**Problem Statement:**

Develop a To-Do List Application for Android that helps users manage their daily tasks. This app should allow users to add, view, mark as complete, and remove tasks in an intuitive way.

**Requirements:**

**Task Addition:** Implement a button on the main screen that navigates to a new screen where users can enter and save a task. Use intents to handle this navigation.

**Task Display:** Display tasks in a list format on the main screen, with each task having a checkbox that users can select to mark the task as completed.

**Menu Options:**

Add a menu option to clear all completed tasks, removing them from the list.

Add a menu option to refresh the list to ensure all items are up-to-date.

Navigation and User Interface: Ensure a user-friendly interface and smooth navigation between screens using intents.

**Solution:**

Here's a **complete step-by-step solution** to build the **To-Do List Application** with detailed explanations for each step. By following these instructions, you'll have a fully functioning application.

**Step 1: Create a New Project**

1. Open Android Studio.

2. Select New Project > Empty Activity.

3. Name your project (e.g., `ToDoListApp`).

4. Choose the language as **Kotlin**.

5. Set the minimum **SDK** (e.g., API 21: Android 5.0).

6. Click **Finish** to create the project.

**Step 2: Design Layouts**

**2.1 Main Screen Layout (activity\_main.xml)**

1. Navigate to: `res/layout/activity\_main.xml`.

2. Replace the existing code with:

```xml

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<androidx.recyclerview.widget.RecyclerView

android:id="@+id/recyclerViewTasks"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_above="@+id/fabAddTask" />

<com.google.android.material.floatingactionbutton.FloatingActionButton

android:id="@+id/fabAddTask"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_alignParentEnd="true"

android:layout\_margin="16dp"

android:src="@drawable/ic\_add"

android:contentDescription="Add Task" />

</RelativeLayout>

**Explanation:**

- `**RecyclerView`** displays the list of tasks.

- `**FloatingActionButton`** is used for task addition.

**2.2 Add Task Screen Layout (activity\_add\_task.xml)**

1. **Navigate to:** `res/layout/activity\_add\_task.xml`.

2. Replace the code with:

```xml

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:padding="16dp">

<EditText

android:id="@+id/editTextTask"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Enter Task"

android:padding="8dp" />

<Button

android:id="@+id/buttonSaveTask"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/editTextTask"

android:layout\_marginTop="16dp"

android:layout\_alignParentEnd="true"

android:text="Save" />

</RelativeLayout>

```

**2.3 Task Item Layout (item\_task.xml)**

**1. Create a New Layout:**

- Navigate to: `res/layout/`.

- Right-click > New > Layout Resource File.

- Name it `item\_task.xml`.

**2. Add the following code:**

```xml

<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/textViewDescription"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Task Description"

android:textSize="16sp" />

<CheckBox

android:id="@+id/checkBoxComplete"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

</LinearLayout>

```

**Explanation:**

- `TextView` displays the task description.

- `CheckBox` lets users mark a task as complete.

---

**Step 3: Create Task Model**

**1. Navigate to:** `app/src/main/java/<your\_package\_name>/`.

**2. Create a New Kotlin File:**

- Right-click on your package > **New > Kotlin File/Class** > Name it `Task`.

**3. Add the following code:**

```kotlin

data class Task(

val description: String,

var isCompleted: Boolean

)

```

**Explanation:**

- `description`: Stores the task's text.

- `isCompleted`: Tracks whether the task is marked as completed.

---

**Step 4: Create RecyclerView Adapter**

1. **Navigate to:** `app/src/main/java/<your\_package\_name>/`.

2. **Create a New Kotlin File:**

- Right-click on your package > **New > Kotlin File/Class** > Name it `TaskAdapter`.

3. **Add the following code:**

```kotlin

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.CheckBox

import android.widget.TextView

import androidx.recyclerview.widget.RecyclerView

class TaskAdapter(private val tasks: MutableList<Task>) :

RecyclerView.Adapter<TaskAdapter.TaskViewHolder>() {

class TaskViewHolder(view: View) : RecyclerView.ViewHolder(view) {

val textViewDescription: TextView = view.findViewById(R.id.textViewDescription)

val checkBoxComplete: CheckBox = view.findViewById(R.id.checkBoxComplete)

}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TaskViewHolder {

val view = LayoutInflater.from(parent.context)

.inflate(R.layout.item\_task, parent, false)

return TaskViewHolder(view)

}

override fun onBindViewHolder(holder: TaskViewHolder, position: Int) {

val task = tasks[position]

holder.textViewDescription.text = task.description

holder.checkBoxComplete.isChecked = task.isCompleted

holder.checkBoxComplete.setOnCheckedChangeListener { \_, isChecked ->

task.isCompleted = isChecked

}

}

override fun getItemCount(): Int = tasks.size

}

```

**Explanation:**

- `tasks`: List of tasks to display.

- `onBindViewHolder`: Binds task data to the `RecyclerView`.

**Step 5: Implement MainActivity**

**1. Open `MainActivity.kt`.**

**2. Replace the code with:**

```kotlin

import android.content.Intent

import android.os.Bundle

import androidx.appcompat.app.AppCompatActivity

import androidx.recyclerview.widget.LinearLayoutManager

import kotlinx.android.synthetic.main.activity\_main.\*

class MainActivity : AppCompatActivity() {

private val tasks = mutableListOf<Task>()

private val taskAdapter = TaskAdapter(tasks)

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

recyclerViewTasks.adapter = taskAdapter

recyclerViewTasks.layoutManager = LinearLayoutManager(this)

fabAddTask.setOnClickListener {

val intent = Intent(this, AddTaskActivity::class.java)

startActivityForResult(intent, 100)

}

}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {

super.onActivityResult(requestCode, resultCode, data)

if (requestCode == 100 && resultCode == RESULT\_OK) {

val taskDescription = data?.getStringExtra("TASK\_DESCRIPTION") ?: return

tasks.add(Task(taskDescription, false))

taskAdapter.notifyDataSetChanged()

}

}

}

```

**Explanation:**

- `fabAddTask`: Opens the add task screen.

- `onActivityResult`: Receives the new task and updates the list.

---

**Step 6: Implement AddTaskActivity**

**1. Create AddTaskActivity:**

- Navigate to `app/src/main/java/<your\_package\_name>/`.

- Right-click on your package > **New > Kotlin Class/File** > Name it `AddTaskActivity`.

**2. Add the following code:**

```kotlin

import android.app.Activity

import android.content.Intent

import android.os.Bundle

import androidx.appcompat.app.AppCompatActivity

import kotlinx.android.synthetic.main.activity\_add\_task.\*

class AddTaskActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_add\_task)

buttonSaveTask.setOnClickListener {

val taskDescription = editTextTask.text.toString()

val intent = Intent().apply {

putExtra("TASK\_DESCRIPTION", taskDescription)

}

setResult(Activity.RESULT\_OK, intent)

finish()

}

}

}

```

**Explanation:**

- Captures task input and sends it back to `MainActivity`.

---

**Step 7: Test and Add Menu Options**

1. Add `Clear Completed` and `Refresh` options to `MainActivity`.

2. Test the app thoroughly.

---

**Step 8: Add Menu Options**

**8.1 Create a Menu Resource File**

**1. Navigate to: `res/menu/`.**

- If the `menu` directory doesn't exist:

- Right-click on `res` > New > Android Resource Directory\*\*.

- Name the directory **menu** and select **Menu** from the dropdown.

- Create a new menu file:

- Right-click on `menu` > **New > Menu Resource File**.

- Name it `main\_menu.xml`.

**2. Add the following code to `main\_menu.xml`:**

```xml

<menu xmlns:android="http://schemas.android.com/apk/res/android">

<item

android:id="@+id/action\_clear\_completed"

android:title="Clear Completed"

android:icon="@drawable/ic\_clear"

android:showAsAction="always" />

<item

android:id="@+id/action\_refresh"

android:title="Refresh"

android:icon="@drawable/ic\_refresh"

android:showAsAction="always" />

</menu>

```

**Explanation:**

- `action\_clear\_completed`: Menu item for clearing completed tasks.

- `action\_refresh`: Menu item for refreshing the task list.

---

**8.2 Inflate Menu in MainActivity**

1. Open `MainActivity.kt`.

2. Override the `onCreateOptionsMenu` and `onOptionsItemSelected` methods:

```kotlin

override fun onCreateOptionsMenu(menu: Menu?): Boolean {

menuInflater.inflate(R.menu.main\_menu, menu)

return true

}

override fun onOptionsItemSelected(item: MenuItem): Boolean {

when (item.itemId) {

R.id.action\_clear\_completed -> {

tasks.removeAll { it.isCompleted }

taskAdapter.notifyDataSetChanged()

return true

}

R.id.action\_refresh -> {

taskAdapter.notifyDataSetChanged()

return true

}

}

return super.onOptionsItemSelected(item)

}

```

**Explanation:**

- `onCreateOptionsMenu`: Loads the `main\_menu.xml` into the app's menu bar.

- `onOptionsItemSelected`:

- Clears completed tasks when `action\_clear\_completed` is selected.

- Refreshes the list when `action\_refresh` is selected.

---

**Step 9: Test the Application**

**9.1 Functionality Checklist**

Ensure the following functionalities work as expected:

**1. Task Addition:**

- Click the **FloatingActionButton** to open the Add Task screen.

- Enter a task and save it.

- Verify the task appears in the list.

**2. Task Display:**

- Verify tasks appear in the `RecyclerView` with a checkbox for completion.

**3. Mark Task as Completed:**

- Check a task and confirm it is marked as completed.

**4. Clear Completed Tasks:**

- Complete a few tasks.

- Select the \*\*Clear Completed\*\* menu option.

- Confirm completed tasks are removed from the list.

**5. Refresh List:**

- Add tasks and verify the list is refreshed when \*\*Refresh\*\* is selected.

---

**9.2 Debugging Tips**

**1. App Crashes:**

- Check for typos in IDs (e.g., `recyclerViewTasks` or `editTextTask`).

- Confirm all layouts and menu files are correctly linked.

**2. Menu Issues:**

- Ensure the `menu` folder exists under `res/`.

- Verify the `onOptionsItemSelected` method handles menu actions properly.

**3. RecyclerView Not Displaying:**

- Verify `taskAdapter` is initialized and assigned correctly.

---

**Step 10: Enhance the App**

**Optional Improvements**

**1. Task Persistence:**

- Use **SharedPreferences**, **SQLite**, or **Room** to save tasks, so they remain available after the app restarts.

**2. Swipe to Delete:**

- Implement swipe gestures using `ItemTouchHelper` to delete tasks.

**3. Dark Mode Support:**

- Add themes for better visual appeal.

**4. Task Due Dates:**

- Enhance tasks with due dates and sorting by deadlines.