# Problem 3

**Exercise Objective:** Create an android app to develop a To-Do-List application.

**Problem Statement 3:** Create an Android application that creates the list of To-Do-tasks. each task would be entered and added in the list on the clicking of event.

**Expected Output:** The app displays the list of To Do tasks.

## XML Layout Files

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:padding="16dp"  
 tools:context=".MainActivity">  
  
 <!-- EditText for entering new tasks -->  
<EditText  
 android:id="@+id/editTextTask"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter a new task"  
 android:inputType="textCapSentences"  
 android:layout\_alignParentTop="true"  
 android:layout\_toStartOf="@+id/buttonAdd"  
 android:minHeight="48dp"  
 android:padding="8dp" />  
  
 <!-- Button to add a new task -->  
<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"  
 android:minHeight="48dp" />  
  
 <!-- ListView to display the tasks -->  
<ListView  
 android:id="@+id/listViewTasks"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_below="@id/editTextTask"  
 android:layout\_marginTop="8dp"  
 android:divider="@android:color/darker\_gray"  
 android:dividerHeight="1dp"/>  
  
</RelativeLayout>

list\_item.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="horizontal"  
 android:padding="8dp"  
 android:gravity="center\_vertical">  
  
 <!-- TextView to display the task description -->  
 <TextView  
 android:id="@+id/textViewTaskDescription"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Sample Task Description"  
 android:textSize="18sp"  
 android:textColor="@android:color/black"  
 android:minHeight="48dp"  
 android:gravity="center\_vertical"/>  
  
 <!-- Button to delete a task -->  
 <Button  
 android:id="@+id/buttonDeleteTask"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Delete"  
 android:layout\_marginStart="8dp"  
 android:minHeight="48dp" />  
  
</LinearLayout>

## Java Files

MainActivity.java

package com.example.todolistapp;  
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ListView;  
import android.widget.Toast; // Import Toast for displaying messages  
import java.util.ArrayList;  
  
public class MainActivity extends AppCompatActivity {  
  
 // Declare UI elements  
 private EditText editTextTask;  
 private Button buttonAdd;  
 private ListView listViewTasks;  
 // ArrayList to hold the tasks  
 private ArrayList<String> tasks;  
 // Custom adapter for the ListView  
 private TaskAdapter adapter;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 // Set the content view to the main layout for this activity  
 setContentView(R.layout.*activity\_main*);  
  
 // Initialize UI elements by finding their IDs from the layout  
 editTextTask = findViewById(R.id.*editTextTask*);  
 buttonAdd = findViewById(R.id.*buttonAdd*);  
 listViewTasks = findViewById(R.id.*listViewTasks*);  
  
 // Initialize the ArrayList for tasks  
 tasks = new ArrayList<>();  
  
 // Initialize the custom TaskAdapter, linking it to the list\_item layout  
 // R.layout.list\_item refers to the layout defined in list\_item.xml for each row  
 adapter = new TaskAdapter(this, R.layout.*list\_item*, tasks);  
  
 // Set the adapter to the ListView  
 listViewTasks.setAdapter(adapter);  
  
 // Set an OnClickListener for the Add button  
 buttonAdd.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 addTask(); // Call the method to add a new task  
 }  
 });  
 }  
  
 */\*\*  
 \* Adds a new task to the list if the input is not empty.  
 \*/* private void addTask() {  
 // Get the text from the EditText field  
 String task = editTextTask.getText().toString().trim(); // Use trim() to remove leading/trailing spaces  
  
 // Check if the task input is not empty  
 if (!task.isEmpty()) {  
 tasks.add(task); // Add the new task to the ArrayList  
 adapter.notifyDataSetChanged(); // Notify the adapter that the data set has changed to refresh the ListView  
 editTextTask.setText(""); // Clear the EditText field after adding the task  
 Toast.*makeText*(this, "Task added!", Toast.*LENGTH\_SHORT*).show(); // Provide user feedback  
 } else {  
 // Show a Toast message if the input is empty  
 Toast.*makeText*(this, "Please enter a task.", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
}

TaskAdaptor.java

package com.example.todolistapp;  
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 android.widget.Toast;  
import androidx.annotation.NonNull;  
import java.util.List;  
  
*/\*\*  
 \* Custom ArrayAdapter for displaying tasks in a ListView.  
 \* This adapter uses a custom layout (list\_item.xml) for each row  
 \* and handles the delete button click for each task.  
 \*/*public class TaskAdapter extends ArrayAdapter<String> {  
  
 // Layout resource ID for each list item  
 private int resourceLayout;  
 // Context from the calling activity  
 private Context mContext;  
 // List of tasks to be displayed  
 private List<String> tasksList;  
  
 */\*\*  
 \* Constructor for the TaskAdapter.  
 \** ***@param*** *context The current context (e.g., MainActivity.this).  
 \** ***@param*** *resource The resource ID for a layout file containing a TextView to use when instantiating views.  
 \** ***@param*** *items The list of task strings to display.  
 \*/* public TaskAdapter(@NonNull Context context, int resource, @NonNull List<String> items) {  
 super(context, resource, items);  
 this.resourceLayout = resource;  
 this.mContext = context;  
 this.tasksList = items; // Store the reference to the original list  
 }  
  
 */\*\*  
 \* Provides a View for an AdapterView (ListView, GridView, etc.)  
 \** ***@param*** *position The position of the item within the adapter's data set.  
 \** ***@param*** *convertView The old view to reuse, if possible.  
 \** ***@param*** *parent The parent that this view will eventually be attached to.  
 \** ***@return*** *A View corresponding to the data at the specified position.  
 \*/* @Override  
 public View getView(final int position, View convertView, @NonNull ViewGroup parent) {  
 View v = convertView;  
  
 // If the view is null, inflate it from the custom list\_item layout  
 if (v == null) {  
 LayoutInflater vi = LayoutInflater.*from*(mContext);  
 v = vi.inflate(resourceLayout, parent, false); // Use parent and attachToRoot=false  
 }  
  
 // Get the task string for the current position  
 final String task = getItem(position);  
  
 if (task != null) {  
 // Find the TextView and Button within the inflated view  
 TextView textViewTaskDescription = v.findViewById(R.id.*textViewTaskDescription*);  
 Button buttonDeleteTask = v.findViewById(R.id.*buttonDeleteTask*);  
  
 // Set the task description text  
 if (textViewTaskDescription != null) {  
 textViewTaskDescription.setText(task);  
 }  
  
 // Set OnClickListener for the delete button  
 if (buttonDeleteTask != null) {  
 buttonDeleteTask.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 // Remove the task from the list and notify the adapter  
 // It's safer to remove using the stored list reference  
 if (tasksList.contains(task)) { // Check if task still exists before removing  
 tasksList.remove(task);  
 notifyDataSetChanged(); // Refresh the ListView to reflect the change  
 Toast.*makeText*(mContext, "Task deleted: " + task, Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 }  
 }  
 return v;  
 }  
}