ITW202: Mobile Application

Unit IV: Developing for Android

Ms. Sonam Wangmo

Gyalpozhing College of Information Technology
Royal University of Bhutan

May 9, 2021

User navigation

Two forms of navigation

Back (temporal) navigation

- Provided by the device's Back button
- Controlled by the Android system back stack

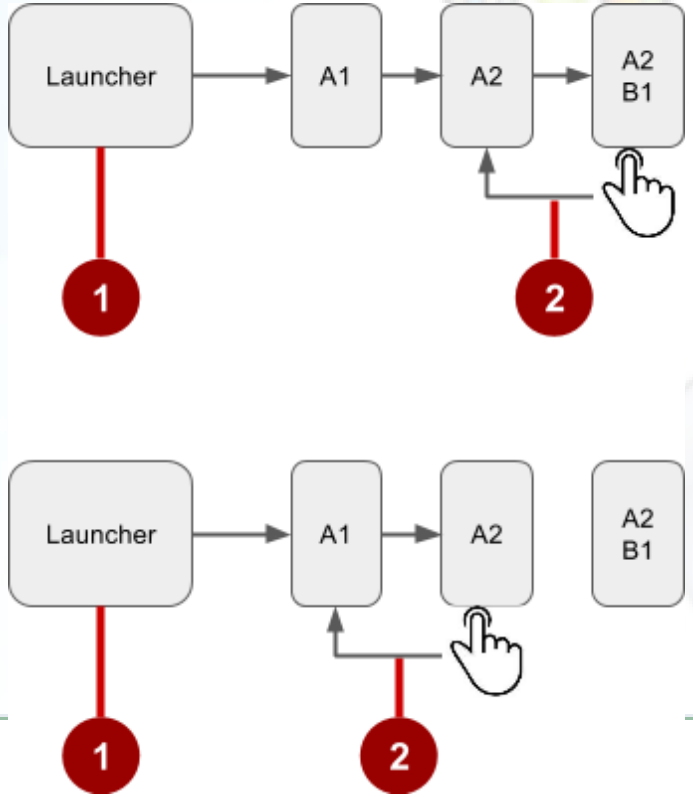
Ancestral (Up) navigation

- Up button provided in app bar
- Controlled by defining parent `Activity` for child `Activity` in the AndroidManifest.xml

Back Navigation

Navigation through history of screens

1. Historys starts from Launcher
2. User clicks the Back button to navigate to previous screens in reverse order



Changing Back button behavior

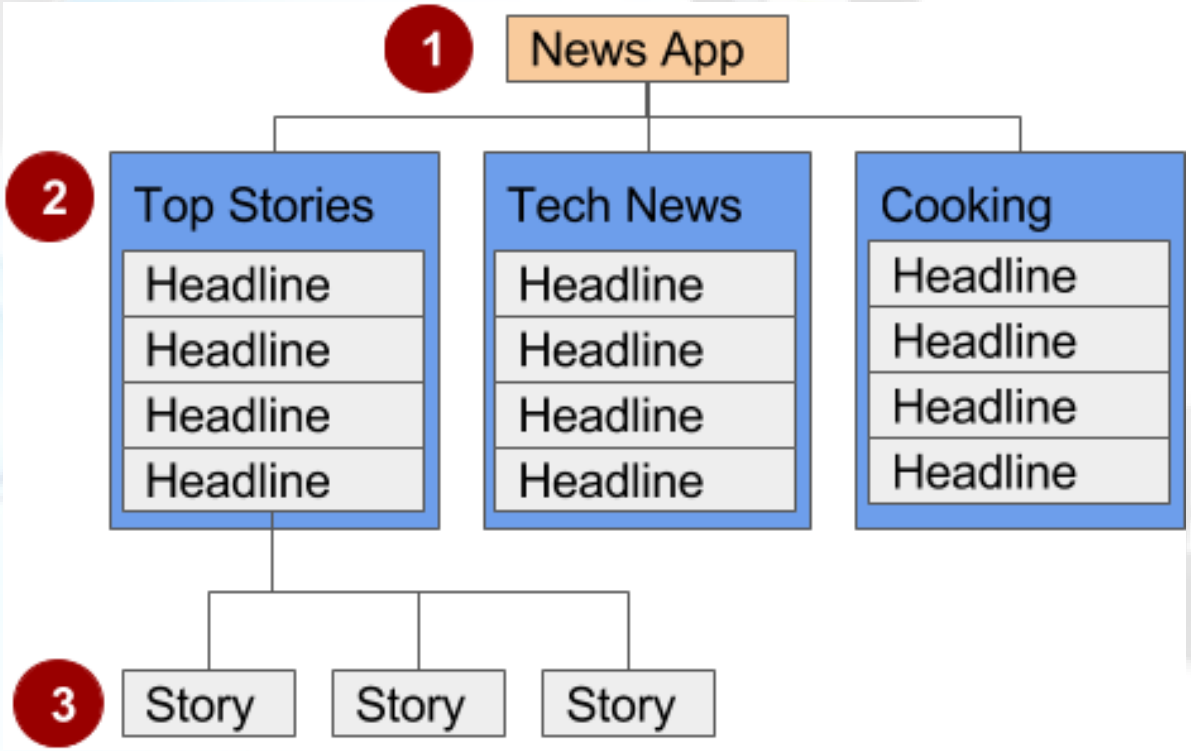
- Android system manages the back stack and Back button.
- There are, however, cases where you may want to override the behavior for the Back button.

```
@Override
public void onBackPressed() {
    // Add the Back key handler here.
    return;
}
```

Hierarchical Navigation

Hierarchical Navigation

An app's screens are typically organized in a parent-child hierarchy, as shown in the figure below:



Hierarchical Navigation

In the figure above:

1. Parent screen
2. First-level child screen siblings
3. Second-level child screen siblings

Hierarchical navigation patterns

- Parent screen—Screen that enables navigation down to child screens, such as home screen and main Activity
- Collection sibling—Screen enabling navigation to a collection of child screens, such as a list of headlines
- Section sibling—Screen with content, such as a story

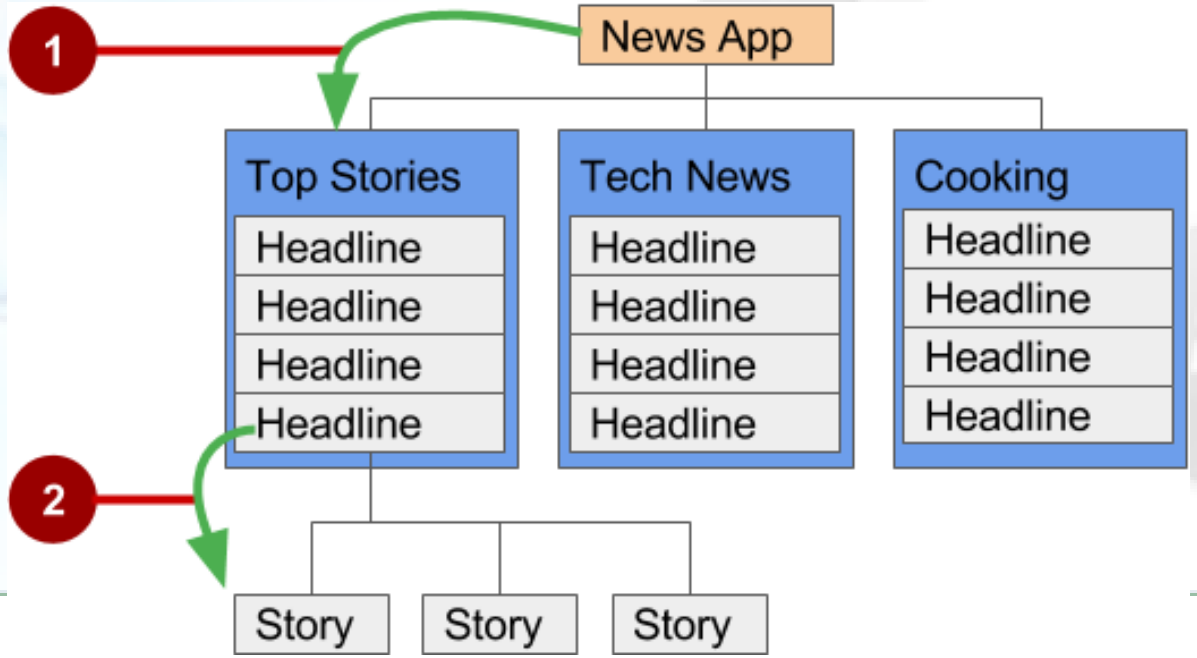
Hierarchical navigation patterns

- Descendant navigation
  - Down from a parent screen to one of its children
  - From a list of headlines—to a story summary—to a story
- Ancestral navigation
  - Up from a child or sibling screen to its parent
  - From a story summary back to the headlines
- Lateral navigation
  - From one sibling to another sibling
  - Swiping between tabbed views

Descendant Navigation

Descendant Navigation

- Down from a parent screen to one of its children
- From the main screen to a list of headlines to a story



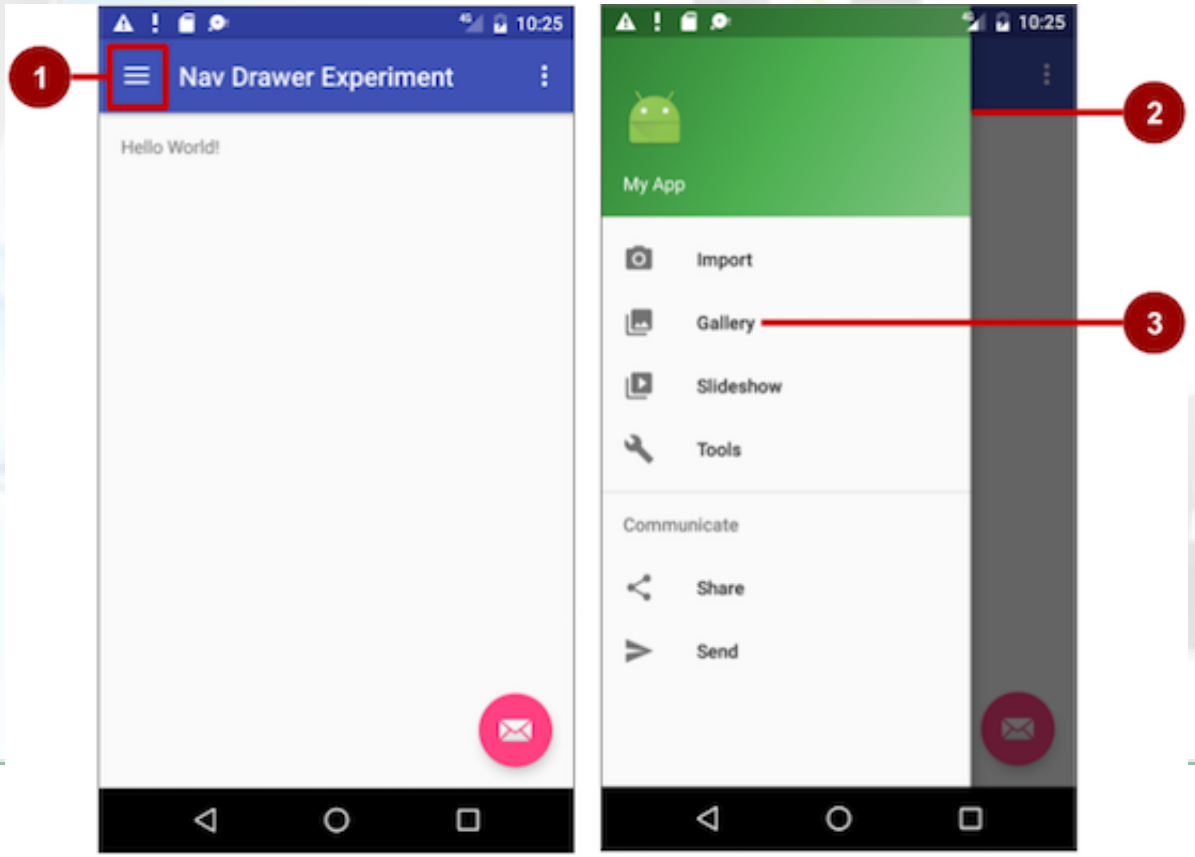
Controls for descendant navigation

- Navigation drawer
- Buttons, image buttons on main screen
- Other clickable views with text and icons arranged in horizontal or vertical rows, or as a grid

Navigation Drawer

Navigation drawer

1. Icon in app bar
2. Header
3. Menu items



Layouts for for navigation drawer

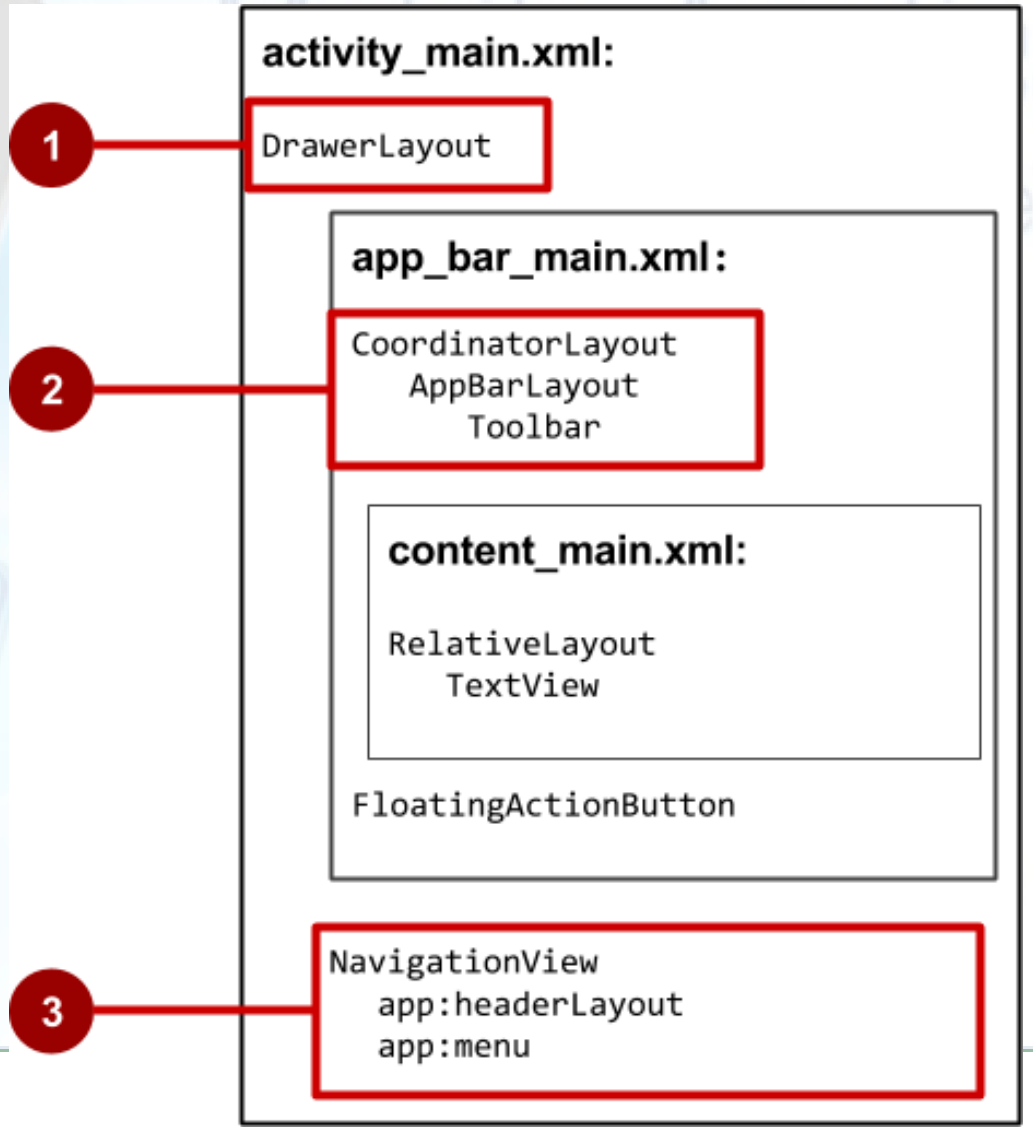
Create layouts:

- A navigation drawer as the Activity layout root ViewGroup
- A navigation View for the drawer itself
- An app bar layout that includes room for a navigation icon button
- A content layout for the Activity that displays the navigation drawer
- A layout for the navigation drawer header

Navigation drawer Activity layout

1. DrawerLayout is root view
2. CoordinatorLayout contains app bar layout with a Toolbar
3. App content screen layout
4. NavigationView with layouts for header and selectable items

Navigation drawer Activity layout



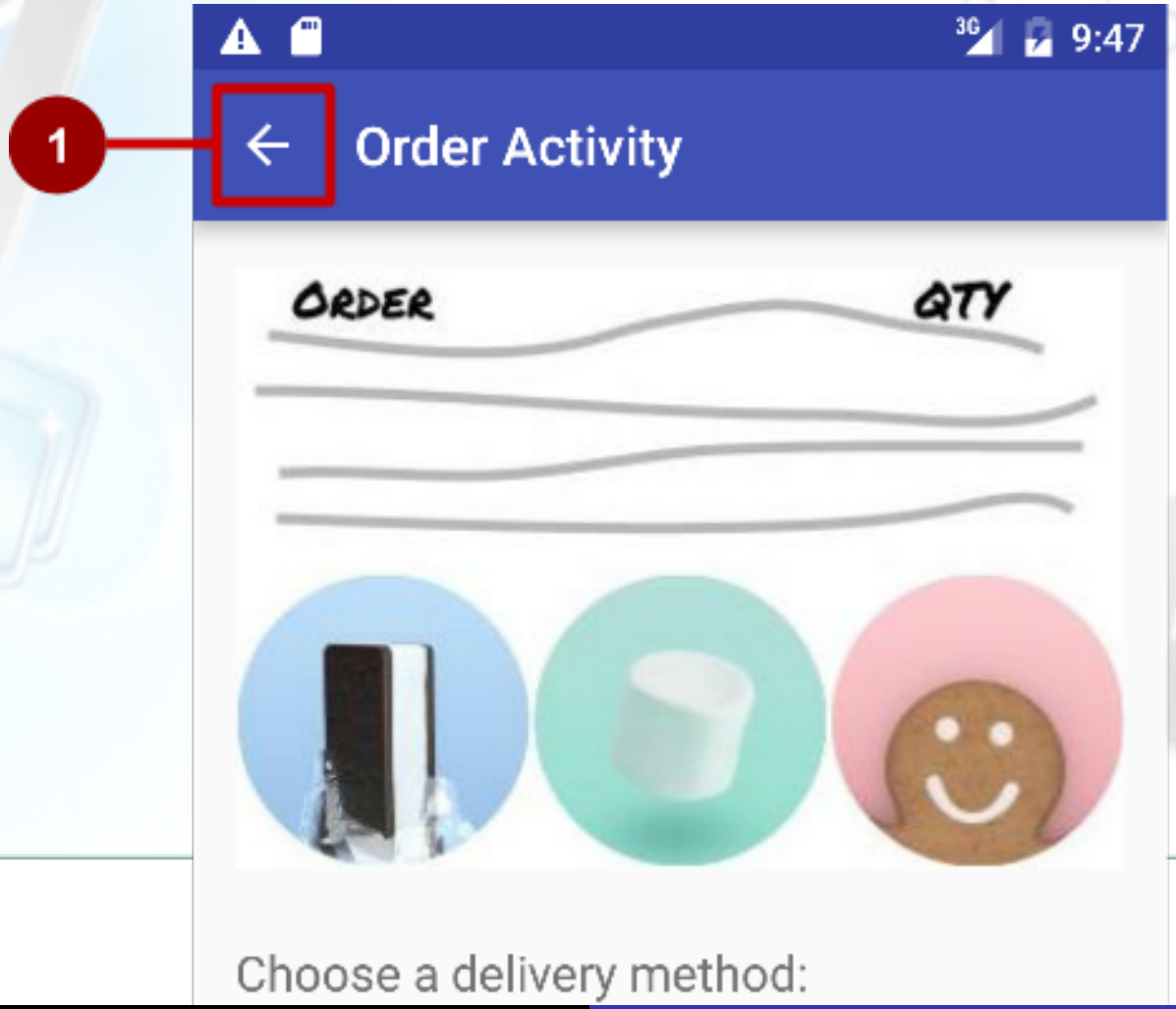
Steps to implement navigation drawer

1. Populate navigation drawer menu with item titles and icons
2. Set up navigation drawer and item listeners in the Activity code
3. Handle the navigation menu item selections

Ancestral Navigation

Ancestral navigation (Up button)

Enable user to go up from a section or child screen to the parent.



Declare parent of child Activity—AndroidManifest

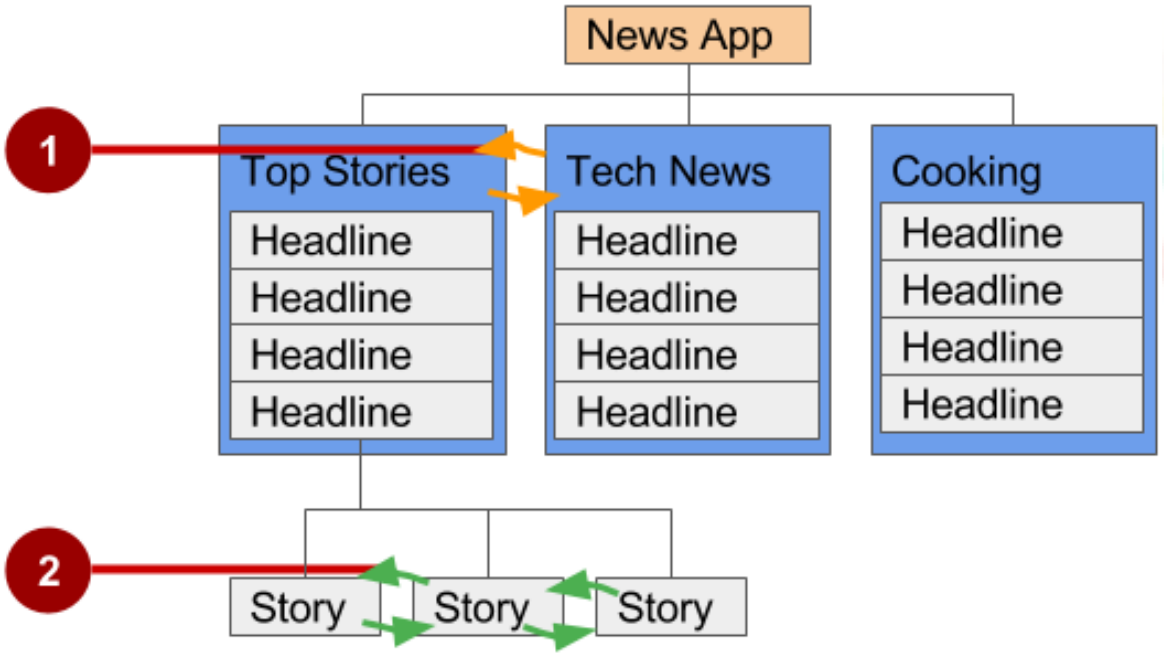
```
<activity android:name=".OrderActivity"
  android:label="@string/title_activity_order"
  android:parentActivityName="com.example.android.
                              optionsmenuorderactivity.MainActivity">
  <meta-data
     android:name="android.support.PARENT_ACTIVITY"
     android:value=".MainActivity"/>
</activity>
```

Lateral Navigation

Tabs and swipes

- Between siblings
- From a list of stories to a list in a different tab
- From story to story under the same tab

Tabs and swipes



Benefits of using tabs and swipes

- A single, initially-selected tab—users have access to content without further navigation
- Navigate between related screens without visiting parent

Steps for implementing tabs

- Define the tab layout using TabLayout
- Implement a Fragment and its layout for each tab
- Implement a PagerAdapter from FragmentPagerAdapter or FragmentStatePagerAdapter
- Create an instance of the tab layout
- Use PagerAdapter to manage screens (each screen is a Fragment)
- Set a listener to determine which tab is tapped

Add tab layout below Toolbar

```
<com.google.android.material.tabs.TabLayout
    android:id="@+id/tab_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/toolbar"
    android:background="?attr/colorAccent"
    android:minHeight="?attr/actionBarSize"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/>
```

Add view pager below TabLayout

```
<androidx.viewpager.widget.ViewPager
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="fill_parent"
    android:layout_below="@+id/tab_layout"
    />
```

Create a tab layout in onCreate()

```
TabLayout tabLayout = findViewById(R.id.tab_layout);
tabLayout.addTab(tabLayout.newTab().setText(R.string.tab_label1));
tabLayout.addTab(tabLayout.newTab().setText(R.string.tab_label2));
tabLayout.addTab(tabLayout.newTab().setText(R.string.tab_label3));

tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);
```

Add the view pager in onCreate()

```
ViewPager viewPager = findViewById(R.id.pager);
PagerAdapter adapter = new PagerAdapter(getSupportFragmentManager(),
                        tabLayout.getTabCount());
viewPager.setAdapter(adapter);
```

Add the listener in onCreate()

```
viewPager.addOnPageChangeListener(new TabLayout
                                .TabLayoutOnPageChangeListener(tabLayout));
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
        viewPager.setCurrentItem(tab.getPosition());
    }
    @Override
    public void onTabUnselected(TabLayout.Tab tab) { }
    @Override
    public void onTabReselected(TabLayout.Tab tab) { }
});
```

THANK YOU