College of Science and Technology

Royal University of Bhutan

Unit II:
Screen Navigation

CTE308- AS2025

Tutor: Pema Galey
#17682761

OUTLINES

- Back navigation
- Hierarchical navigation
  - Up navigation
  - Descendant navigation
  - Navigation drawer for descendant navigation
  - Lists and carousels for descendant navigation
  - Ancestral navigation
  - Lateral navigation

Two Forms of Navigation

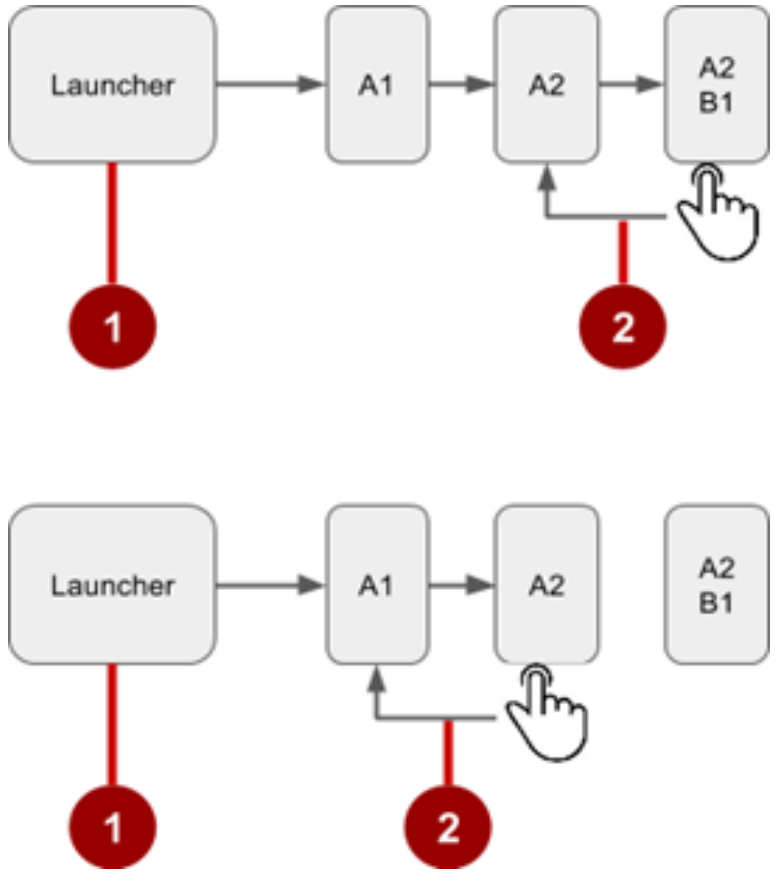
Temporal or back navigation

- provided by the device's back button
- controlled by the Android system's back stack

- Ancestral or up navigation
  - provided by the app's action bar
  - controlled by defining parent-child relationships between activities in the Android manifest

Navigation through History of Screen

1. Temporal or back navigation
2. User clicks the back button to navigate to the previous screens in reverse order.



Changing Back Button Behaviour

- Android system manages the back stack and Back button
- If in doubt, don't change
- Only override, if necessary to satisfy user expectation

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

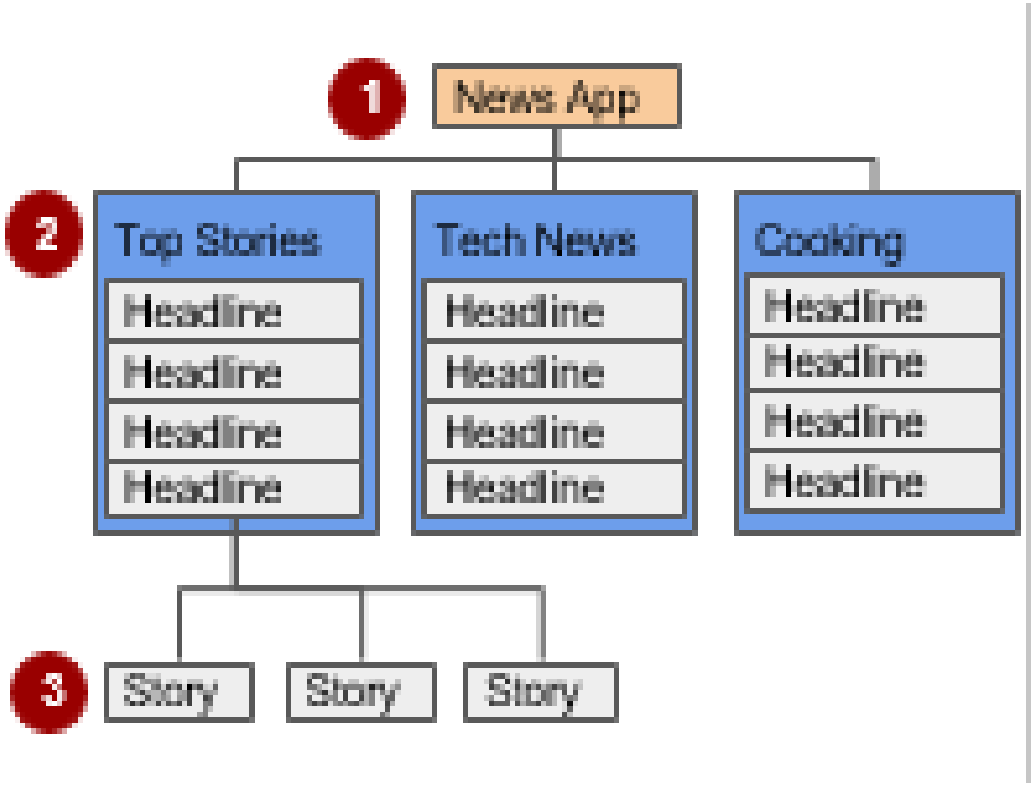
HIERARCHICAL NAVIGATION

- **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

Example: Hierarchical Navigation

1. Parent screen
2. Children: collection siblings
3. Children: section siblings

- Use activities or fragments to implement a hierarchy

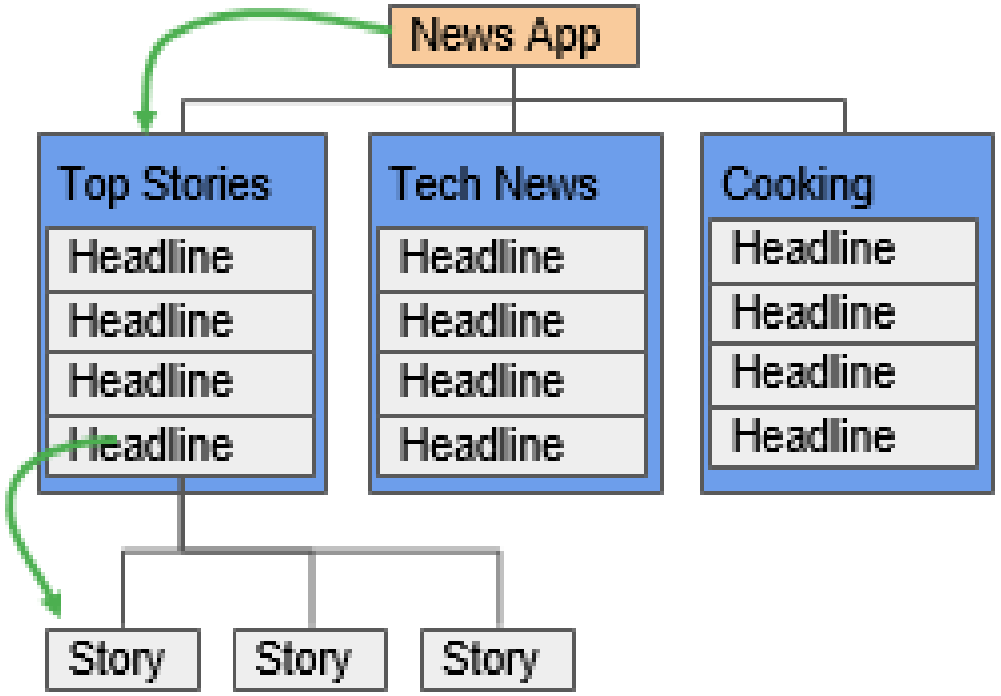


Types of Hierarchical Navigation

- 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

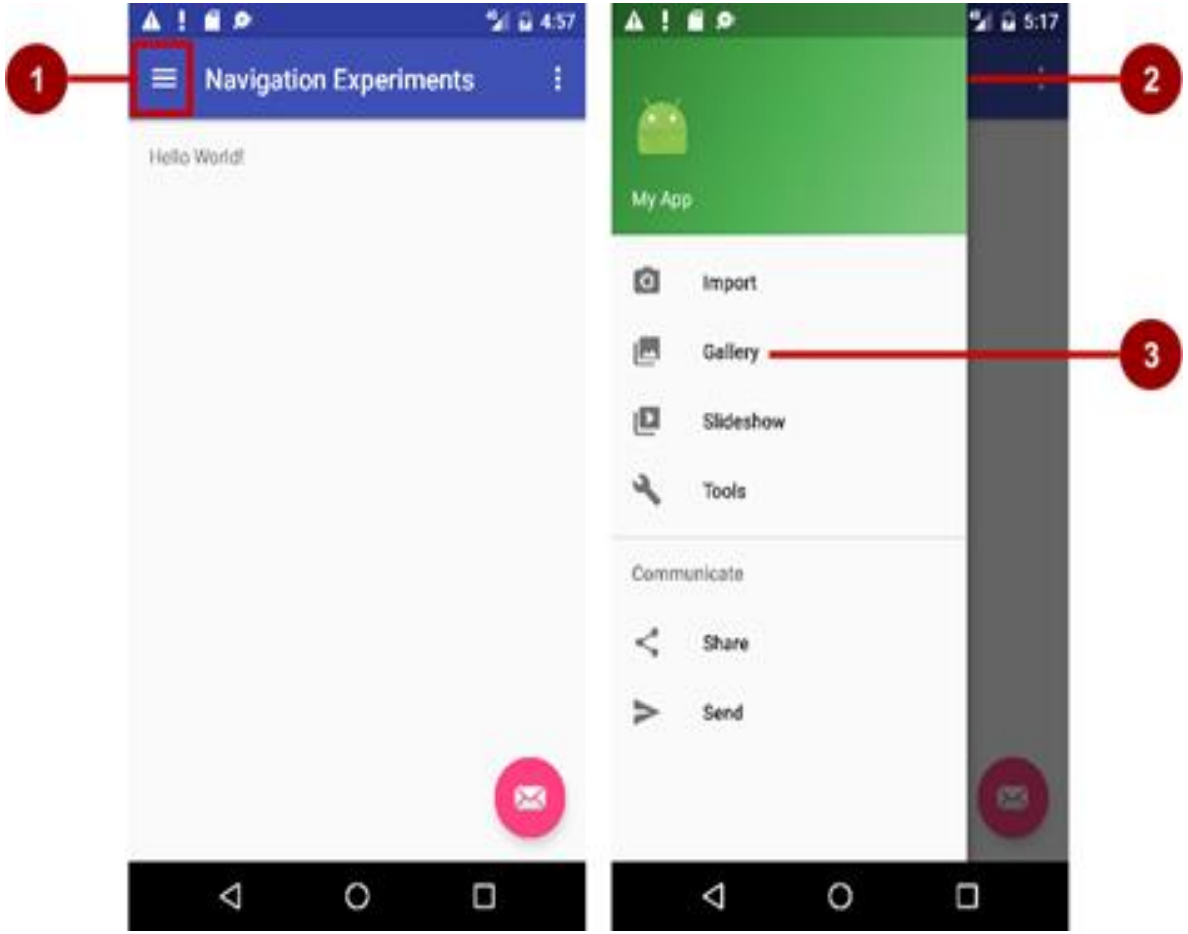
Descendant Navigation

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



Navigation Drawer for Descendant Navigation

1. Icon in App Bar
2. Header
3. Menu Items

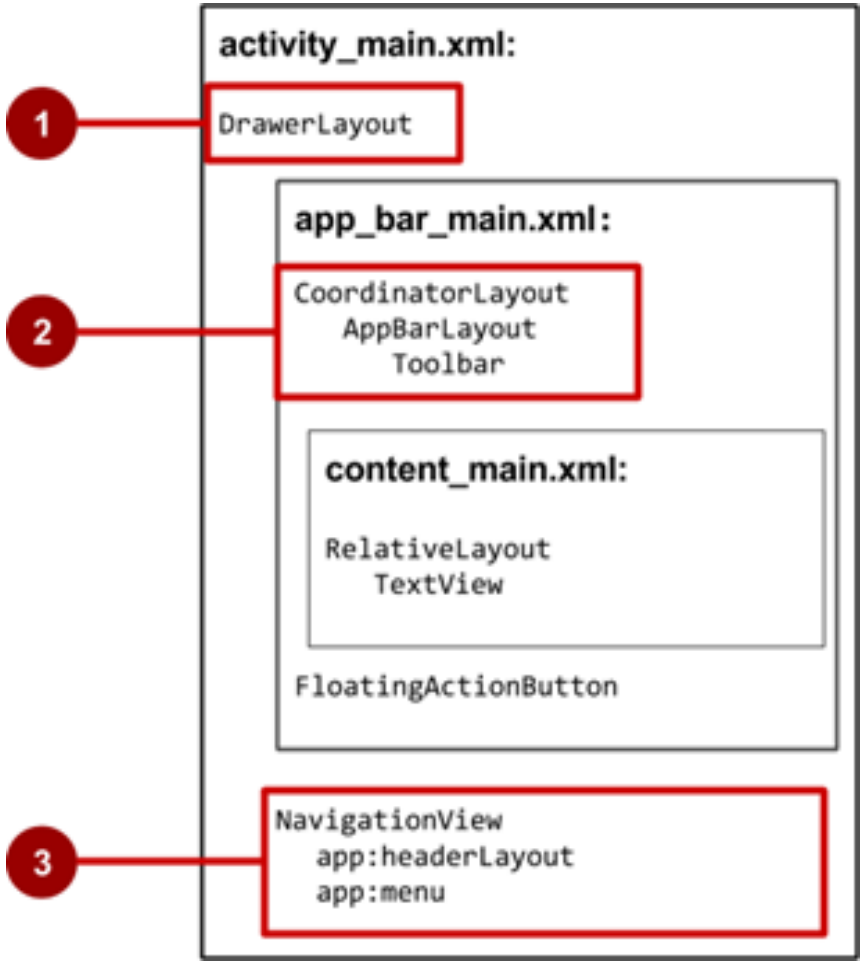


Steps for Navigation Drawer

1. Create layouts for drawer, drawer header, drawer menu items, app bar, activity screen contents
2. Add navigation drawer and item listeners to activity code
3. Handle the navigation drawer menu item selections

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



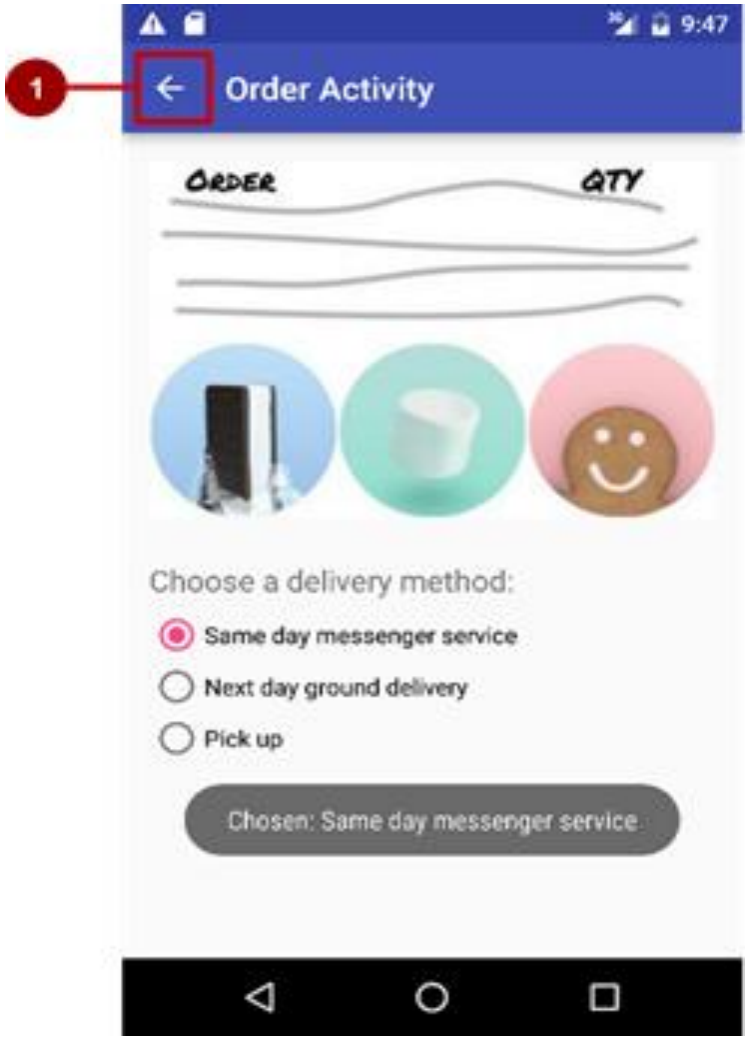
Other Descendant Navigation Patterns

1. Vertical List, such as RecyclerView
2. Vertical grid, such as GridView
3. Multi-level menus, such as the Options menu
4. Master/Detail navigation flow

Ancestral Navigation

- Up from a child or sibling screen to its parent
- From a story summary back to the headlines

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

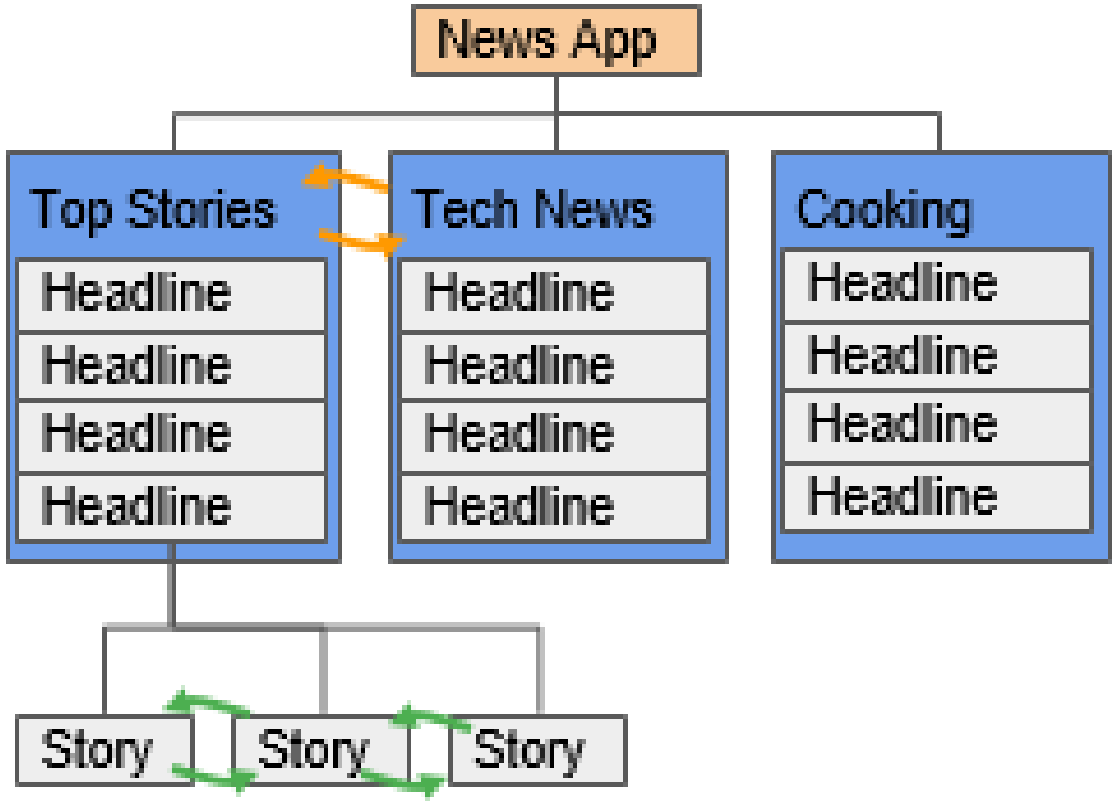


Declare activity's parent in AndroidManifest file

```
<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>
```

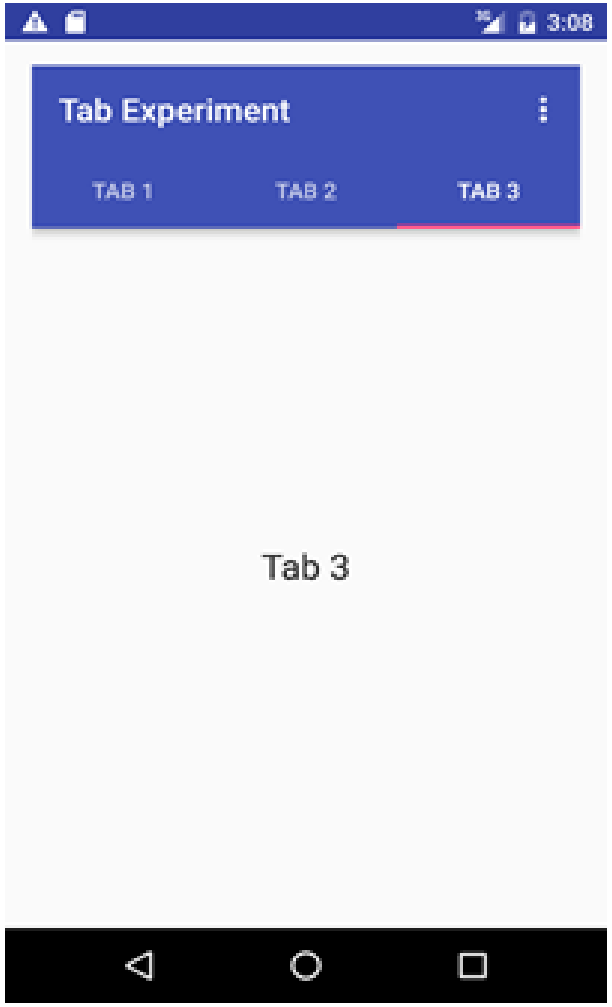
Lateral Navigation

- From one sibling to another sibling
- Swiping between tabbed views



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



Best practices with Tabs

- Layout horizontally
- Run along top of screen
- Persistent across related screens
- Switching should not be treated as history

Tabs Implementing Steps

1. Define the tab layout using **TabLayout**
2. Implement a **fragment** and its layout for each tab
3. Implement a **PagerAdapter** from **FragmentPagerAdapter** or **FragmentStatePagerAdapter**
4. Create an instance of the tab layout
5. Manage screen views in fragments
6. Set a **listener** to determine which tab is tapped.

Add tab layout below Toolbar

```
<android.support.design.widget.TabLayout
    android:id="@+id/tab_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/toolbar"
    android:background="?attr/colorPrimary"
    android:minHeight="?attr/actionBarSize"

android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/
>
```

Add view pager below TabLayout

```
<android.support.v4.view.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 = (TabLayout)
findViewById(R.id.tab_layout);
tabLayout.addTab(tabLayout.newTab().setText("Tab 1"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 2"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 3"));
tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);
```

Add the view pager in onCreate()

```
final ViewPager viewPager = (ViewPager)
findViewById(R.id.pager);
final 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!