LabW02 – Handling Interactions

Objectives:

1. Use clicks event listener for ListView
2. Use UI element dialog
3. Use Activity navigation
4. Experience Cross-Platform app development [Optional]

Tasks:

1. Build a simple mobile app: ToDoList
2. Handle long click event for an item in ListView
3. Pop up a dialog
4. Handle item clicking in ListView with Intent and another Activity
5. Add time information to your ToDoList app
6. Develop Cross-Platform app with Flutter [Optional]

Good mobile apps provide useful functionality and an easy-to-use interface. The user interface is made up of various Graphical User Interface (GUI) components and typically waits for user interaction.

In this tutorial, we will learn about various GUI components, their associated events, and how to handle events on those components.

Task 1: Build a simple ToDoList App

1. Start a new Android Studio project.

2. In the '**New Project'** window:
   - Under the "**Phone and Tablet**" tab, create a new empty activity by selecting "No Activity".
   - Click Next

3. On the '**No Activity**' screen, enter the following:
   - **Name:** ToDoList
   - **Package name:** comp5216.sydney.edu.au.helloworld
   - **Save location:** (leave as is or change to your preferred location)
   - **Language:** Java
   - **Minimum SDK:** API 24: Android 7.0 (Nougat)
   - **Build configuration language:** Groovy DSL (build.gradle)
   - Click Finish

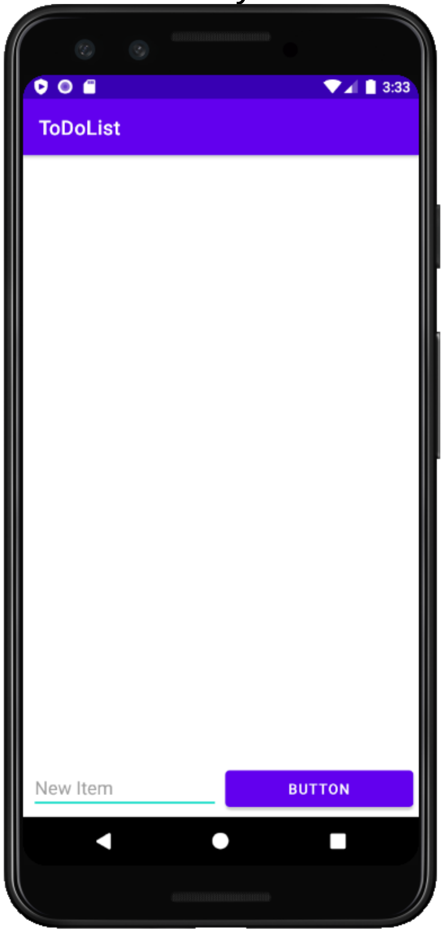
4. In 'activity_main.xml':
   - Drag these widgets onto the main_layout (please refer to the screenshot below)
     - Text → Plain Text
     - Buttons → Button
     - Legacy → ListView
     - Open the XML layout file and edit the following:
       - Assign hint to EditText: New Item
       - Assign ID to views:
         - ListView -> lstView
         - EditText -> txtNewItem
         - Button -> btnAddItem
   - Specify the necessary constraint for each of the widget to position a given widget relative to another one e.g. app:layout_constraintBottom_toBottomOf.

Refer to
https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout to learn more about ConstraintLayout which allows you to position and size widgets in a flexible way.

5. Add the activity intent in the manifest file, you could refer or use the code below.

```
    <activity
        android:name=".MainActivity"
        android:exported="true"
        android:theme="@style/Theme.AppCompat">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```

6. Run it. You should only see an empty list, text input and button. Refer next for a sample screenshot and activity_main.xml. Clicking the button does not have any effect.



```
<Button
    android:id="@+id/btnAddItem"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="4dp"
    android:layout_marginEnd="5dp"
    android:text="Button"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/txtNewItem"
/>

<ListView
    android:id="@+id/lstView"
    android:layout_width="390dp"
    android:layout_height="560dp"
    android:layout_marginBottom="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<EditText
    android:id="@+id/txtNewItem"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="6dp"
    android:layout_marginEnd="6dp"
    android:layout_marginStart="8dp"
    android:ems="10"
    android:hint="New Item"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/btnAddItem"
    app:layout_constraintStart_toStartOf="parent" />
```

7. Add the code in MainActivity.java (**keep the original imports and package**). Read the comments inside the code.

```
package comp5216.sydney.edu.au.todolist;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    // Define variables
    ListView listView;
    ArrayList<String> items;
    ArrayAdapter<String> itemsAdapter;
    EditText addItemEditText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Use "activity_main.xml" as the layout
        setContentView(R.layout.activity_main);

        // Reference the "listView" variable to the id "lstView" in the layout
        listView = (ListView) findViewById(R.id.lstView);
        addItemEditText = (EditText) findViewById(R.id.txtNewItem);

        // Create an ArrayList of String
        items = new ArrayList<String>();
        items.add("item one");
        items.add("item two");

        // Create an adapter for the list view using Android's built-in item layout
        itemsAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
items);

        // Connect the listView and the adapter
        listView.setAdapter(itemsAdapter);
    }
}
```

8. Run the project.

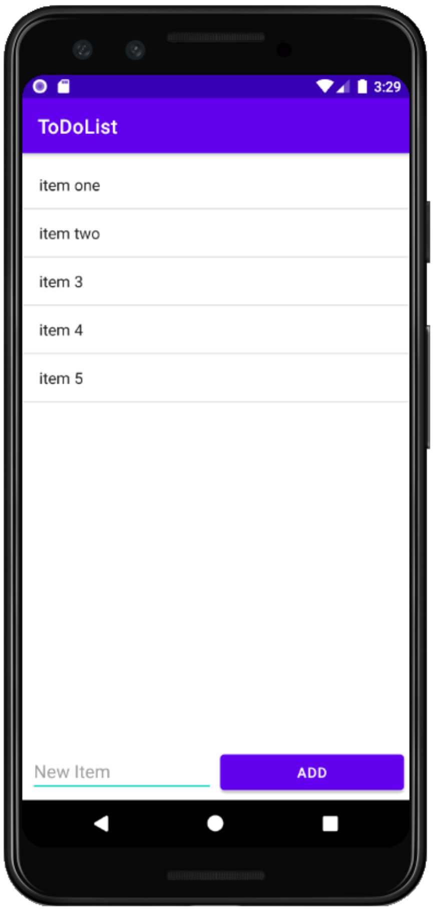
9. Add button click handler as follows:

```
public void onAddItemClick(View view) {
    String toAddString = addItemEditText.getText().toString();
    if (toAddString != null && toAddString.length() > 0) {
        itemsAdapter.add(toAddString); // Add text to list view adapter
        addItemEditText.setText("");
    }
}
```

10. Link the method to the button. Edit the activity_main.xml layout file:
    1) Update the button text to 'Add'
    2) Add 'android:onClick' attribute to the method name 'onAddItemClick'

```
<Button
        android:id="@+id/btnAddItem"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="5dp"
        android:layout_marginBottom="4dp"
        android:text="Add"
        android:onClick="onAddItemClick"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/txtNewItem" />
```

11. Enjoy running your **ToDoList** Android app! If you get it right, you should see something like below. Add as many new items as you like to test it.



Task 2: Handling item long clicking in ListView

Currently, nothing happens when an item is clicked in the ListView. We will first implement. "Deleting an item" by long clicking it.

1. Add the following method to the MainActivity.

   It sets two listeners for LongClick and Click events.

```
private void setupListViewListener() {
    listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
        public boolean onItemLongClick(AdapterView<?> parent, View view, final int
position, long rowId)
        {
            Log.i("MainActivity", "Long Clicked item " + position);
            return true;
        }
    });

    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
            String updateItem = (String) itemsAdapter.getItem(position);
            Log.i("MainActivity", "Clicked item " + position + ": " + updateItem);
        }
    });
}
```

2. Call this method at the end of the onCreate() method

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    XXXXXX
    XXXXXX

    // Setup listView listeners
    setupListViewListener();
}
```

3. Run it. Click and long click a ToDoItem in the list view.

4. Now update the listener for onItemLongClickListener.
   Add two lines of code to remove an item from the ArrayList and notify the ListView adapter to update the list. Run it.

```
        Log.i("MainActivity", "Long Clicked item " + position);
        items.remove(position); // Remove item from the ArrayList
        itemsAdapter.notifyDataSetChanged(); // Notify listView adapter to update list
        return true;
```

Run your code, and long click a ToDoItem, you should able to see the following line printed on the Android Studio Logcat:

```
comp5216.sydney.edu.au.todolist I/MainActivity: Long Clicked item 1
```

Task 3: Popping up a dialog

1. Add a dialog to let user confirm the delete operation. Replace the existing **OnItemLongClick** method.

```
listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    public boolean onItemLongClick(AdapterView<?> parent, View view, final int
position, long rowId)
    {
        Log.i("MainActivity", "Long Clicked item " + position);
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setTitle(R.string.dialog_delete_title)
                .setMessage(R.string.dialog_delete_msg)
                .setPositiveButton(R.string.delete, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialogInterface, int i) {
                        items.remove(position); // Remove item from the ArrayList
                        itemsAdapter.notifyDataSetChanged(); // Notify listView adapter
to update the list
                    }
                })
                .setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialogInterface, int i) {
                        // User cancelled the dialog
                        // Nothing happens
                    }
                });

        builder.create().show();
        return true;
    }
});
```

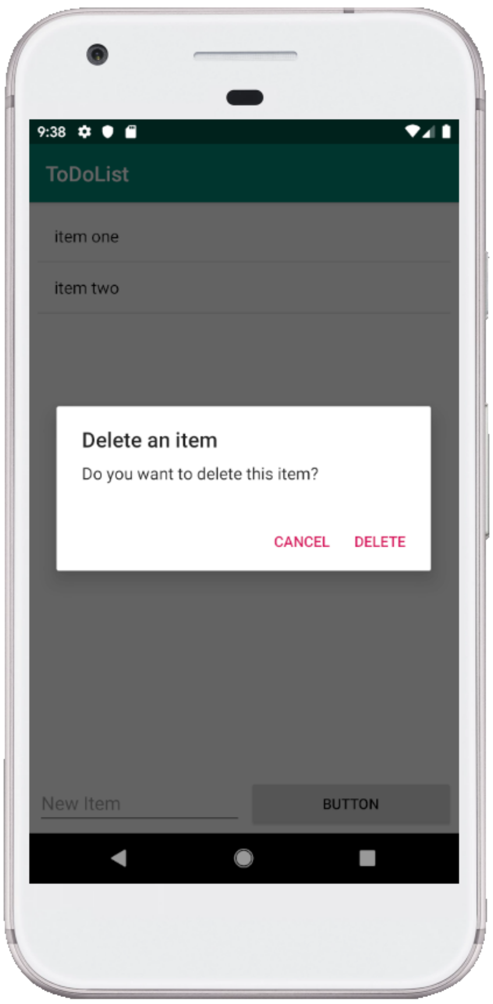
You will also need to add the following String resources into "**res/values/strings.xml**"

```
<string name="dialog_delete_title">Delete an item</string>
<string name="dialog_delete_msg">Do you want to delete this item?</string>
<string name="delete">Delete</string>
<string name="cancel">Cancel</string>
<string name="edit">Edit</string>
```

Run it and long click an item.

2. In order to further customise a dialog, such as adding a third button, or using your own layout, please refer to the following tutorial: https://developer.android.com/guide/topics/ui/dialogs

   If you do the above steps right, you should able to see the following screenshot once you long click a ToDoItem:



Task 4: Handling item clicking in ListView with Intent and another Activity

When clicking an item (NOT a long click), we should open another Activity to edit the item.

1. Copy the EditToDoItemActivity.java to the same “**src**” folder as the MainActivity.
   This Activity receives the data from the MainActivity when an item is clicked, and send back the updated content to the MainActivity. Read the inline comments for details.

2. Copy **activity_edit_item.xml** to the “**res/layout**” folder. This is the layout for EditToDoItemActivity

3. Update the onItemClick method in MainActivity to open the

EditToDoItemActivity

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    String updateItem = (String) itemsAdapter.getItem(position);
    Log.i("MainActivity", "Clicked item " + position + ": " + updateItem);

    Intent intent = new Intent(MainActivity.this, EditToDoItemActivity.class);
    if (intent != null) {
        // put "extras" into the bundle for access in the edit activity
        intent.putExtra("item", updateItem);
        intent.putExtra("position", position);

        // brings up the second activity
        mLauncher.launch(intent);
        itemsAdapter.notifyDataSetChanged();
    }
}
```

When an item is clicked, it creates an Intent to start another activity and wait for its result. You need to register a request to start an activity for result and register the result callback in the MainActivity.

4. When the EditToDoItemActivity finishes, MainActivity will receive the result by checking the result code first, and then uses its result's data.
   Add the following method into MainActivity:

```
// Register a request to start an activity for result and register the result callback
ActivityResultLauncher<Intent> mLauncher = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        result -> {
            if (result.getResultCode() == RESULT_OK) {
                // Extract name value from result extras
                String editedItem = result.getData().getExtras().getString("item");
                int position = result.getData().getIntExtra("position", -1);
                items.set(position, editedItem);
                Log.i("Updated item in list ", editedItem + ", position: " + position);

                // Make a standard toast that just contains text
                Toast.makeText(getApplicationContext(), "Updated: " + editedItem,
Toast.LENGTH_SHORT).show();
                itemsAdapter.notifyDataSetChanged();
            }
        }
);
```

To learn more about using intents to create flows, please read:
https://github.com/codepath/android_guides/wiki/Using-Intents-to-Create-Flows

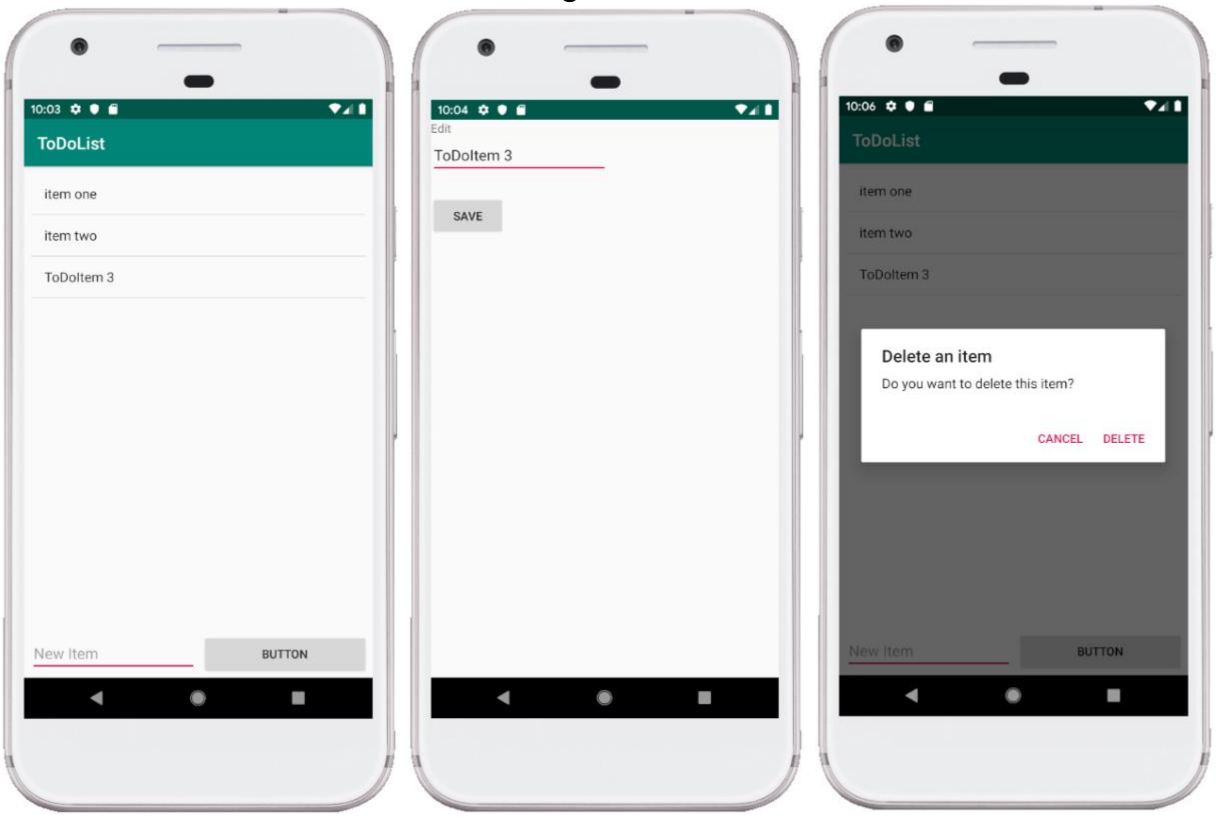
You may also notice the use of Toast. It is a simple pop-up notification method in Android. Learn more at: https://developer.android.com/guide/topics/ui/notifiers/toasts

5. Lastly, the application manifest must include the new EditToDoItemActivity, otherwise this Activity cannot be used.

Open **AndroidManifest.xml** and add the following code inside the <Application> tag.

```
<activity
    android:name=".EditToDoItemActivity"
    android:label="@string/app_name" >
</activity>
```

6. Run it. Test the above steps. If all the steps are correct, your app now are able to handle the click and long click events:



Delete an item

Do you want to delete this item?

Task 5: Add time information to your ToDoList app
In this task, you are required to add time information and display it every time you add a ToDoItem to your ListView. In order to achieve this, you may need to use the "Date" class. For detailed information of the Date class, please refer to https://developer.android.com/reference/java/util/Date

Task 6: Using Flutter [Optional]

Flutter is a tool introduced by Google in 2018 for cross-platform Android and iOS mobile apps development. It is considered as one of the most efficient tools to build platform independent apps quickly, with a single codebase for both Android and iOS. This means that you can use one programming language and one codebase to create two different apps (for Android and iOS).

Flutter is an open-source mobile application development framework, available for download from: https://flutter.dev/docs/get-started/install

To get started, select the operating system on which you are installing Flutter:

1. Flutter for Windows install: https://flutter.dev/docs/get-started/install/windows
2. Flutter for MacOS install: https://flutter.dev/docs/get-started/install/macos
3. Flutter for Linux install: https://flutter.dev/docs/get-started/install/linux

Please refer to the following URL for more details on Flutter: https://flutter.dev/