MODUL 4
Building App Navigation

THEME DESCRIPTION

This module covers 3 main primary navigation patterns: bottom navigation, the navigation drawer, and tabbed navigation. These navigation patterns are used to build user-friendly app navigation. Through guided theory and practice, you will learn how each of these patterns works so that users can easily access your app's content. This module will also focus on making the user aware of where they are in the app and which level of your app's hierarchy they can navigate to.

WEEKLY LEARNING OUTCOME (SUB-LEARNING OUTCOME)

Students will know how to use the 3 primary navigation patterns: bottom navigation, the navigation drawer, tabbed navigation and understand how they work with the app bar to support navigation.

TOOLS/SOFTWARE USED

- Android Studio

PRACTICAL STEPS

Part 1 - Navigation drawer

1. Open Android Studio and click **New Project.**
2. Choose the **Empty Views Activity** to start with.
3. Name your project **"LAB_WEEK_04"**.
4. Set the minimum SDK to **"API 24: Android 7.0 (Nougat)"**.
5. Click **Finish**, and let your android application build itself.
6. In this part, we will be focusing on how we can use the **Navigation Drawer** in Android. In order to use the **Navigation Drawer**, we need to add the necessary library to our application. Add the code below to the **dependencies** section in **build.gradle.kts (Module :app)** and **sync** your gradle files.

```
implementation("androidx.navigation:navigation-fragment-ktx:2.5.3")
implementation("androidx.navigation:navigation-ui-ktx:2.5.3")
```

7. Now update your **strings.xml** to the code below.

```
<resources>
```

```
    <string name="app_name">LAB_WEEK_04</string>

    <string name="app_title">Coffee App</string>

    <string name="nav_header_desc">Navigation header</string>

    <string name="menu_coffee">Coffee List</string>
    <string name="menu_favorites">Favorite Coffee</string>

    <string name="affogato_title">AFFOGATO</string>
    <string name="affogato_desc">Espresso poured on a vanilla ice cream. Served in a
cappuccino cup.</string>
    <string name="americano_title">AMERICANO</string>
    <string name="americano_desc">Espresso with added hot water (100-150 ml). Often
served in a cappuccino cup. (The espresso is added into the hot water rather than
all the water being flowed through the coffee that would lead to over
extraction.)</string>
    <string name="latte_title">CAFFE LATTE</string>
    <string name="latte_desc">A tall, mild \'milk coffee\' (about 150-300 ml). An
espresso with steamed milk and only a little milk foam poured over it. Serve in a
latte glass or a coffee cup. Flavoured syrup can be added.</string>

    <string name="coffee_list">Coffee List</string>
    <string name="coffee_favorites">My Favorite Coffee</string>
    <string name="cafe_list">Cafe List</string>

    <string name="bottom_menu_coffee">Coffee</string>
    <string name="bottom_menu_cafe">Cafe</string>

    <string name="starbucks_title">STARBUCKS</string>
    <string name="janjijiwa_title">JANJI JIWA</string>
    <string name="kopikenangan_title">KOPI KENANGAN</string>
</resources>
```

8. Update your **themes.xml** to the code below.

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.LAB_WEEK_04"
parent="Theme.Material3.DayNight.NoActionBar">
        <!-- Customize your light theme here. -->
        <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
    </style>

    <style name="Theme.NavigationDrawer.NoActionBar"
parent="Theme.AppCompat.Light.NoActionBar">
```

```
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
    </style>
    <style name="Theme.NavigationDrawer.AppBarOverlay"
        parent="ThemeOverlay.AppCompat.Dark.ActionBar" />
    <style name="Theme.NavigationDrawer.PopupOverlay"
        parent="ThemeOverlay.AppCompat.Light" />

    <style name="Theme.LAB_WEEK_04" parent="Base.Theme.LAB_WEEK_04" />
</resources>
```

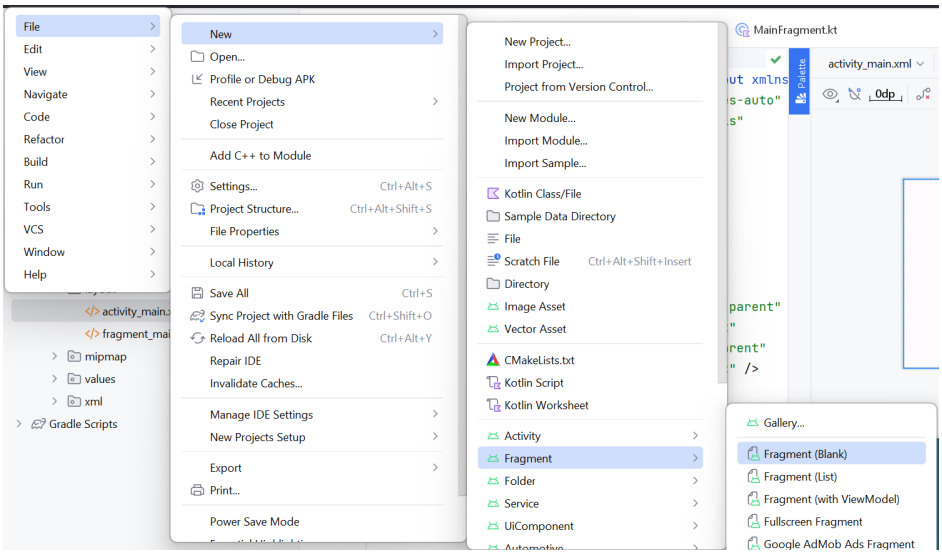
9. And update your **colors.xml** to the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="gray">#939393</color>
    <color name="white">#FFFFFFFF</color>
    <color name="light_brown">#C2663C</color>
</resources>
```

10. We won't be needing the **Action Bar** in our application, it'll be provided by the **Navigation Drawer Layout**. Update your activity in your **AndroidManifests.xml** to the code below.

```
<activity
    android:name=".MainActivity"
    android:exported="true"
    android:theme="@style/Theme.NavigationDrawer.NoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

11. Next, we will be using the same layout as the previous **Coffee App** from your **LAB_WEEK_03**. Create 2 new blank fragments, name it **ListFragment** and **DetailFragment**, and copy both layouts and kotlin files from **LAB_WEEK_03** to the current **LAB_WEEK_04** project.



12. Here's what your **fragment_list.xml** should look like.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ListFragment"
    android:padding="20dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="30sp"
            android:text="@string/coffee_list"/>
        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:layout_marginVertical="20dp"
            android:background="?android:attr/dividerVertical" />
        <TextView
            android:id="@+id/affogato"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
            android:padding="10dp"
            android:text="@string/affogato_title"/>
        <TextView
            android:id="@+id/americano"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="10dp"
            android:text="@string/americano_title"/>
        <TextView
            android:id="@+id/latte"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="10dp"
            android:text="@string/latte_title"/>
    </LinearLayout>
</ScrollView>
```

13. Here's what your **fragment_detail.xml** should look like.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp"
    tools:context=".DetailFragment">
    <TextView
        android:id="@+id/coffee_title"
        android:textStyle="bold"
        android:gravity="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:text="@string/affogato_title"/>
    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:layout_marginVertical="20dp"
        android:background="?android:attr/dividerVertical" />
    <TextView
        android:id="@+id/coffee_desc"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        tools:text="@string/affogato_desc" />
</LinearLayout>
```

14. Here's what your **ListFragment.kt** should look like.

```
class ListFragment : Fragment() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_list, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val coffeeList = listOf<View>(
            view.findViewById(R.id.affogato),
            view.findViewById(R.id.americano),
            view.findViewById(R.id.latte)
        )

        coffeeList.forEach{ coffee ->
            val fragmentBundle = Bundle()
            fragmentBundle.putInt(COFFEE_ID, coffee.id)
            coffee.setOnClickListener(
                Navigation.createNavigateOnClickListener(
                    R.id.action_listFragment_to_detailFragment,
fragmentBundle)
            )
        }
    }

    companion object {
        const val COFFEE_ID = "COFFEE_ID"
```

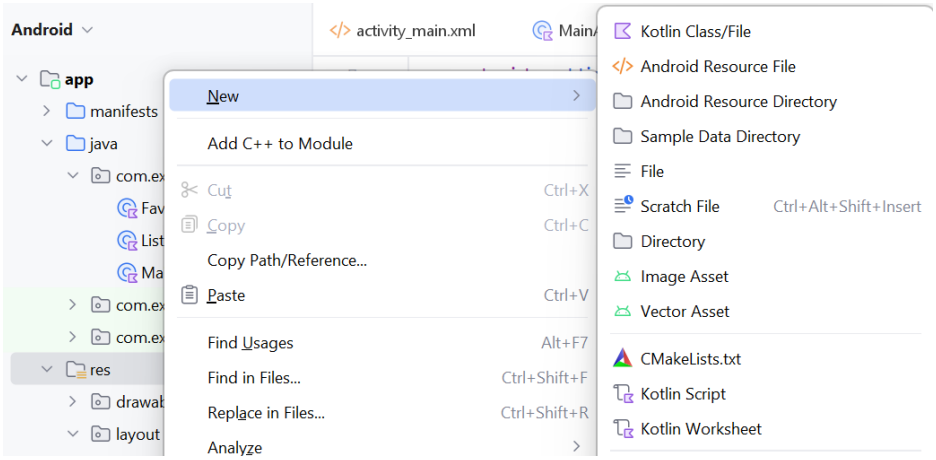
```
    }
}
```

15. Here's what your **DetailFragment.kt** should look like.

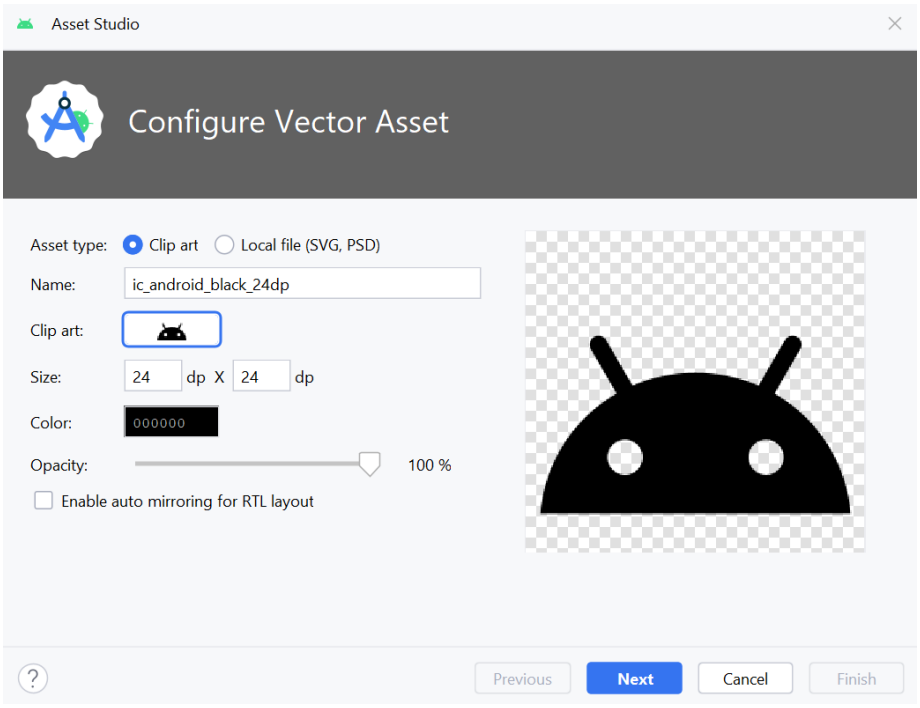
```
class DetailFragment : Fragment() {
    private val coffeeTitle: TextView?
        get() = view?.findViewById(R.id.coffee_title)
    private val coffeeDesc: TextView?
        get() = view?.findViewById(R.id.coffee_desc)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_detail, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val coffeeId = arguments?.getInt(COFFEE_ID, 0) ?: 0
        setCoffeeData(coffeeId)
    }

    fun setCoffeeData(id: Int){
        when(id){
            R.id.affogato -> {
                coffeeTitle?.text = getString(R.string.affogato_title)
                coffeeDesc?.text = getString(R.string.affogato_desc)
            }
            R.id.americano -> {
                coffeeTitle?.text = getString(R.string.americano_title)
                coffeeDesc?.text = getString(R.string.americano_desc)
            }
            R.id.latte -> {
                coffeeTitle?.text = getString(R.string.latte_title)
                coffeeDesc?.text = getString(R.string.latte_desc)
```

```
            }
        }
    }

    companion object {
        private const val COFFEE_ID = "COFFEE_ID"
    }
}
```

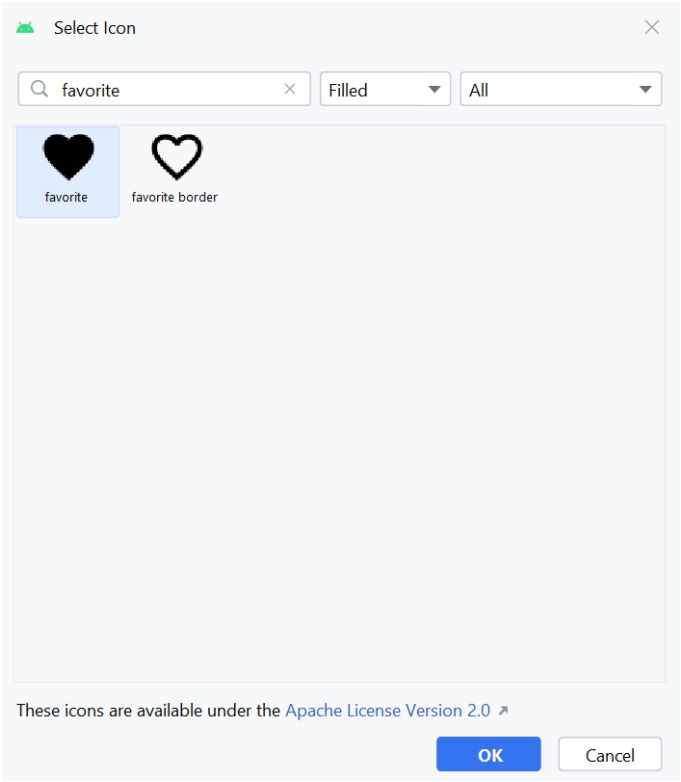
16. Next, create another fragment called **FavoritesFragment**. After that we need to update the layout for **fragment_favorites.xml**. But before that, we also want to add a **favorite icon** for every coffee presented in the list. Therefore we need to make a **Vector Asset**. To do this, right click the **res folder > New > Vector Asset.**



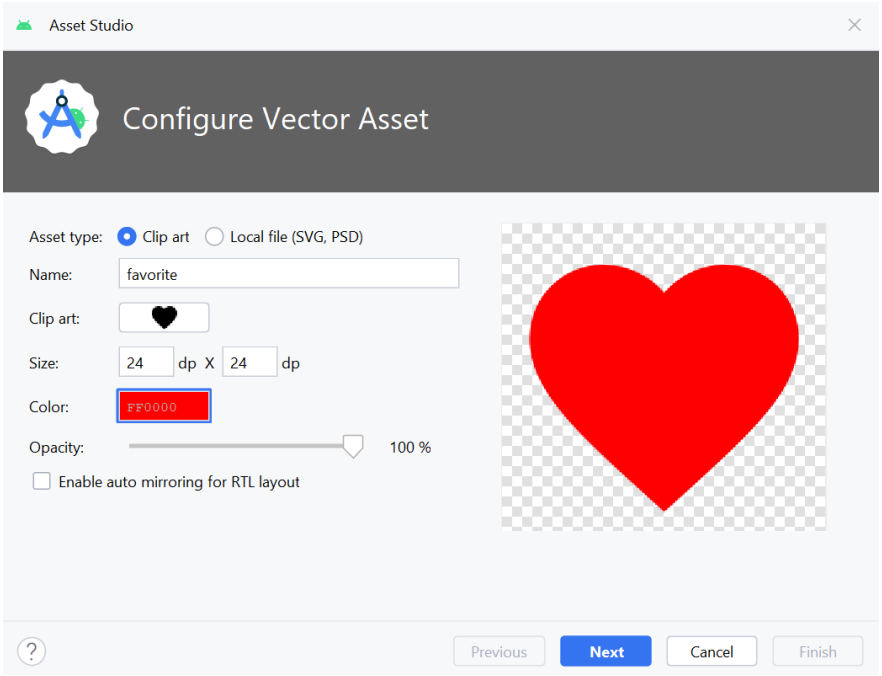
17. A menu should pop up and It should look like this.



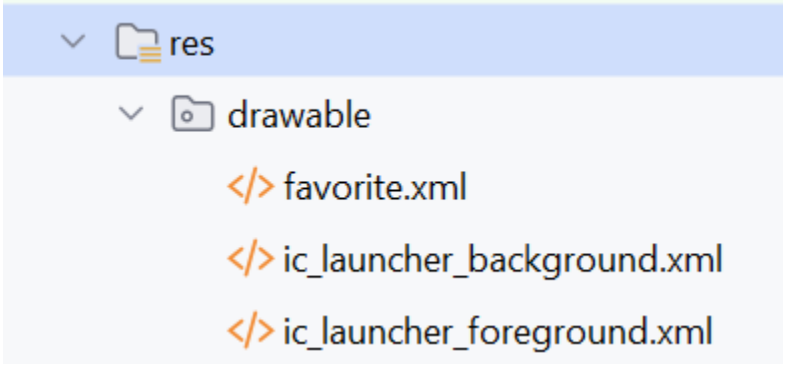
18. Click on the **Clip art** and type in the search box **"Favorite"**, choose the **Favorite Icon** and click **OK**.



19. Name the icon “**Favorite”** and choose **Red** for the color. Click **Next** then **Finish**.



20. Now you can check your **drawable folder** and you should see the **favorite.xml** file in there.

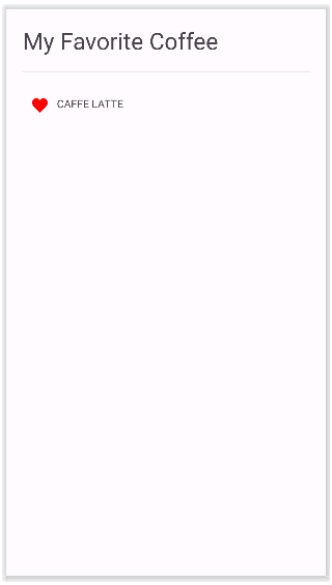


21. Next, update your **fragment_favorites.xml** to the code below.

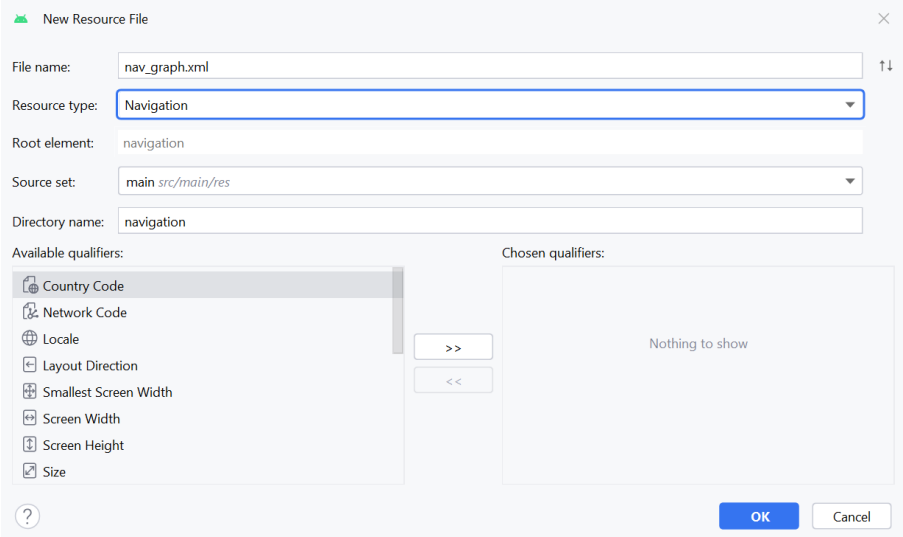
```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FavoritesFragment"
    android:padding="20dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="30sp"
            android:text="@string/coffee_favorites"/>
        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:layout_marginVertical="20dp"
            android:background="?android:attr/dividerVertical" />
        <TextView
            android:id="@+id/latte"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="10dp"
            android:text="@string/latte_title"
            app:drawableStartCompat ="@drawable/favorite"
            android:drawablePadding="10dp"/>
    </LinearLayout>
</ScrollView>
```

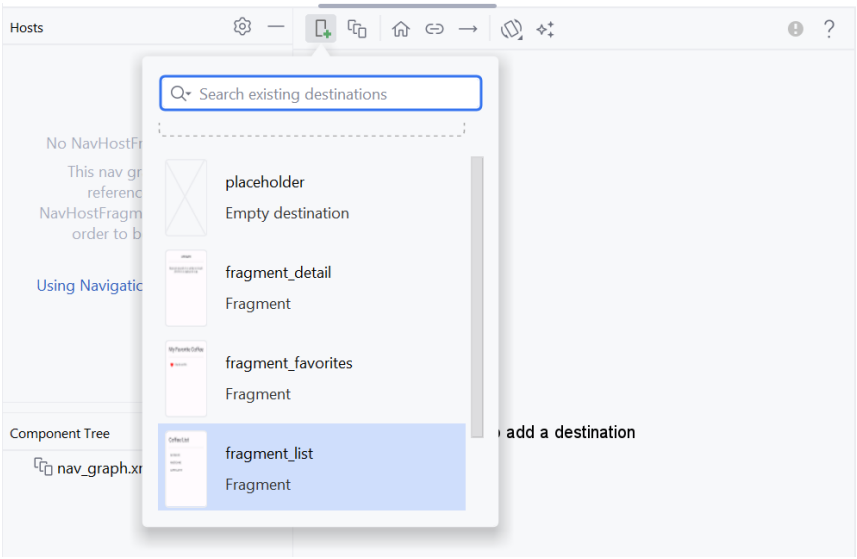
22. Notice the **app:drawableStartCompat** that places our previously created vector image on the left side of the **TextView**.



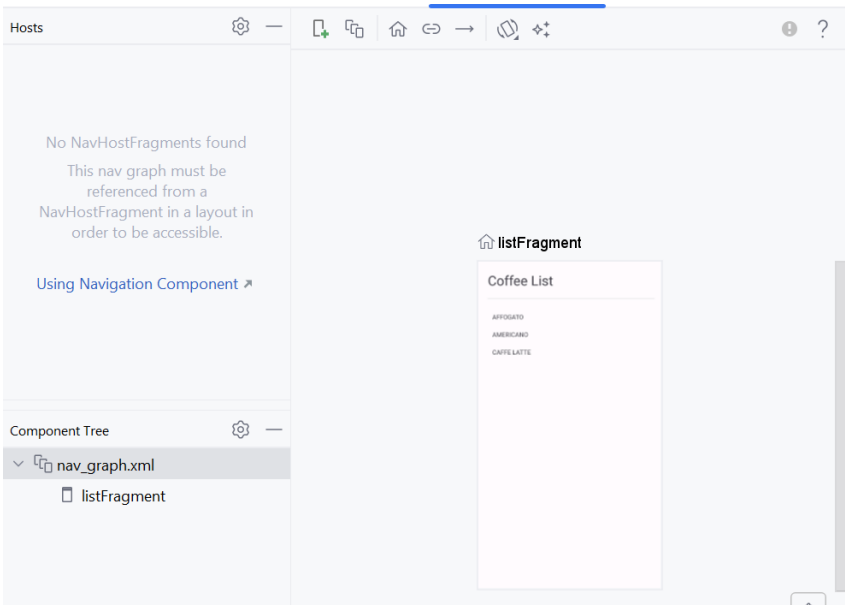
23. Next, let's make a **Navigation** using the previously learned **Jetpack Navigation**. Name your navigation **nav_graph.xml**.



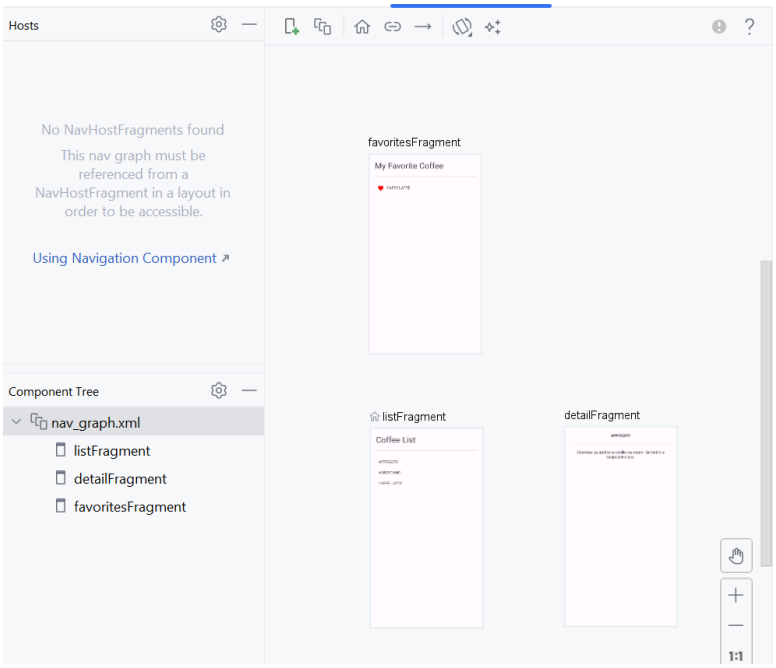
24. Before, you edited the **nav_graph.xml** file using the **code mode**. Now we will be using an easier way to do it and that's with **design mode**. First we need to add our fragments into our board. Click on **New Destination** and choose the **fragment_list** to start with.



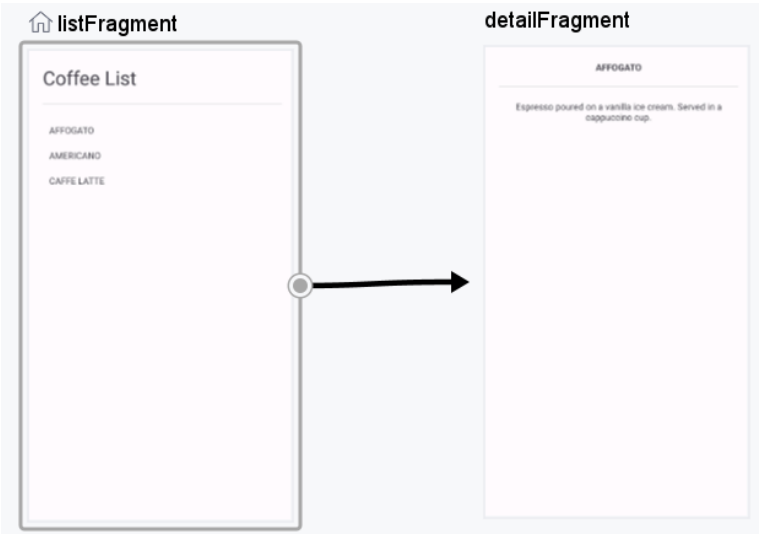
25. **listFragment** should appear on the board.



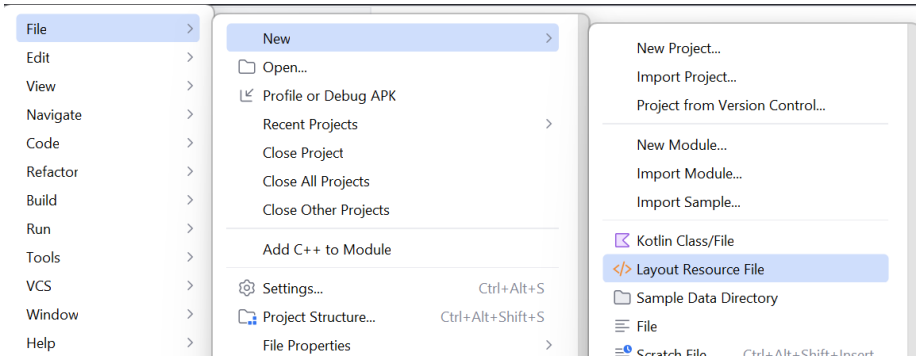
26. Do this with **favoritesFragment** and **detailFragment** too. Your board should now look like this.



27. Now we need to make a **connection** between the fragments. We can start off by connecting **listFragment** to **detailFragment**. **Drag** a line from **listFragment** to **detailFragment** to make one.



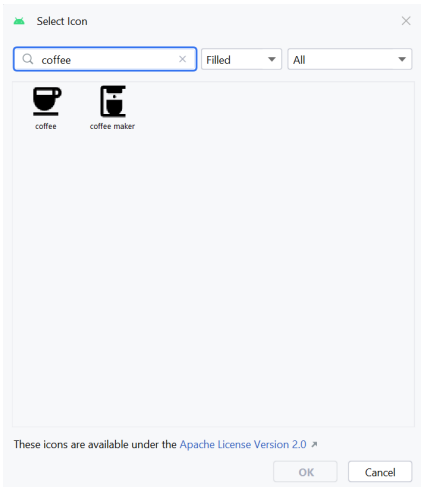
28. Our navigation is done, next we need to create a **Nav Host Fragment** to host all of our fragments. Go to **File > New > Layout Resource File**. Name your layout file **"content_main.xml"** and click **Finish**.



29. **Nav Host Fragment** is created using the **FragmentContainerView** layout. Update your **content_main.xml** to the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/nav_host_fragment"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:defaultNavHost="true"
    app:navGraph="@navigation/nav_graph" />
```

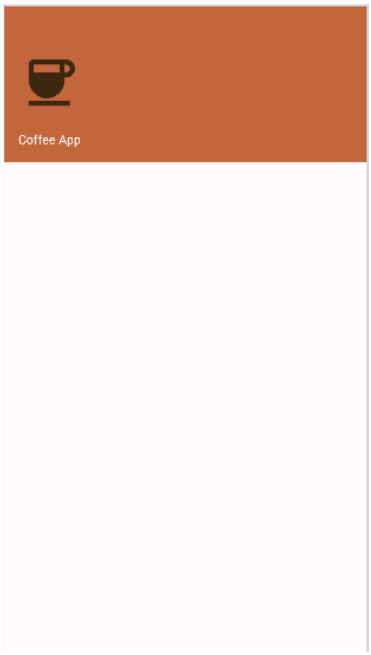
30. Notice the **app:navGraph** is pointing at the **nav_graph.xml** that we just created.
31. Next, let's create the **Navigation Drawer** layout. Start off by creating the header for our **Navigation Drawer.** Create a new layout again (**File > New > Layout Resource File**) with the name **"nav_header_main.xml"**. After that, create a new **Vector Asset** and search for **"Coffee"**.



32. Name the icon **“coffee”** and set the color to **#3E2811**, then click **Next** > **Finish**.
33. Next, update your **nav_header_main.xml** to the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="176dp"
   android:background="@color/light_brown"
   android:gravity="bottom"
   android:orientation="vertical"
   android:paddingStart="16dp"
   android:paddingTop="16dp"
   android:paddingEnd="16dp"
   android:paddingBottom="16dp"
   android:theme="@style/ThemeOverlay.AppCompat.Dark">
   <ImageView
       android:id="@+id/imageView"
       android:layout_width="70dp"
       android:layout_height="100dp"
       android:contentDescription="@string/nav_header_desc"
       android:paddingTop= "8dp"
       app:srcCompat="@drawable/coffee" />
   <TextView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:paddingTop= "8dp"
       android:text="@string/app_title"
       android:textAppearance= "@style/TextAppearance.AppCompat.Body1" />
</LinearLayout>
```

34. So far your navigation header should look like this.

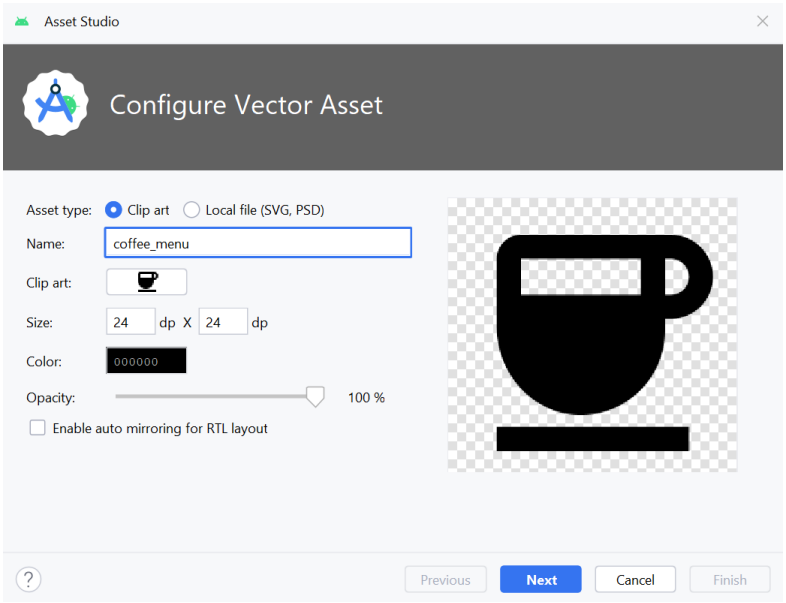


35. Next, let's create the **app bar** for our application. Create a new **layout resource file** and name it **"app_bar_main.xml"**. Update it to the code below.

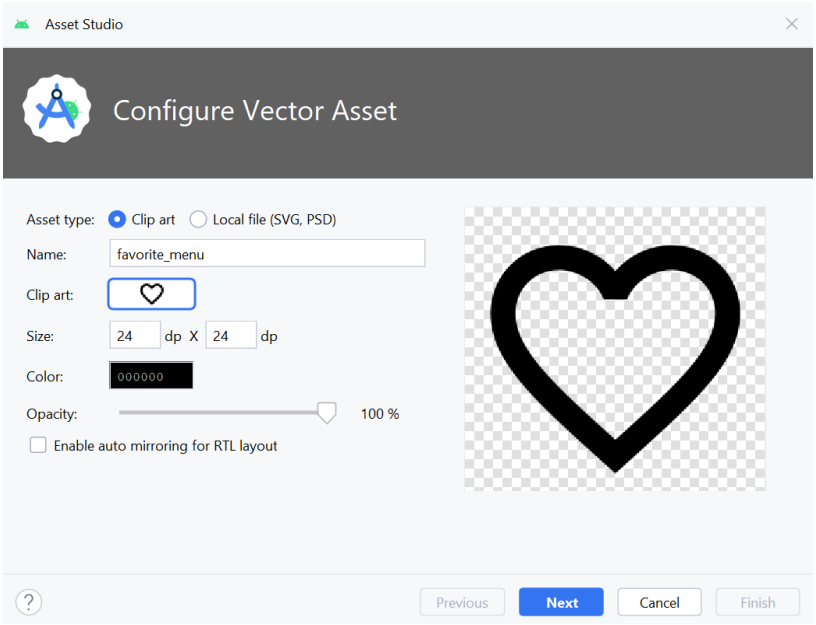
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context=".MainActivity">
   <com.google.android.material.appbar.AppBarLayout
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:theme= "@style/Theme.NavigationDrawer.AppBarOverlay">
           <androidx.appcompat.widget.Toolbar
               android:id="@+id/toolbar"
               android:layout_width="match_parent"
               android:layout_height="?attr/actionBarSize"
               android:background="@color/light_brown"
               app:popupTheme="@style/Theme.NavigationDrawer.PopupOverlay" />
   </com.google.android.material.appbar.AppBarLayout>
   <include
       android:id="@+id/include"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
```

```
        layout="@layout/content_main"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"/>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

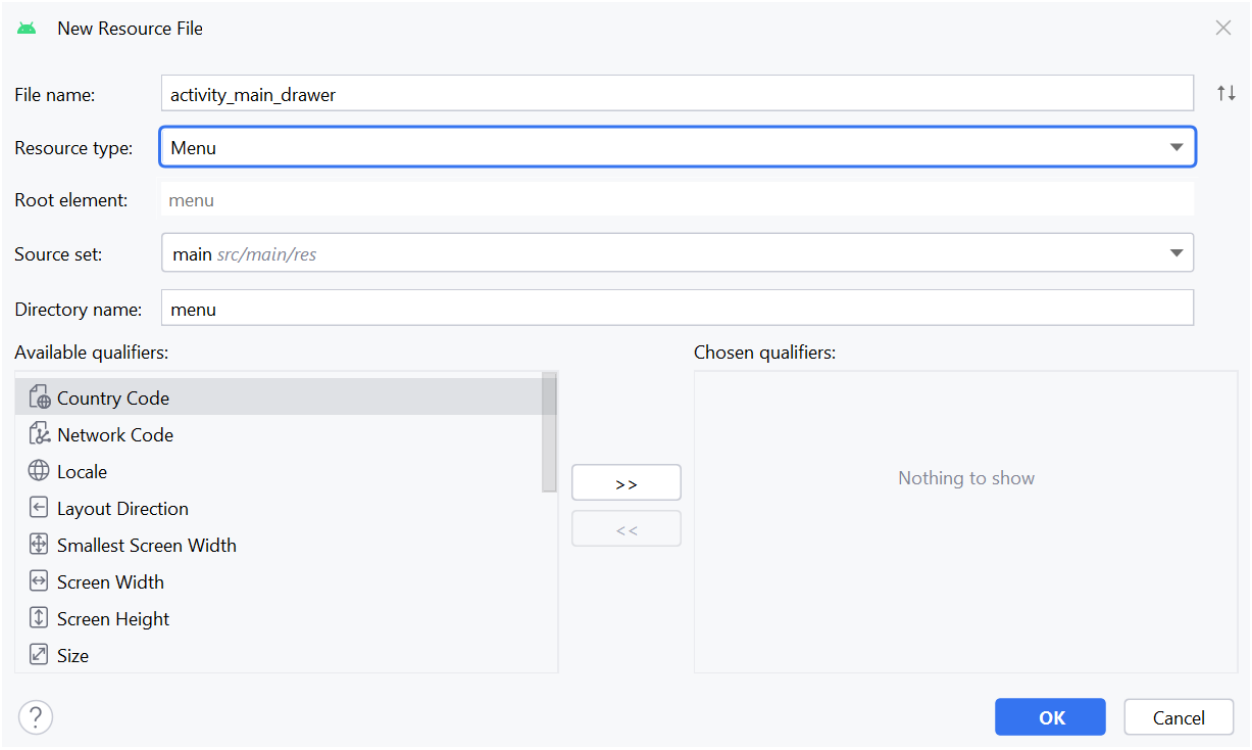
36. The above code basically integrates the **main body layout** of the app with the **app bar** that appears above it. You may also notice that **content_main.xml** is placed separately from the **app bar**.
37. We've created the **header** for our **Navigation Drawer**, we've also created the **app bar** for our application. Next, let's create the **Menu** for our **Navigation Drawer**. Before that, let's add the necessary icons for each menu. For the first icon, create a **Coffee Vector Asset** with the name **"coffee_menu"** and set the color to **black**.



38. For the second icon, create a **Favorite Vector Asset** with the name **"favorite_menu"** and set the color to **black**.



39. Now let's create the **Menu Layout.** Go to **File > New > Android Resource File**. Name the file **"activity_main_drawer"** and set the resource type to **Menu**.



40. Edit your **activity_main_drawer.xml** to the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:showIn="navigation_view">
    <group
        android:id="@+id/menu_top"
        android:checkableBehavior="single">
        <item
            android:id="@+id/listFragment"
            android:icon="@drawable/coffee_menu"
            android:title="@string/menu_coffee" />
    </group>
    <group
        android:id="@+id/menu_bottom"
        android:checkableBehavior="single">
        <item
            android:id="@+id/favoritesFragment"
            android:icon="@drawable/favorite_menu"
            android:title="@string/menu_favorites" />
    </group>
</menu>
```

41. Notice that each **item** has the same **ID** as the **navigation ID** in **nav_graph.xml**. With this, each item is automatically linked to that specific fragment without any extra work.

42. Next, let's update our **activity_main.xml** to the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">
    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <com.google.android.material.navigation.NavigationView
```

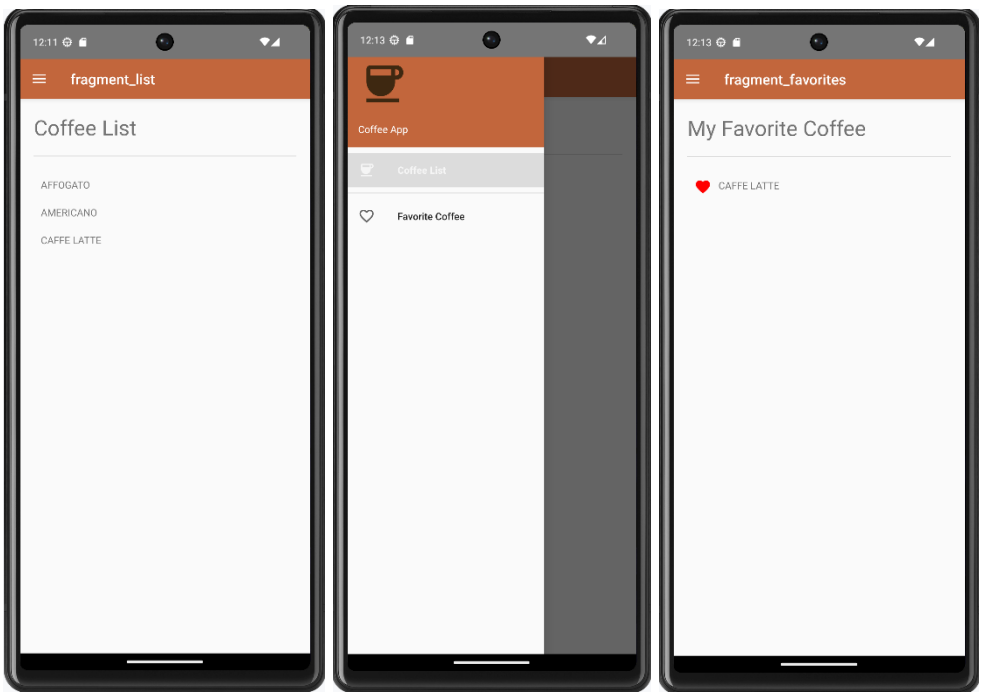
```
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />
</androidx.drawerlayout.widget.DrawerLayout>
```

43. Notice **app:headerLayout** corresponds to our previously created **navigation header** and **app:menu** corresponds to our previously created **drawer menu**.
44. Lastly, let's update the logic for our app in **MainActivity.kt**.

```
package com.example.lab_week_04
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.navigation.findNavController
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.*
import com.google.android.material.navigation.NavigationView

class MainActivity : AppCompatActivity() {
   private lateinit var appBarConfiguration: AppBarConfiguration
   override fun onCreate(savedInstanceState: Bundle?) {
       super.onCreate(savedInstanceState)
       setContentView(R.layout.activity_main)
       setSupportActionBar(findViewById(R.id.toolbar))
       val navHostFragment =
supportFragmentManager.findFragmentById(R.id.nav_host_fragment)
               as NavHostFragment
       val navController = navHostFragment.navController
       //Creating top level destinations
       //and adding them to the draw
       appBarConfiguration = AppBarConfiguration(
           setOf(
               R.id.listFragment, R.id.favoritesFragment
           ), findViewById(R.id.drawer_layout)
       )
       setupActionBarWithNavController(navController, appBarConfiguration)
               findViewById<NavigationView>(R.id.nav_view)
                   ?.setupWithNavController(navController)
   }
   override fun onSupportNavigateUp(): Boolean {
       val navController = findNavController(R.id.nav_host_fragment)
       return navController.navigateUp(appBarConfiguration) ||
```

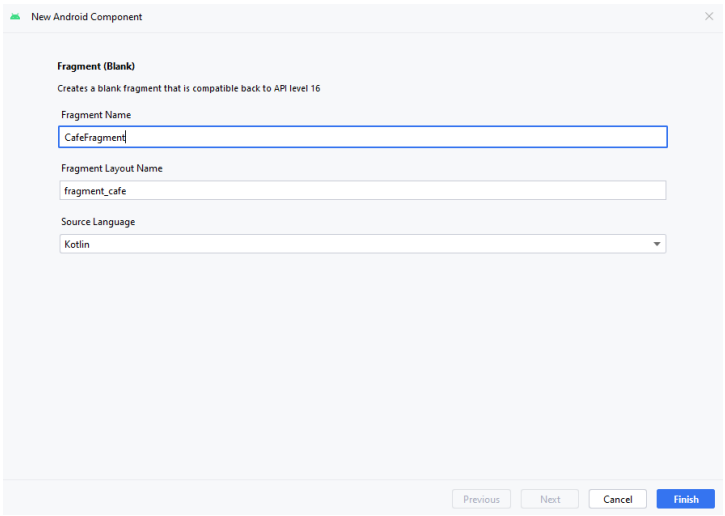
```
super.onSupportNavigateUp()
    }
}
```

45. You're all set to go! Run your application and it should work as intended.



Part 2 - Bottom Navigation

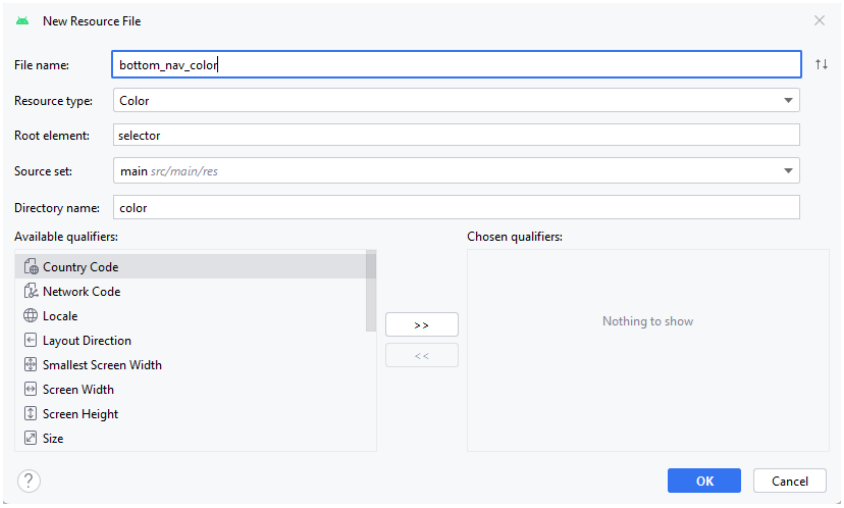
1. Continue your "**LAB_WEEK_04"** project.
2. In this part, we will be focusing on how you can make a **Bottom Navigation** in Android. First, create a new **Blank Fragment** called **CafeFragment**.



3. Update your **fragment_cafe.xml** to the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".CafeFragment"
android:padding="20dp">
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">
   <TextView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:textSize="30sp"
       android:text="@string/cafe_list"/>
   <View
       android:layout_width="match_parent"
       android:layout_height="1dp"
       android:layout_marginVertical="20dp"
       android:background="?android:attr/dividerVertical" />
   <TextView
       android:id="@+id/affogato"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:padding="10dp"
       android:text="@string/starbucks_title"/>
   <TextView
       android:id="@+id/americano"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:padding="10dp"
       android:text="@string/janjijiwa_title"/>
   <TextView
       android:id="@+id/latte"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:padding="10dp"
       android:text="@string/kopikenangan_title"/>
</LinearLayout>
</ScrollView>
```

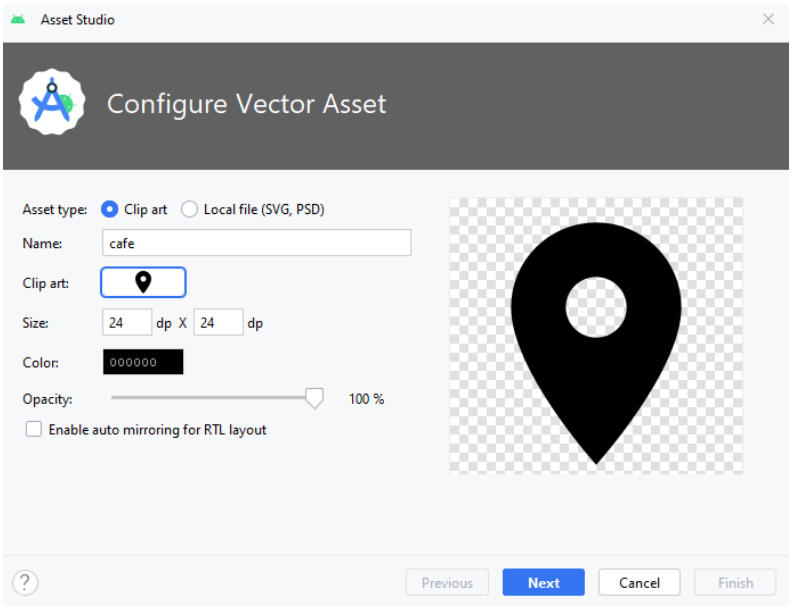
4. Before we add our **Bottom Navigation**, we need to set the style behavior of our **Bottom Navigation Menu**. Go to **File > New > Android Resource File**. Set the **Resource Type** to **Color** and set the name to **"bottom_nav_color"**.



5. Edit the **bottom_nav_color.xml** to the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:state_checked="true" android:color="@color/black" />
   <item android:state_checked="false" android:color="@color/gray"/>
</selector>
```

6. The code above sets the color for the **Selected Menu** to **Black** and sets the color for the **Unselected Menu** to **Gray**.
7. Other than the colors, let's add 1 more **Vector Asset** for our cafe menu. Use the "**location on"** icon and name the icon **"cafe"**.



8. Next, let's add our **Bottom Navigation**. Go to **app_bar_main.xml** and update it to the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context=".MainActivity">
   <com.google.android.material.appbar.AppBarLayout
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:theme= "@style/Theme.NavigationDrawer.AppBarOverlay">
           <androidx.appcompat.widget.Toolbar
               android:id="@+id/toolbar"
               android:layout_width="match_parent"
               android:layout_height="?attr/actionBarSize"
               android:background="@color/light_brown"
               app:popupTheme="@style/Theme.NavigationDrawer.PopupOverlay" />
   </com.google.android.material.appbar.AppBarLayout>
   <include
       android:id="@+id/include"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       layout="@layout/content_main"
```

```
        app:layout_behavior="@string/appbar_scrolling_view_behavior"/>
    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_nav"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:background="?android:attr/windowBackground"
        app:itemTextColor="@color/bottom_nav_color"
        app:itemIconTint="@color/bottom_nav_color"
        app:menu="@menu/bottom_nav_menu"
        app:labelVisibilityMode="labeled"/>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

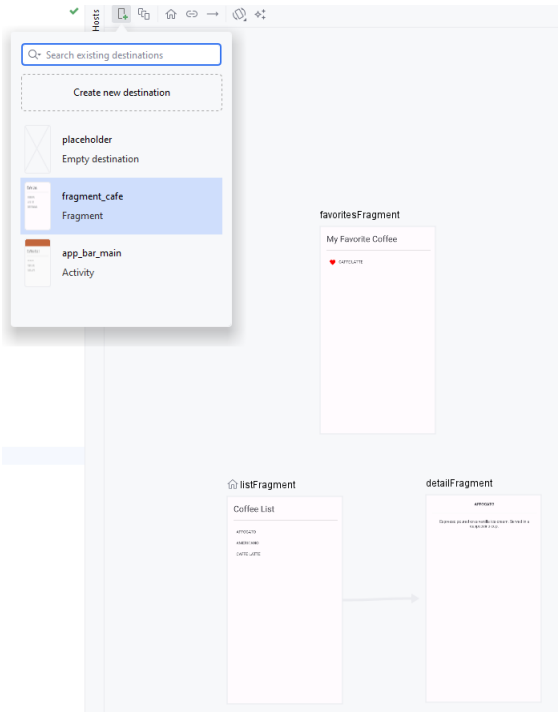
9. The code above added the **Bottom Navigation** as the navigation in **app_bar_main.xml**. The **Coffee Menu** navigates you to the **List Fragment** and the **Cafe Menu** navigates you to the **Cafe Fragment**. You may also notice 2 important attributes from **BottomNavigation**.
    - `app:menu="@menu/bottom_nav_menu"` sets the menu items of the bottom navigation.
    - `app:labelVisibilityMode="labeled"` sets the display mode for the bottom navigation. There are 4 modes: auto, selected, labeled, unlabeled.

10. Our layouts are done. Now let's update our **Kotlin Files**. Go to **MainActivity.kt** and update it to the code below.

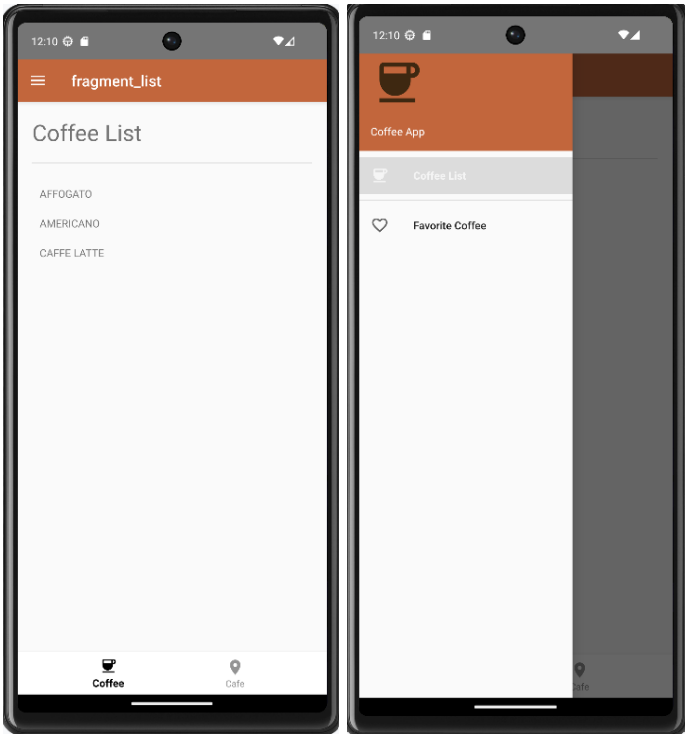
```
class MainActivity : AppCompatActivity() {
    private lateinit var appBarConfiguration: AppBarConfiguration
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        setSupportActionBar(findViewById(R.id.toolbar))
        val navHostFragment =
supportFragmentManager.findFragmentById(R.id.nav_host_fragment)
                as NavHostFragment
        val navController = navHostFragment.navController
        //Creating top level destinations
        //and adding them to the draw
        appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.listFragment, R.id.favoritesFragment, R.id.cafeFragment
            ), findViewById(R.id.drawer_layout)
        )
        setupActionBarWithNavController(navController, appBarConfiguration)
```

```
        findViewById<NavigationView>(R.id.nav_view)
            ?.setupWithNavController(navController)

        // Added this part only
        findViewById<BottomNavigationView>(R.id.bottom_nav)
            ?.setupWithNavController(navController)
    }
    override fun onSupportNavigateUp(): Boolean {
        val navController = findNavController(R.id.nav_host_fragment)
        return navController.navigateUp(appBarConfiguration) ||
super.onSupportNavigateUp()
    }
}
```

11. The code above only adds the **setupWithNavController** for the **BottomNavigationView** and sets the necessary **Nav Controller**. The rest of the code is the same as before.
12. Lastly, because we added a new fragment, let's update our **nav_graph.xml** and add the **Cafe Fragment** into the board.



13. Now run your application, and the **Bottom Navigation** should work alongside with the **Navigation Drawer.**

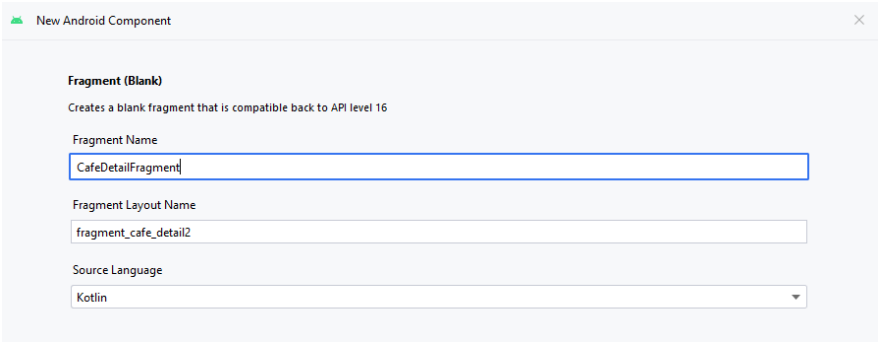


Part 3 - Tabbed Navigation

1. Continue your "**LAB_WEEK_04"** project.
2. In this part, we will be focusing on how you can make a **Tabbed Navigation** in Android. First, let's implement a library for our application. Add the code below to the **dependencies** section in **build.gradle.kts (Module :app)** and **sync** your gradle files.

```
implementation("androidx.viewpager2:viewpager2:1.0.0")
```

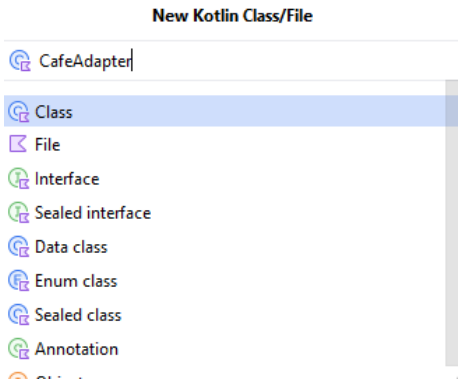
3. Create a new **Blank Fragment** called **CafeDetailFragment**.



4. We will implement the **Tabbed Navigation** inside our **Cafe Fragment** so that we can see the individual description of each cafe. Update your **fragment_cafe.xml** to the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
   <com.google.android.material.tabs.TabLayout
       app:layout_constraintTop_toTopOf="parent"
       android:id="@+id/tab_layout"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       app:tabMode="fixed"/>
   <androidx.viewpager2.widget.ViewPager2
       app:layout_constraintTop_toBottomOf="@id/tab_layout"
       android:id="@+id/view_pager"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

5. The code above has 2 **important** key components:
   - `com.google.android.material.tabs.TabLayout` - this is where the **tabs** will be placed.
   - `androidx.viewpager2.widget.ViewPager2` - this is where the **content** of the **chosen tab** will be placed.
6. Our layouts are done, now let's update our **Kotlin Files**. In order to make the **Tabbed Navigation**, we need something called an **Adapter**. This **Adapter** is used to populate the **View Pager** that we've just created in our layout. Right click the **Res Folder > New > Kotlin File** and set the name to **CafeAdapter**.



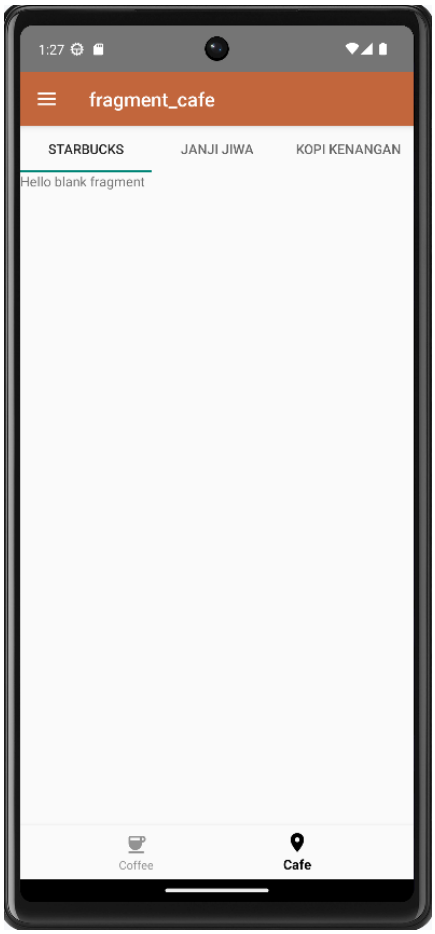
7. Now update your **CafeAdapter.kt** to the code below.

```
val TABS_FIXED = listOf(
   R.string.starbucks_title,
   R.string.janjijiwa_title,
   R.string.kopikenangan_title,
)

class CafeAdapter(fragmentManager: FragmentManager, lifecycle: Lifecycle) :
FragmentStateAdapter(fragmentManager, lifecycle) {

   override fun getItemCount(): Int {
       return TABS_FIXED.size
   }
   override fun createFragment(position: Int): Fragment
   {
       return CafeDetailFragment()
   }
}
```

8. Lastly, we just need to link our **Cafe Adapter** to our **View Pager**. Update your **CafeFragment.kt** to the code below.

```
class CafeFragment : Fragment() {

   override fun onCreate(savedInstanceState: Bundle?) {
       super.onCreate(savedInstanceState)
   }

   override fun onCreateView(
       inflater: LayoutInflater, container: ViewGroup?,
       savedInstanceState: Bundle?
   ): View? {
       // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_cafe, container, false)
   }

   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
       super.onViewCreated(view, savedInstanceState)
       val viewPager = view.findViewById<ViewPager2>(R.id.view_pager)
       val tabLayout = view.findViewById<TabLayout>(R.id.tab_layout)
       val adapter = CafeAdapter(childFragmentManager, lifecycle)
       viewPager.adapter = adapter
       TabLayoutMediator(tabLayout, viewPager) { tab, position ->
           tab.text = resources.getString(TABS_FIXED[position])
       }.attach()
   }
}
```

9. Run your application, and the **Tabbed Navigation** should work in your **Cafe Fragment**.



ASSIGNMENT

Continue your **LAB_WEEK_04** project. You may notice that the **Tabbed Navigation** content is **not changing** when you **switch** the tab. Implement a way to **change** the **content** of **each tab** so that each cafe can have their own description.

Here's some help:

1. Update your **strings.xml** to the code below.

```
<resources>
    <string name="...">...</string>

    <string name="starbucks_title">STARBUCKS</string>
```

```
   <string name="starbucks_desc">Starbucks Corporation is an American
multinational chain of coffeehouses and roastery reserves headquartered in
Seattle, Washington. It is the world\'s largest coffeehouse
chain.</string>
   <string name="janjijiwa_title">JANJI JIWA</string>
   <string name="janjijiwa_desc">It is undeniable that Janji Jiwa outlets have
spread to various corners. Janji Jiwa is a local coffee brand that is popular among
students, students, workers and even families. Carrying the jargon "coffee from the
heart", Janji Jiwa is committed to serving coffee with a classic taste for coffee
lovers.</string>
   <string name="kopikenangan_title">KOPI KENANGAN</string>
   <string name="kopikenangan_desc">At Kopi Kenangan, their dream is to serve high
quality coffee, made with the freshest local ingredients to customers across
Indonesia - and the rest of the world</string>

</resources>
```

2. Update your **fragment_cafe_detail.xml** to the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".CafeDetailFragment"
   android:padding="30dp">
   <TextView
       android:id="@+id/content_description"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:text="" />
</FrameLayout>
```

3. Update your **CafeDetailFragment.kt** to the code below.

```
private const val TAB_CONTENT = "TAB_CONTENT"

class CafeDetailFragment : Fragment() {
   private var content: String? = null
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let {
            content = it.getString(TAB_CONTENT)
        }
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_cafe_detail, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        view.findViewById<TextView>(R.id.content_description)
            ?.text = content
    }

    companion object {
        fun newInstance(content: String) =
            CafeDetailFragment().apply {
                arguments = Bundle().apply {
                    putString(TAB_CONTENT, content)
                }
            }
    }
}
```

4. Next, you only need to modify your **CafeAdapter.kt** and **CafeFragment.kt**. Goodluck!