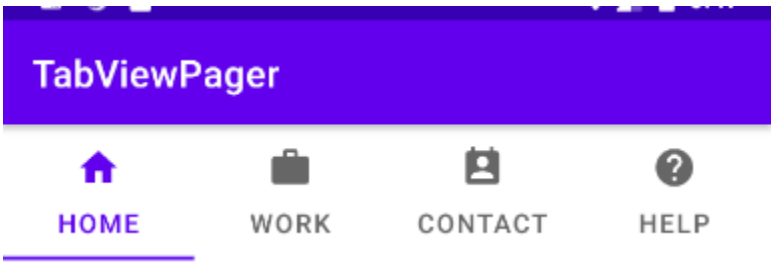
TabLayout Step by Implantation to Design Tabs with Swipe Views

Create a New Project with an Empty Activity

Add the following dependency on your build.gradle plugins

```
// Get xml ids to Kotlin Code
id 'kotlin-android-extensions'
```

Problem Requirement: Design your code with 4 Fragments for the given Tabs with TabLayout and ViewPager2



Step 1: Go to your activity_main.xml and remove the existing TextView Component.

Need to add TabLayout and ViewPager2 inside the Root Constraint Layout. Refer 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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.google.android.material.tabs.TabLayout
        android:id="@+id/tlayout"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </com.google.android.material.tabs.TabLayout>

    <androidx.viewpager2.widget.ViewPager2
        android:id="@+id/vpager"
        android:layout_width="0dp"
```

```
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/tlayout">

    </androidx.viewpager2.widget.ViewPager2>

</androidx.constraintlayout.widget.ConstraintLayout>
```

If you want Linear Layout, go with the below code,

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.google.android.material.tabs.TabLayout
        android:id="@+id/tab_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <androidx.viewpager2.widget.ViewPager2
        android:id="@+id/pager"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />

</LinearLayout>
```

Step 2: Add Vector Asset in the Drawable folder.

The drawable images are added from Vector Asset. To add pictures from the API library, **Right click drawable\New\Vector Asset and add four images as mentioned below on the Tabs.**

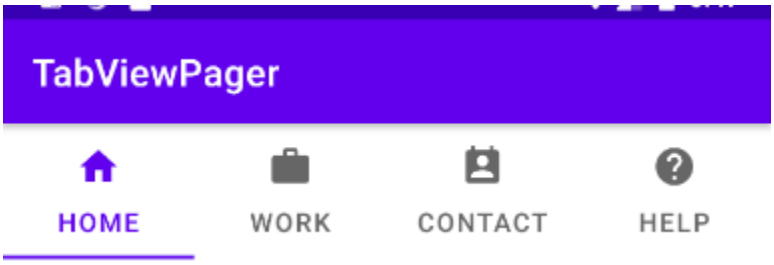
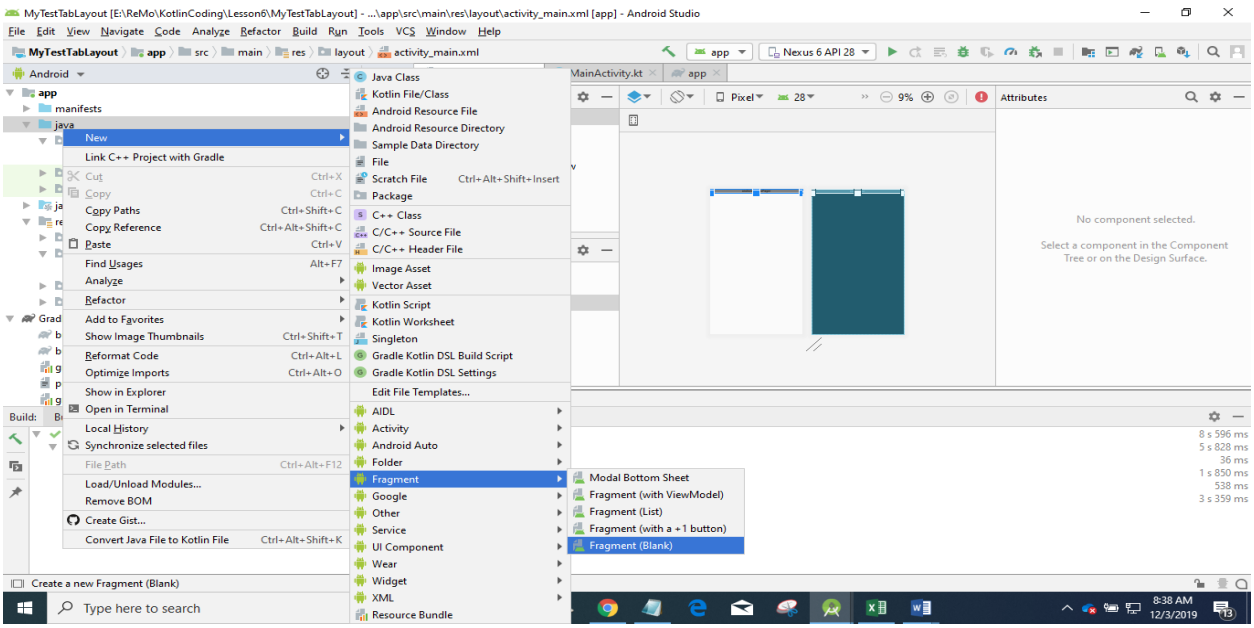


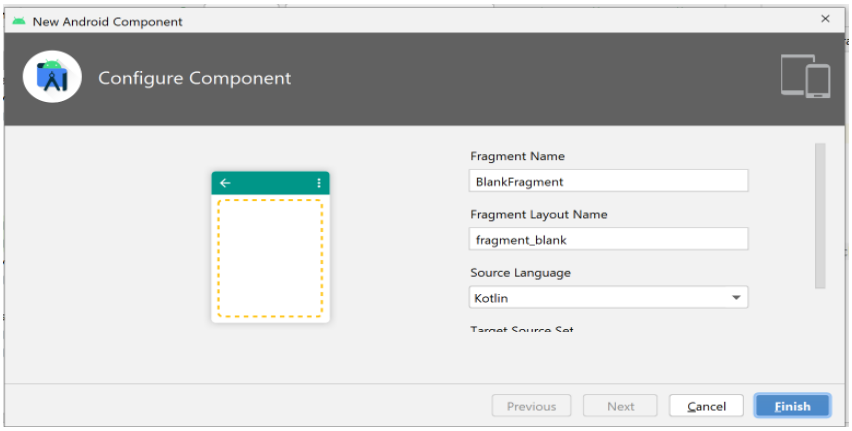
Image Asset Studio helps you create various types of icons at different densities and shows you exactly where they'll be placed in your project.

Vector Asset Studio that helps you add material icons and import Scalable Vector Graphic (SVG) and Adobe Photoshop Document (PSD) files into your project as vector drawable resources. Using vector drawables instead of bitmaps reduces the size of your APK because the same file can be resized for different screen densities without loss of image quality.

Step 3: Create a new Blank Fragment as per the screenshots.



In the next screen, Give the name of your Fragment as HomeFragment. **Click Finish**, Your Gradle Project take some time to Sync.



Fragment loaded with several lines of code. Only keep the below code and delete the remining codes.

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

Step 4: Similar way creates another three fragments and named as

- WorkFragment
- ContactFragmnet
- HelpFragment

Refer the Fragments Codes

```
class WorkFragment : Fragment(R.layout.fragment_work)


class HelpFragment : Fragment(R.layout.fragment_help)


class ContactFragment : Fragment(R.layout.fragment_contact
)
```

After completing the above steps, you will get four Fragements with its Layout.

Do the necessary changes in all Layout to show the UI according to the requirements. Similarly create for other three Fragments.

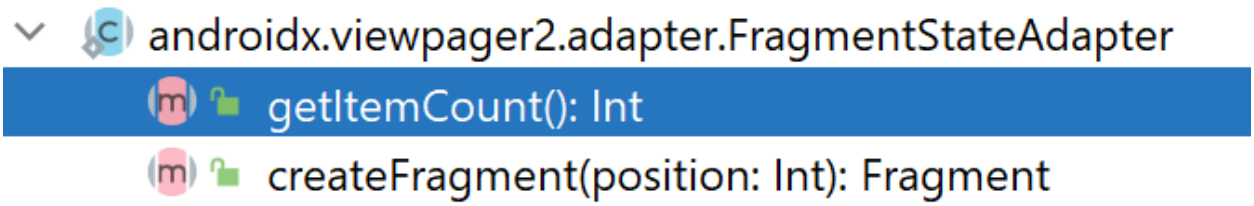
```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/teal_200"
    tools:context=".ContactFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Contact Fragment"
        android:textColor="@color/purple_700"
        android:textSize="25sp" />
</FrameLayout>
```

Step 5: Create your Adapter class inherit from FragmentStateAdapter and pass the argument of FragmentActivity to your class and the parent class.

```
// FragmentStateAdapter handles saving and restoring of fragment's
state.
//Base class for activities that want to use the support-based
Fragments.
```

```
class MyPageAdapter(fragmentActivity:FragmentActivity) :
FragmentStateAdapter(fragmentActivity) {
}
```

Step 6: You will get the error to implement the below two methods as mentioned below



Step 6: Implement both the methods

```
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentActivity
import androidx.viewpager2.adapter.FragmentStateAdapter

class MyPageAdapter(fragmentActivity:FragmentActivity) :
FragmentStateAdapter(fragmentActivity) {
    override fun getItemCount() = 4 // We have 4 fragments

    // Provide a new Fragment associated with the specified
position.
    override fun createFragment(position: Int): Fragment {
        return when (position) {
            0 -> HomeFragment()
            1 -> WorkFragment()
            2 -> ContactFragment()
            3 -> HelpFragment()
            else -> Fragment()

        }
    }
}
```

Step 6: Do the implementation in MainActivity.xml

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.google.android.material.tabs.TabLayout
import com.google.android.material.tabs.TabLayoutMediator
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        // Create an object for the Adapter Class
        setContentView(R.layout.activity_main)
        val myPageAdapter = MyPageAdapter(this)
        // Set the Adapter to your Viewpager UI
        vpager.adapter = myPageAdapter
        // Will align the space according to the Screen size to
equally spread
        tlayout.tabGravity = TabLayout.GRAVITY_FILL
        /* Setting up Tab Layout with the ViewPageg2 is handled by
the TabLayoutMediator class
       * by passing your tablayout id and viewpager id*/
        TabLayoutMediator(tlayout,vpager){tab,position->
            when(position){
                0->{
                    tab.text="Home"
                    tab.setIcon(R.drawable.home)
                }
                1->{
                    tab.text="Work"
                    tab.setIcon(R.drawable.work)
                }
                2->{
                    tab.text="Contact"
                    tab.setIcon(R.drawable.contact)
                }
                3->{
                    tab.text = "Help"
                    tab.setIcon(R.drawable.help)
                }
            }
        }.attach()
    }
    }
```