Experiment 6: To-do List Application (Part 1)

1. Objective

Start building a basic to-do list application that allows users to add items to a list. This experiment will cover basic ListView usage and custom adapters.

2. Steps to Complete the Experiment

1. Set Up the Project:

   Create a new Android project in Android Studio with an Empty Activity. Name the project as you see fit, for instance, "ToDoListApp".

2. Design the UI:

   The main activity layout will need an EditText for inputting new to-do items, a Button to add the entered item to the list, and a ListView or RecyclerView to display the list of to-do items.

```
<?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"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/editTextTask"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter a new task"
        android:inputType="textPersonName"
        android:layout_alignParentTop="true"
        android:layout_toStartOf="@+id/buttonAdd"/>

    <Button
        android:id="@+id/buttonAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"/>

    <ListView
        android:id="@+id/listViewTasks"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/editTextTask"
        android:divider="@android:color/darker_gray"
        android:dividerHeight="1dp"/>

</RelativeLayout>
```

3. Initialize UI Components:

   In MainActivity.java, initialize the UI components (the EditText, Button, and ListView/RecyclerView) using findViewById.

4. Set Up the Data Structure:

   Create an ArrayList<String> to hold the to-do items. This will serve as the data source for your ListView/RecyclerView.

5. Create an Adapter:

   For a ListView, use an ArrayAdapter<String> to link your ArrayList to the ListView.

   For a RecyclerView, you'll need to create a custom Adapter class and a ViewHolder class.

6. Handle Item Addition:

   Set an OnClickListener for the add button. When the button is clicked, retrieve the text from the EditText, add it to the ArrayList, and notify the adapter of the change to update the UI.

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


import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
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 ArrayAdapter<String> 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        ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, tasks);
        listViewTasks.setAdapter(adapter);

        buttonAdd.setOnClickListener(new
View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                String task = editTextTask.getText().toString();
                if (!task.isEmpty()) {
                    tasks.add(task);
                    adapter.notifyDataSetChanged(); // Refresh the ListView
                    editTextTask.setText(""); // Clear the EditText
                }
            }
        });
    }
}
```

7. Optional UI Enhancements:

   Add functionality to remove items from the list on long press or by adding a delete button in each list item. This might require a custom adapter if you're using a ListView.

   Improve the app's appearance with Material Design components and custom styles.

8. Testing:

   Run your application and test adding new items to the list. Ensure that the UI updates correctly when new items are added.

3. Explanation:

Initialization:

The EditText for input (editTextTask), the Button for adding tasks (buttonAdd), and the ListView for displaying tasks (listViewTasks) are initialized using their respective IDs.

ArrayList and ArrayAdapter:

An ArrayList<String> named tasks is used to store the list of tasks.

An ArrayAdapter<String> named adapter connects the tasks ArrayList with the ListView. The adapter uses a simple built-in list item layout (simple_list_item_1).

Adding Tasks:

An OnClickListener is set on the buttonAdd to handle the click events. When the button is clicked, it retrieves the text from editTextTask, adds it to the tasks ArrayList if it's not empty, notifies the adapter to refresh the ListView, and clears the EditText.