Lecture 6

User Interface

ScrollView, ActionBar, Toolbar

Course: **Mobile App Development**

By: Tho C. Mai
Nha Trang University

User Interface

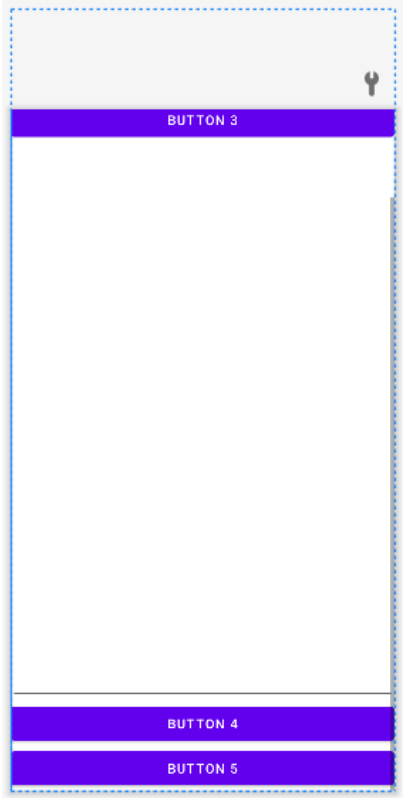
- How ViewGroups and Layouts can be used to lay out your views and organize your application screen
- How to adapt and manage changes in screen orientation
- How to create the UI programmatically

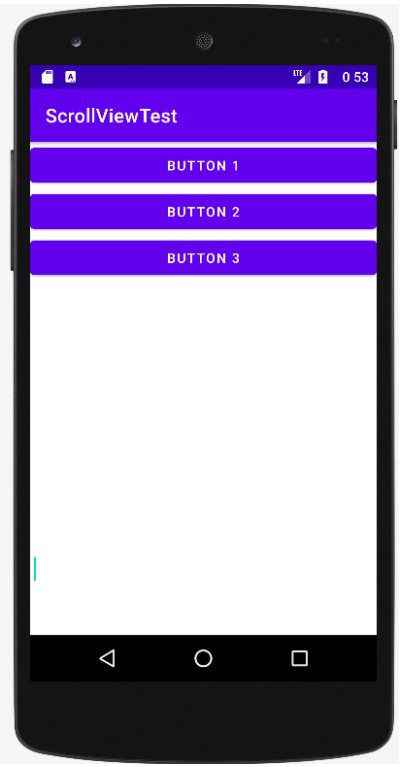
User Interface

- ScrollView
- Adapting to Display Orientation
- Action Bar
- Creating UI Programmatically

ScrollView

- A ScrollView is a special type of FrameLayout in that it enables users to scroll through a list of views that occupy more space than the physical display
- The ScrollView can contain only one child view or ViewGroup, which normally is a LinearLayout



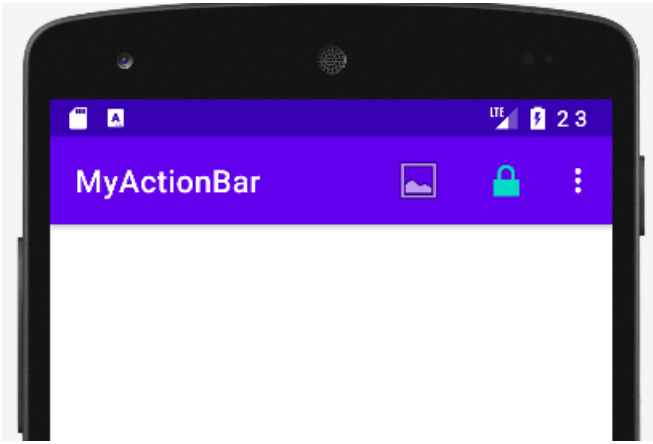


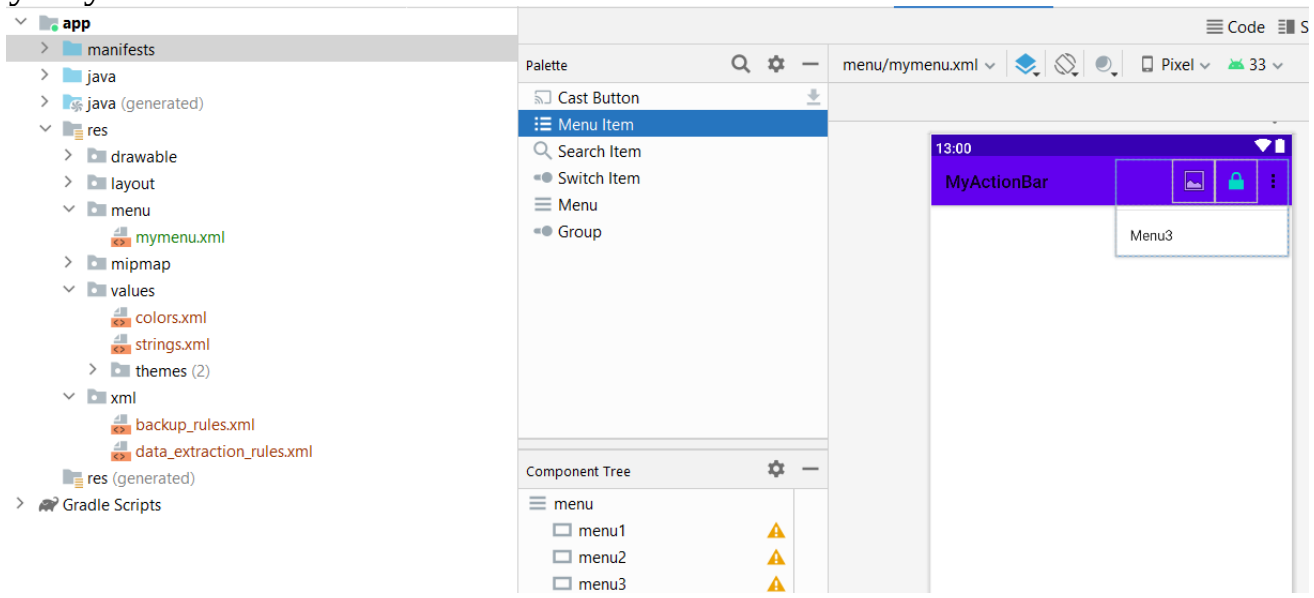
```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:focusable="true"
        android:focusableInTouchMode="true">
    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 1" />
    <Button
        android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 2" />
```

```
    <Button
        android:id="@+id/button3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 3" />
     <EditText
        android:id="@+id/txt"
        android:layout_width="fill_parent"
        android:layout_height="600dp" />
     <Button
        android:id="@+id/button4"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 4" />
     <Button
        android:id="@+id/button5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 5" />
    </LinearLayout>
</ScrollView>
```

Action Bar

- Action Bar displays the application icon and the activity title
- Optionally, on the right side of the Action Bar are action items
- Using Android Studio, create a new Android project and name it **MyActionBar**
- In res folder, creat a new resource file (type = menu), named: **mymenu**
  - Modify mymenu.xml as follow:





Adding Action Items

```
<?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/menu1"
      android:icon="@android:drawable/ic_menu_gallery"
      android:title="Menu 1"
      app:showAsAction="ifRoom" />
   <item
      android:id="@+id/menu2"
      android:icon="@android:drawable/ic_lock_lock"
      android:title="Menu 2"
      app:showAsAction="always" />
   <item
      android:id="@+id/menu3"
      android:checkable="false"
      android:icon="@android:drawable/ic_lock_silent_mode"
      android:title="Menu3"
      app:showAsAction="collapseActionView" />
</menu>
```

Adding Action Items

- Modify, add **bold lines** to the **MainActivity**.java file as follows

```
package com.maicuongtho.myactionbar;
....
public class MainActivity extends AppCompatActivity {

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

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId())
  {
     case android.R.id.home:
        onBackPressed();                          return true;
     case R.id.menu1:
        Toast.makeText(this, "You clicked on Item 1",
                Toast.LENGTH_LONG).show();        break;
     case R.id.menu2:
        Toast.makeText(this, "You clicked on Item 2",
                Toast.LENGTH_LONG).show();        break;
     case R.id.menu3:
        Toast.makeText(this, "You clicked on Item 3",
            Toast.LENGTH_LONG).show();            break;
     default:break;
  }
  return super.onOptionsItemSelected(item);
}
```

Toolbar

- Toolbar is a kind of ViewGroup that can be placed in the XML layouts of an activity.
- It was introduced by the Google Android team during the release of Android Lollipop(API 21).
- The Toolbar is basically the advanced successor of the ActionBar.
  - It is much more flexible and customizable in terms of appearance and functionality.
  - Unlike ActionBar, its position is not hardcoded i.e., not at the top of an activity.
  - Developers can place it anywhere in the activity according to the need just like any other View
- One can use the Toolbar in the following two ways:
  1) Use as an ActionBar
  2) Use a standalone Toolbar

Toolbar (Use as an ActionBar)

- Using Android Studio, create a new Android project and name it **MyToolbarAsActionBar**
- Step 1: Open the activity_main.xml file, Adding Toolbar in activity_main.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    …
    tools:context=".MainActivity">
      <com.google.android.material.appbar.AppBarLayout
        android:layout_width="0dp“          android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
          <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"     android:layout_height="wrap_content"
            android:background="?attr/colorPrimary"
            android:minHeight="?attr/actionBarSize"
            android:theme="?attr/actionBarTheme"
            tools:layout_editor_absoluteX="0dp"
            tools:layout_editor_absoluteY="0dp" />
      </com.google.android.material.appbar.AppBarLayout>
…
```

Toolbar (Use as an ActionBar ..)

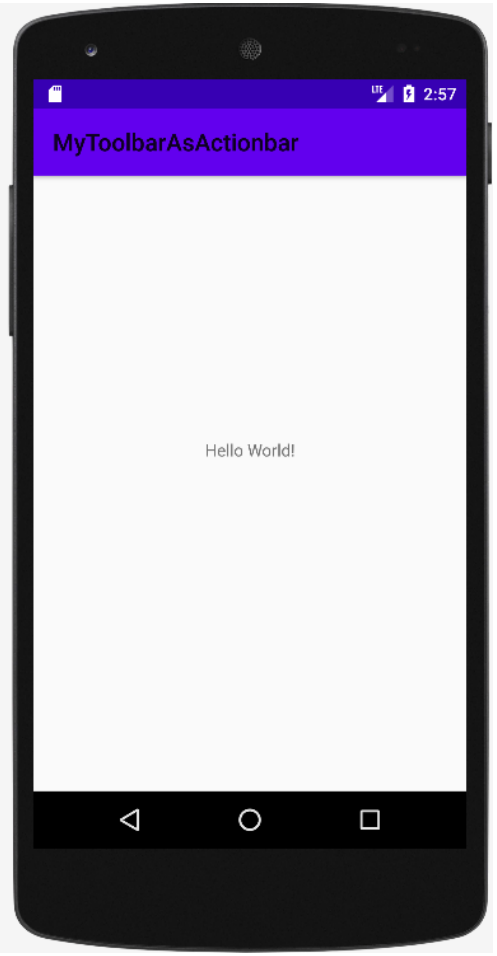
- Step 2: Remove ActionBar by Changing **bold line**, in res/valued/themes/themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.MyToolbarAsActionbar" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Toolbar (Use as an ActionBar ..)

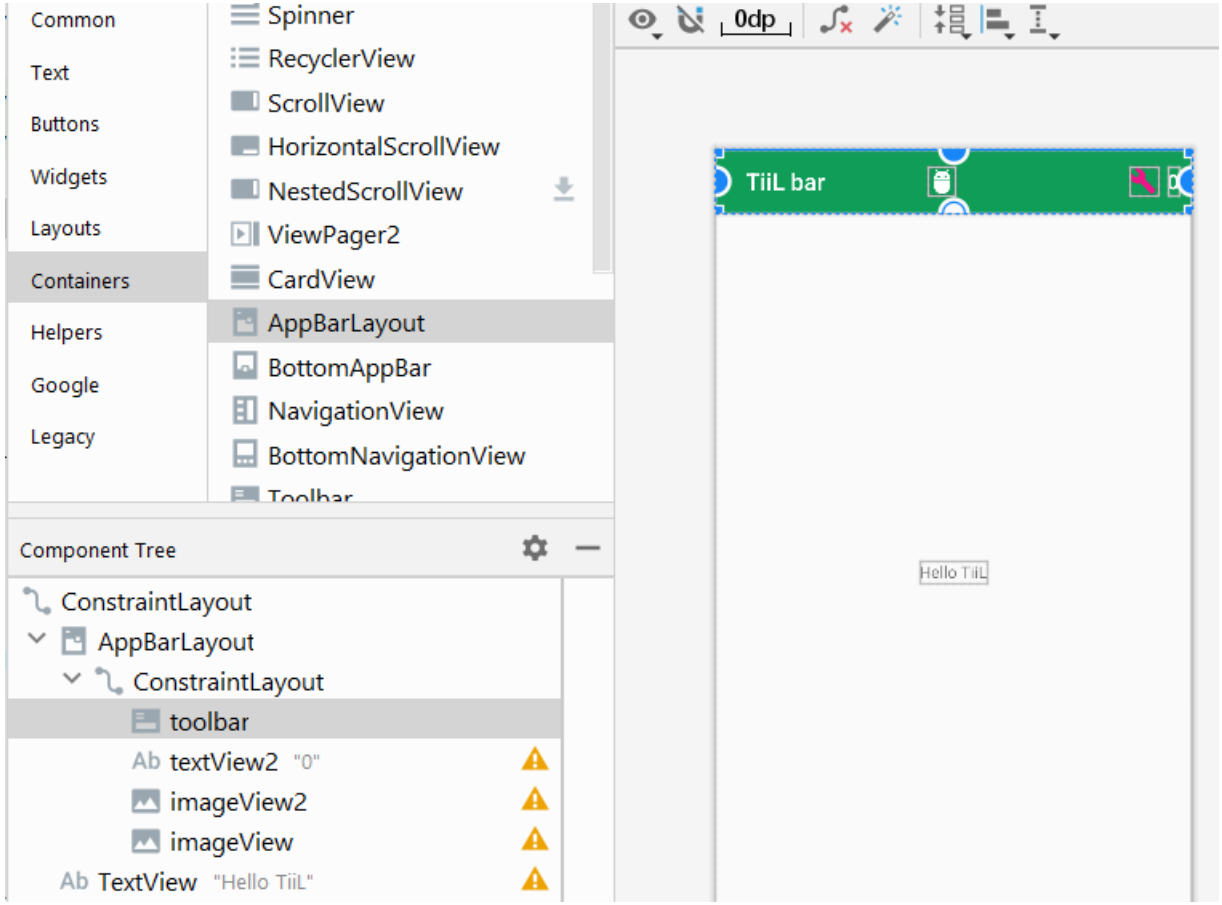
- Step 3: Using Toolbar as ActionBar
  - The Toolbar will not display the application title unless it is declared as an ActionBar.

```
package com.maicuongtho.mytoolbarasactionbar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // assigning ID of the toolbar to a variable
      Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
      // using toolbar as ActionBar
      setSupportActionBar(toolbar);
   }
}
```



Toolbar (Use as an ActionBar ..)

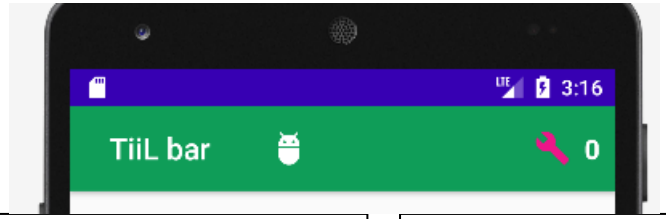
- Step 4: Customize the Toolbar




```
..
   tools:context=".MainActivity">
   <!-- AppBar layout for using Toolbar as AppBar -->
   <com.google.android.material.appbar.AppBarLayout
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent">

      <androidx.constraintlayout.widget.ConstraintLayout
         android:layout_width="match_parent"
         android:layout_height="match_parent">

            [CUSTOMIZE code]

      </androidx.constraintlayout.widget.ConstraintLayout>
   </com.google.android.material.appbar.AppBarLayout>
```

Toolbar (Use as an ActionBar ..)

- Customize the Toolbar : Customize code

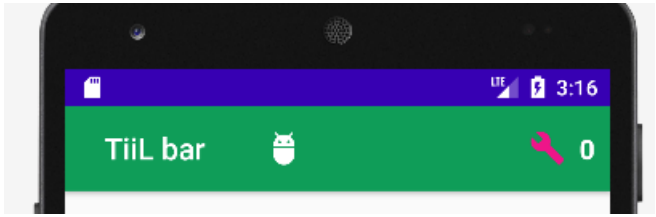



```
<!-- ToolBar widget -->
<androidx.appcompat.widget.Toolbar
   android:id="@+id/toolbar"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="#0F9D58"
   android:minHeight="?attr/actionBarSize"
   android:theme="?attr/actionBarTheme"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toTopOf="parent"
   app:title="  TiiL bar"
   app:titleTextColor="#ffff" />
```

```
<!-- Right most TextView -->
<TextView
   android:id="@+id/textView2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_marginEnd="10dp"
   android:text="0"
   android:textColor="#ffff"
   android:textSize="18sp"
   android:textStyle="bold"
   app:layout_constraintBottom_toBottomOf="parent"
   app:layout_constraintEnd_toEndOf="@+id/toolbar"
   app:layout_constraintTop_toTopOf="parent" />
```

Toolbar (Use as an ActionBar ..)

- Customize the Toolbar : Customize code




```
<!-- Right most ImageView -->
<ImageView
   android:id="@+id/imageView2"
   android:layout_width="wrap_content"
   android:layout_height="24dp"
   android:layout_marginEnd="9dp"
   app:layout_constraintBottom_toBottomOf="parent"
   app:layout_constraintEnd_toStartOf="@+id/textView2"
   app:layout_constraintTop_toTopOf="parent"
   app:srcCompat="@drawable/build_icon" />
```

```
<!-- ImageView beside title of ToolBar -->
<ImageView
   android:id="@+id/imageView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_marginEnd="150dp"
   app:layout_constraintBottom_toBottomOf="parent"
   app:layout_constraintEnd_toStartOf="@+id/imageView2"
   app:layout_constraintTop_toTopOf="parent"
   app:srcCompat="@drawable/android_icon"
   app:tint="#ffff" />
```

Toolbar (Use as an ActionBar ..)

- Step 5: Adding a logo and some more customization (bold lines)

```
   // assigning ID of the toolbar to a variable
   Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
   // using toolbar as ActionBar
   setSupportActionBar(toolbar);
   // Display application icon in the toolbar
   getSupportActionBar().setDisplayShowHomeEnabled(true);
   getSupportActionBar().setLogo(R.drawable.app_icon);
   getSupportActionBar().setDisplayUseLogoEnabled(true);
// assigning ID of textView2 to a variable
ImageView img2= (ImageView)findViewById(R.id.imageView2);       img2.setClickable(true);
img2.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      Toast.makeText(view.getContext(),  "You clicked on Build Icon", Toast.LENGTH_SHORT).show();
   }
});
```