Experiment 7: To-do List Application (Part 2)

1. Objective

Enhance the to-do list application by allowing users to delete and edit items. This will introduce more advanced concepts of ListView manipulation and event handling.

2. Steps to Complete the Experiment

1. Enhance the UI Layout:

   If not already implemented in Experiment 6, consider adding a more sophisticated layout for each item in the to-do list, possibly through a custom XML layout file for the ListView items or RecyclerView ViewHolder. This could include a more prominent display of each task and a delete button for removing tasks.

```
<?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="horizontal"
    android:padding="8dp">

    <TextView
        android:id="@+id/textViewTaskDescription"
```

```
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Sample Task"
        android:textSize="18sp"
        android:gravity="center_vertical"/>

    <Button
        android:id="@+id/buttonDeleteTask"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Delete"
        android:layout_gravity="center_vertical"/>

</LinearLayout>
```

2. Implement Task Deletion:

   Implement functionality to remove a task from the list. This can be done in various ways, such as by swiping the item away or by pressing a delete button next to each task. For a ListView, you might need a custom adapter to include a delete button in each list item. For a RecyclerView, this functionality can be more seamlessly integrated with the help of ItemTouchHelper.

   Ensure that the deletion updates the list displayed to the user in real-time.

TaskAdapter.java

```
package com.yourpackage.name; // Ensure this matches your actual
package name
```

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.TextView;
import androidx.annotation.NonNull;
import java.util.List;

public class TaskAdapter extends ArrayAdapter<String> {
    private int resourceLayout;
    private Context mContext;

    public TaskAdapter(@NonNull Context context, int resource,
List<String> items) {
        super(context, resource, items);
        this.resourceLayout = resource;
        this.mContext = context;
    }

    @Override
    public View getView(int position, View convertView,
ViewGroup parent) {
        View v = convertView;
```

```
        if (v == null) {
            LayoutInflater vi;
            vi = LayoutInflater.from(mContext);
            v = vi.inflate(resourceLayout, null);
        }

        String p = getItem(position);

        if (p != null) {
            TextView                    tt                    =
v.findViewById(R.id.textViewTaskDescription);
            Button bt = v.findViewById(R.id.buttonDeleteTask);
            if (tt != null) {
                tt.setText(p);
            }
            if (bt != null) {
                bt.setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        remove(p); // Remove the item from the
list
                        notifyDataSetChanged(); // Notify the
adapter to refresh the list
                    }
                });
```

```
            }
        }

        return v;
    }
}
```

MainActivity.java

```
package com.yourpackage.name; // Change this to your actual
package name

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

public class MainActivity extends AppCompatActivity {

    private EditText editTextTask;
    private Button buttonAdd;
    private ListView listViewTasks;
    private ArrayList<String> tasks;
    private TaskAdapter adapter;

    @Override
```

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

        editTextTask = findViewById(R.id.editTextTask);
        buttonAdd = findViewById(R.id.buttonAdd);
        listViewTasks = findViewById(R.id.listViewTasks);

        tasks = new ArrayList<>();
        adapter = new TaskAdapter(this, R.layout.list_item,
tasks);
        listViewTasks.setAdapter(adapter);

        buttonAdd.setOnClickListener(v -> {
            String task = editTextTask.getText().toString();
            if (!task.isEmpty()) {
                tasks.add(task);
                adapter.notifyDataSetChanged(); // Refresh the
ListView
                editTextTask.setText(""); // Clear the EditText
            }
        });
    }
}
```

3. Persist Tasks (Optional):

If you wish to retain the tasks even after the application is closed, consider implementing a form of persistence, such as using SharedPreferences, a database (SQLite), or a file system. This step might require more advanced handling, including converting the list of tasks into a format suitable for storage and retrieving/updating this data on app startup/shutdown.

4. Enhance Task Interaction:

   Besides deletion, you could add functionalities like editing a task (by tapping on it to bring up an edit dialog) or marking a task as completed (changing its appearance, like striking through the text).

5. UI and UX Improvements:

   Refine the user interface with Material Design components for a more polished look and feel. Add animations for adding/removing tasks for a smoother user experience.

   Implement empty state views, for example, a text or image that appears when the list is empty, guiding users to add new tasks.

6. Testing:

   Thoroughly test all new functionalities to ensure they work as expected. Pay special attention to the deletion process and any persistence mechanisms implemented to ensure data integrity.

3. Explanation

LinearLayout: This is the root layout for each list item. It's set to horizontal orientation, so the task description and delete button are side by side.

TextView (textViewTaskDescription): Displays the task description. It takes up most of the layout space (layout_weight="1"), ensuring the delete button does not push it out of view.

Button (buttonDeleteTask): When clicked, this button will trigger the deletion of the associated task from the list.

TaskAdapter: This custom adapter handles the inflation of the list_item.xml layout for each item in the to-do list. It also manages the deletion of tasks when the delete button is clicked.

MainActivity: Initializes the TaskAdapter with the tasks ArrayList and sets it on the ListView. The add button's functionality remains the same, adding tasks to the list and refreshing the view.