# CS 473 - MDP
# Mobile Device Programming



Maharishi International University

# CS 473 - MDP
# Mobile Device Programming

MS.CS Program

Department of Computer Science

**Renuka Mohanraj , Ph.D.**

Maharishi International
University

# CS 473 – MDP
# Mobile Device Programming

## Lesson-7
## Menus, Fragments and Tab layout with Swipe views

Maharishi International University

# Wholeness

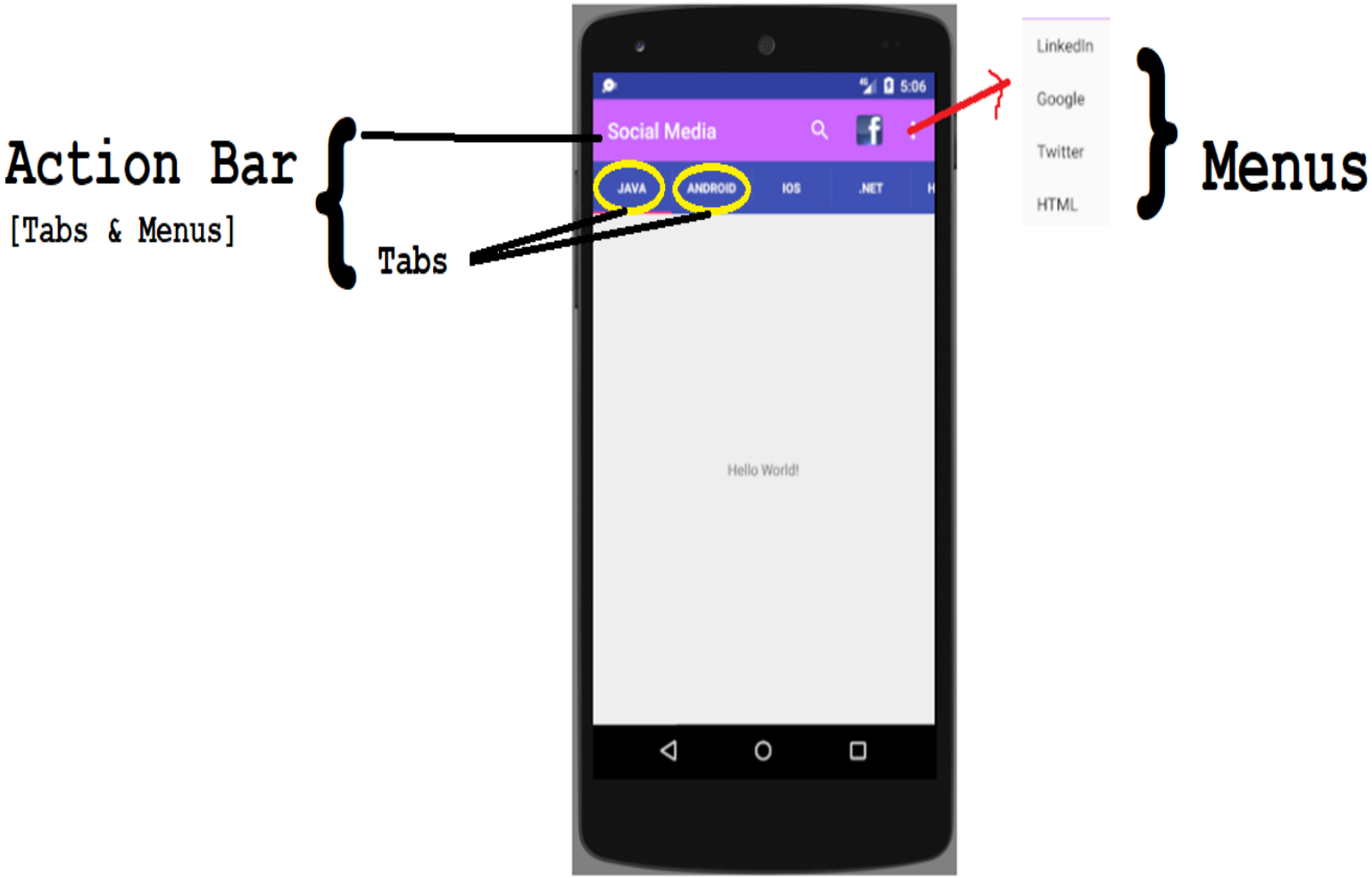
- Your app's user interface is everything that the user can see and interact with. Android provides a variety of pre-built UI components such as menus, fragments, tabs and swipes that allow you to build the graphical user interface for your app. *The ultimate provider of tools for the creation of beautiful and functional content is pure intelligence itself; all creativity arises from this field's self-interacting dynamics.*

# Contents

- Day 1
  - Options Menu
  - Contextual Menus/ Floating Menu
  - Popup menus
- Day 2
  - Fragments
  - Tab Layouts and Swipe Views
  - Material Design

# Option Menu

- Option Menus are the primary menus of android.
- They can be used for settings, search, delete item etc.

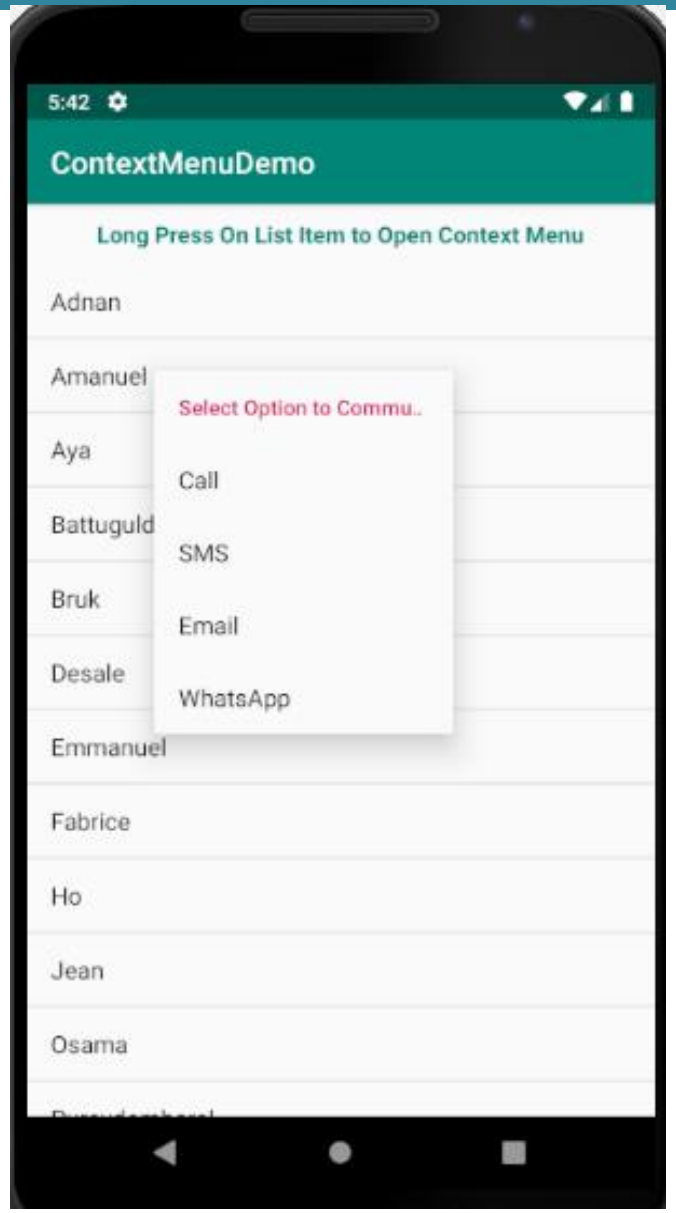


# Popup Menu

- Popup Menu displays a list of items in a modal popup window that is anchored to the view.
- The popup menu will appear below the view if there is a room or above the view in case if there is no space and it will be closed automatically when we touch outside of the popup.

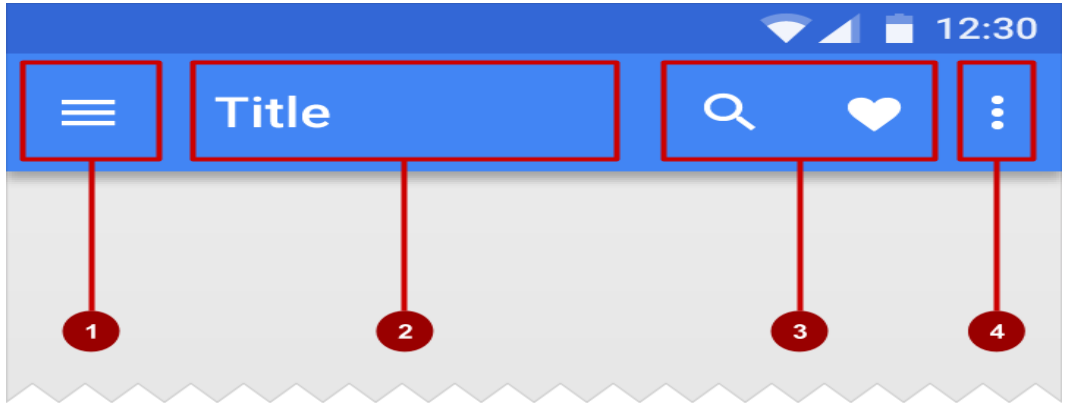
# Context Menu / Floating Menu

- Context Menu Android context menu appears when user press long click on the element.
- The action performs on context menu affect only on the selected content.
- Context Menu can be implemented on any view, but it is mostly used with items of ListView, GridView or other view collections.



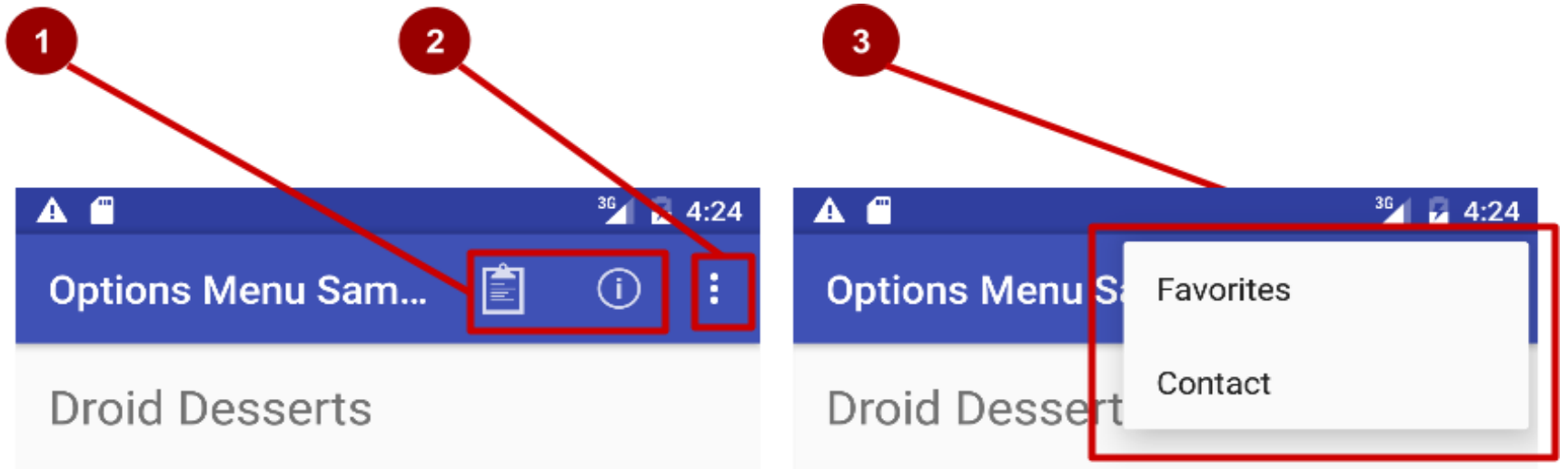
# What is the App Bar?

Bar at top of each screen—(usually) the same for all screens




1. Navigation icon to open navigation drawer
2. Title of current activity
3. Icons for options menu items
4. Action overflow button for the rest of the options menu

# What is the options menu?




- Action icons in the app bar for important items (1)
- Tap the three dots, the "action overflow button" to see the options menu (2)
- Appears in the right corner of the app bar (3) for navigating to other actions

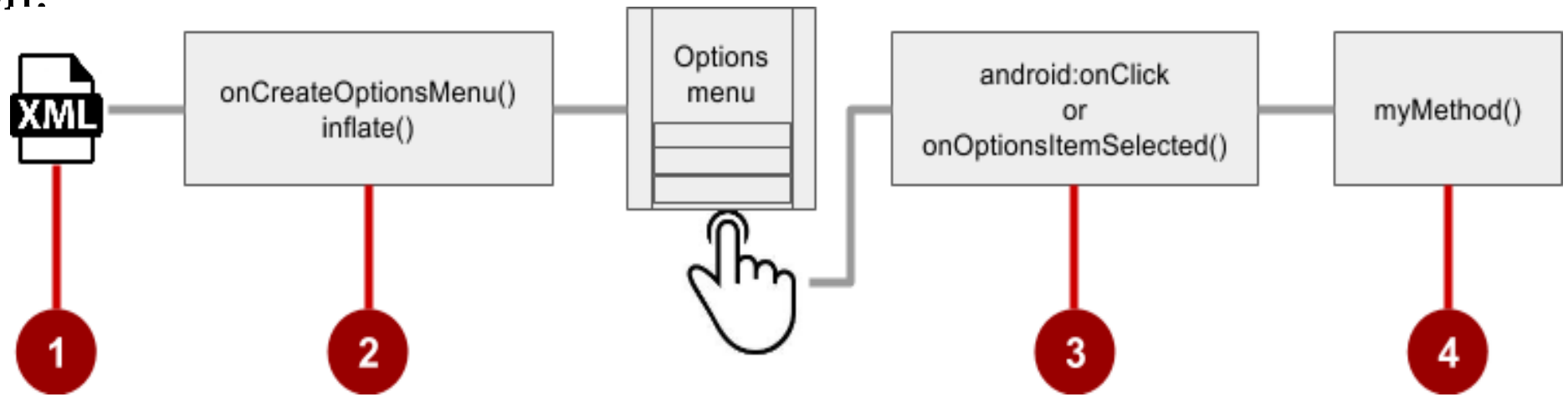
# 1. Adding Options Menu

# Steps to implement options menu

1. XML menu resource (menu_main.xml)
2. Implement onCreateOptionsMenu() to inflate the menu

3. To perform event handling on menu items, you need to override onOptionsItemSelected() method of Activity class.

Refer https://developer.android.com/guide/topics/ui/menus.html to knows more information.

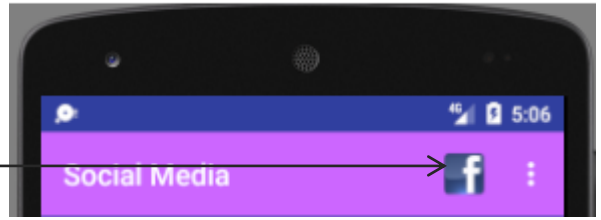


# Step 1 : Create menu resource

- Menus in Android are usually xml resource files.
- Write click on app→res→New→resource directory and choose the resource type as menu then click OK.
- Menu file : To create a menus xml file, Just right click on menu folder New->Resource file, give the file name and then click OK.
- Edit the menu directly in xml
- Menus are defined within <menu> Menu name <menu> tag pairs.
- Each menu item is defined in an <item> Item name<item/> Tag
- Submenus are defined as <menu> within an <item>. Only one level of sub-menu is allowed.

# Adding menu items

```
<menu    xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto">
<!--name space is needed to use app:showAsAction="ifRoom“-->
<item
        android:id="@+id/m1"
        android:title="Facebook"
        android:icon="@drawable/fb_icon"
        app:showAsAction="ifRoom"/>
<!-- Here app is name space name which is defined in the menu header part-->
</menu>
```



# Adding menu items

- Items are added to the menu using the <item> element. Each action item is described using a separate <item>.
- The <item> element has several attributes you can use, here are some of the most common ones:

| | |
|---|---|
| android:id | Gives the item a unique ID. You need this in order to refer to the item in your activity code. |
| android:icon | The item's icon. This is a drawable resource. |
| android:title | The item's text. This may not get displayed if your item has an icon if there's not space in the action bar for both. If the item appears in the action bar's overflow, only the text will be displayed. |
| android: orderInCategory | An integer value that helps Android decide the order in which items should appear in the action bar. |

# Adding menu items

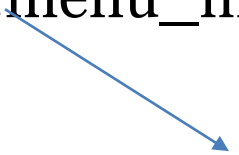
- **app:showAsAction**: The showAsAction attribute is used to say how you want the item to appear in the action bar.
- If you are not using this attribute, always occurred in the overflow dot menu. As an example, you can use it to get an item to appear in the overflow rather than the main action bar, or to place an item on the main action bar only if there's room. The attribute can take the following values:

| "ifRoom" | Place the item in the action bar if there's space. If there's not space, put it in the overflow. |
|---|---|
| "withText" | Include the item's title text. |
| "never" | Put the item in the overflow area, and never in the main action bar. |
| "always" | Always place the item in the main area of the action bar. This value should be used sparingly; if you apply this to many items, they may overlap each other. |

# Step 2 : Inflate options menu

- Override onCreateOptionsMenu() in main activity
- Must import android.View.Menu, android.View.Menuinflater, and android.View.Menuitem

```
class MainActivity : AppCompatActivity() {
 ....
@SuppressLint("RestrictedApi") // Must add to get icon visible
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
if (menu is MenuBuilder) {
        menu.setOptionalIconsVisible(true) // To get icon visible in overflow
    }
getMenuInflater().inflate(R.menu.menu_main, menu);
      return true;
    }
}
```



Name of the menu xml resource file

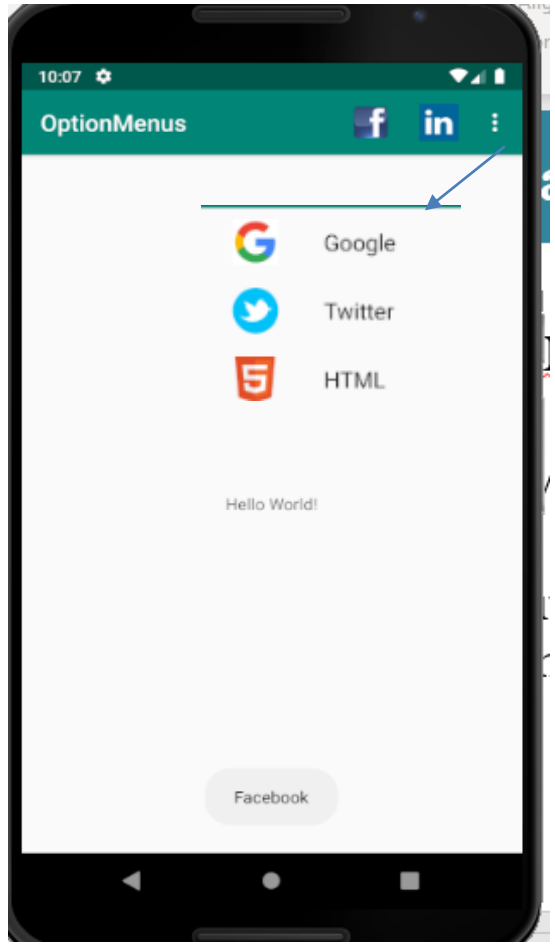
# Step 3 : Override onOptionsItemSelected()

- The onOptionsItemSelected() method takes one attribute, a MenuItem object that represents the item on the action bar that was clicked.
- You can use the MenuItem's getItemId() method to get the ID of the item on the action bar that was clicked so that you can perform an appropriate action, such as starting a new activity.

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    // Handle presses on the action bar menu items
    when (item.itemId) {
        R.id.action_cut -> {
            text_view.text = "Cut"
            return true
        }
        R.id.action_copy -> {
            text_view.text = "Copy"
            return true
        }
        R.id.action_paste -> {
            text_view.text = "Paste"
            return true
        }
        R.id.action_new -> {
            text_view.text = "New"
            return true
        }
    }
    return super.onOptionsItemSelected(item)
}
```

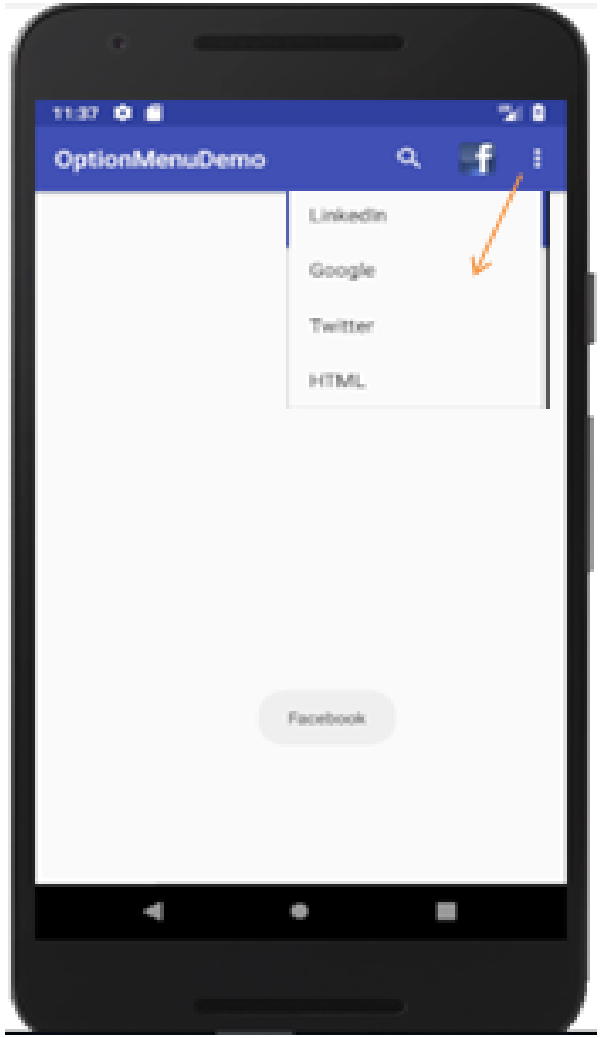
# Hands on Example 1 : OptionMenu

- Refer Lesson6\OptionMenuDemo folder.
- The Appbar shows menu item icons
- Overflow dot button shows additional option menus.



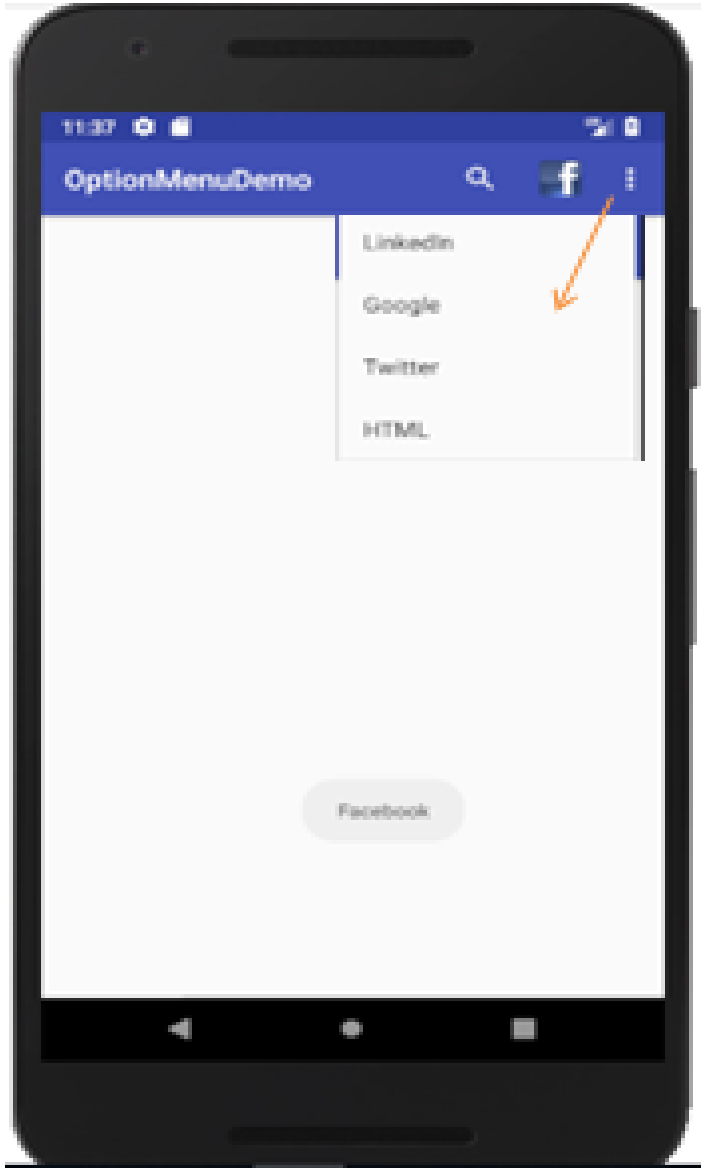
# Hands on Example 1 : main_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto">
   xmlns:app="http://schemas.android.com/apk/res-auto">
   xmlns:app="http://schemas.android.com/apk/res-auto">
   <item
      android:id="@+id/m1"
      android:title="Facebook"
      android:icon="@drawable/fb_icon"
      app:showAsAction="ifRoom|withText"/>
    <item
      android:id="@+id/m2"
      android:title="LinkedIn"
      android:icon="@drawable/lin_icon"
      app:showAsAction="ifRoom"/>
```



# Hands on Example 1 : main_menu.xml

```
<item
      android:id="@+id/m3"
      android:title="Google"
      android:icon="@drawable/g_logo"
      app:showAsAction="ifRoom"/>
   <item
      android:id="@+id/m4"
      android:title="Twitter"
      android:icon="@drawable/twitter_logo"
      app:showAsAction="ifRoom"/>
<item
      android:id="@+id/m5"
      android:title="HTML"
      android:icon="@drawable/html_logo"
      app:showAsAction="ifRoom"/>
</menu>
```



# Hands on Example 1 :MainActivity.kt

```
import android.content.Context
import androidx.appcompat.app.AppCompatActivity
import com.example.optionmenuexample.databinding.ActivityMainBinding
import android.os.Bundle
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
            binding = ActivityMainBinding.inflate(layoutInflater)
            setContentView(binding.root)     }
```

# Hands on Example 1 :MainActivity.kt

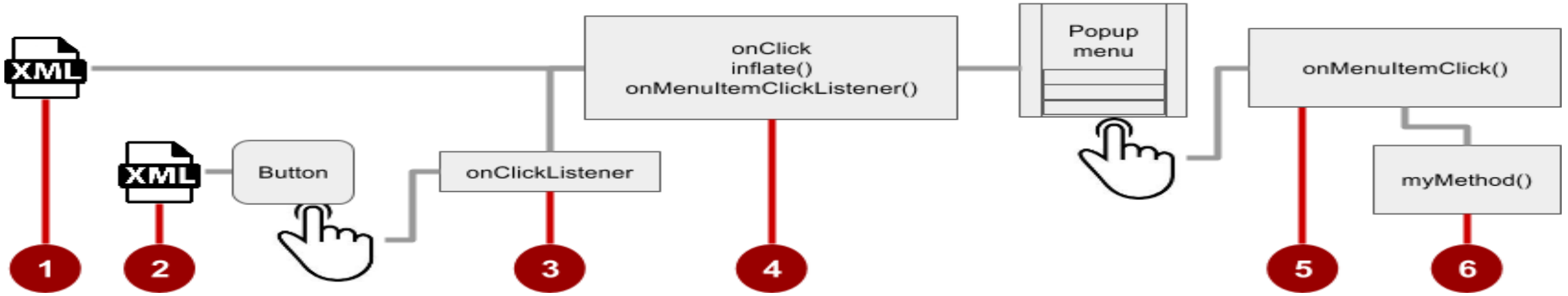
```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
      menuInflater.inflate(R.menu.main_menu,menu)
      }
      return super.onCreateOptionsMenu(menu)
   }

 override fun onOptionsItemSelected(item:MenuItem):Boolean {
      Toast.makeText(
         applicationContext,
         item.title.toString(),
         Toast.LENGTH_LONG).show()
      return super.onOptionsItemSelected(item)
   }}
```

Refer: Drag and drop design using Menu Editor, read Chapter 38 – Creating and managing Overflow menus on Android from Kotlin Android Studio Essentials text book

# 2. Popup Menus

- Vertical list of items anchored to a view. You must follow these steps




1. Create XML menu resource file and assign appearance and position attributes
2. Add `Button` for the popup menu icon in the XML activity layout file
3. Assign `onClickListener` to the button
4. Override `onClick()` to inflate the popup and register it with `onMenuItemClickListener()`
5. Implement `onMenuItemClick()`
6. Create a method to perform an action for each popup menu item

# activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    android:id="@+id/rl"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="My text color is changing."
        android:textSize="50sp"
        android:textStyle="bold"
        android:fontFamily="sans-serif-condensed"/>
    <Button
        android:id="@+id/bt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="214dp"
        android:fontFamily="sans-serif-condensed"
        android:text="Click to get Popup."
        android:textSize="20sp"
        android:textStyle="bold" />
</RelativeLayout>
```

# popup_menu

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/red"
        android:title="Red"/>
    <item
        android:id="@+id/green"
        android:title="Green"/>
    <item
        android:id="@+id/blue"
        android:title="Blue"/>
</menu>
```

# MainActivity.kt

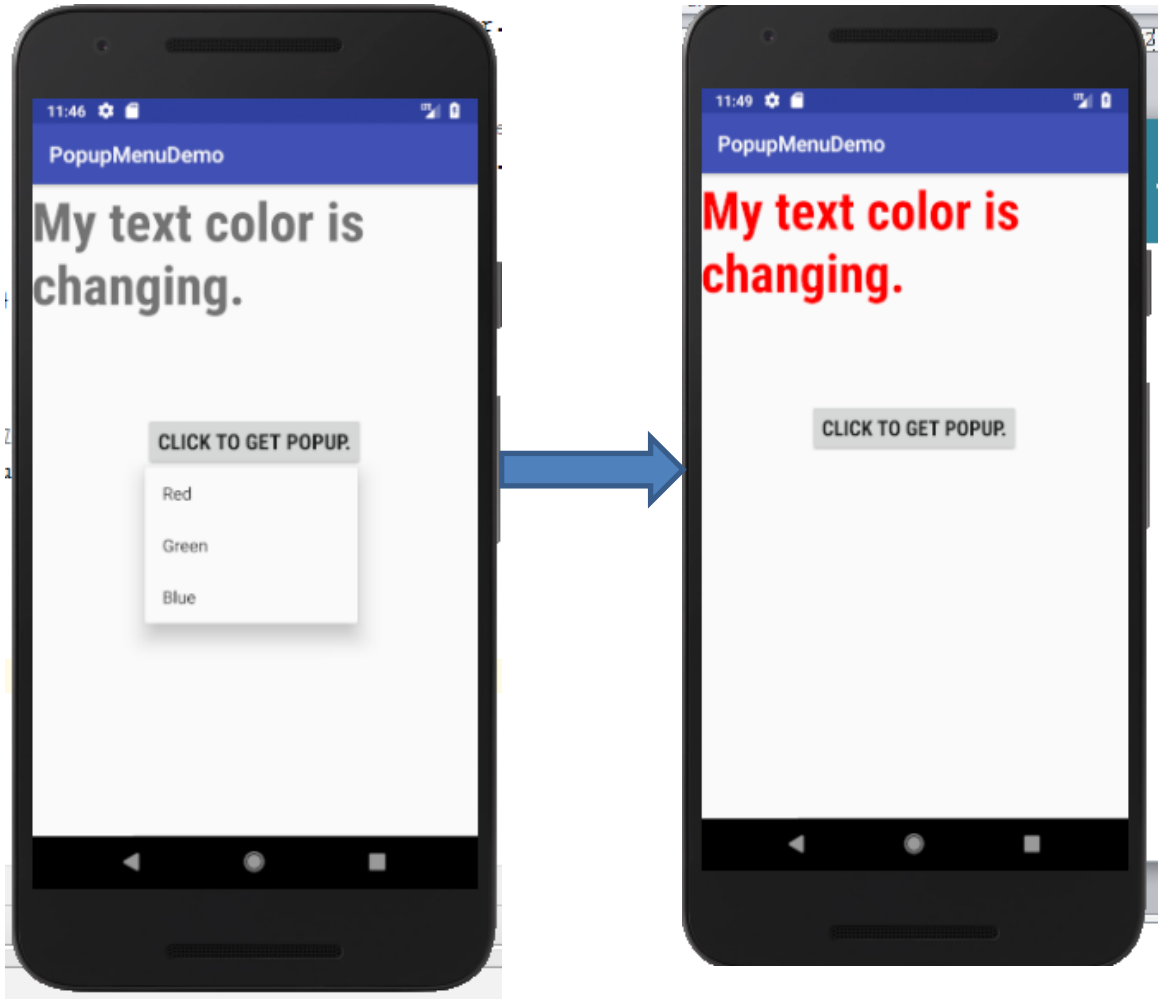
```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
            binding = ActivityMainBinding.inflate(layoutInflater)
            setContentView(binding.root)
// Set click listener to the button
 binding.bt.setOnClickListener {
            // Initialize a new instance of popup menu
            val popupMenu = PopupMenu(this, binding.bt)
            // Inflate the popup menu
            popupMenu.menuInflater.inflate(R.menu.mypopup, popupMenu.menu)
```

# MainActivity.kt

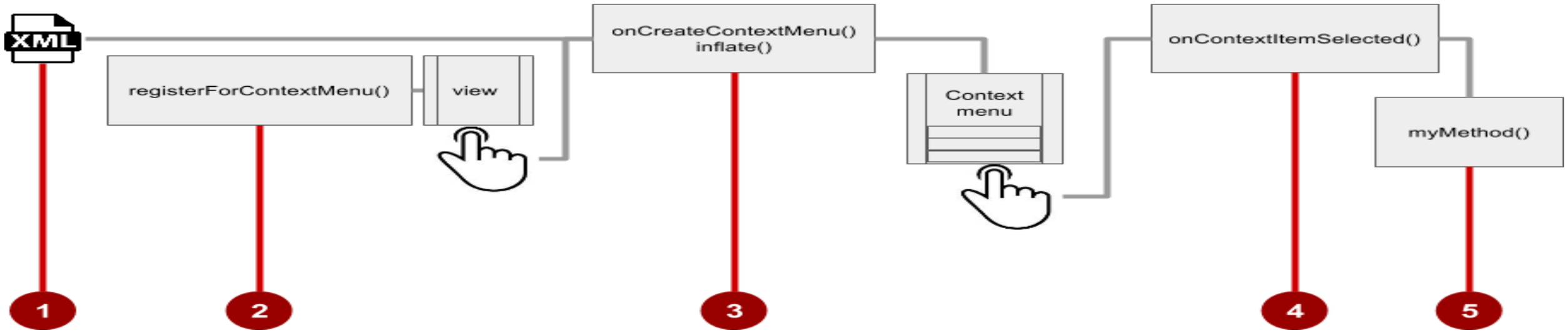
```
    // Set a click listener for menu item click
    popupMenu.setOnMenuItemClickListener { menuItem ->
        when (menuItem.itemId) {
            // Handle the menu items here
            R.id.red -> { // Set the text color to red
                tv.setTextColor(Color.RED)
                true  ^setOnMenuItemClickListener
            }
            R.id.green -> { // Set the text color to green
                tv.setTextColor(Color.GREEN)
                true  ^setOnMenuItemClickListener
            }
            R.id.blue -> { // Set the text color to blue
                tv.setTextColor(Color.BLUE)
                true  ^setOnMenuItemClickListener
            }
            else -> false  ^setOnMenuItemClickListener
        }
    } // Finally, show the popup menu
    popupMenu.show()
}
```

# Hands on Example - 2

- Popup menu is fixed for the button control.
- If the user select the Red option, text color will change in Red, similarly for other options.
- Refer **Lesson6\ PopupMenuDemo**



# 3. Context Menu Steps




1. Create XML menu resource file and assign appearance and position attributes
2. Register view to use a context menu using `registerForContextMenu()`
3. Implement `onCreateContextMenu()` in the activity or fragment to inflate the menu
4. Implement `onContextItemSelected()` to handle menu item clicks
5. Create a method to perform an action for each context menu item

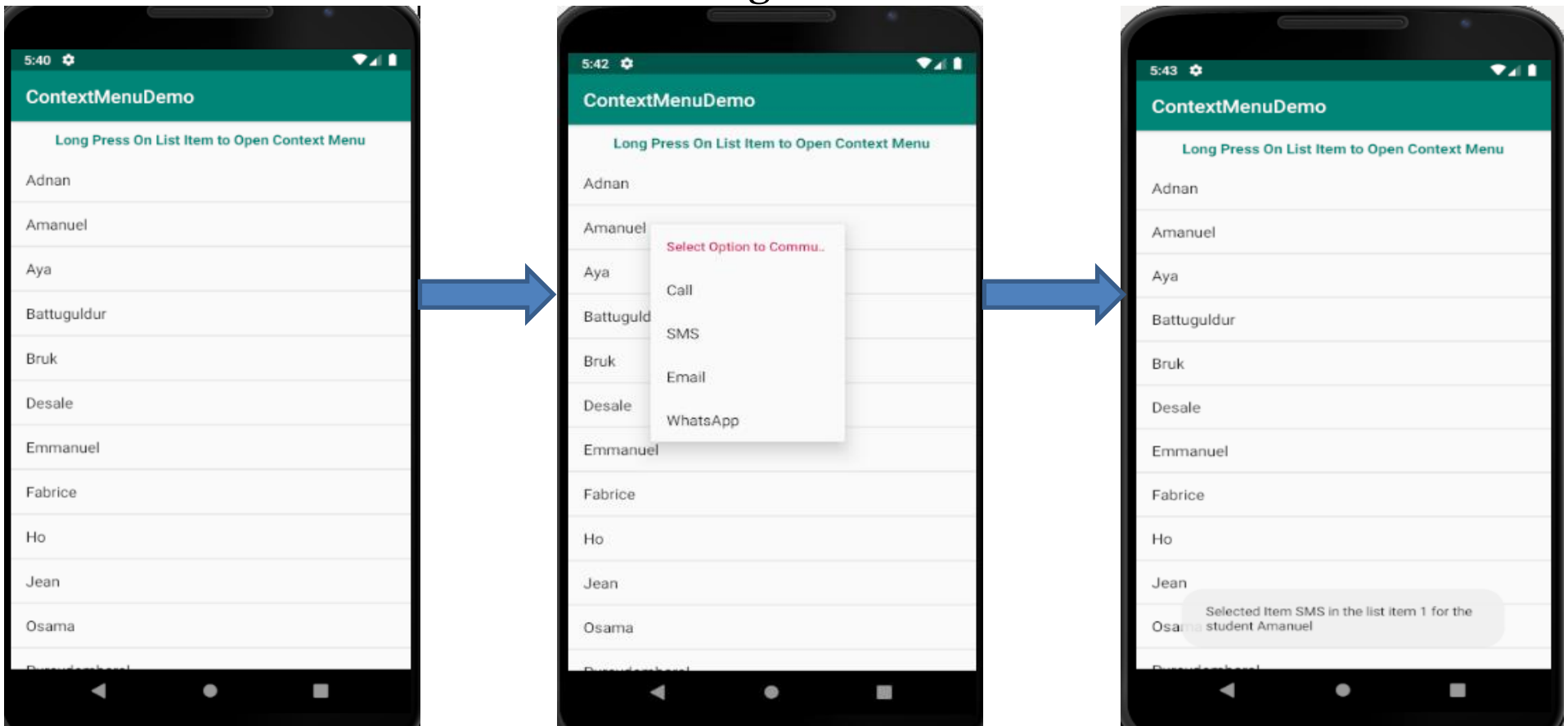
# Hands on Example 3 – ContextMenu Demo

Context Menu – Demo shows how programmatically  add menus.

List of  Students

Performing a long Press selecting context menu

Get Toast after



# MainActivity.kt

```
class MainActivity : AppCompatActivity() {
private lateinit var binding: ActivityMainBinding
    private val callid:Int =1       private val smsid:Int =2     private val mailid:Int =3
    private val whatsid:Int =4
    lateinit var students: Array<String>
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
       setContentView(binding.root)
students = arrayOf("Adnan", "Amanuel", "Aya","Battuguldur", "Bruk", "Desale",   "Emmanuel",   "Fabrice", "Ho",
"Jean",  "Osama", "Purevdemberel", "Resty", "Samiksha", "Sibtain",    "Xingke",  "Yared",   "Yousef", "Ahmed")
        //Creating Adapter
        val adp = ArrayAdapter(this, android.R.layout.simple_list_item_1, students)
        //Set Adapter to ListView
     binding.lv.adapter = adp
        //Register ListView for Context Menu. Accepts a view argument that should show a //context menu.
        registerForContextMenu(binding.lv)
    }
```

# MainActivity.kt

```
override fun onCreateContextMenu(menu: ContextMenu?, v: View?, menuInfo: ContextMenu.ContextMenuInfo?) {
    super.onCreateContextMenu(menu, v, menuInfo)
    //Set Header of Context Menu
    menu!!.setHeaderTitle("Select Option to Communicate")
    menu.add(0, callid, 0, "Call")
    menu.add(0, smsid, 1, "SMS")
    menu.add(0, mailid, 2, "Email")
    menu.add(0, whatsid, 3, "WhatsApp")
    /*   menu.add get 4 Parameters
        1. groupId if you want to add multiple Group than for every group Id is Different Here
          we have only One Group so We take 0(Zero) as GroupId
        2. menu id for Item Id
        3. Set Order of Our Item(Position Of Item) if you Change order of Call to 1 and SMS to zero
            than in Menu SMS Display First.
        4. Title to Display on Context menu
     */
}
```

# MainActivity.kt

```
override fun onContextItemSelected(item: MenuItem?): Boolean {
    //Get Order of Selected Item such as 0,1,2,3
    val selectedItemOrder = item!!.order
    //Get Title Of Selected Item such as Call, SMS, Email and WhatsApp
    val selectedItemTitle = item.title
    // Get the id of the menu selected
    val id = item.itemId
    //To get Name of student Click on ListView
    val info = item.menuInfo as AdapterView.AdapterContextMenuInfo
    val listPosition = info.position
    val name = students[listPosition]
    Toast.makeText(this,
        "Context Menu $selectedItemTitle in the order $selectedItemOrder with the id $id " +
            "for the student $name",
      Toast.LENGTH_LONG).show()
    return true
  }}
```

# Adding Search Bar

## Refer : ContextMenuDemo

Need to add menu for SearchBar in your menu.xml layout file.

```
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:android="http://schemas.android.com/apk/res/android">

   <item
      app:showAsAction="always"
      android:icon="@android:drawable/ic_menu_search"
      android:id="@+id/menu_item_search"
      android:title="Search"
      app:actionViewClass="android.widget.SearchView" />
</menu>
```

# Adding Search Bar

Override fun onCreateOptionsMenu() to inflate the Search menu into your activity by writing the below code

```
override fun onCreateOptionsMenu(menu: Menu):Boolean {
    // Inflate the menu; this adds items to the action bar if it is present.
    menuInflater.inflate(R.menu.main, menu)
    // get the currently set action view for this menu item which returns View and cast it as a SearchView
    val searchView = menu.findItem(R.id.menu_item_search).actionView as SearchView
    // Set Search bar hint
    searchView.queryHint= "Search"
    // Gets information about a searchable activity (Activity exist, searchable activity or null)
    searchView.setSearchableInfo(searchManager.getSearchableInfo(componentName))
   /* Listener to perform the search based on the types text, need to implement
     SearchView.OnQueryTextListener */
    searchView.setOnQueryTextListener(this)
    return super.onCreateOptionsMenu(menu)
  }
```

# Adding Search Bar

need to implement interface SearchView.OnQueryTextListener and override the below methods

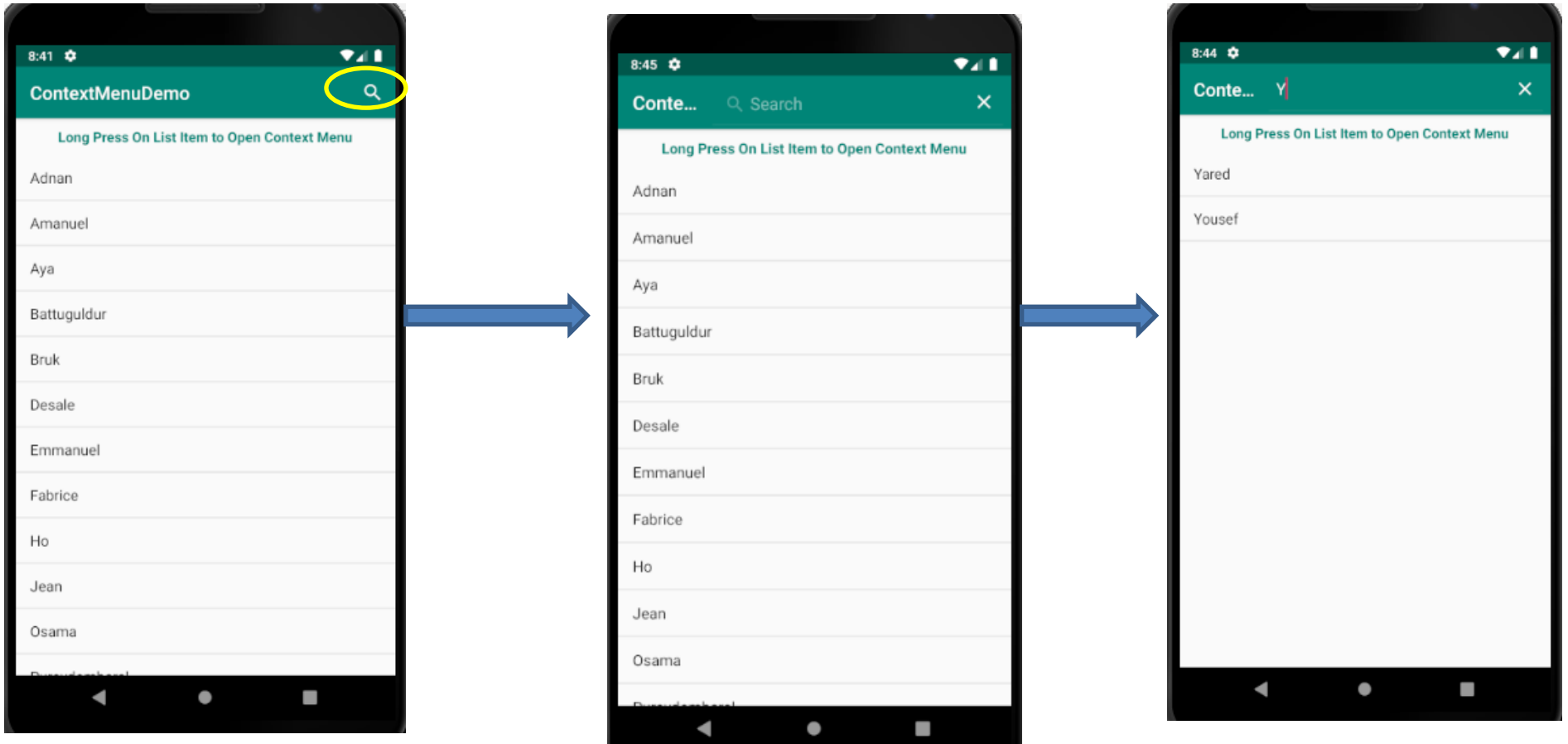
```
// Filter the text from the List items
    override fun onQueryTextChange(newText: String?): Boolean {
     // adp is an ArrayAdapter for the ListView to filter the searched data
      adp?.filter?.filter(newText)
        return true
    }
   override fun onQueryTextSubmit(query: String?): Boolean {
        return false
    }
```

# Example-3 is used to add SearchBar

ListView
With SearchBar

After Clicking
Search Icon

Filter the Text
typed from the
ListView



# Main Point 1

Menus are a common user interface component in many types of applications. To provide a familiar and consistent user experience, you should use the Menu APIs to present user actions and other options in your activities. Science of Consciousness: TM is a common interface to perform greater activities and taking right decision in life like how we choose the menus and corresponding actions in your android device immediately.