Utilizing the Action Bar in Android

Android ActionBar is a menu bar that runs across the top of the activity screen in android. Android ActionBar can contain menu items which become visible when the user clicks the “menu” button.

In general an `ActionBar` consists of the following four components:

- **App Icon**: App branding logo or icon will be displayed here
- **View Control**: A dedicated space to display Application title. Also provides option to switch between views by adding spinner or tabbed navigation
- **Action Buttons**: Some important actions of the app can be added here
- **Action Overflow**: All unimportant action will be shown as a menu

Android ActionBar Menu

The simplest way to get toolbar icons and action overflow items into the action bar is by creating menu XML resource file found in **res/menu** folder. We can add menu items in the raw xml file present in the folder as follows:

Utilizing the Action Bar in Android

menu_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/item1"
          android:title="Add"
          android:icon="@drawable/icon1"
          app:showAsAction="always"/>
    <item android:id="@+id/item2"
          android:title="Battery"
          android:icon="@drawable/icon2"
          app:showAsAction="always"/>
    <item android:id="@+id/item3"
          android:title="About"
          app:showAsAction="never"/>
    <item android:id="@+id/item4"
          android:title="Setting"
          app:showAsAction="never"/>
    <item android:id="@+id/item5"
          android:title="Add"
          android:icon="@drawable/icon1"
          app:showAsAction="always"/>
</menu>
```

There are four things that are needed to be configured for every menu item.

1. **android:id**: attribute specifies the id of the menu item. This works like ids anywhere else in the Android app. An android:id value starting with a @+id/ will create a constant in the R.menu constant collection
2. **android:title**: attribute value contains the title of the menu item
3. **android:icon**: attribute references an icon in the drawable directories
4. **android:showAsAction**: This attribute indicates how the given item should be portrayed in the action bar.

   We can choose from any of the flags mentioned below:

- **always** to keep it in the ActionBar at all times
- **ifRoom** to keep it only if space is available
- **never** this means that the menu item will not be placed in the ActionBar as an icon. It will only be visible when the menu button is clicked, in the menu that's popping up
- **|withText** : we can append this to either always or ifRoom, to indicate that the toolbar button to be both the icon and the title, not just the icon

Note that **always** is not guaranteed to be a toolbar button – if you ask for 100 always items, you will not have room for all of them. However, `always` items get priority for space in the action bar over `ifRoom` items.

Inflating the Menu Into the Android ActionBar

In order for the menu items defined in the menu XML file to be displayed, you need to inflate the menu file. We do so inside the `onCreateOptionsMenu()` method of the activity where we wish to add the ActionBar. Here is the code snippet:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if
it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

The **R.menu.menu_main** parameter is the constant referring to the menu XML file. The menu parameter is the menu into which we want to inflate the menu items.

To find out when the user taps on one of these things, we'll need to override `onOptionsItemSelected()` from the MainActivity as shown below:

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {

    switch(item.getItemId())
    {
        case R.id.item1:
            Toast.makeText(this,"phone is pressed",Toast.LENGTH_LONG).show();
            return true;
        case R.id.item2:
            Toast.makeText(this,"Battery is pressed",Toast.LENGTH_LONG).show();
            return true;
        case R.id.item3:
            Toast.makeText(this,"About is pressed",Toast.LENGTH_LONG).show();
            return true;
```

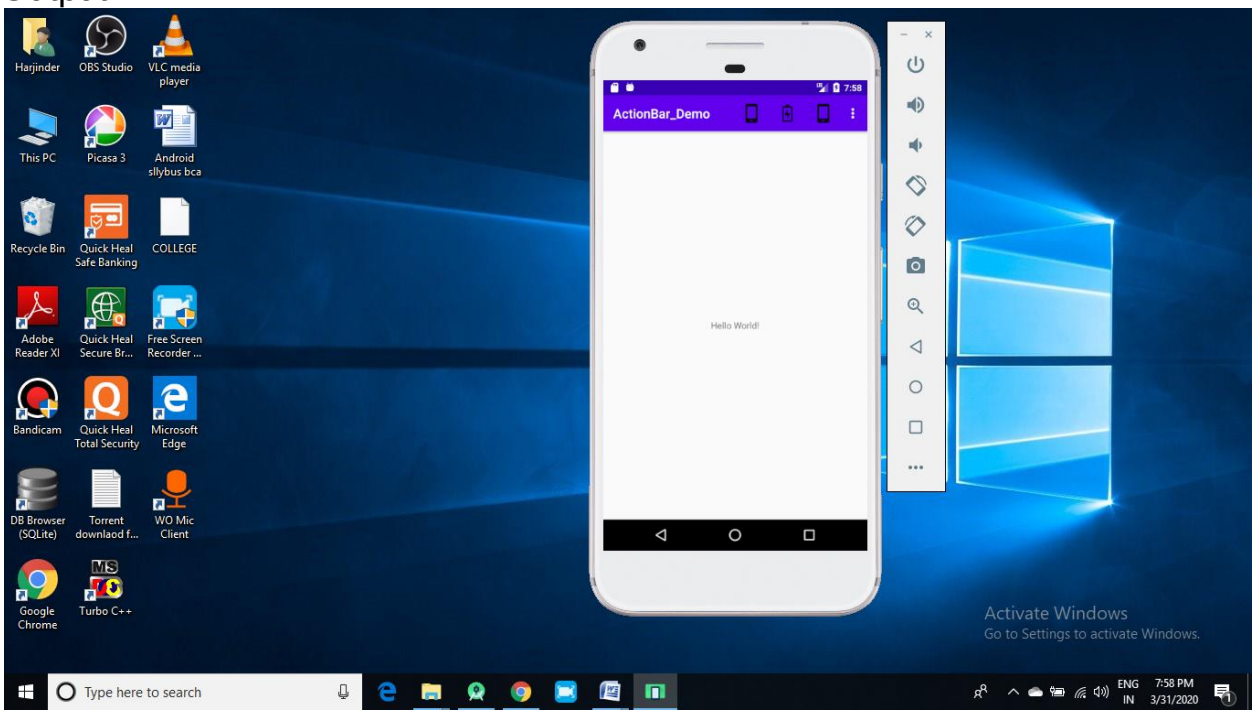
Utilizing the Action Bar in Android

```
        case R.id.item4:
            Toast.makeText(this,"Setting is pressed",Toast.LENGTH_LONG).show();
            return true;
        case R.id.item5:
            Toast.makeText(this,"phone 3 is pressed",Toast.LENGTH_LONG).show();
            return true;
    }
   return super.onOptionsItemSelected(item);
}
```

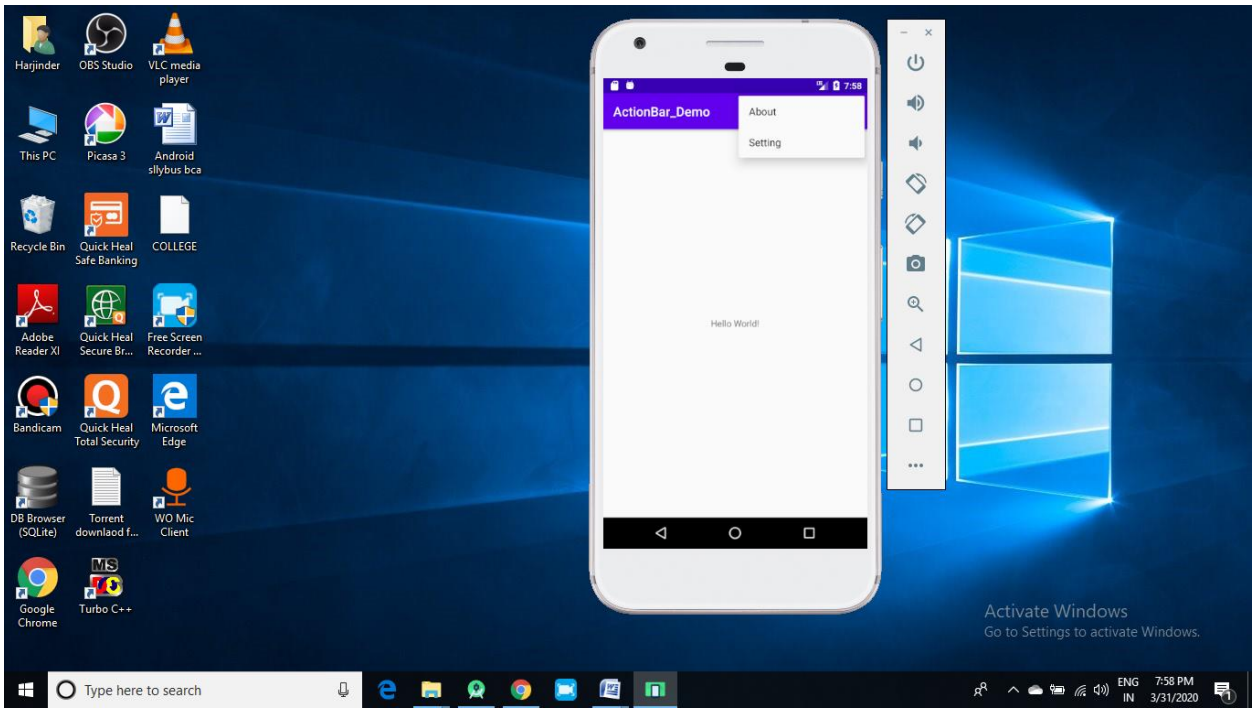
The `styles.xml` file is defined as follows:

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
    </style>

</resources>
```

Output:



Utilizing the Action Bar in Android



Complete Source code Link:

https://github.com/Pythotech/Utilizing-Action-Bar-in-Android