Menus & Dialogs in Android (Kotlin)

Topic: Menus & Dialogs in Android
Learning Objectives

- Create and handle **Options Menus** (top-right menu in toolbar)
- Use **Context Menus** (appears when you long-press an item)
- Create **AlertDialogs** (pop-up message boxes for confirmation, alerts, etc.)

Concept Overview
1. Options Menu

- Appears in the **App Bar (three dots icon)**.
- Commonly used for global app actions like *Settings, Help, About*.
- Defined in res/menu/ folder as an XML file (e.g., menu_main.xml).
- Handled in onOptionsItemSelected() in your Activity.

2. Context Menu

- Opens when user **long-presses** an item (like text or list element).
- Best for item-specific actions like *Edit, Delete, Share*.

3. AlertDialog

- Used to show **pop-up dialogs** for confirmation or alerts.
- Includes title, message, and buttons (e.g., OK, Cancel).

Note

Component	Purpose	Trigger
Options Menu	App-wide actions like Settings, Help	Top-right (three dots)
Context Menu	Actions for specific views	Long press
AlertDialog	Pop-up confirmation or alerts	Button/menu action

Example App: “Menu & Dialog Demo”
This app demonstrates:

- An Options Menu with “About” and “Exit”
- A Context Menu when long-pressing text
- An AlertDialog confirmation before exit

Step-by-Step Implementation

Step 1: Create a New Project
Name: MenuDialogDemo
Language: Kotlin
Minimum SDK: API 21 or above.

Step 2: Layout (activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textViewInfo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Long press me for Context Menu"
        android:textSize="20sp"
        android:padding="20dp" />
</LinearLayout>
```

Step 3: Create Menu XML (res/menu/menu_main.xml)

```
Create folder if not exists:
res → New → Android Resource Directory → type: menu → OK
Then inside it create menu_main.xml:
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/menu_about"
        android:title="About" />
    <item
        android:id="@+id/menu_exit"
        android:title="Exit" />
</menu>
```

Step 4: Kotlin Code (MainActivity.kt)

```
package com.example.menudialogdemo

import android.os.Bundle
```

```
import android.view.ContextMenu
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private lateinit var textViewInfo: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        textViewInfo = findViewById(R.id.textViewInfo)

        // Register textView for context menu
        registerForContextMenu(textViewInfo)
    }

    // ---------- OPTIONS MENU ----------
    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.menu_main, menu)
        return true
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        return when (item.itemId) {
            R.id.menu_about -> {
                Toast.makeText(this, "Menu & Dialog Demo App", Toast.LENGTH_SHORT).show()
                true
            }
            R.id.menu_exit -> {
                showExitDialog()
                true
            }
            else -> super.onOptionsItemSelected(item)
        }
    }

    // ---------- CONTEXT MENU ----------
    override fun onCreateContextMenu(
        menu: ContextMenu?,
        v: View?,
        menuInfo: ContextMenu.ContextMenuInfo?
```

```
    ) {
        super.onCreateContextMenu(menu, v, menuInfo)
        menu?.setHeaderTitle("Choose an Action")
        menu?.add(0, v?.id ?: 0, 0, "Copy Text")
        menu?.add(0, v?.id ?: 0, 1, "Share Text")
    }

    override fun onContextItemSelected(item: MenuItem): Boolean {
        return when (item.title) {
            "Copy Text" -> {
                Toast.makeText(this, "Text Copied!", Toast.LENGTH_SHORT).show()
                true
            }
            "Share Text" -> {
                Toast.makeText(this, "Sharing Text...", Toast.LENGTH_SHORT).show()
                true
            }
            else -> super.onContextItemSelected(item)
        }
    }

    // ---------- ALERT DIALOG ----------
    private fun showExitDialog() {
        val builder = AlertDialog.Builder(this)
        builder.setTitle("Exit App")
        builder.setMessage("Are you sure you want to exit?")
        builder.setPositiveButton("Yes") { _, _ ->
            finish()
        }
        builder.setNegativeButton("No", null)
        builder.show()
    }
}
```

Step 5: Run the App

Tap the **three-dot menu** → choose *About* or *Exit*
Long-press the **TextView** → shows a **Context Menu**
Choose "Exit" → shows an **AlertDialog** for confirmation

Activity

- In "About" → show a **Dialog** with developer info.
- In Context Menu → actually **copy to clipboard** or **share via Intent**.
- Add **icons** to menu items using android:icon.

Extend the same "Menu & Dialog Demo" app to include a **Custom Dialog**

The user can enter their **name and email**, and the app displays it in a Toast message.

Custom Dialogs in Android
Learning Objectives

- Create and use a **Custom AlertDialog** with your own XML layout.
- Access and handle input values inside the dialog.
- Show user-entered data dynamically.

Note:

Component	Purpose	Example
AlertDialog	Show alerts, confirmations	Exit confirmation
Custom Dialog	Gather user input	Name & Email form
Options Menu	App-wide actions	About, Exit, Info
Context Menu	View-specific actions	Copy/Share text

Step 1: Add a New Menu Option
Open your menu_main.xml and add one more item for the custom dialog:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item
      android:id="@+id/menu_about"
      android:title="About" />
   <item
      android:id="@+id/menu_userinfo"
      android:title="Enter Info" /> <!-- 🆕 Custom dialog option -->
   <item
      android:id="@+id/menu_exit"
      android:title="Exit" />
</menu>
```

Step 2: Create a Custom Layout
Create a new file:
res/layout/dialog_userinfo.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical"
```

```
    android:padding="20dp">

    <EditText
        android:id="@+id/etName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your name"
        android:inputType="textPersonName"
        android:layout_marginBottom="10dp" />

    <EditText
        android:id="@+id/etEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your email"
        android:inputType="textEmailAddress" />

</LinearLayout>
```

Step 3: Update MainActivity.kt

Now, handle the new menu option and show the custom dialog.

```
package com.example.menudialogdemo

import android.os.Bundle
import android.view.ContextMenu
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private lateinit var textViewInfo: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        textViewInfo = findViewById(R.id.textViewInfo)
        registerForContextMenu(textViewInfo)
    }
```

```
// ---------- OPTIONS MENU ----------
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.menu_main, menu)
    return true
}

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return when (item.itemId) {
        R.id.menu_about -> {
            Toast.makeText(this, "Menu & Dialog Demo App", Toast.LENGTH_SHORT).show()
            true
        }
        R.id.menu_userinfo -> {
            showCustomDialog() // New
            true
        }
        R.id.menu_exit -> {
            showExitDialog()
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}

// ---------- CONTEXT MENU ----------
override fun onCreateContextMenu(
    menu: ContextMenu?,
    v: View?,
    menuInfo: ContextMenu.ContextMenuInfo?
) {
    super.onCreateContextMenu(menu, v, menuInfo)
    menu?.setHeaderTitle("Choose an Action")
    menu?.add(0, v?.id ?: 0, 0, "Copy Text")
    menu?.add(0, v?.id ?: 0, 1, "Share Text")
}

override fun onContextItemSelected(item: MenuItem): Boolean {
    return when (item.title) {
        "Copy Text" -> {
            Toast.makeText(this, "Text Copied!", Toast.LENGTH_SHORT).show()
            true
        }
        "Share Text" -> {
            Toast.makeText(this, "Sharing Text...", Toast.LENGTH_SHORT).show()
            true
        }
```

```
            else -> super.onContextItemSelected(item)
        }
    }

    // ---------- ALERT DIALOG ----------
    private fun showExitDialog() {
        val builder = AlertDialog.Builder(this)
        builder.setTitle("Exit App")
        builder.setMessage("Are you sure you want to exit?")
        builder.setPositiveButton("Yes") { _, _ -> finish() }
        builder.setNegativeButton("No", null)
        builder.show()
    }

    // ---------- CUSTOM DIALOG ----------
    private fun showCustomDialog() {
        val dialogView = layoutInflater.inflate(R.layout.dialog_userinfo, null)
        val etName = dialogView.findViewById<EditText>(R.id.etName)
        val etEmail = dialogView.findViewById<EditText>(R.id.etEmail)

        val builder = AlertDialog.Builder(this)
        builder.setTitle("Enter Your Info")
        builder.setView(dialogView)
        builder.setPositiveButton("Submit") { _, _ ->
            val name = etName.text.toString()
            val email = etEmail.text.toString()
            Toast.makeText(this, "Name: $name\nEmail: $email", Toast.LENGTH_LONG).show()
        }
        builder.setNegativeButton("Cancel", null)
        builder.show()
    }
}
```

Step 4: Run the App

- Tap the **three-dot menu**
- Choose **Enter Info** → a pop-up dialog appears
- Type name and email → click *Submit*
  The info appears in a Toast.

Student Activity

- Validate input fields before showing Toast.
- Store user input in **SharedPreferences** and display on app start.
- Use **Material Design AlertDialog** for a modern look.

Extend the same "Menu & Dialog Demo" app to include **SharedPreferences**

The user's name and email entered in the dialog are *saved* and *automatically displayed* next time the app opens.

SharedPreferences with Custom Dialogs

- Store user input locally using **SharedPreferences**.
- Retrieve and display saved data when the app restarts.
- Combine **Dialogs + Menus + Persistent Storage** in one app.

Step 1: Full Updated Code (MainActivity.kt)

```
package com.example.menudialogdemo

import android.content.Context
import android.os.Bundle
import android.view.ContextMenu
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private lateinit var textViewInfo: TextView
    private val PREFS_NAME = "UserPrefs"
    private val KEY_NAME = "name"
    private val KEY_EMAIL = "email"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        textViewInfo = findViewById(R.id.textViewInfo)
        registerForContextMenu(textViewInfo)

        // ☑ Load saved data when app opens
        loadUserData()
    }

    // ---------- OPTIONS MENU ----------
```

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

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return when (item.itemId) {
        R.id.menu_about -> {
            Toast.makeText(this, "Menu & Dialog Demo App", Toast.LENGTH_SHORT).show()
            true
        }
        R.id.menu_userinfo -> {
            showCustomDialog()
            true
        }
        R.id.menu_exit -> {
            showExitDialog()
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}

// ---------- CONTEXT MENU ----------
override fun onCreateContextMenu(
    menu: ContextMenu?,
    v: View?,
    menuInfo: ContextMenu.ContextMenuInfo?
) {
    super.onCreateContextMenu(menu, v, menuInfo)
    menu?.setHeaderTitle("Choose an Action")
    menu?.add(0, v?.id ?: 0, 0, "Copy Text")
    menu?.add(0, v?.id ?: 0, 1, "Share Text")
}

override fun onContextItemSelected(item: MenuItem): Boolean {
    return when (item.title) {
        "Copy Text" -> {
            Toast.makeText(this, "Text Copied!", Toast.LENGTH_SHORT).show()
            true
        }
        "Share Text" -> {
            Toast.makeText(this, "Sharing Text...", Toast.LENGTH_SHORT).show()
            true
        }
        else -> super.onContextItemSelected(item)
    }
}
```

```
// ---------- ALERT DIALOG ----------
private fun showExitDialog() {
    val builder = AlertDialog.Builder(this)
    builder.setTitle("Exit App")
    builder.setMessage("Are you sure you want to exit?")
    builder.setPositiveButton("Yes") { _, _ -> finish() }
    builder.setNegativeButton("No", null)
    builder.show()
}

// ---------- CUSTOM DIALOG ----------
private fun showCustomDialog() {
    val dialogView = layoutInflater.inflate(R.layout.dialog_userinfo, null)
    val etName = dialogView.findViewById<EditText>(R.id.etName)
    val etEmail = dialogView.findViewById<EditText>(R.id.etEmail)

    val builder = AlertDialog.Builder(this)
    builder.setTitle("Enter Your Info")
    builder.setView(dialogView)
    builder.setPositiveButton("Save") { _, _ ->
        val name = etName.text.toString()
        val email = etEmail.text.toString()
        if (name.isNotEmpty() && email.isNotEmpty()) {
            saveUserData(name, email)
            textViewInfo.text = "Welcome, $name!\nEmail: $email"
            Toast.makeText(this, "Data Saved!", Toast.LENGTH_SHORT).show()
        } else {
            Toast.makeText(this, "Please enter both fields!", Toast.LENGTH_SHORT).show()
        }
    }
    builder.setNegativeButton("Cancel", null)
    builder.show()
}

// ---------- SHARED PREFERENCES ----------
private fun saveUserData(name: String, email: String) {
    val sharedPref = getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE)
    val editor = sharedPref.edit()
    editor.putString(KEY_NAME, name)
    editor.putString(KEY_EMAIL, email)
    editor.apply()
}

private fun loadUserData() {
    val sharedPref = getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE)
    val name = sharedPref.getString(KEY_NAME, null)
    val email = sharedPref.getString(KEY_EMAIL, null)
```

```
        if (name != null && email != null) {
            textViewInfo.text = "Welcome back, $name!\nEmail: $email"
        } else {
            textViewInfo.text = "No user info saved yet."
        }
    }
}
```

Step 2: activity_main.xml

Make sure your layout has a **TextView** to display info.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:orientation="vertical"
    android:gravity="center"
    android:padding="16dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textViewInfo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="No user info saved yet."
        android:textSize="18sp"
        android:padding="10dp" />
</LinearLayout>
```

Step 3: dialog_userinfo.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="20dp">

    <EditText
        android:id="@+id/etName"
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:hint="Enter your name"
    android:inputType="textPersonName"
    android:layout_marginBottom="10dp" />

  <EditText
    android:id="@+id/etEmail"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter your email"
    android:inputType="textEmailAddress" />
</LinearLayout>
```

Step 4: menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item
    android:id="@+id/menu_about"
    android:title="About" />
  <item
    android:id="@+id/menu_userinfo"
    android:title="Enter Info" />
  <item
    android:id="@+id/menu_exit"
    android:title="Exit" />
</menu>
```

Working of app:

1. The user selects **Enter Info** from the menu.
2. A dialog appears to enter name and email.
3. When clicking **Save**, info is saved in **SharedPreferences**.
4. On app restart, the info is automatically loaded and shown.

Output Behavior

- First launch → "No user info saved yet."
- Enter info → dialog appears → save → Toast "Data Saved!"
- Relaunch app → "Welcome back, <PRIVATE_PERSON>!"

Student Activity

- **Add a feature next** so that the user can **edit or clear the saved data** from the same menu.
- Complete a perfect small CRUD (Create, Read, Update, Delete) using SharedPreferences.