

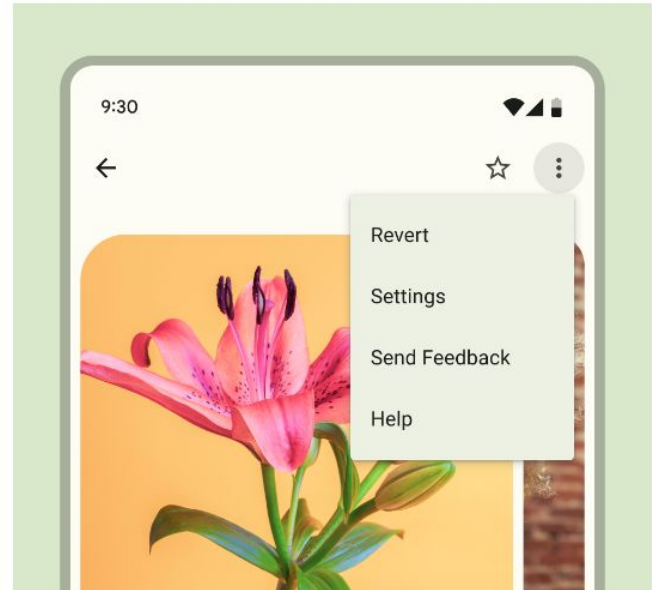
# Menus in Android

# Contents

- What is a menu?
- menu from xml
- menu via code
- Sub menu
- Option Menu
- Context menu
- **Pop Up Menu**
- ActionBar
- **ActionBar & Tabs**
- **View Pager**
- **Action Bar & View Pager**

# What is a menu?

- Menus are a common user interface component in many types of apps.
- With the help of menu, users can experience a smooth and consistent experience throughout the application.



# What is a menu?

- Three fundamental types of menus:
  - **Options menu:** The options menu is the primary collection of menu items for an activity. It's where you place actions that have a global impact on the app, such as "Search," "Compose email," and "Settings."
  - **Context menu:** A context menu is a floating menu that appears when the user performs a touch & hold on an element. It provides actions that affect the selected content or context frame. The contextual action mode displays action items that affect the selected content in a bar at the top of the screen and lets the user select multiple items.
  - **Popup menu:** A popup menu displays a vertical list of items that's anchored to the view that invokes the menu. Actions in a popup menu don't directly affect the corresponding content—that's what contextual actions are for. Rather, the popup menu is for extended actions that relate to regions of content in your activity.

# Define a menu in xml

- Android provides a standard XML format to define menu items.
- Instead of building a menu in your activity's code, define a menu and all its items in an XML menu resource.
- Create an XML file inside your project's res/menu/ directory
  - Create a new folder **menu** inside of our project directory (res/menu) to define the menu and also add a new XML file to build the menu
  - Right click on menu-> New -> Menu Resource file
  - Give a name -> ok

# Define a menu in xml

`<menu>`

Defines a `Menu`, which is a container for menu items. A `<menu>` element must be the root node for the file, and it can hold one or more `<item>` and `<group>` elements.

`<item>`

Creates a `MenuItem`, which represents a single item in a menu. This element can contain a nested `<menu>` element to create a submenu.

`<group>`

An optional, invisible container for `<item>` elements. It lets you categorize menu items so they share properties, such as active state and visibility. For more information, see the [Create a menu group](#) section.

# Define a menu in xml

`android:id`

A resource ID that's unique to the item, which lets the app recognize the item when the user selects

`android:icon`

A reference to a drawable to use as the item's icon.

`android:title`

A reference to a string to use as the item's title.

`android:showAsAction`

The specification for when and how this item appears as an action item in the app bar.

# Sub menu

- If we want to add a submenu in menu item, then we need to add a `<menu>` element as the child of an `<item>`
- Submenus are useful when your app has a lot of functions that can be organized into topics.



# Menu via code

- To use the menu in your activity, `_inflate_` the menu resource, converting the XML resource into a programmable object using **`MenuInflater.inflate()`**
- You can declare items for the menu from your Activity subclass or a Fragment subclass.
- If both your activity and your fragments declare items for the menu, the items are combined in the UI.
- The activity's items appear first, followed by those of each fragment, in the order in which the fragments are added to the activity.
- If necessary, you can reorder the menu items with the `android:orderInCategory` attribute in each `<item>` you need to move.

```

<?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="@string/create_new_project" />
    <item
        android:id="@+id/item2"
        android:title="@string/open_a_project"/>
    <item
        android:id="@+id/item3"
        android:title="@string/help"/>

    <group >
        <item
            android:id="@+id/gi1"
            android:title="@string/group_item1" />
        <item
            android:id="@+id/gi2"
            android:title="@string/group_item2" />
    </group>
    <item android:title="@string/submenu">
        <menu>
            <item android:title="@string/submenu_item1" />
            <item android:title="@string/submenu_item2" />
        </menu>
    </item>
</menu>

```

14:00

OptionsMenuApp

Create New Project

Open a Project

Help

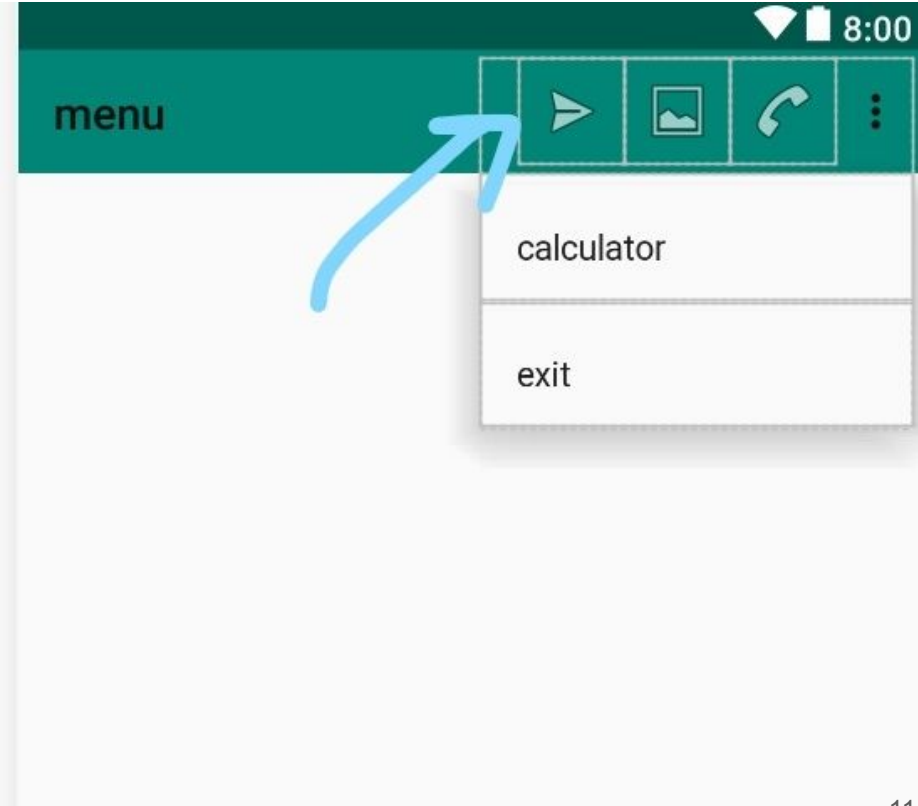
Group Item1

Group Item2

submenu ▶

# Options Menu

- To specify the options menu for an activity, override `onCreateOptionsMenu()`.
- Fragments provide their own `onCreateOptionsMenu()` callback.
- In this method, you can inflate your menu resource, defined in XML, into the Menu provided in the callback.
- You can also add menu items using `add()` and retrieve items with `findItem()` to revise their properties with MenuItem APIs.



# Options Menu

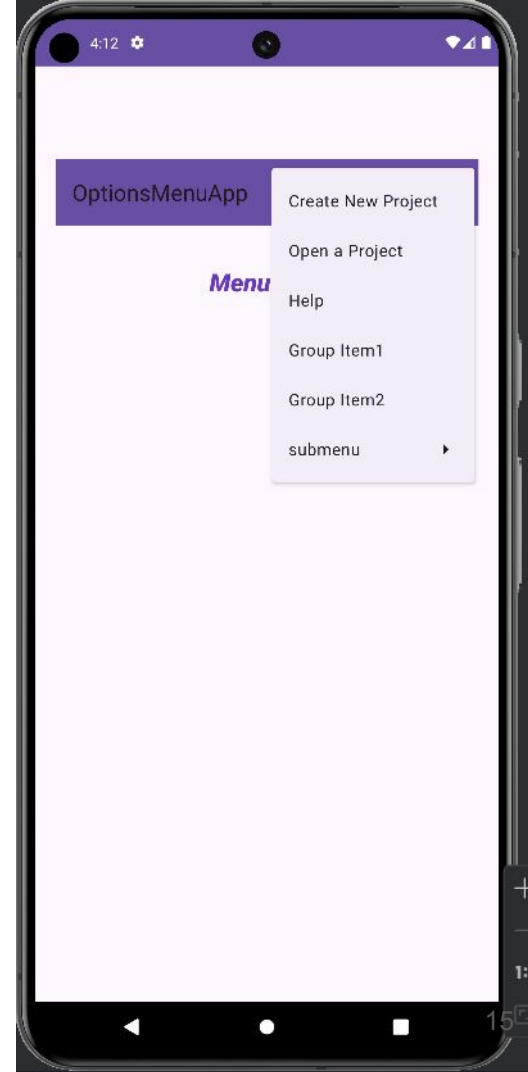
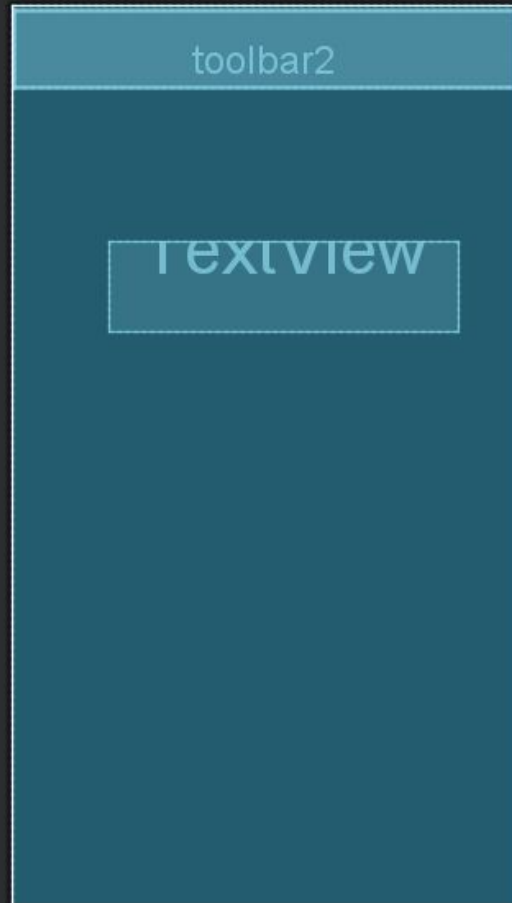
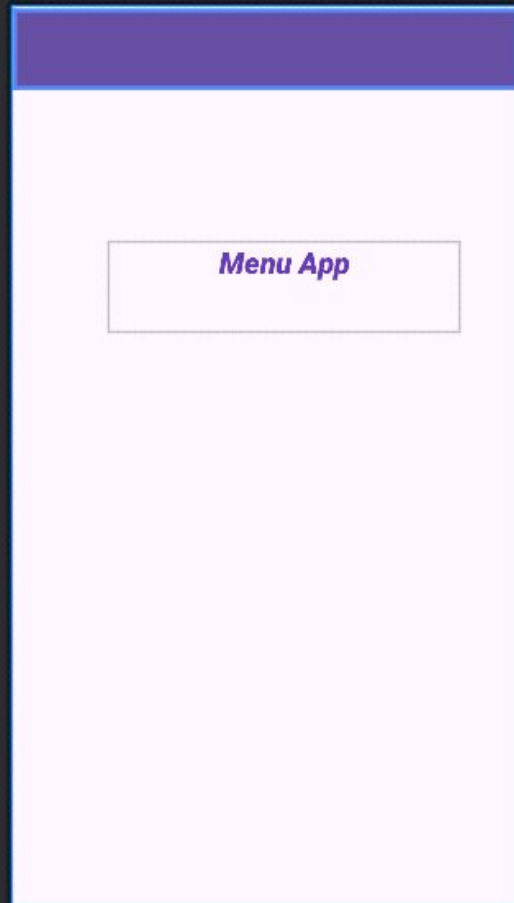
- When the user selects an item from the options menu, including action items in the app bar, the system calls your activity's `onOptionsItemSelected()` method.
- This method passes the `MenuItem` selected.
- You can identify the item by calling `getItemId()`, which returns the unique ID for the menu item, defined by the `android:id` attribute in the menu resource or with an integer given to the `add()` method.
- You can match this ID against known menu items to perform the appropriate action.

# Options Menu

- When you successfully handle a menu item, return true.
- If you don't handle the menu item, call the superclass implementation of `onOptionsItemSelected()`.
- The default implementation returns false.
- If your activity includes fragments, the system first calls `onOptionsItemSelected()` for the activity, then for each fragment in the order the fragments are added, until one returns true or all fragments are called.

# Options Menu

- If you want to modify the options menu based on events that occur during the activity lifecycle, you can do so in the `onPrepareOptionsMenu()` method.
- This method passes you the `Menu` object as it currently exists so you can modify it, such as by adding, removing, or disabling items.
- Fragments also provide an `onPrepareOptionsMenu()` callback.
- The options menu is considered always open when menu items are presented in the app bar.
- When an event occurs and you want to perform a menu update, call `invalidateOptionsMenu()` to request that the system call `onPrepareOptionsMenu()`.



# Options Menu

```
import android.view.Menu;
```

```
import android.view.MenuInflater;
```

```
import android.view.MenuItem;
```

```
import android.widget.Toast;
```

```
import androidx.appcompat.widget.Toolbar;
```



# Options Menu

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar2);  
    setSupportActionBar(toolbar);  
}
```

# Options Menu

```
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.menu_example, menu);  
    return true;  
}
```

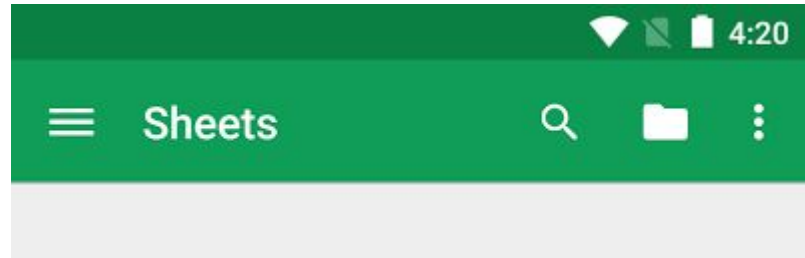
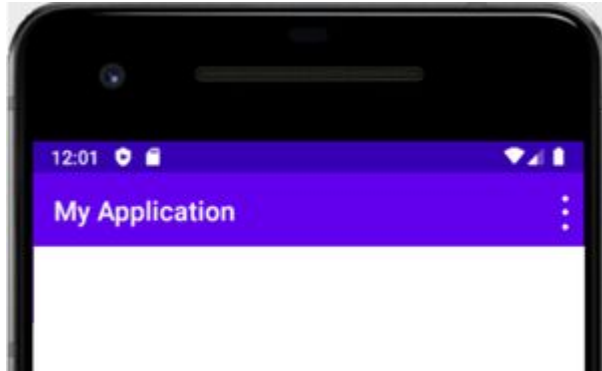
```
public boolean onOptionsItemSelected(MenuItem item) {  
    // Handle item selection.  
    if (item.getItemId() == R.id.item1)  
        Toast.makeText(context: MainActivity.this, text: " Create a New Project", Toast.LENGTH_LONG).show();  
    if (item.getItemId() == R.id.item2)  
        Toast.makeText(context: MainActivity.this, text: " Open a Project", Toast.LENGTH_LONG).show();  
    if (item.getItemId() == R.id.item3)  
        Toast.makeText(context: MainActivity.this, text: " Help", Toast.LENGTH_LONG).show();  
    if (item.getItemId() == R.id.gi1)  
        Toast.makeText(context: MainActivity.this, text: " Group Item 1", Toast.LENGTH_LONG).show();  
    if (item.getItemId() == R.id.gi2)  
        Toast.makeText(context: MainActivity.this, text: " Group Item 2", Toast.LENGTH_LONG).show();  
    if (item.getItemId() == R.id.s1)  
        Toast.makeText(context: MainActivity.this, text: " Sub menu Item 1", Toast.LENGTH_LONG).show();  
    if (item.getItemId() == R.id.s2)  
        Toast.makeText(context: MainActivity.this, text: " Sub menu Item 2", Toast.LENGTH_LONG).show();  
    return super.onOptionsItemSelected(item);  
}
```

# App Bar/ Action Bar

- App bar, also known as the action bar, is one of the most important design elements in your app's activities
- Provides a visual structure and interactive elements that are familiar to users.
- Using the app bar makes your app consistent with other Android apps, letting users quickly understand how to operate your app and have a great experience.
- Action bar displays the title for the activity on one side and an overflow menu on the other.
- Provides useful information to users and gives Android apps a consistent look and feel.

# App Bar/ Action Bar

- Android **Toolbar** widget as an app bar.
- There are other ways to implement an app bar.
  - Some themes set up an **ActionBar** as an app bar by default.
  - Using the AppCompat Toolbar makes it easier to set up an app bar that works on the widest range of devices.
- It also gives you room to customize your app bar later in your app's development.

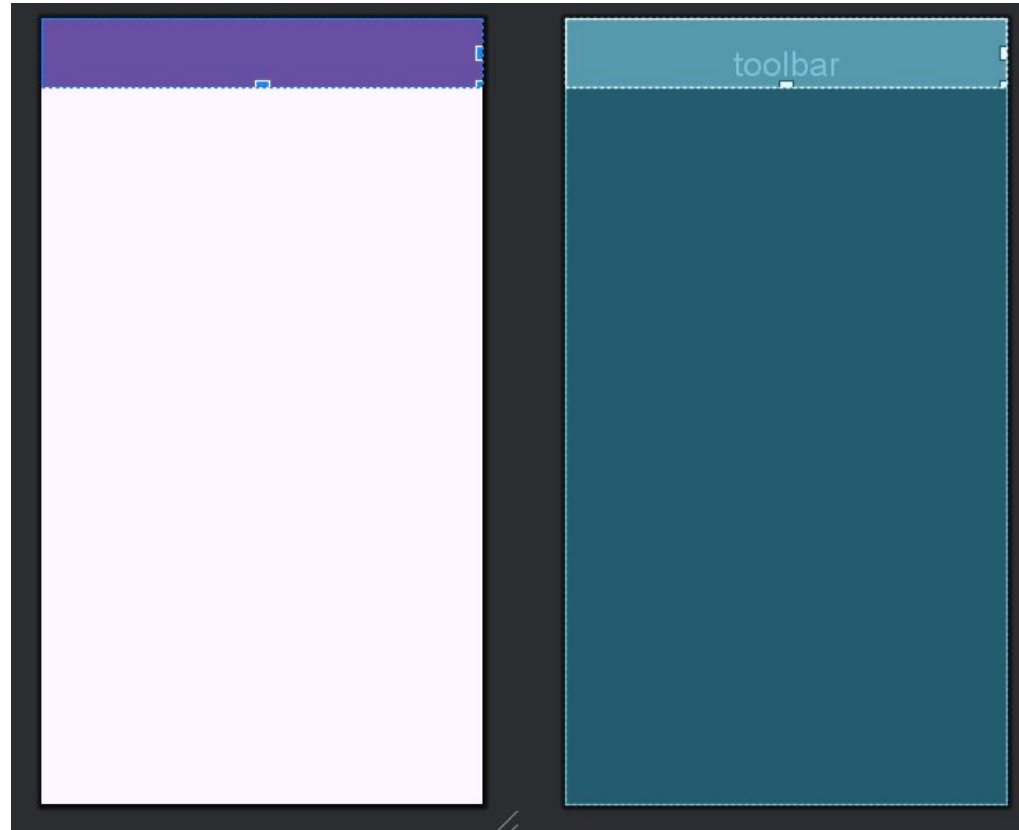
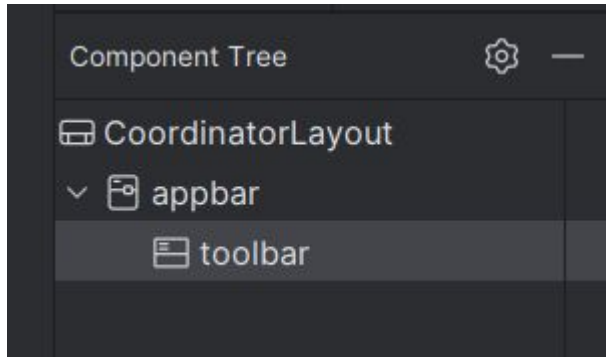


# App Bar/ Action Bar

- Create a New Project with Empty Views Activity.
- Create res/menu directory.
- Add a new Menu resource file
- Add images for action items
  - res -> New-> Image Asset -> Icon Type (Action Bar and Tab Icons) -> create a name for image -> Clip art(asset type) -> Click on Clip art icon -> search and select image -> Next -> Finish.
  - Icons present in res/drawable/
- Design the menu file
  - For each item, android:icon="@drawable/image\_name"
  - app:showAsAction="ifRoom"

# App Bar/ Action Bar

- Design activity\_main.xml
- Insert AppBarLayout from palette
- Insert a Toolbar into AppBarLayout, Give id to Toolbar



# App Bar/ Action Bar

- Write menu code inside MainActivity.java
- Edit the onCreate() to insert the toolbar
  - `setSupportActionBar(toolbar);`

```
<?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/create"
        android:icon="@drawable/new_folder"
        android:title="@string/create_new_folder"
        app:showAsAction="ifRoom" />

    <item
        android:id="@+id/search"
        android:icon="@drawable/search"
        android:title="@string/search"
        app:showAsAction="ifRoom" />

    <item
        android:id="@+id/settings"
        android:icon="@drawable/settings"
        android:title="@string/settings"
        app:showAsAction="ifRoom" />

</menu>
```





# Context Menu

- Context menu is like a floating menu
- Arises when the user has long-pressed or clicked on an item
- Implement functions that define the specific content or reference frame effect.
- Touch & hold on a view create context menu



**Long Press Me!!!**



Component Tree



layout

Ab textview "@string/long\_...

# Context Menu

```
import android.view.ContextMenu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.TextView;  
import android.graphics.Color;  
import androidx.constraintlayout.widget.ConstraintLayout;
```

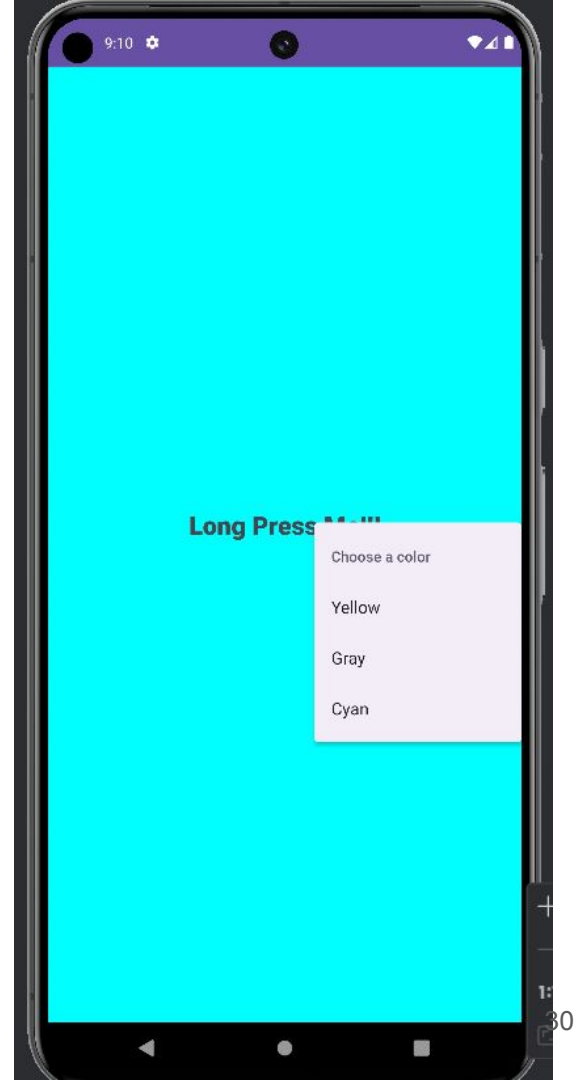
```
public class MainActivity extends AppCompatActivity {  
    2 usages  
    TextView textView;  
    4 usages  
    ConstraintLayout layout;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        // Link those objects with their respective id's that we have given in .XML file  
        textView = (TextView) findViewById(R.id.textview);  
        layout=(ConstraintLayout) findViewById(R.id.layout);  
        // here you have to register a view for context menu  
        // you can register any view like listview, image view, textview, button etc  
        registerForContextMenu(textView);  
    }  
}
```

# Context Menu

```
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {  
    super.onCreateContextMenu(menu, v, menuInfo);  
    // you can set menu header with title icon etc  
    menu.setHeaderTitle("Choose a color");  
    // add menu items  
    menu.add( groupId: 0, v.getId(), order: 0, title: "Yellow");  
    menu.add( groupId: 0, v.getId(), order: 0, title: "Gray");  
    menu.add( groupId: 0, v.getId(), order: 0, title: "Cyan");  
}
```

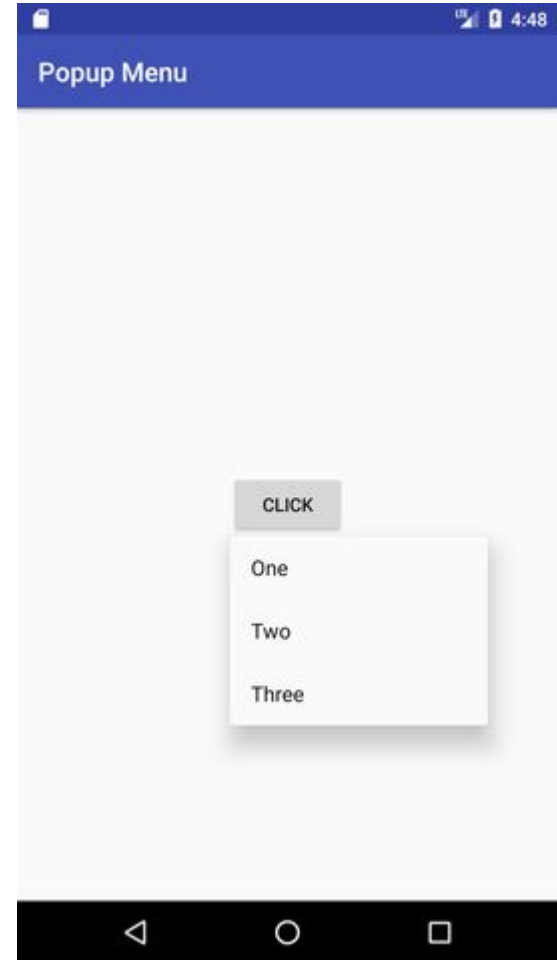
# Context Menu

```
// menu item select listener
@Override
public boolean onContextItemSelected(MenuItem item) {
    if (item.getTitle() == "Yellow") {
        layout.setBackgroundColor(Color.YELLOW);
    } else if (item.getTitle() == "Gray") {
        layout.setBackgroundColor(Color.GRAY);
    } else if (item.getTitle() == "Cyan") {
        layout.setBackgroundColor(Color.CYAN);
    }
    return true;
}
```



# Pop Up Menus

- A Popup Menu displays a Menu in a modal popup window anchored to a View.
- The popup will appear below the anchor view if there is room, or above it if there is not.
- Touching outside of the popup will dismiss it.



# Pop Up Menus

- Make a popup menu anchored to a Button and on click, the popup menu will appear, and on a touch of the popup menu item, a Toast message will be shown.
- Create a New Project. Insert a Button on the UI. (**onclick:showPopUp**)
- Add some color attributes in order to enhance the app bar.
  - Go to app > res > values > colors.xml and add the following color attributes.

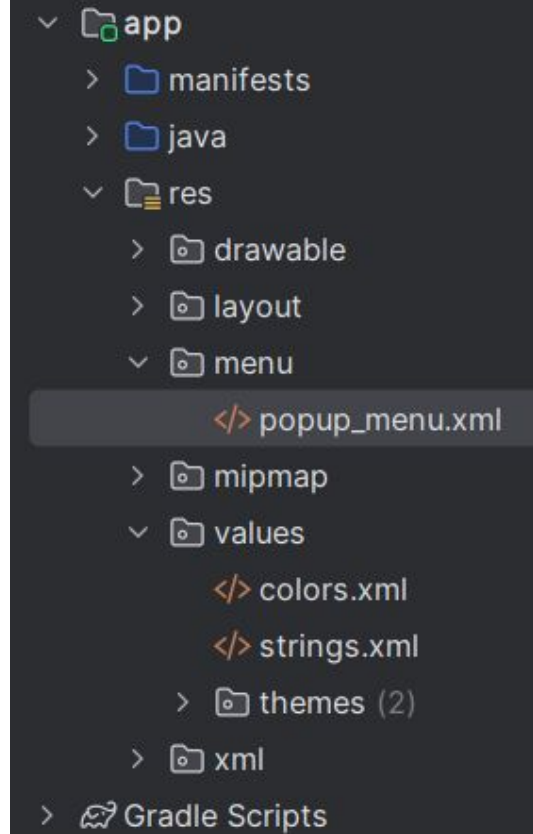
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFF</color>
    <color name="colorPrimary">#0F9D58</color>
    <color name="colorPrimaryDark">#16E37F</color>
    ⚡ <color name="colorAccent">#03DAC5</color>
</resources>
```

```
android:onClick="showPopUp"
```



# Pop Up Menus

- Create a new folder inside /res named 'menu'
- Create a new menu resource file inside the menu folder.
- Design the menu.



```
<?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"
      android:title="Select a Language">

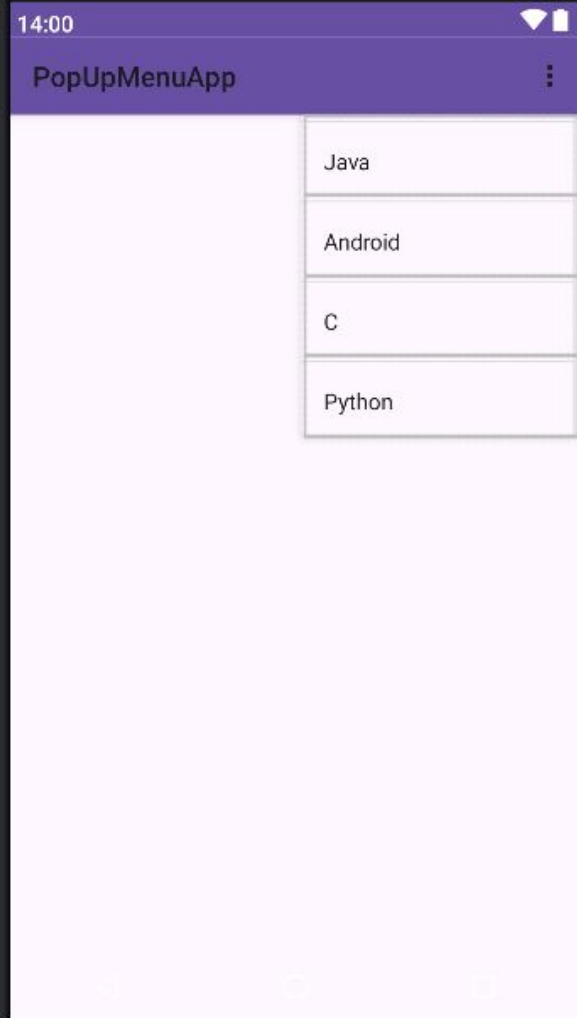
    <item
        android:id="@+id/item1"
        android:title="Java" />

    <item
        android:id="@+id/item2"
        android:title="Android" />

    <item
        android:id="@+id/item3"
        android:title="C" />

    <item
        android:id="@+id/item4"
        android:title="Python" />

</menu>
```



## Write the Java Code (Inflate the menu to main activity)

```
import android.view.MenuInflater;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.PopupMenu;  
import android.widget.Toast;
```

```
Button button;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
}
```

```
public void showPopUp(View v)
{
    button=(Button) findViewById(R.id.button);
    PopupMenu p = new PopupMenu(context: MainActivity.this, button);
    p.getMenuInflater().inflate(R.menu.popup_menu, p .getMenu());
    p.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
        public boolean onMenuItemClick(MenuItem item) {
            Toast.makeText(context: MainActivity.this,item.getTitle(),
                           Toast.LENGTH_SHORT).show();
            return true;
        }
    });
    p.show();
}
```