NAME: SAKSHI VINAYAK KITTURKAR
CSU ID: 2860273
HOMEWORK 5

Modify the Action Bar app by altering the behavior when each action item is clicked in the following ways:
When item1 through item4 is clicked, a TextView should be displayed at the upper-left, upper-right, bottom-left, and button-right on the screen, respectively.
When item5 is clicked, all existing TextViews that have been added to the screen should be cleared.

= I have created app in the LANDSCAPE MODE where you can find 5 items on top of the screen when you click item 1 the TextView will appear on the upper-left corner on the screen when you click item 2 the TextView will appear on the upper-right corner on the screen when you click item 3 the TextView will appear on the bottom left side of the screen and when you click item 4 the TextView will appear on the bottom-right corner and when you click the last item that is item 5 you will find all the TextViews are been cleared.

In layout: MainActivity.java This Android Java code defines an activity with a layout containing four TextView elements. It sets up an ActionBar and includes a menu with items that control the visibility of these TextViews. When a menu item is selected, it triggers actions like showing or hiding specific TextViews. For instance, selecting "item1" makes textView1 visible, and selecting "item5" hides all TextViews.
ActionBar Initialization

```
ActionBar actionBar = getSupportActionBar();
```

Menu Inflation

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   getMenuInflater().inflate(R.menu.main, menu);
   return super.onCreateOptionsMenu(menu);
}
```

Handling Menu Item Selection uses a switch syayement to determine which menu item was selected manipulates the visibility of TextViews based on the selected menu item.

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
   switch (item.getItemId()) {
      case R.id.item1:
         textView1.setVisibility(View.VISIBLE);
         break;
      case R.id.item2:
         textView2.setVisibility(View.VISIBLE);
         break;
      case R.id.item3:
         textView3.setVisibility(View.VISIBLE);
         break;
      case R.id.item4:
         textView4.setVisibility(View.VISIBLE);
         break;
```

```
        case R.id.item5:
            textView1.setVisibility(View.GONE);
            textView2.setVisibility(View.GONE);
            textView3.setVisibility(View.GONE);
            textView4.setVisibility(View.GONE);
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

In layout: ExampleInstrumentedTest.java this instrumental test verifies that the context of the app under test has the expected package name it serves as a basic example of how to write an instrumental.

Test Class Declaration
Declaration of the test class 'ExampleInstrumentedTest' and Use of @RunWith(AndroidJUnit4.class) indicating the use of the AndroidJUnit4 test runner.

```
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
```

App Context Retrieval Retrieval of the context of the app under test using InstrumentationRegistry.

```
Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
```

Assertion An assertion verifying that the package name of the app under test is equal to "com.example.homework5".

```
assertEquals("com.example.homework5", appContext.getPackageName());
```

In layout: ExampleUnitTest.java This code is an instrumental test ensuring that the package name of the app under test matches the expected value. The test uses the AndroidJUnit4 test runner, retrieves the app context, and performs an assertion on the package name. The comments offer a link to relevant testing documentation for further information.

Test Class Declaration

```
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
```

Test Method

```
@Test
public void useAppContext() {
```

App Context Retrieval

```
Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
```

Assertion

```
assertEquals("com.example.homework5", appContext.getPackageName());
```

In layout: AndroidMainfest.xml This AndroidManifest.xml file defines essential information about the application, including its name, icon, theme, main activity, and additional configuration options. It provides the system with necessary metadata about the application's structure and behavior.

XML Declaration: Specifies the version and encoding of the XML file.

```
<?xml version="1.0" encoding="utf-8"?>
```

Manifest Element: The root element for the AndroidManifest.xml file, defining XML namespaces for Android and tools.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
```

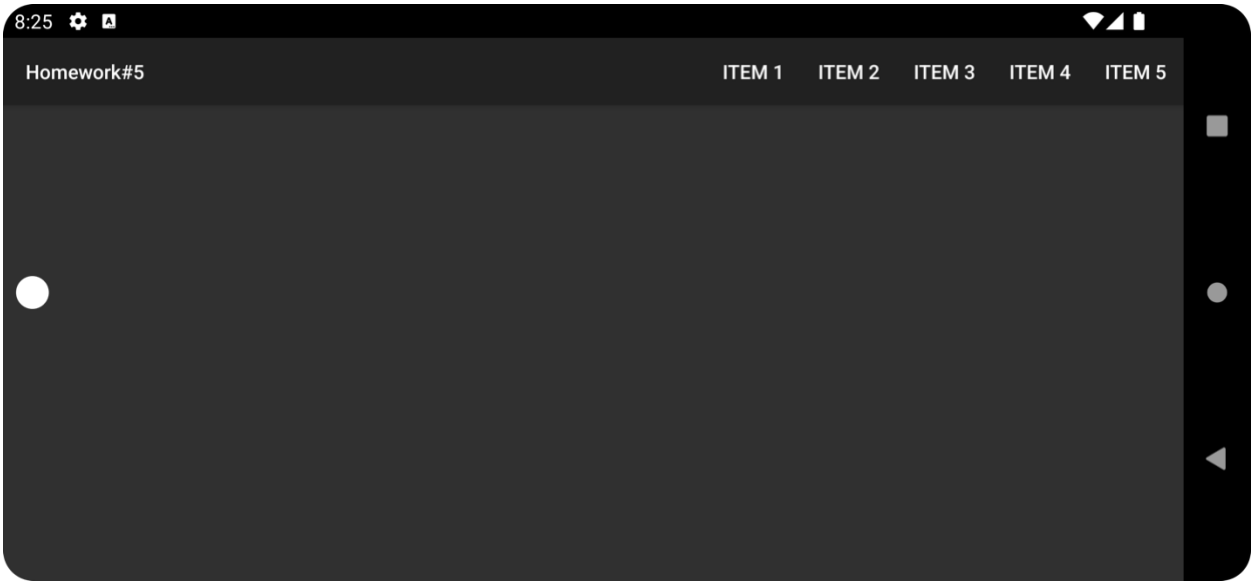
Application Element: Describes the properties and configuration of the application.

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/Theme.AppCompat"
    tools:targetApi="31">
```

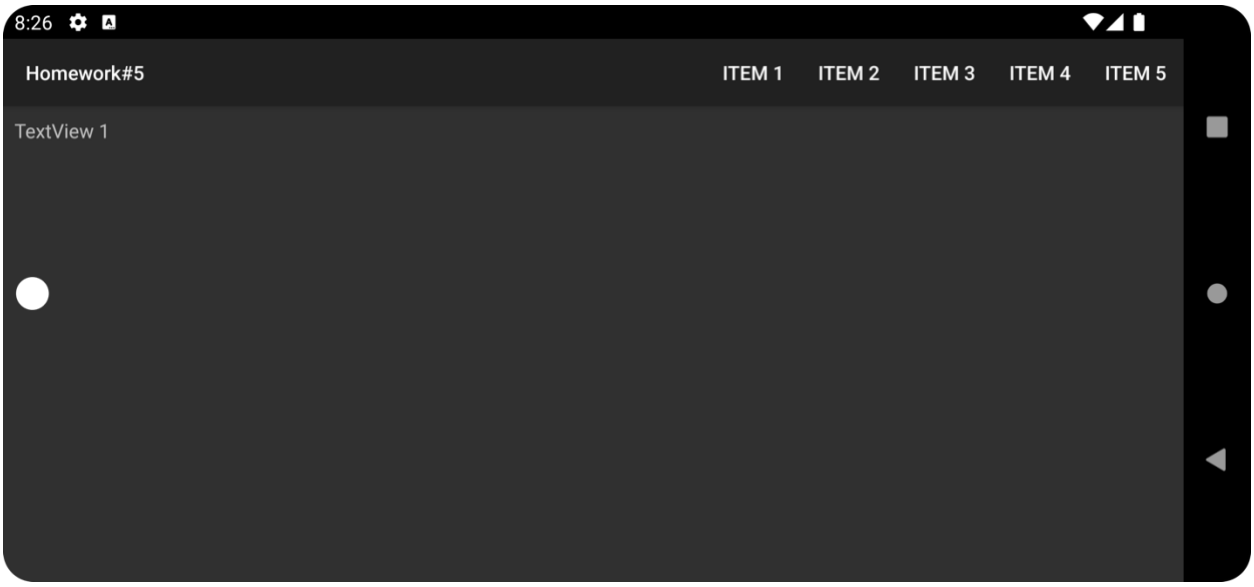
Activity Element: Describes the properties and configuration of an activity.

```
<activity
    android:name=".MainActivity"
    android:screenOrientation="landscape"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

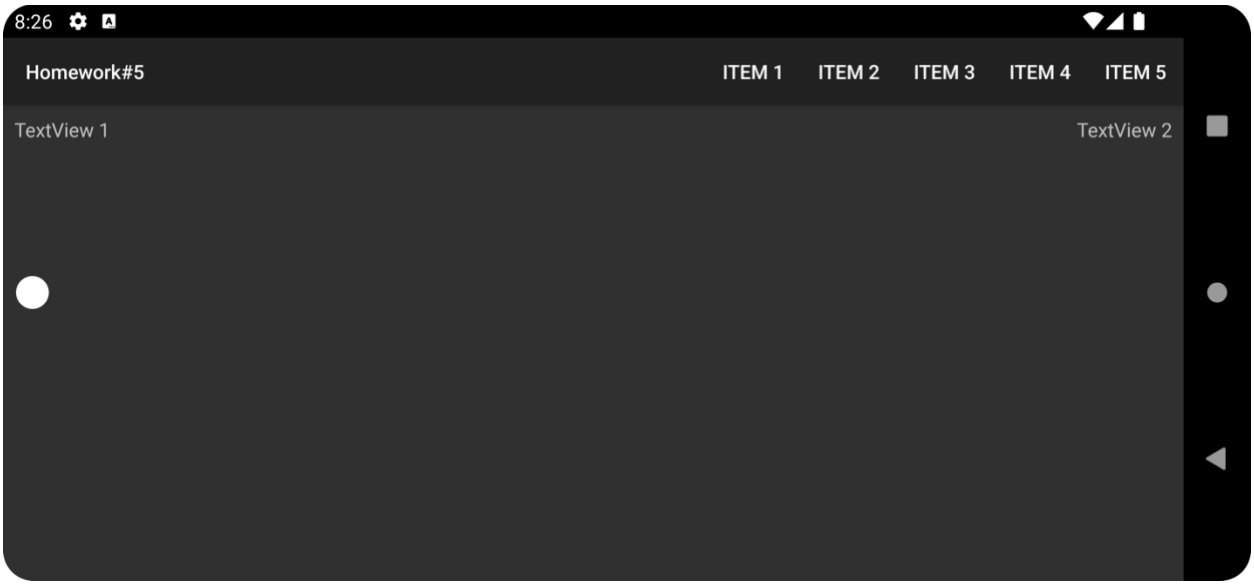
Home screen:



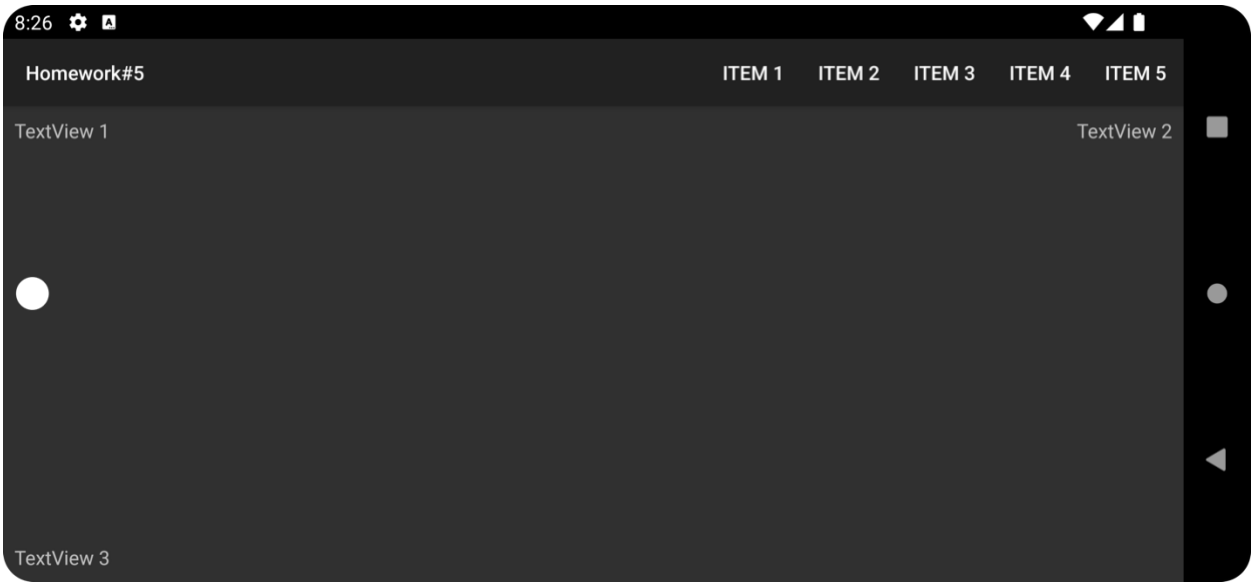
Screen with selection of ITEM1:



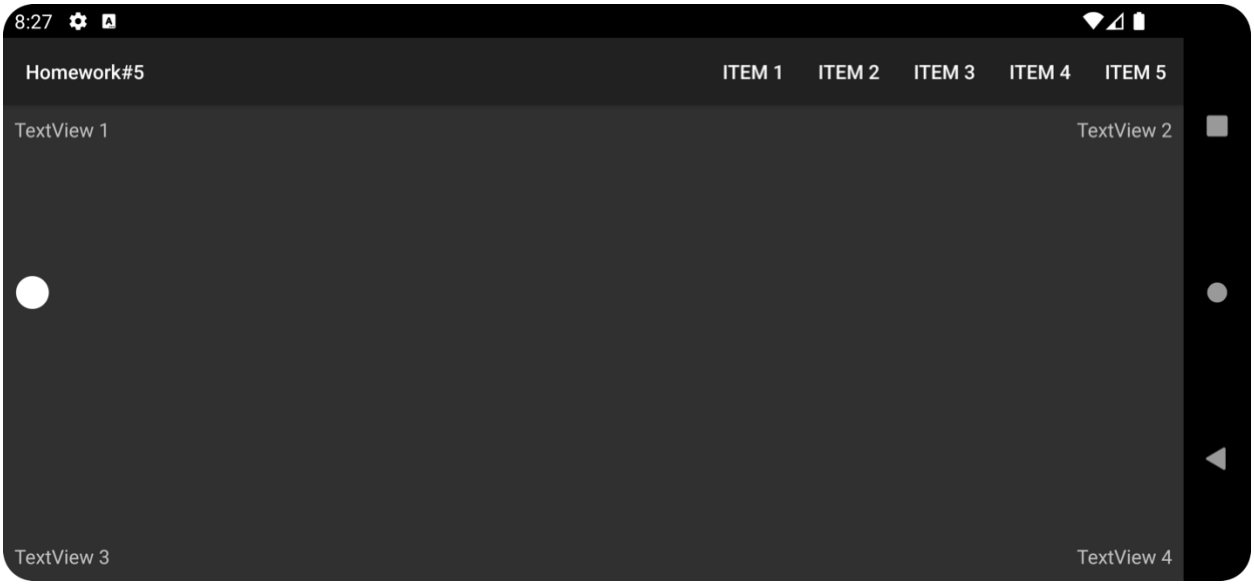
Screen with selection of ITEM2:



Screen with selection ofITEM3:



Screen with selection ofITEM4:



Screen with selection ofITEM5: