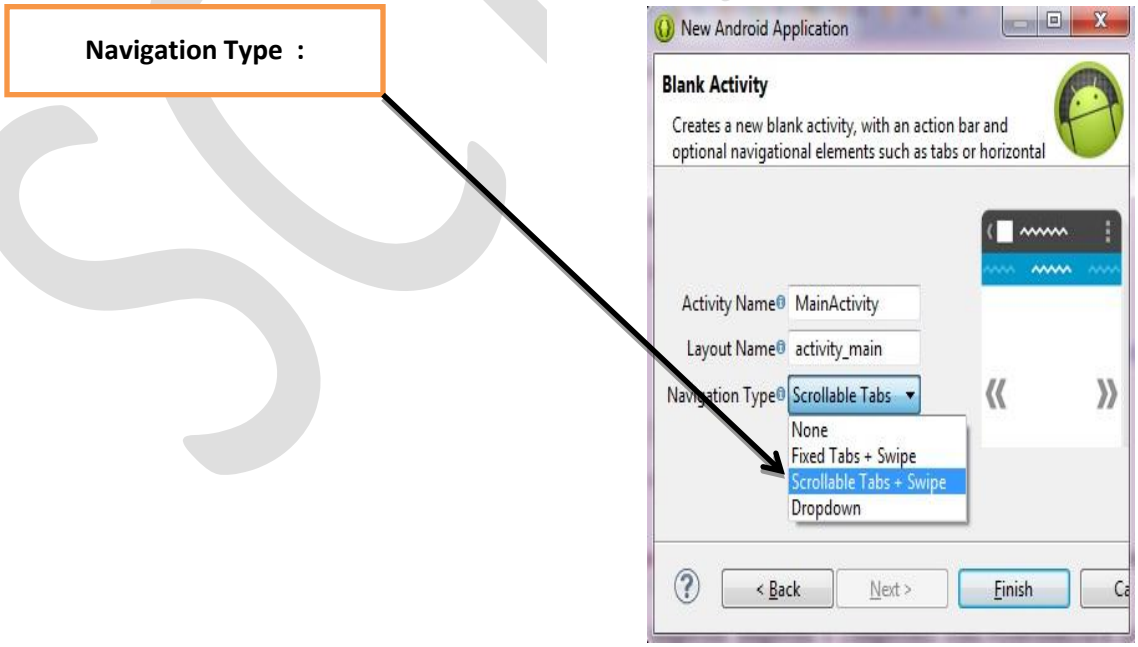
Android SwipeTab (Touch Gesture)

Objective: Swipe Views are an efficient way to allow the user to laterally navigate among related data items spread across multiple panes using a simple touch gesture (swipes/flings), making access to data faster and much more enjoyable. Think of the android home screen where you can swipe across multiple sibling screens or the Facebook or twitter app with multiple screens (and their respective tabs) where you can just swipe to navigate through them. The entire experience is super interactive and fun. They're basically equivalent to slideshows/carousels which has sections with/without tabs (or similar controls to navigate) in the web development arena.

Technical Specification: In this Application we have 2 Activities and corresponding xml layout files and 1 java file.

1. Create a new android project with the



Android provides us with the **ViewPager** class that allows the user to flip left and right through pages of data. To use it, you'll have to put the **<ViewPager>** element (from the support library) inside your XML layout file that'll contain multiple child views (which will be the different pages).

```
<android.support.v4.view.ViewPagerxmlns:android="http://schemas.android.com/apk/res/and
roid"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/pager"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
```

PagerTitleStrip is a non-interactive indicator of the current, next, and previous pages of a ViewPager. It is intended to be used as a child view of a ViewPager widget.This title strip will display the currently visible page title, as well as the page titles for adjacent pages.

```
<android.support.v4.view.PagerTitleStrip
android:id="@+id/pager_title_strip"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_gravity="top"
android:background="#33b5e5"
android:paddingBottom="4dp"
android:paddingTop="4dp"
android:textColor="#fff"/>
```

Add appropriate images for all tabs in the res>drawable-hdpi folder. (eg. android.jpg, windows.jpg etc.)

2. **Make XML file in res>values folder called "Array.xml" which will contain the two different arrays of tab title and tab description.**

3. Create new activity which will **extends** the **Fragment**

   This is responsible for inflating the objects into the fragment which are our images and description in a fragment of the xml layout.

   Create two static objects like:

```
public static final String ImageIDKey = "imagekey";
public static final String DescriptionKey = "descriptionkey";
```

   For accessing the key through all the java files.

   The XML file of this activity contains one <**ImageView**> for showing the image of a tab and one <**TextView**> for showing the description of the tab.

4. Make a class file (.java) in package which is going to **extends** **FragmentPagerAdapter.**

   FragmentPagerAdapter – Each page is a fragment and all of them are kept in memory, hence best to use when there's a fixed small set of screens to be navigated through. If loads of fragments are shoved in using this adapter then it could lead to a laggy user experience. It works even better when the content of the fragments are static than something that constantly changes or gets updated.

Code Snippet of Activity

```
public DevicePageAdaptor(FragmentManager fm, Context context) {
	super(fm);
	// TODO Auto-generated constructor stub

	Resources resources = context.getResources();

	devices = resources.getStringArray(R.array.devices);
	deviceDescription = resources.getStringArray(R.array.device_description);
}

@Override
public Fragment getItem(int position) {
	// TODO Auto-generated method stub

	Bundle bundle = new Bundle();

bundle.putString(DeviceFragment.DescriptionKey,deviceDescription[position]);
	bundle.putInt(DeviceFragment.ImageIDKey,getDeviceIamgeID(position));

	DeviceFragment deviceFragment = new DeviceFragment();
	deviceFragment.setArguments(bundle);



	return deviceFragment;
}

privateint getDeviceIamgeID(int position){

	int id=0;
	switch (position)
	{

	case 0:
		id = R.drawable.windows;
		break;

	case 1:
		id = R.drawable.android;
		break;
```

```
case 2:
                id = R.drawable.ios;
                break;
        }

        return id;
}


@Override
public CharSequence getPageTitle(int position) {
        // TODO Auto-generated method stub
        returndevices[position];
}


@Override
publicint getCount() {
        // TODO Auto-generated method stub
        returndevices.length;
}
```

3rd Tab

Output: