

LAPORAN PRAKTIKUM
PENGEMBANGAN APLIKASI BERGERAK
PRAKTIKUM 06 – BOTTOM NAVIGATION



L0122034
BINTANG HARIDA RAMADHAN

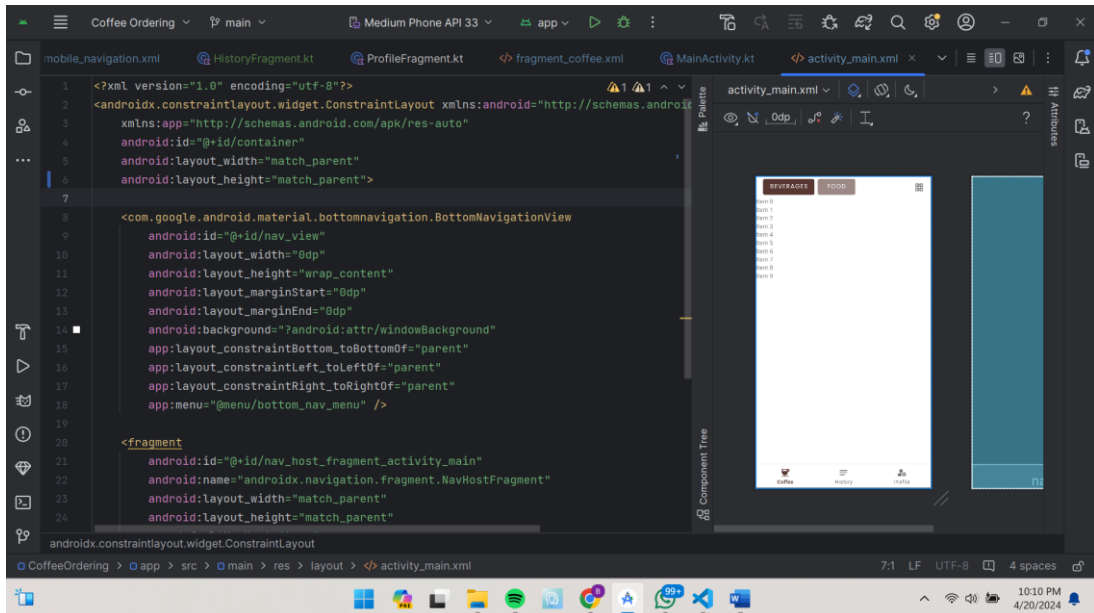
PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA
UNIVERSITAS SEBELAS MARET

2024

BAB I

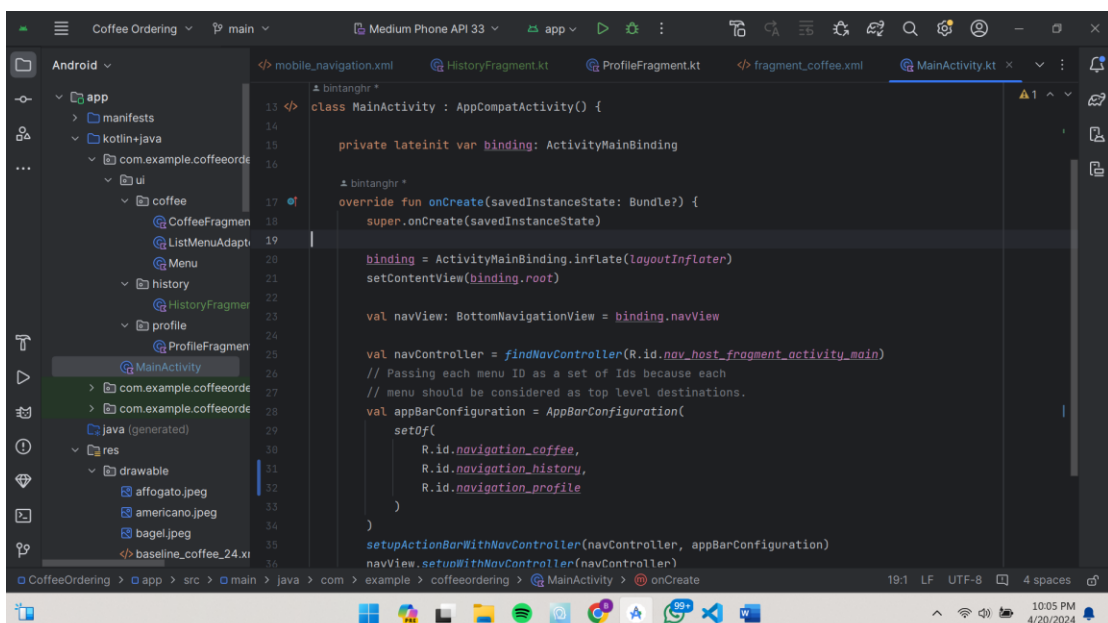
SOURCE CODE

1. File “activity_main.xml”



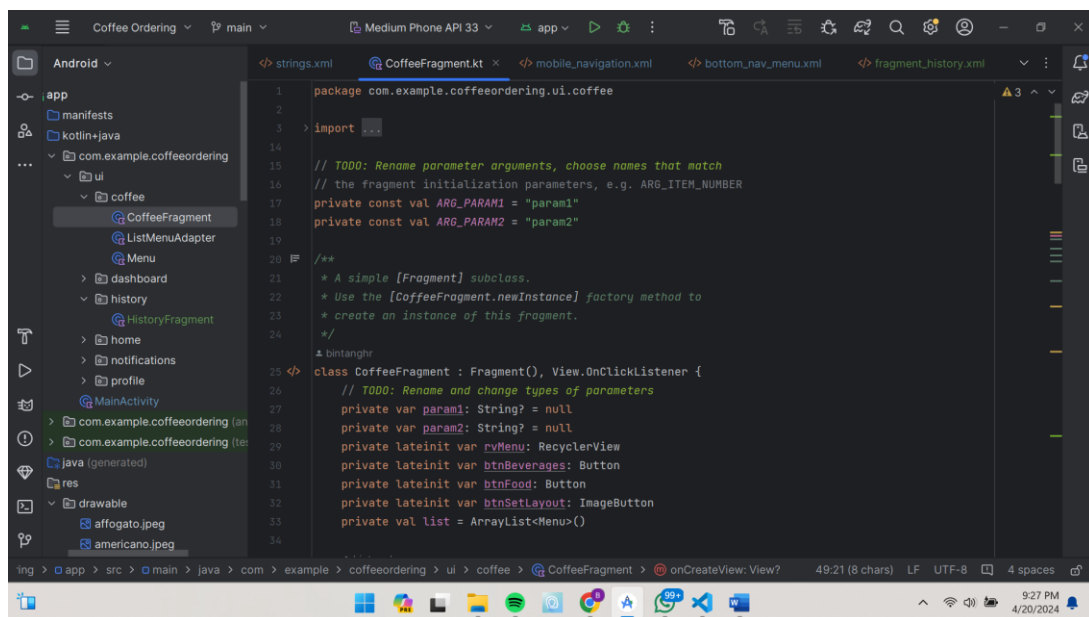
Pada file ini terdapat komponen *bottom navigation*. Komponen tersebut juga memanggil file “menu/bottom_nav_menu.xml”, yang berguna untuk memasukkan masing masing item ke dalam *bottom navigation*. Selain itu, juga terdapat komponen *nav_host_fragment* yang berfungsi sebagai host yang mengatur fragment-fragment lainnya. Fragment yang sedang aktif akan diletakkan disini, untuk navigation graph dari fragment berada pada file *mobile_navigation.xml*.

2. File “MainActivity.kt”



File ini berisi logika penerapan dari *navigation* tersebut. *AppBarConfiguration* adalah daftar ID yang merujuk ke item-item menu dalam *BottomNavigation*. Ini digunakan untuk mengkonfigurasi tampilan *AppBar* agar sesuai dengan menu tersebut. Fungsi *setupActionBarWithNavController* digunakan untuk menyesuaikan judul *AppBar* dengan *Fragment* yang sedang ditampilkan. Sedangkan fungsi *setupWithNavController* berfungsi agar *Bottom Navigation* dapat menampilkan *fragment* yang sesuai ketika salah satu menu dipilih.

3. File “CoffeeFragment”



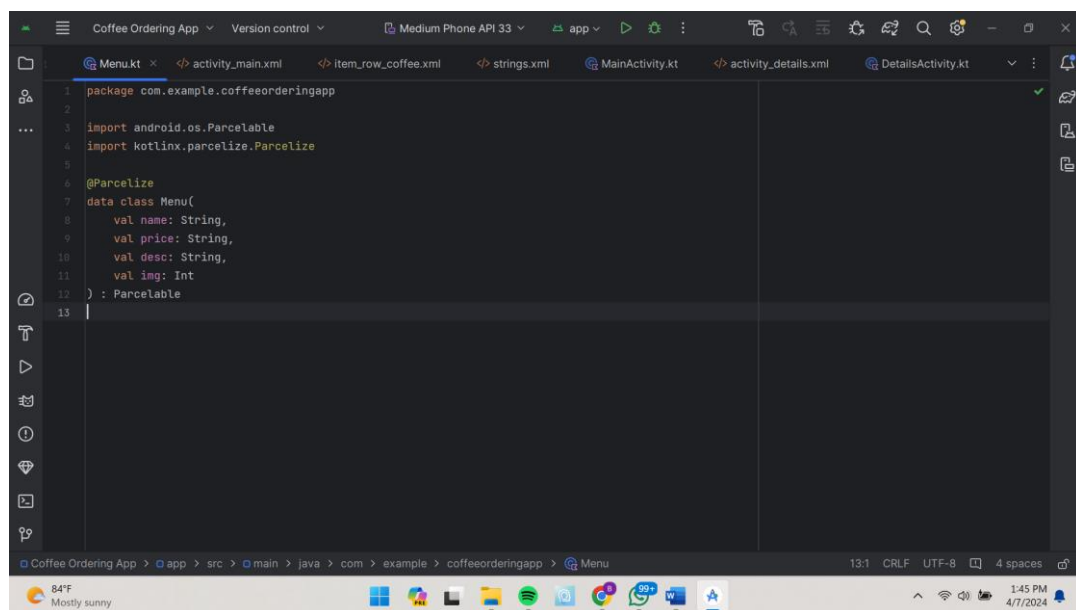
Pada file ini akan menentukan penggunaan *recycler view*. Pada bagian awal definisi class dideklarasikan beberapa *lateinit variable* untuk menampung view yang ada di file “*fragment_coffee.xml*” sesuai dengan tipe view masing masing, selain itu juga dideklarasikan array yang akan digunakan untuk menampung list menu. Masing masing *variable view* diinisialisasikan dengan memanggil fungsi *findViewById()* yang disertai dengan argumen *id* yang sesuai dengan masing masing view. Untuk *Button* diimplementasikan method *setOnClickListener(this)* yang akan digunakan untuk penentuan event saat tombol tersebut ditekan. Setelah itu, variable “list” akan diisi dengan hasil *return value* dari fungsi *getCoffeeList()* sebagai default, karena disini terdapat menu minuman dan makanan. Terakhir, akan dipanggil fungsi *showRecyclerView()*.

Pada fungsi *getCoffeeList()* dan *getFoodList()* diinisialisasikan beberapa variable untuk menampung nama menu, harga, deskripsi, dan foto dengan mendapatkannya dari resource dengan *id* yang sesuai. Selanjutnya akan dilakukan

looping dari seluruh data tersebut untuk dimasukkan pada list. Terakhir, list tersebut akan digunakan sebagai *return value* yang akan digunakan untuk list yang ada di fungsi utama *onCreate()*. Pada fungsi *showRecyclerView()* akan diatur layoutnya dengan *layoutManager* dan menginisialisasikan adapter yang akan diimplementasikan oleh *recycler view*.

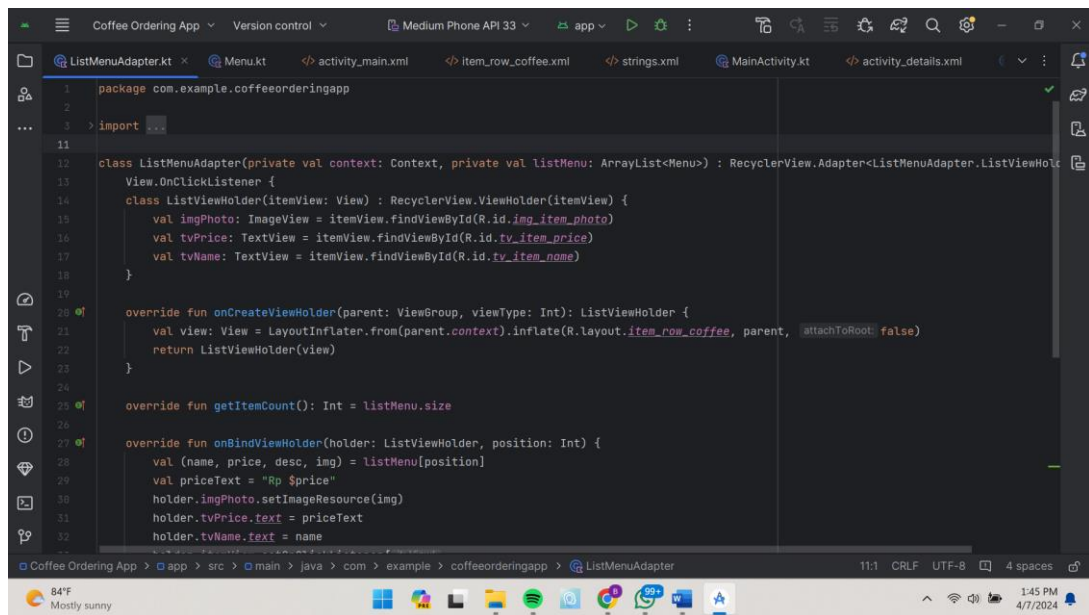
Pada fungsi *onClick* akan diberikan masing masing event bergantung pada tombol mana yang ditekan. Untuk tombol makanan, warna background dari tombol akan ditukar dengan warna yang sedang aktif. Kemudian akan menghapus seluruh list saat ini dan memberikan list yang baru dengan *return value* dari fungsi *getFoodList()* untuk mendapatkan list makanan. Terakhir, akan dipanggil fungsi *showRecyclerView*. Begitupun sebaliknya pada tombol minuman, yang akan memberikan list baru dengan *return value* dari fungsi *getCoffeeList()*.

4. File “Menu.kt”



Class ini digunakan untuk membuat instance object dari masing masing menu. Menu yang dimaksud disini adalah menu makanan dan menu minuman. *@Parcelable* dan *kotlinx.parcelize.Parcelize* dideklarasikan agar class tersebut dapat dibagi bagikan.

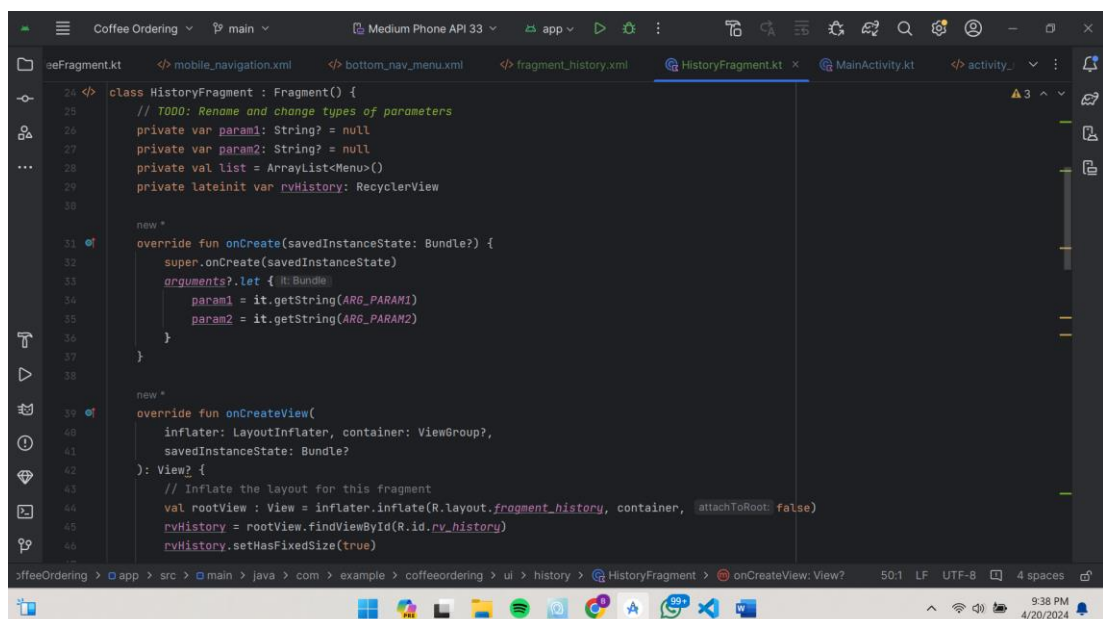
5. File “ListMenuAdapter.kt”

The screenshot shows the ListMenuAdapter.kt file in an IDE. The code defines a class ListMenuAdapter that implements RecyclerView.Adapter<RecyclerView.ViewHolder>. It has a constructor taking Context and ArrayList<Menu> as parameters. Inside, there is a class ViewHolder that implements RecyclerView.ViewHolder and contains references to ImageView, TextView, and another TextView. The adapter overrides onCreateViewHolder, getItemCount, and onBindViewHolder. onCreateViewHolder inflates the item_row_coffee layout. getItemCount returns the size of the listMenu. onBindViewHolder binds the data from listMenu to the ViewHolder's views.

```
1 package com.example.coffeeorderingapp
2
3 import androidx.recyclerview.widget.RecyclerView
4
5 class ListMenuAdapter(private val context: Context, private val listMenu: ArrayList<Menu>) : RecyclerView.Adapter<RecyclerView.ViewHolder>() {
6     class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
7         val imgPhoto: ImageView = itemView.findViewById(R.id.img_item_photo)
8         val tvPrice: TextView = itemView.findViewById(R.id.tv_item_price)
9         val tvName: TextView = itemView.findViewById(R.id.tv_item_name)
10    }
11
12    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
13        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_row_coffee, parent, attachToRoot: false)
14        return ViewHolder(view)
15    }
16
17    override fun getItemCount(): Int = listMenu.size
18
19    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
20        val (name, price, desc, img) = listMenu[position]
21        holder.tvName.text = name
22        holder.tvPrice.text = price
23        holder.imgPhoto.setImageResource(img)
24    }
25 }
```

Pada class ini diberikan parameter konstruktor *context* dan *listMenu*. Di dalam definisi class tersebut terdapat class *ViewHolder* yang akan menampung masing masing dari *view* di dalam *card list*. Pada method *onBindViewHolder* akan diambil data dari posisi yang sesuai. data data tersebut selanjutnya akan digunakan untuk memberi nilai *variable view*. Dari masing masing list tersebut diberikan *onClickListener* untuk *intent* menuju *DetailsActivity*. Sebelum *intent* dimulai, data dari masing masing item akan dikirimkan ke *DetailsActivity* agar data tersebut dapat digunakan disana.

6. File “HistoryFragment.kt”

The screenshot shows the HistoryFragment.kt file in an IDE. The code defines a class HistoryFragment that extends Fragment. It has private variables for param1, param2, list, and rvHistory. It overrides onCreate and onCreateView. onCreate gets the parameters from the Bundle. onCreateView inflates the fragment_history layout and sets the rvHistory variable to the RecyclerView found in the layout.

```
1 class HistoryFragment : Fragment() {
2     // TODO: Rename and change types of parameters
3     private var param1: String? = null
4     private var param2: String? = null
5     private val list = ArrayList<Menu>()
6     private lateinit var rvHistory: RecyclerView
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        arguments?.let {
11            param1 = it.getString(ARG_PARAM1)
12            param2 = it.getString(ARG_PARAM2)
13        }
14    }
15
16    override fun onCreateView(
17        inflater: LayoutInflater, container: ViewGroup?,
18        savedInstanceState: Bundle?
19    ): View? {
20        // Inflate the layout for this fragment
21        val rootView: View = inflater.inflate(R.layout.fragment_history, container, attachToRoot: false)
22        rvHistory = rootView.findViewById(R.id.rv_history)
23        rvHistory.setHasFixedSize(true)
24    }
25 }
```

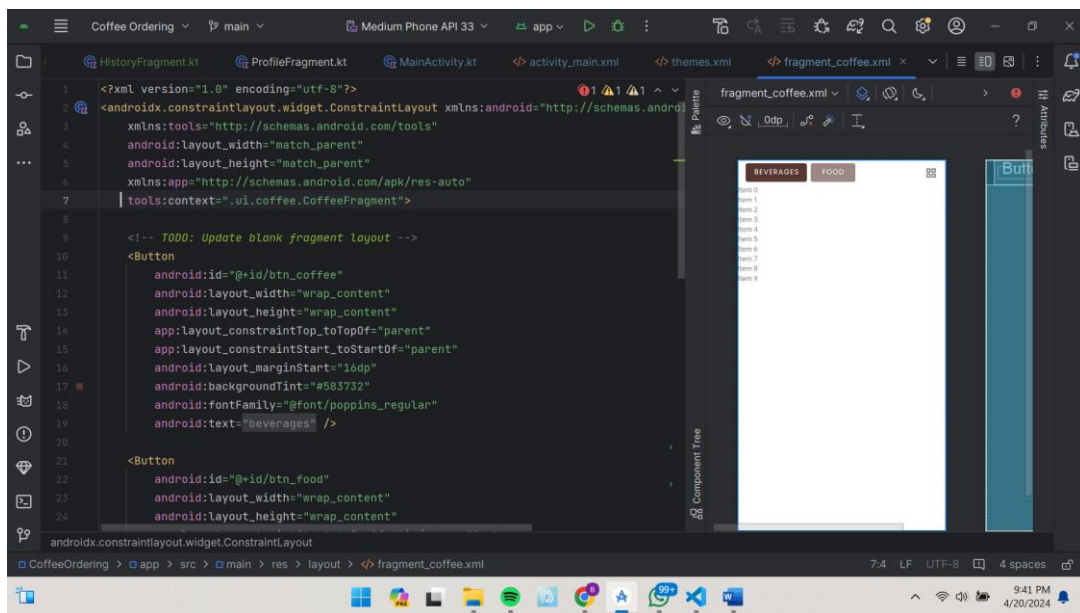
Pada file ini juga akan menggunakan *recycler view* yang sama dengan sebelumnya, hanya berbeda data saja. Untuk data history disimpan pada “string.xml”

yang berisi kategori (makanan atau minuman) dan id menu tersebut. Dari data tersebut akan dilakukan percabangan, jika kategorinya adalah makanan, maka akan diambil seluruh data makanan dengan indeks dari id yang disimpan tersebut. Jika kategorinya adalah minuman, maka akan diambil seluruh data minuman dengan indeks dari id yang disimpan tersebut. Masing masing data tersebut akan diisimpan di dalam *list* kemudian akan digunakan untuk argumen pemanggilan fungsi *List Adapter*.

7. File “ProfileFragment.kt”

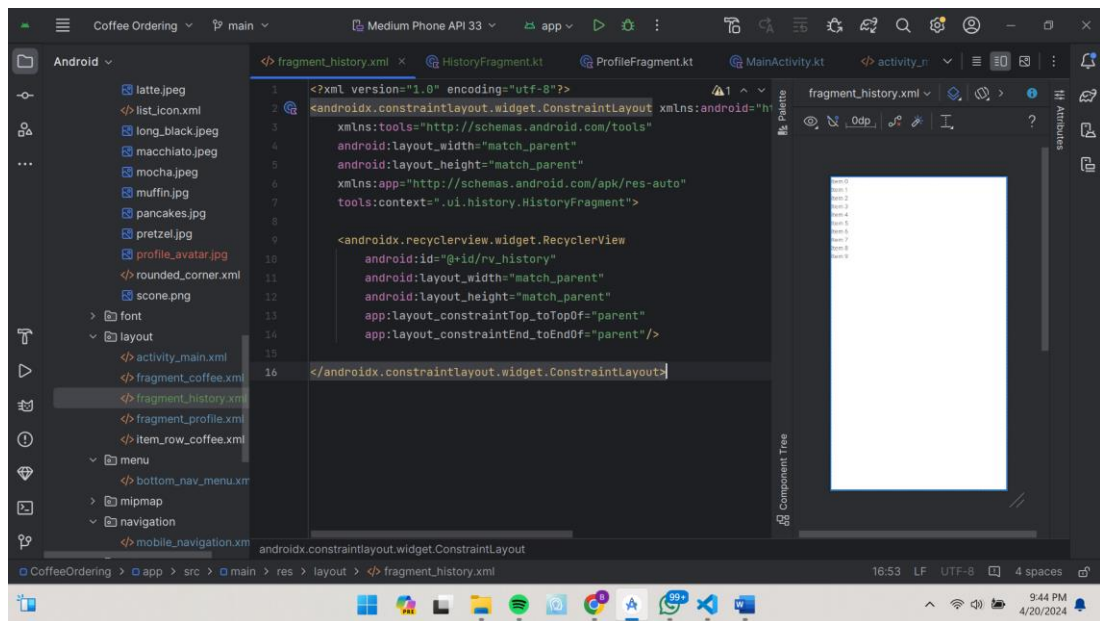
Disini tidak terdapat perubahan apa apa (masih sama dengan default), karena pada fragment ini hanya menampilkan tambilan statis saja.

8. File “fragment_coffee.xml”



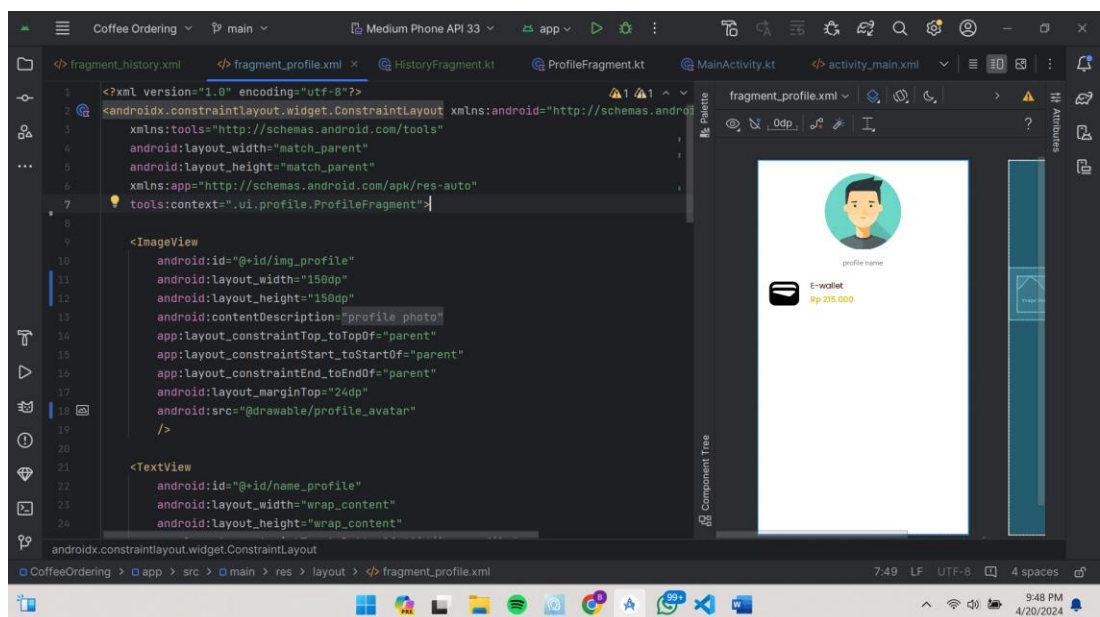
Layout pada fragment ini akan menampilkan beberapa *button* dan *rececycler view* dengan masing masing *id*. Fragment ini nantinya akan digunakan sebagai fragment utama saat aplikasi dimulai.

9. File “fragment_history.xml”



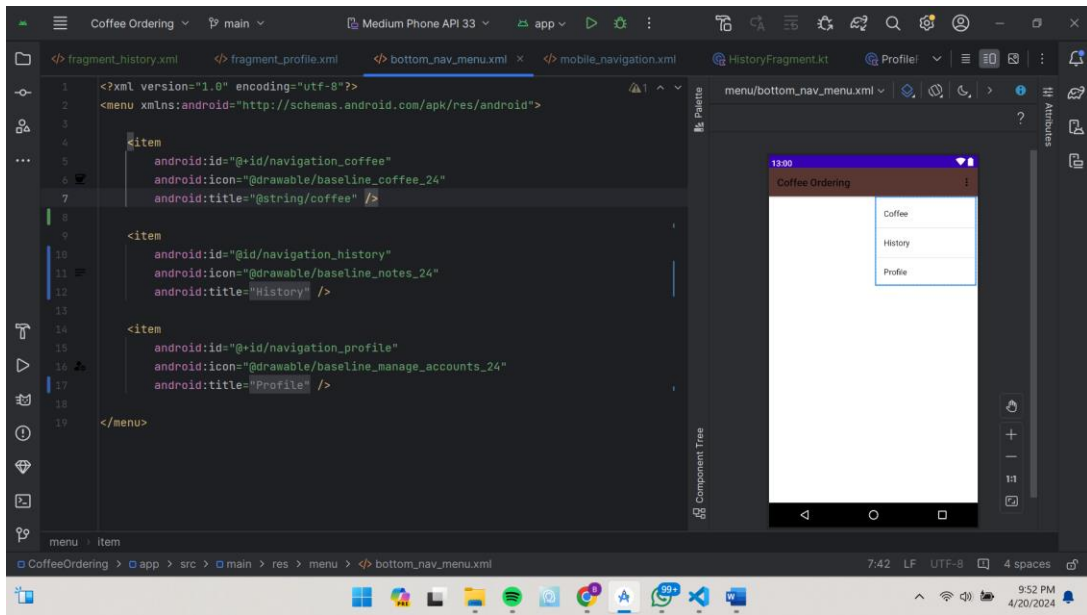
Layout dari fragment ini akan menampilkan *recycler view* saja yang akan menampilkan data data history pembelian yang dimiliki oleh pengguna.

10. File “fragment_profile.xml”



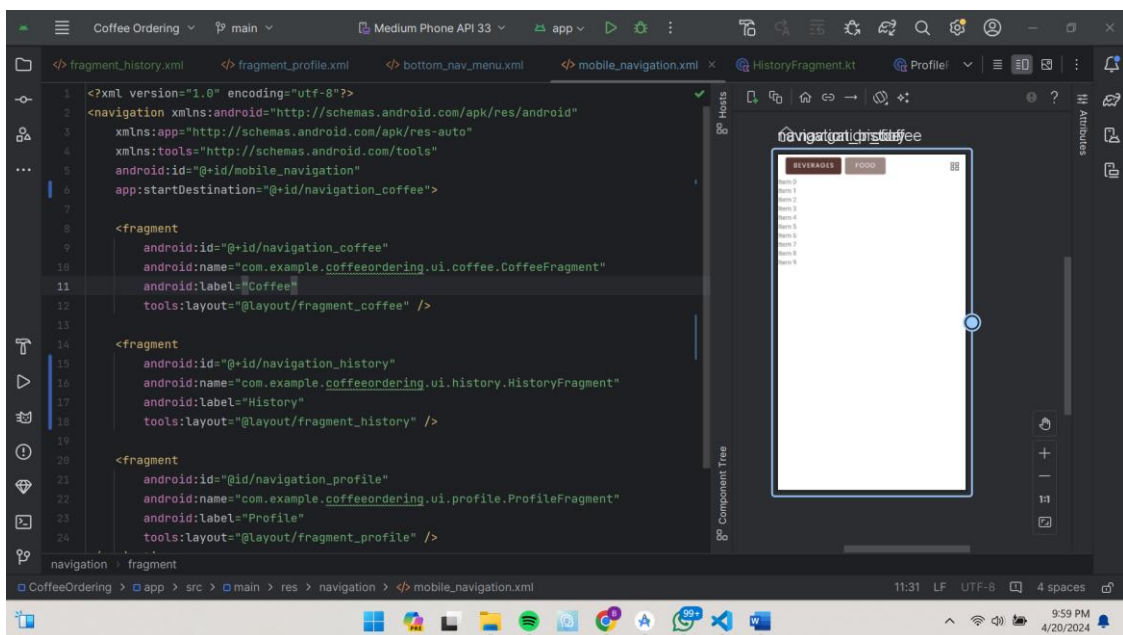
Layout dari fragment ini akan menampilkan profil foto, nama, dan saldo pengguna. *Asset* vektor berbentuk dompet tersebut didapatkan dengan import *vector asset* pada folder */drawable*.

11. File “bottom_nav_menu.xml”



File ini akan menampilkan bottom navigation menu. Masing masing dari item navigation tersebut diberikan id, icon, dan title untuk memudahkan dalam interaksi pengguna.

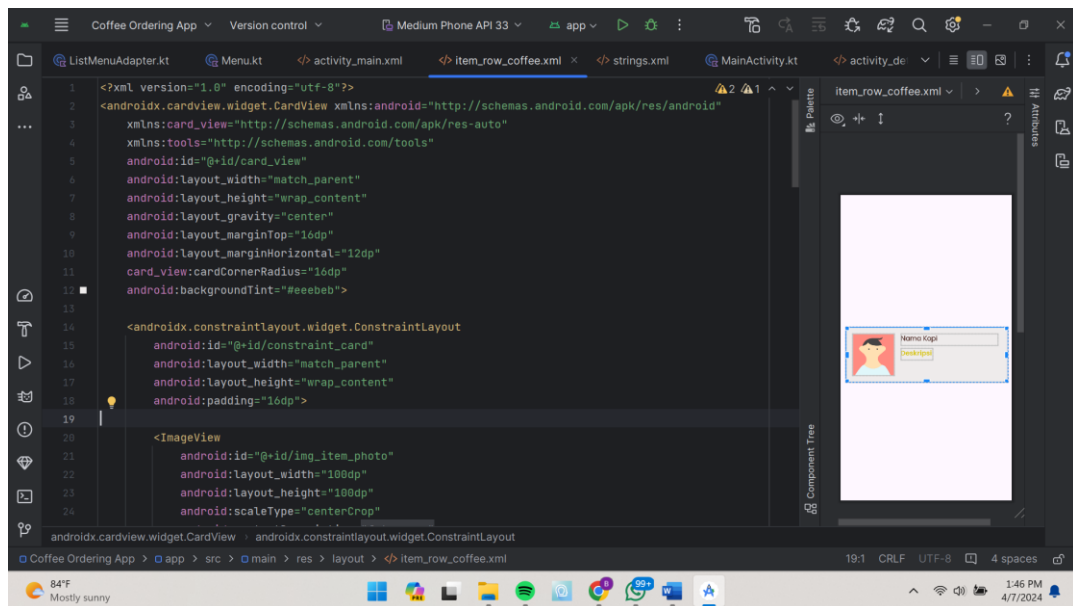
12. File “mobile_navigation.xml”



File ini akan menentukan navigasi dari masing masing fragment. Aplikasi akan mengarahkan ke fragment dengan id yang sesuai dengan item navigation bar yang sedang aktif atau setelah ditekan. Masing masing fragment dideklarasikan dengan id, path file, label, dan layout. Fragment destination tersebut antara lain *navigation_coffee*,

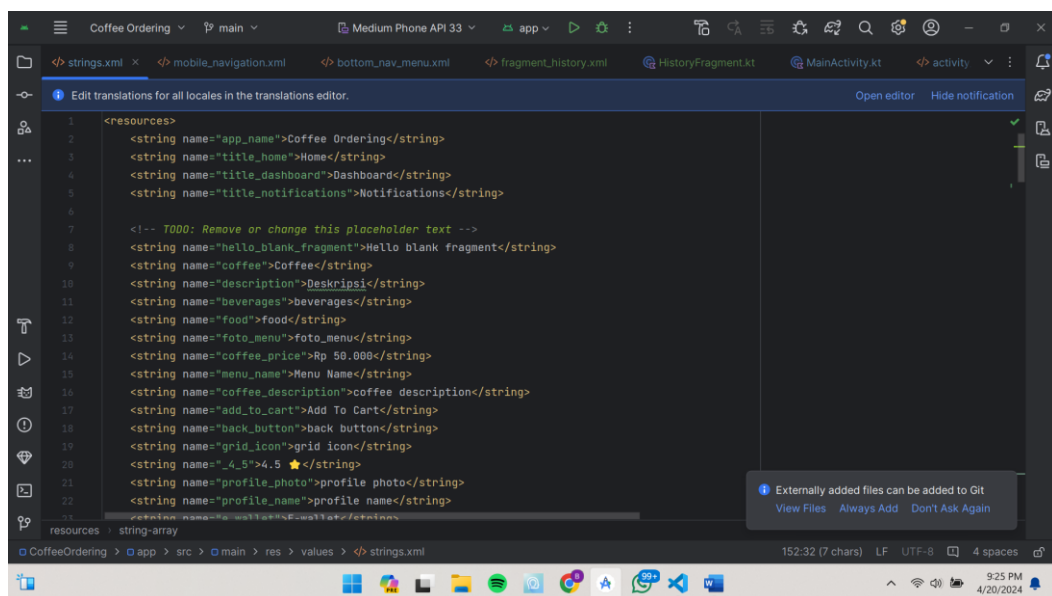
navigation_history, dan *navigation_profile*. Id dari fragment-fragment ini akan sesuai dengan id yang ada di *menu bottom navigation*.

13. File “item_row_coffee.xml”



Pada file ini digunakan *CardView*, di dalamnya terdapat lagi *ConstraintLayout* yang menampung *ImageView* dan *TextView*. Constraint dari masing masing view diberikan *constraint* yang sesuai dengan tata letak yang diinginkan.

14. File “string.xml”



Di dalam file ini diinisialisasikan daftar nama menu, deskripsi, harga, dan foto. Masing masing dari daftar tersebut diberikan nama yang nantinya akan disesuaikan penggunaannya pada *MainActivity*.

BAB II

USER INTERFACE

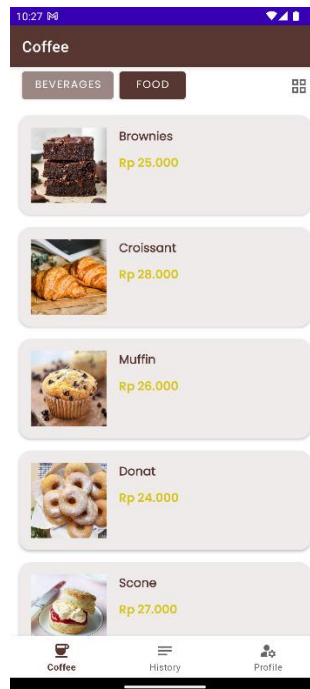
1. Coffee Fragment

- Beverages



Pada bagian ini ditampilkan berbagai macam list menu minuman (sebagai default).

- Food



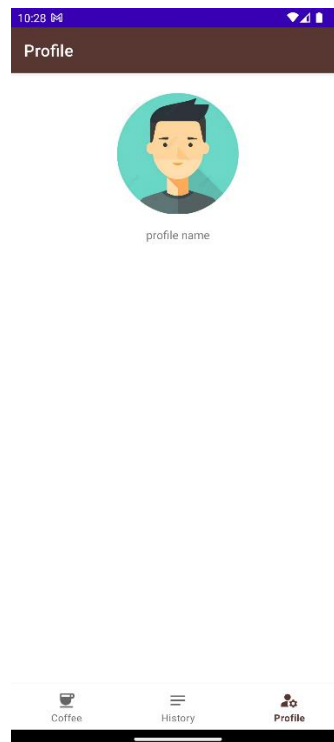
Ketika user menekan tombol “food” maka *RecyclerView* akan menampilkan berbagai macam list menu makanan.

2. History Fragment



Fragment ini akan menampilkan *recycler view* yang berisi history dari pengguna.

3. Profile Fragment



Fragment ini berisi data diri pengguna.

BAB III

KESIMPULAN

Activity adalah layar utama dalam aplikasi, sedangkan *Fragment* adalah potongan-potongan kecil dari layar tersebut. *Fragment* memungkinkan aplikasi untuk membagi tampilan menjadi bagian-bagian yang lebih kecil dan dapat digunakan kembali di berbagai bagian aplikasi. Dengan menggunakan *Fragment*, developer dapat membuat antarmuka pengguna yang lebih dinamis dan mudah diubah.

Bottom Navigation Bar adalah komponen aplikasi *Android* yang ditempatkan pada bagian bawah layar. Komponen ini berisi beberapa item navigasi yang memungkinkan pengguna untuk beralih dengan mudah antara beberapa tujuan atau bagian penting dalam aplikasi. Biasanya setiap item dilengkapi dengan ikon dan teks singkat untuk memberikan informasi visual tentang tujuan atau bagian yang dipilih, serta menyoroti item yang sedang aktif untuk memudahkan pemahaman pengguna tentang posisi mereka dalam aplikasi.