emirates",
                    website = "https://www.garuda-
16  indonesia.com/id/id//",
                ),
17              CardProp(
18                  title = "Lion Air",
19                  desc = "Lion Air adalah maskapai penerbangan
20  swasta terbesar di Indonesia yang berdiri pada tahun 1999.
    Fokus utamanya adalah layanan penerbangan berbiaya rendah (low-
    cost carrier) dengan rute domestik dan internasional. Lion Air
    terkenal dengan jaringan penerbangan yang luas dan harga tiket
    yang kompetitif.",
                    ImageURL = "https://cdn.plnspttrs.net/20446/pk-
21  lhg-lion-air-boeing-747-
    412_PlanespottersNet_612707_2702071c86_o.jpg",
                    Wiki =
22  "https://id.wikipedia.org/wiki/Lion_Air",
                    tglBerdiri = "15 November 1999",
```

```
                armada = "100",
                rute = "Arab Saudi, China, Malaysia",
                website = "https://www.lionair.co.id/",
            ),
            CardProp(
                title = "Citilink",
                desc = "Citilink adalah anak perusahaan dari
Garuda Indonesia yang beroperasi sebagai maskapai berbiaya
rendah. Didirikan pada tahun 2001, Citilink menawarkan
penerbangan domestik dan regional dengan konsep layanan yang
modern, dinamis, dan lebih santai untuk menarik kalangan muda
dan pelancong bisnis.",
                ImageURL = "https://cdn.plnspttrs.net/42179/pk-
gaf-citilink-atr-72-600-72-
212a_PlanespottersNet_1657140_83de625cfa_o.jpg",
                Wiki =
"https://id.wikipedia.org/wiki/Citilink",
                tglBerdiri = "16 Juli 2001",
                armada = "57",
                rute = "Australia, Malaysia, Papua Nugini,
Singapura, Timor Leste, China",
                website = "https://www.citilink.co.id/",
            ),
            CardProp(
                title = "Super Air Jet",
                desc = "Super Air Jet adalah maskapai baru di
Indonesia yang mulai beroperasi pada tahun 2021. Mengusung
konsep \"new lifestyle airline\", Super Air Jet fokus melayani
segmen anak muda dengan harga terjangkau, desain modern, dan
rute-rute domestik populer.",
                ImageURL = "https://cdn.plnspttrs.net/14570/pk-
std-super-air-jet-airbus-a320-232-
wl_PlanespottersNet_1755960_137445c980_o.jpg",
                Wiki =
"https://www.superairjet.com/en/about.php",
                tglBerdiri = "1 March 2021",
                armada = "61",
                rute = "Malaysia",
                website =
"https://checkin.superairjet.com/dx/IUCI/",
            ),
            CardProp(
                title = "Batik Air",
                desc = "Batik Air adalah maskapai layanan penuh
(full-service) dari Lion Air Group yang didirikan pada tahun
2013. Batik Air menawarkan fasilitas premium seperti makanan
dalam penerbangan dan hiburan di kursi, serta menghubungkan
berbagai kota besar di Indonesia dan Asia.",
                ImageURL = "https://cdn.plnspttrs.net/35032/pk-
lug-batik-air-airbus-a320-214-
wl_PlanespottersNet_1693958_5bbffba586_o.jpg",
                Wiki =
```

```
    "https://id.wikipedia.org/wiki/Batik_Air",
53                  tglBerdiri = "10 Juni 2012",
54                  armada = "65",
55                  rute = "Australia, Arab Saudi, China, India,
    Malaysia, Singapura, Thailand",
56                  website = "https://www.batikair.com.my/",
57              ),
58              CardProp(
59                  title = "Pelita Air",
60                  desc = "Pelita Air adalah maskapai yang awalnya
    fokus pada penerbangan carter dan layanan energi (minyak dan
    gas), namun sejak 2022 mulai mengembangkan penerbangan reguler
    domestik. Sebagai anak usaha Pertamina, Pelita Air membawa
    konsep layanan penerbangan yang nyaman dan profesional.",
61                  ImageURL = "https://cdn.plnspttrs.net/22490/pk-
    pwd-pelita-air-service-airbus-a320-214-
    wl_PlanespottersNet_1698319_901e5c6a31_o.jpg",
62                  Wiki =
    "https://id.wikipedia.org/wiki/Pelita_Air",
63                  tglBerdiri = "24 Januari 1970",
64                  armada = "33",
65                  rute = "Indonesia",
66                  website = "https://www.pelita-air.com/",
67              )
68          )
69      }
70  }
```

Tabel 2 Source Code Jawaban Soal 1 AirlineData

3. models/CardProp.kt

```
1   package com.android.modul4.models
2
3   data class CardProp(
4       val title: String,
5       val desc: String,
6       val ImageURL: String,
7       val Wiki: String,
8       val tglBerdiri: String,
9       val armada: String,
10      val rute: String,
11      val website:String
12  )
```

Tabel 3 Source Code Jawaban Soal 1 UIComponent

4. screens/cardDetailScreen.kt

```
1   package com.android.modul4.screens
2
3   import android.content.Intent
4   import android.net.Uri
```

```
5  import android.util.Log
6  import androidx.compose.foundation.layout.Arrangement
7  import androidx.compose.foundation.layout.Column
8  import androidx.compose.foundation.layout.Row
9  import androidx.compose.foundation.layout.Spacer
10 import androidx.compose.foundation.layout.fillMaxWidth
11 import androidx.compose.foundation.layout.height
12 import androidx.compose.foundation.layout.padding
13 import androidx.compose.foundation.rememberScrollState
14 import
   androidx.compose.foundation.shape.RoundedCornerShape
15 import androidx.compose.foundation.verticalScroll
16 import androidx.compose.material3.Button
17 import androidx.compose.material3.Text
18 import androidx.compose.runtime.Composable
19 import androidx.compose.ui.Alignment
20 import androidx.compose.ui.Modifier
21 import androidx.compose.ui.platform.LocalContext
22 import androidx.compose.ui.unit.dp
23 import androidx.navigation.NavController
24 import com.android.modul4.Desc
25 import com.android.modul4.DetailRow
26 import com.android.modul4.Img
27 import com.android.modul4.Title
28 import com.android.modul4.viewmodel.CardViewModel
29
30 @Composable
31 fun DetailPage(itemTitle: String, navController:
   NavController, viewModel: CardViewModel) {
32     val context = LocalContext.current
33     val detailMaskapai =
   viewModel.getDetailByTitle(itemTitle)
34
35     if (detailMaskapai != null) {
36         Column(modifier = Modifier
37             .padding(16.dp)
38             .verticalScroll(rememberScrollState()),
39             horizontalAlignment =
   Alignment.CenterHorizontally
40         ) {
41             Img(detailMaskapai.ImageURL, 400)
42             Title(itemTitle)
43             Spacer(modifier = Modifier.height(16.dp))
44             detailMaskapai?.let {
45                 DetailRow("Tanggal Berdiri", ":
   ${it.tglBerdiri}")
46                 DetailRow("Armada", ": ${it.armada}")
```

```
47                  DetailRow("Rute Tujuan", ": ${it.rute} ")
48              }
49              Spacer(modifier = Modifier.height(16.dp))
50              Desc(detailMaskapai.desc)
51              Spacer(modifier = Modifier.height(16.dp))
52
53              Row(modifier =
    Modifier.fillMaxWidth(),horizontalArrangement =
    Arrangement.SpaceBetween) {
54                  Button(onClick = {
55                      Log.d("Detail", "tombol website
    ${itemTitle} ditekan")
56                      val intent =
    Intent(Intent.ACTION_VIEW, Uri.parse(detailMaskapai?.let
    {it.website}))
57                      context.startActivity(intent)
58                  }, modifier = Modifier.padding(horizontal
    = 6.dp),
59                      shape = RoundedCornerShape(8.dp)
60                  ) { Text("Web $itemTitle") }
61                  Button(onClick = {
62                      navController.navigate("card_list")
    }, shape = RoundedCornerShape(8.dp)) { Text("Kembali") }
63              }
64          }
65      }
66      else {
67          Column(modifier = Modifier.padding(16.dp)) {
68              Text("Data tidak ditemukan untuk
    \"$itemTitle\"")
69              Button(onClick = {
    navController.navigate("card_list") }) {
70                  Text("Kembali")
71              }
72          }
73      }
74  }
```

5. screens/cardListScreen.kt

```
1   package com.android.modul4.screens
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.util.Log
6   import androidx.compose.foundation.background
7   import androidx.compose.foundation.layout.Arrangement
```

```
8  import androidx.compose.foundation.layout.Column
9  import androidx.compose.foundation.layout.Row
10 import androidx.compose.foundation.layout.Spacer
11 import androidx.compose.foundation.layout.fillMaxWidth
12 import androidx.compose.foundation.layout.height
13 import androidx.compose.foundation.layout.padding
14 import androidx.compose.foundation.lazy.LazyColumn
15 import
   androidx.compose.foundation.shape.RoundedCornerShape
16 import androidx.compose.material3.Button
17 import androidx.compose.material3.Card
18 import androidx.compose.material3.CardDefaults
19 import androidx.compose.material3.Text
20 import androidx.compose.runtime.Composable
21 import androidx.compose.runtime.collectAsState
22 import androidx.compose.runtime.getValue
23 import androidx.compose.ui.Modifier
24 import androidx.compose.ui.graphics.Color
25 import androidx.compose.ui.platform.LocalContext
26 import androidx.compose.ui.unit.dp
27 import androidx.navigation.NavController
28 import com.android.modul4.Desc
29 import com.android.modul4.Img
30 import com.android.modul4.Title
31 import com.android.modul4.viewmodel.CardViewModel
32
33 @Composable
34 fun CardList( navController: NavController, viewModel:
   CardViewModel) {
35     val CardList by viewModel.cardList.collectAsState()
36     val context = LocalContext.current
37     val bgcard = Color(0xFF7AE2CF)
38     val bgcolor = Color(0xFFFFFDF6)
39
40     Title("Maskapai-maskapai Penerbangan di Indonesia")
41     LazyColumn(
42         modifier = Modifier
43             .padding(top = 50.dp)
44             .background(bgcolor)
45     ){
46         items(CardList.size) { index ->
47             val property = CardList[index]
48             Card(
49                 modifier = Modifier
50                     .fillMaxWidth()
51                     .padding(6.dp)
52                     .height(200.dp),
```

```
53                  colors =
    CardDefaults.cardColors(containerColor = bgcard),
54              ) {
55                  Row(
56                      modifier = Modifier
57                          .padding(6.dp),
58                  ) {
59                      Img(property.ImageURL, 180)
60                      Column {
61                          Title(property.title)
62                          Desc(if(property.desc.length >
    80) property.desc.take(80) + "..." else property.desc)
63                          Spacer(modifier =
    Modifier.weight(1f))
64                          Row(
65                              horizontalArrangement =
    Arrangement.End,
66                              modifier = Modifier
67                                  .fillMaxWidth()
68                          ) {
69                              Button(onClick = {
70
    viewModel.selectCard(property, "wiki")
71                                  val intent =
    Intent(Intent.ACTION_VIEW, Uri.parse(property.Wiki))
72
    context.startActivity(intent)
73                              }, modifier =
    Modifier.padding(horizontal = 6.dp),
74                                  shape =
    RoundedCornerShape(8.dp)
75                              ) { Text("Wiki") }
76                              Button(onClick = {
77
    viewModel.selectCard(property, "Detail")
78
    Log.d("cardListScreen","pindah ke halaman detail dari
    item ${property.title} ")
79
    navController.navigate("detail/${property.title}")
80                              }, shape =
    RoundedCornerShape(8.dp)) { Text("Detail") }
81                          }
82                      }
83                  }
84              }
85          }
```

```
86      }
87  }
```

6. screens/CardViewModel.kt

```
1   package com.android.modul4.viewmodel
2
3   import android.util.Log
4   import androidx.lifecycle.ViewModel
5   import com.android.modul4.data.sourceData
6   import com.android.modul4.models.CardProp
7   import kotlinx.coroutines.flow.MutableStateFlow
8   import kotlinx.coroutines.flow.StateFlow
9
10  class CardViewModel(private val dataSource: sourceData) :
    ViewModel() {
11      private val _cardList =
    MutableStateFlow<List<CardProp>>(emptyList())
12      val cardList: StateFlow<List<CardProp>> = _cardList
13
14      private val _selectCard =
    MutableStateFlow<CardProp?>(null)
15      val selectedCard: StateFlow<CardProp?> = _selectCard
16
17      init { loadData() }
18
19      private fun loadData() {
20          val cards = dataSource.loadCardProps()
21          _cardList.value = cards
22          Log.d("CardViewMode","Card yang tampil ada
    ${cards.size} ")
23      }
24      fun getDetailByTitle(Title: String): CardProp? {
25          return _cardList.value.find { it.title == Title }
26      }
27      fun selectCard(card: CardProp, name: String) {
28          _selectCard.value = card
29          Log.d("CardViewModel", "tombol ${name}
    ${card.title} ditekan")
30      }
31  }
```

7. screens/CardViewModelFactory.kt

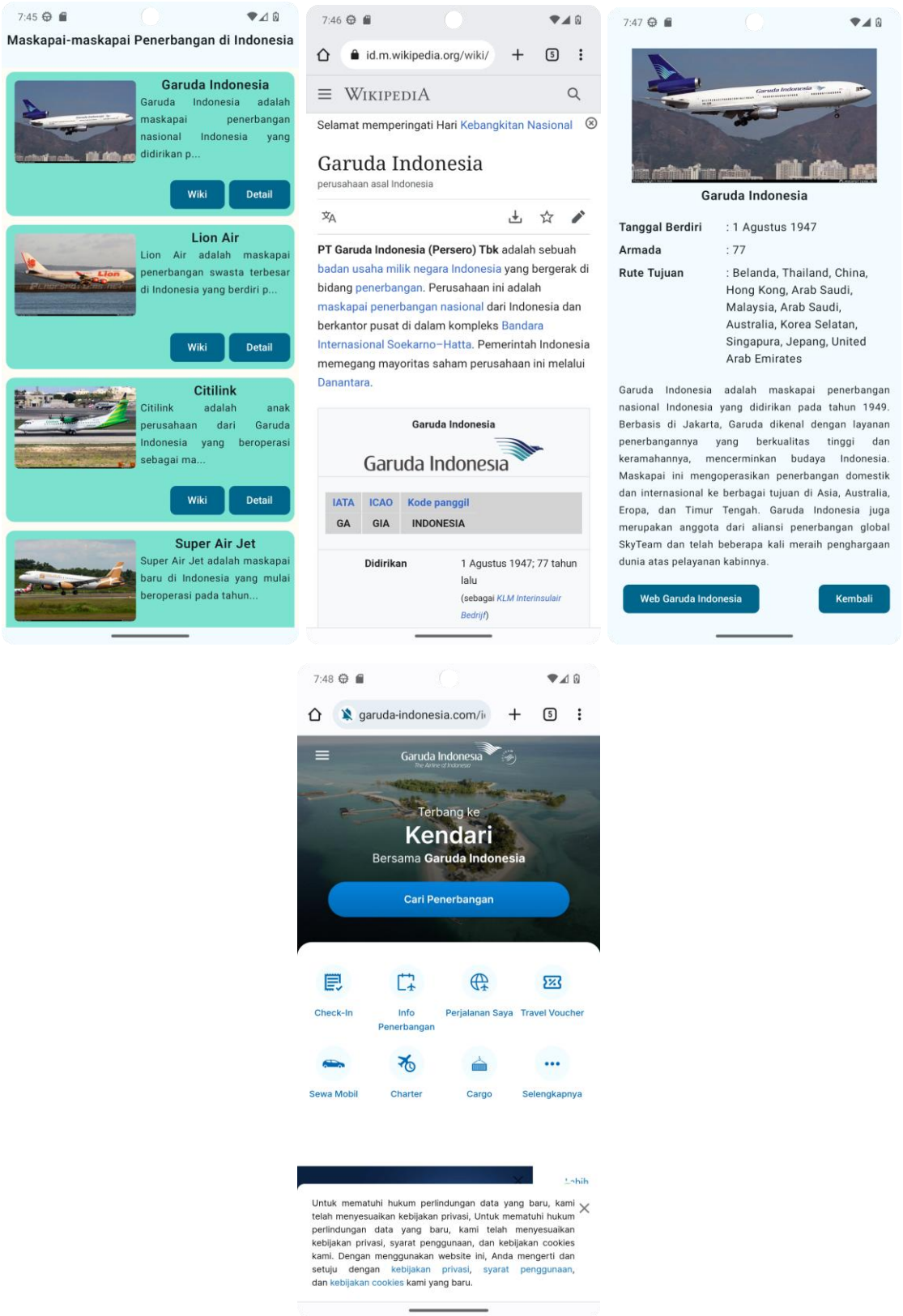
```
1  package com.android.modul4.viewmodel
2
3  import androidx.lifecycle.ViewModel
4  import androidx.lifecycle.ViewModelProvider
5  import com.android.modul4.data.sourceData
6
7  class CardViewModelFactory(private val dataSource:
   sourceData) : ViewModelProvider.Factory {
8      override fun <T : ViewModel> create(modelClass:
   Class<T>): T {
9          if
   (modelClass.isAssignableFrom(CardViewModel::class.java))
   {
10             return CardViewModel(dataSource) as T
11         }
12         throw IllegalArgumentException("Unknown ViewModel
   class")
13     }
14 }
```

8. UIComponent.kt

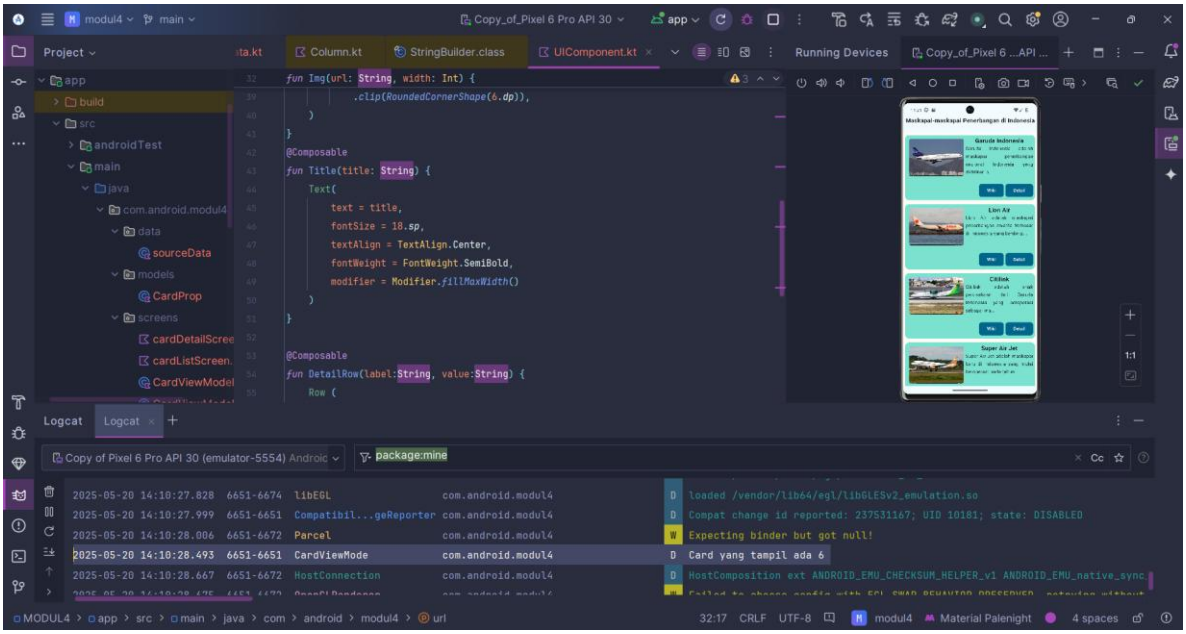
```
1  package com.android.modul3
2
3  import android.content.Intent
4  import android.net.Uri
5  import androidx.compose.foundation.layout.Row
6  import androidx.compose.foundation.layout.fillMaxWidth
7  import androidx.compose.foundation.layout.padding
8  import androidx.compose.foundation.layout.width
9  import androidx.compose.foundation.shape.RoundedCornerShape
10 import androidx.compose.material3.Button
11 import androidx.compose.material3.Text
12 import androidx.compose.runtime.Composable
13 import androidx.compose.ui.Modifier
14 import androidx.compose.ui.draw.clip
15 import androidx.compose.ui.text.font.FontWeight
16 import androidx.compose.ui.text.style.TextAlign
17 import androidx.compose.ui.unit.dp
18 import androidx.compose.ui.unit.sp
19 import
   com.bumptech.glide.integration.compose.ExperimentalGlideComposeApi
20 import com.bumptech.glide.integration.compose.GlideImage
21
22 @Composable
23 fun Desc(desc: String) {
24     Text(
```

```
25          text = desc,
26          fontSize = 14.sp,
27          textAlign = TextAlign.Justify
28      )
29  }
30  @OptIn(ExperimentalGlideComposeApi::class)
31  @Composable
32  fun Img(url: String, width: Int) {
33      GlideImage(
34          model = url,
35          contentDescription = "My Image",
36          modifier = Modifier
37              .width(width.dp)
38              .padding(6.dp)
39              .clip(RoundedCornerShape(6.dp)),
40      )
41  }
42  @Composable
43  fun Title(title: String) {
44      Text(
45          text = title,
46          fontSize = 18.sp,
47          textAlign = TextAlign.Center,
48          fontWeight = FontWeight.SemiBold,
49          modifier = Modifier.fillMaxWidth()
50      )
51  }
52
53  @Composable
54  fun DetailRow(label:String, value:String) {
55      Row (
56          modifier = Modifier
57              .fillMaxWidth()
58              .padding(vertical = 4.dp)
59      ) {
60          Text(
61              text = label,
62              modifier = Modifier.width(150.dp),
63              fontWeight = FontWeight.SemiBold
64          )
65          Text(value)
66      }
67  }
```

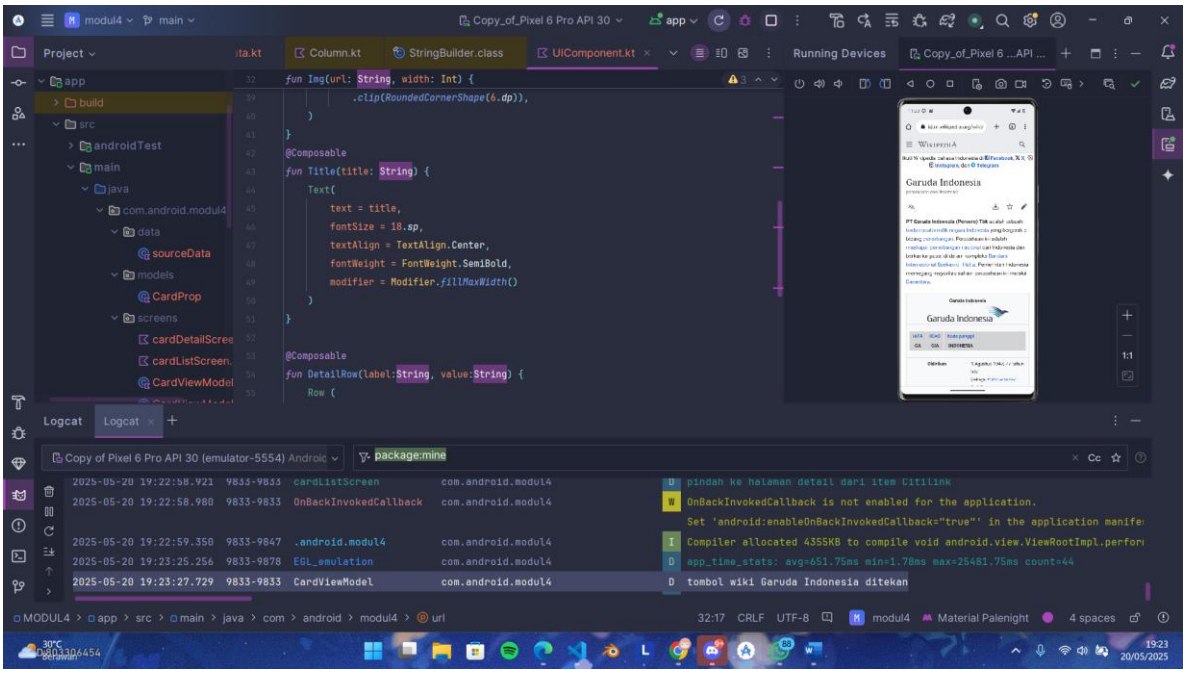
B. Output Program



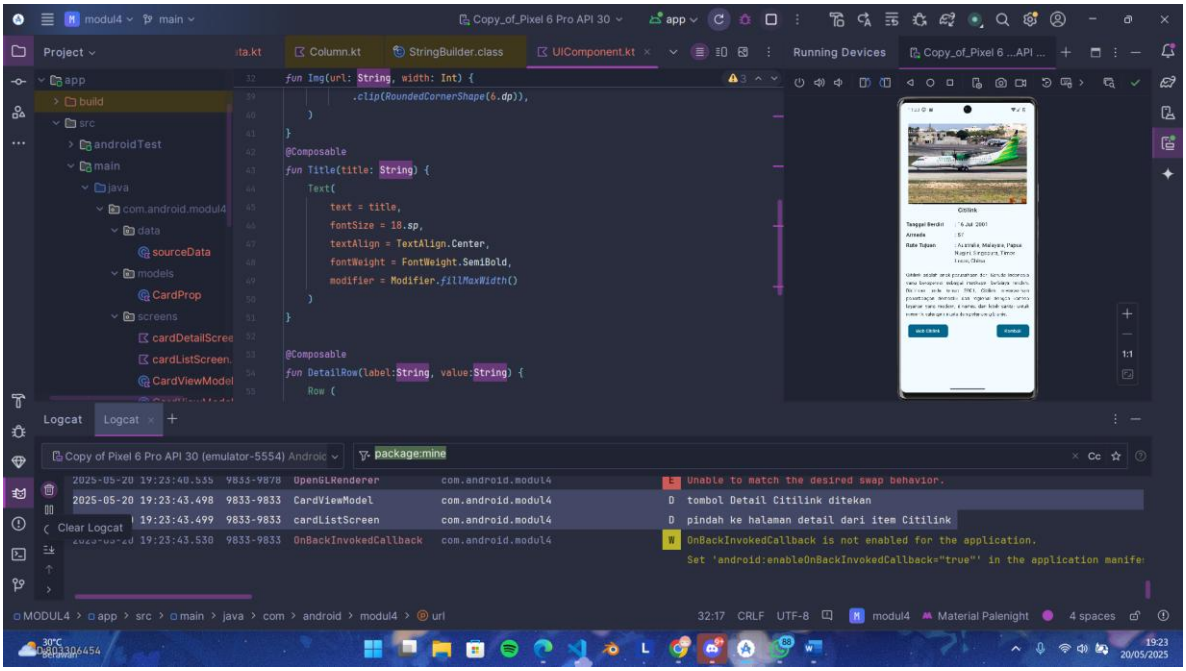
Gambar 1. Screenshot Hasil Jawaban Soal 1



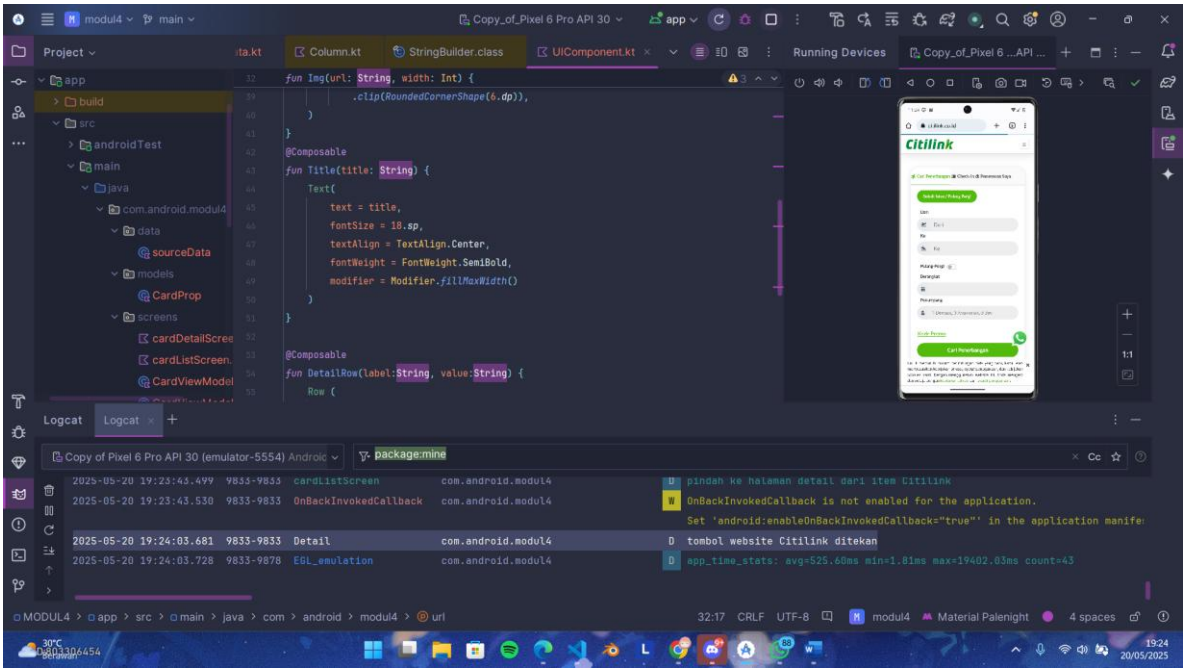
Gambar 1. screenshot saat data item masuk list



Gambar 2. screenshot saat tombol explicit intent ditekan



Gambar 3. screenshot data dari list yang dipilih ketika ke halaman detail



C. Pembahasan

1. MainActivity.kt:

- Pada baris 1, **package com.android.modul4** pendeklarasian nama package file Kotlin.

- Pada baris 3-20, **import** berfungsi untuk mengimpor package atau kelas untuk mengakses komponen dan fungsi dari android SDK dan jetpack compose.
- Pada baris 22, **class MainActivity : ComponentActivity(),** merupakan titik mula yang menjadi kelas utama dan akan dijalankan pertama kali saat aplikasi dibuka
- Pada baris 23, **override fun onCreate(savedInstanceState: Bundle?),** berfungsi untuk menimpa (override) fungsi oncreate dari ComponentActivity.
- Pada baris 24, **super.onCreate(savedInstanceState),** berfungsi untuk memanggil superclass dari fungsi oncreate untuk memastikan bahwa proses inisialisasi standar dari Android dijalankan sebelum logika saya dijalankan.
- Pada baris 25, **enableEdgeToEdge(),** berfungsi agar tampilan aplikasi dapat menggunakan seluruh layar dari status bar sampai navigation bar atau fullscreen layout.
- Pada baris 26, **setContent(),** digunakan untuk menampilkan UI berbasis jetpack compose ke dalam activity
- Pada baris 27, **Modul4Theme,** merupakan fungsi yang berisi tema custom yang membungkus seluruh UI untuk memberikan style yang konsisten.
- Pada baris 28, **Display(),** merupakan fungsi yang dibuat untuk menampilkan aplikasi scrollable list ke dalam activity.
- Pada baris 34, **@composable,** merupakan annotation untuk menandai bahwa fungsi yang dibuat dibawahnya merupakan fungsi UI di jetpack compose.
- Pada baris 35, **fun Display(),** merupakan fungsi yang dibuat untuk menampilkan semua composable yang ingin ditampilkan ke dalam activity, seperti CardList() dan DetailPage() lalu menerima parameter navController.
- Pada baris 36-50, **val** merupakan keyword untuk membuat suatu variable bersifat immutable yang menampung nilai dari berbagai tipe data.
- Pada baris 36, **val navController = rememberNavController()** berfungsi untuk membuat dan menyimpan instance dari NavController
- Pada baris 37, **val viewModelFactory = CardViewModelFactory(sourceData())** berfungsi untuk membuat objek dari factory untuk membuat instance dari CardViewModel
- Pada baris 38, **val viewModel: CardViewModel = viewModel(factory = viewModelFactory)** berfungsi untuk memanggil fungsi viewModel(), sehingga viewModel dapat digunakan untuk mengakses data dan fungsi.

- Pada baris 40, **scaffold()**, merupakan komponen layout yang menjadi kerangka dasar dari tampilan yang mempermudah dalam mengatur strktur UI karena sudah disediakan slot-slot bawannya.
- Pada baris 41, **NavHost(),** berfungsi untuk menyediakan wadah untuk navigasi antar composable screen berdasarkan rute yang ditentukan
    - Pada baris 42, **navController** merupakan controller yang bertanggung jawab atas navigasi.
    - Pada baris 43, **startDestination** merupakan rute awal yang ditampilkan ketika aplikasi dijalankan.
    - Pada baris 44, modifier = **Modifier** berfungsi untuk menerapkan modifikasi terhadap tampilan komponen, seperti *.padding* untuk memberikan jarak dalam component.
- Pada baris 46 dan 49, **composable(),** berfungsi untuk mendefinisikan rute, seperti di kasus ini ada "card_list" dan "detail"
- Pada baris 47, **CardList(navController, ViewModel),** merupakan fungsi untuk menampilkan list yang dapat scroll sekaligus menjadi halaman utama aplikasi dan menerima parameter navController dan ViewModel.
- pada baris 50, **itemTitle** berfungsi untuk mengambil parameter di navigasi melalui backstackentry.
- pada baris 51, **DetailPage(itemTitl, navController, ViewModel)** merupakan fungsi untuk menampilkan halaman detail dari aplikasi dan menerima beberapa parameter.

2. data/sourceData.kt

- Pada baris 1, **package com.android.modul4.data** pendeklarasian nama package file Kotlin.
- Pada baris 3, **import com.android.modul4.models.CardProp** berfungsi untuk mengimport data class dari file CardProp.
- Pada baris 5, **class sourceData** berfungsi untuk membuat sebuah class dengan nama sourceData.
- Pada baris 6, **fun loadCardProps(): List<CardProp>** merupakan fungsi dengan tujuan untuk mengembalikan list dari objek CardProp yang berisikan data-data
- Pada baris 7, **return listOf<CardProp>** merupakan list immutable yang akan dikembalikan

3. models/CardProp.kt

- Pada baris 1, **package com.android.modul4.models** pendeklarasian nama package file Kotlin.
- Pada baris 3, **data class** meerupakan jenis kelas untuk menyimpan struktur data.
- Pada baris 4-11, **val** merupakan keyword untuk membuat suatu variable bersifat immutable yang menampung nilai dari berbagai tipe data.

4. screens/cardDetailScreen.kt

- Pada baris 1, **package com.android.modul4.screens** pendeklarasian nama package file Kotlin.
- Pada baris 3-28, **import** berfungsi untuk mengimpor package atau kelas untuk mengakses komponen dan fungsi dari android SDK dan jetpack compose.
- Pada baris 31, **fun DetailPage(navController: NavController, viewModel: CardViewModel),** merupakan fungsi untuk menampilkan detail dari card yang dipilih dan menerima parameter berupa navController dan viewModel.
- Pada baris 32, **LocalContext.current** berfungsi untuk mendapatkan context dari acrivity saat ini dalam fungsi composable dan juga digunakan untuk menampilkan Toast sebagai notifikasi dari peringatan jika user salah input.
- Pada baris 33, **val detailMaskapai = viewModel.getDetailByTitle(itemTitle)** berfungsi untuk memanggil fungsi viewModel lalu dapatkan detail data berdasarkan judul.
- Pada baris 34, **if** merupakan percabangan yang jika kondisinya terpenuhi maka isinya akan dieksekusi.
- Pada baris 36 dan 67, **Column()** merupakan komponen layout mirip seperti scaffold, hanya saja Column digunakan untuk menyusun elemen UI secara vertical.
    - Pada baris 36 dan 67, **modifier** merupakan parameter dalam Column() untuk menerapkan modifikasi terhadap tampilan komponen, seperti .*padding* untuk memberikan jarak dalam componen dan .*verticalScroll* agar bisa di scroll
    - Pada baris 65, **horizontalAlignment** berfungsi untuk menentukan layout secara horizontal.
- Pada baris 41, **Img()** merupakan fungsi yang saya buat sendiri untuk memudahkan dalam memuat gambar.
- Pada baris 42, **Title()** merupakan fungsi yang saya buat sendiri untuk memudahkan dalam membuat judul.

- Pada baris 43-51, **Spacer()** berfungsi untuk memberikan jarak kosong biasanya secara vertical dalam layout.
- Pada baris 50, **Desc()** merupakan fungsi yang saya buat sendiri untuk memudahkan dalam membuat deskripsi.
- Pada baris 53, **Row** merupakan komponen layout yang berlawanan dari Column, Dimana jika Column menyusun elemen UI secara Vertical, maka Row menyusun elemen UI secara horizontal.
    - Pada baris 53, **modifier** berfungsi untuk menerapkan modifikasi terhadap tampilan komponen, seperti *.padding* untuk memberikan jarak dalam component, *.fillMaxWidth()* untuk membuat lebar component lebarnya selayar.
    - Pada baris 53, **horizontalArrangement** berfungsi untuk memberi jarak maksimal antar elemen.
- Pada baris 54 dan 61, **Button()** merupakan komponen UI yang berfungsi untuk membuat tombol pada aplikasi.
    - Pada baris 54 dan 61, **onClick** merpakan parameter dalam Button() untuk memberikan aksi yang akan dijalankan ketika tombol ditekan.
    - Pada baris 55, **Log.d("Detail", "tombol website ${itemTitle} ditekan")** berfungsi untuk mencetak pesan ke logcat untuk debugging.
    - Pada baris 56, **val intent = Intent(Intent.ACTION_VIEW, Uri.parse(detailMaskapai?.let {it.website}))** berfungsi untuk membuat intent secara eksplisit untuk membuka URL di browser.
    - Pada baris 58, **modifier** merupakan parameter dalam TextField untuk menerapkan modifikasi terhadap tampilan komponen, seperti *.fillMaxWidth* agar column lebar Button mengisi ukuran layar.
    - Pada baris 59 dan 62, **shape** merupakan parameter dalam Button() untuk mengatur sudut agar dapat membulat.

5. screens/cardListScreen.kt

- Pada baris 1, **package com.android.modul4.screens** pendeklarasian nama package file Kotlin.
- Pada baris 3-31, **import** berfungsi untuk mengimpor package atau kelas untuk mengakses komponen dan fungsi dari android SDK dan jetpack compose.
- Pada baris 34, **fun CardList(navController: NavController, viewModel: CardViewModel),** merupakan fungsi untuk menampilkan card-card dalam jumlah

banyak yang dapat di scroll dan menerima parameter berupa navController dan viewModel.

- Pada baris 35, **val CardList by viewModel.cardList.collectAsState()** berfungsi untuk mengamati stateflow dan dikonversi menjadi state compose, lalu akan mengambil nilai .value dari collectAsState lalu akan gunakan CardList untuk UI.
- Pada baris 36, **LocalContext.current** berfungsi untuk mendapatkan context dari acrivity saat ini dalam fungsi composable dan juga digunakan untuk menampilkan Toast sebagai notifikasi dari peringatan jika user salah input.
- Pada baris 40 dan 61, **Title()** merupakan fungsi yang saya buat sendiri untuk memudahkan dalam membuat judul.
- Pada baris 41, **LazyColumn()** berfungsi untuk menampilkan daftar item secara vertical yang dapat di scroll tapi hanya yang terlihat di layar.
    - Pada baris 42, **modifier** merupakan parameter dalam Column() untuk menerapkan modifikasi terhadap tampilan komponen, seperti *.padding* untuk memberikan jarak dalam componen dan *.background* untuk memberikan warna latar.
- Pada baris 46, **item()** berfungsi untuk membuat item di dalam LazyColumn sebanyak jumlah data.
- pada baris 48, **Card()** berfungsi untuk membuat sebuah kartu.
    - pada baris 49, **modifier** merupakan parameter dalam Column() untuk menerapkan modifikasi terhadap tampilan komponen, seperti *.padding* untuk memberikan jarak dalam component, *.height* untuk tinggi component, dan *.fillMaxWidth()* untuk membuat lebar component lebarnya selayar.
    - pada baris 53, **colors** merupakan parameter dalam Card() untuk mengatur warna background (containerColor).
- Pada baris 55 dan 64, **Row** merupakan komponen layout yang berlawanan dari Column, Dimana jika Column menyusun elemen UI secara Vertical, maka Row menyusun elemen UI secara horizontal.
    - Pada baris 56 dan 66, **modifier** berfungsi untuk menerapkan modifikasi terhadap tampilan komponen, seperti *.padding* untuk memberikan jarak dalam component, *.fillMaxWidth()* untuk membuat lebar component lebarnya selayar.
    - Pada baris 65, **horizontalArrangement** berfungsi untuk memberi jarak maksimal antar elemen.

- Pada baris 59, **Img()** merupakan fungsi yang saya buat sendiri untuk memudahkan dalam memuat gambar.
- Pada baris 60, **Column()** merupakan komponen layout mirip seperti scaffold, hanya saja Column digunakan untuk menyusun elemen UI secara vertical.
- Pada baris 62, **Desc()** merupakan fungsi yang saya buat sendiri untuk memudahkan dalam membuat deskripsi.
- Pada baris 63, **Spacer()** berfungsi untuk memberikan jarak kosong biasanya secara vertical dalam layout.
- Pada baris 69 dan 76, **Button()** merupakan komponen UI yang berfungsi untuk membuat tombol pada aplikasi.
    - Pada baris 69 dan 76, **onClick** merpakan parameter dalam Button() untuk memberikan aksi yang akan dijalankan ketika tombol ditekan.
    - Pada baris 73, **modifier** merupakan parameter dalam TextField untuk menerapkan modifikasi terhadap tampilan komponen, seperti *.fillMaxWidth* agar column lebar Button mengisi ukuran layar.
    - Pada baris 74 dan 80, **shape** merupakan parameter dalam Button() untuk mengatur sudut agar dapat membulat.

6. screens/CardViewModel.kt

- Pada baris 1, **package com.android.modul4.viewModel** pendeklarasian nama package file Kotlin.
- Pada baris 3-8, **import** berfungsi untuk mengimpor package atau kelas untuk mengakses komponen dan fungsi dari android SDK dan jetpack compose.
- Pada baris 10, **class CardViewModel(private val dataSource: sourceData) : ViewModel()** berfungsi untuk membuat kelas CardViewModel yang menerima parameter dataSource
- Pada baris 11, **private val _cardList = MutableStateFlow<List<CardProp>>(emptyList())** berfungsi untuk mendeklarasikan stateflow secara mutable yang berisi list objek CardProp, lalu emptyList() untuk menandakan bahwa data awalnya kosong
- Pada baris 12, **val cardList: StateFlow<List<CardProp>> = _cardList** berfungsi untuk memberikan akses agar UI dapat membaca dari _cardList
- Pada baris 14, **private val _selectCard = MutableStateFlow<CardProp?>(null)** berfungsi untuk menyimpan satu objek CardProp yang dipilih oleh user

- Pada baris 15, **val selectedCard: StateFlow<CardProp?> = _selectCard** berfungsi untuk memberikan akses agar UI dapat membaca isi dari _selectCard
- Pada baris 17, **init { loadData() }** berfungsi untuk menjalankan fungsi loadData() saat CardViewModel dibuat
- Pada baris 19, **private fun loadData()** fungsi yang dibuat untuk mengambil data dari dataSource lalu akan disimpan ke _CardList agar UI mendapatkan data terbaru
- Pada baris 20, ***val* cards = dataSource.loadCardProps()** berfungsi untuk mengambil data dari dataSource
- Pada baris 21, **_cardList.value = cards** berfungsi untuk menyimpan data yang diambil tadi ke dalam _cardList
- Pada baris 22 dan 29, **Log.d()** berfungsi untuk mencetak pesan ke logcat untuk debugging.
- Pada baris 19, **fun getDetailByTitle(Title: String): CardProp?** Fungsi yang dibuat untuk mencari objek cardProp berdasarkan judul
- Pada baris 19, **return _cardList.value.find { it.title == Title }** berfungsi untuk mengembalikan nilai _cardList yang titlenya sama
- Pada baris 27, **fun selectCard(card: CardProp, name: String)** fungsi yang dibuat untuk menyimpan data card ke dalam _selectCard, sebagai tanda bahwa user memilih card tersebut

7. screens/CardViewModelFactory.kt

- Pada baris 1, **package com.android.modul4.viewmodel** pendeklarasian nama package file Kotlin.
- Pada baris 3-5, **import** berfungsi untuk mengimpor package atau kelas untuk mengakses komponen dan fungsi dari android SDK dan jetpack compose.
- Pada baris 7, **class CardViewModelFactory(private val dataSource: sourceData) : ViewModelProvider.Factory** berfungsi untuk membuat kelas CardViewModelFactory untuk mengimplementasikan interface ViewModelProvider.Factory.
- Pada baris 8, **override *fun* <T : ViewModel> create(modelClass: Class<T>): T** berfungsi untuk melakukan override terhadap method create() lalu parameter modelClass akan menyatakan kelas ViewModel apa yang ingin dibuat.
- Pada baris 9, **if (modelClass.isAssignableFrom(CardViewModel::class.java))** berfungsi untuk mengecek apakah class yang diminta oleh modelclass apakah adalah CardViewModel atau subclassnya

- Pada baris 10, **return CardViewModel(dataSource) as T** berfungsi untuk membuat instance CardViewModel menggunakan dataSource dan dikembalikan sebagai dengan tipe T
- Pada baris 10, **throw IllegalArgumentException("Unknown ViewModel class")** berfungsi untuk melemparkan error jika ternyata modeclassnya bukan CardViewModel

8. UIComponent.kt

- Pada baris 1, **package com.android.modul3** pendeklarasian nama package file Kotlin.
- Pada baris 3-20, **import** berfungsi untuk mengimpor package atau kelas untuk mengakses komponen dan fungsi dari android SDK dan jetpack compose.
- Pada baris 22-53, **@Composable** merupakan annotation untuk menandai bahwa fungsi yang dibuat dibawahnya merupakan fungsi UI di jetpack compose.
- Pada baris 23, **fun Desc(desc: String)** merupakan fungsi yang berisikan Text dengan style tertentu untuk deskripsi.
- pada baris 24-65, **Text()** merupakan komponen UI yang berfunsi menampilkan teks.
    - pada baris 2-61, **text** merupakan isi dari teks yang akan ditampilkan
    - pada baris 26 dan 46, **fontSize** merupakan ukuran dari font yang akan ditampilkan
    - pada baris 27 dan 47, **textAlign** untuk membuat apakah teks berada di kiri, Tengah , atau kanan
    - pada baris 48 dan 63, **fontWeight** merupakan ketebalan dari teks yang akan ditampilkan
    - pada baris 49 dan 62, **modifier** berfungsi untuk menerapkan modifikasi terhadap tampilan komponen, seperti *.fillMaxWidth* agar column lebar Button mengisi ukuran layar.
- Pada baris 30, **@OptIn(ExperimentalGlideComposeApi::*class*)** berfungsi untuk memberitahukan compiler bahwa fungsi composable dibawahnya masih menggunakan API yang bersifat eksperimental untuk menghindari error saat kompilasi.
- Pada baris 32, **fun Img(url: String, width: Int)** merupakan fungsi yang berisikan gambar dengan style yang sudah ditentukan.
- Pada baris 33, **GlideImage** merupakan library glide yang digunakan untuk menampilakn gambar dar internet melalui URL
    - Pada baris 34, **model** merupakan tempat Dimana URL diletakkan

    - Pada baris 35, **contentDescription** merupakan deskripsi dari gambar yang akan ditampilkan
    - Pada baris 36, **modifier** merupakan parameter dalam TextField untuk menerapkan modifikasi terhadap tampilan komponen, seperti *.width* untuk lebar, *.height* untuk tinggi, dan *.clip* untuk mengatur lengkungan gambar.
- Pada baris 43, **fun Title(title: String)** merupakan fungsi uang bersikan Text dengan style yang telah ditentukan untuk judul.
- Pada baris 33, **fun DetailRow(label:String, value:String)** merupakan fungsi untuk berisikan sepasang Text untuk halaman detail.
- Pada baris 55, **Row** merupakan komponen layout yang berlawanan dari Column, Dimana jika Column menyusun elemen UI secara Vertical, maka Row menyusun elemen UI secara horizontal.
    - Pada baris 94-124, **modifier** merupakan parameter dalam TextField untuk menerapkan modifikasi terhadap tampilan komponen, seperti *.fillMaxWidth* agar column lebar Row mengisi ukuran layar.

Jawaban e.

1. Penjelasan Debugger serta cara pakainya
    - Debugger adalah sebuah tool atau alat di android studio yang digunakan untuk menemukan atau memperbaiki bug dalam kode.
    - Cara pakai debugger:
        - Pertama, menentukan breakpoint dari kode kita. Biasanya dengan menekan nomor baris code hingga ada bulatan merah
        - Kedua, menjalankan aplikasi dalam mode debug dengan menekan tombol seperti serangga di bagian atas dekat tombol run
        - Ketiga, setelahnya program akan terhenti tepat di breakpoint yang sudah di atur tadi, Dimana disini kita dapat melihat informasi seperti nilai variable, informasi mengenai fungsi, serta status thread.
2. Penjelasan step into, step over, step out
    - Step Into, berfungsi untuk masuk kedalam fungsi yang dipanggil di baris yang kita beri breakpoint tadi.
    - Step over, berfungsi saat eksekusi terhenti di breakpoint kita dapat melanjutkan ke baris selanjutnya dengan menekan step over
    - Step Out, berfungsi untuk keluar dari dalam fungsi yang kita masuki dengan step into tadi

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/AppleCraft2005/kuliah-santuy/tree/main/semesterIV/Pemrograman-Mobile

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

- Application Class merupakan komponen global yang aplikasi android untuk menyimpan dan mengatur state atau juga konfigurasi dari aplikasi yang dibutuhkan di seluruh bagian aplikasi.
- Application class juga berfungsi untuk:
  - Inisiasi secara global, Dimana application class berfunsi sebagai tempat yang sesuai atau ideal untuk menjalankan library seperti firebase, retrofit dan lain lain yang hanya akan dieksekusi sekali dalam satu lifecycle aplikasi.
  - Menyimpan state secara global, Dimana application class berfungsi sebagai tempat penyimpanan state aplikasi untuk kebutuhan seperti autentikasi yang membutuhkan tokennya dan data preferensi pengguna yang diperlukan komponen aplikasi
  - Mengelola lifecycle aplikasi, Dimana application class dapat digunakan untuk melakukan override pada onCreate() untuk melakukan eksekusi sesuatu pada saat aplikasi dijalankan pertama kali