Simple To- Do List application

Description:

The goal of this project is to develop a fully functional to-do list application that provides users with a simple and effective way to manage their tasks. The application is designed by using language “kotlin” in android studio platform.

How to setup the application:

Step 1: Download the android studio

Step 2: Install Android Studio

Step 3: Configure Android SDK

Step 4: Setup virtual device(Emulator)

Step 5: Create a New project

Step 6: Build and run your App

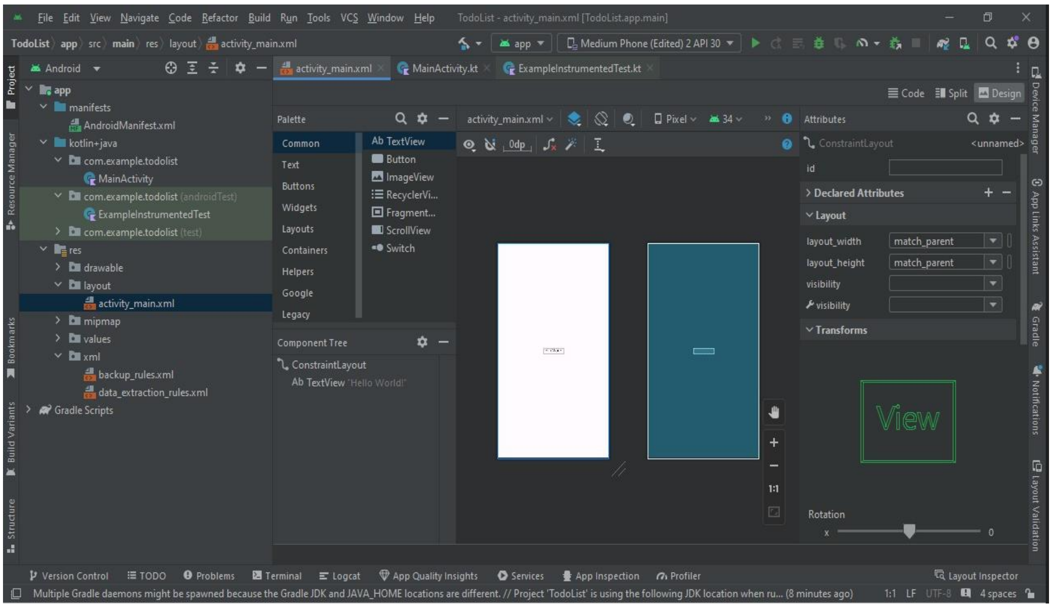
How to Build and Run the simple To-Do List application:

Step 1: Create a new android studio project

- Open android studio
- Select “Start a new android studio project”

Step 2: Design the user interface (UI)

- Open the res/layout/activity_main.xml file.
- Design the layout using XML, including components like EditText for task input, Button for adding tasks, and a RecyclerView to display tasks.
- Customize the layout according to your preferences.



TodoList - activity_main.xml [TodoList.app.main]

TodoList 〉 app 〉 src 〉 main 〉 res 〉 layout 〉 activity_main.xml

app | Medium Phone (Edited) 2 API 30

Component Tree

ConstraintLayout

Ab TextView "Hello World!"

Attributes

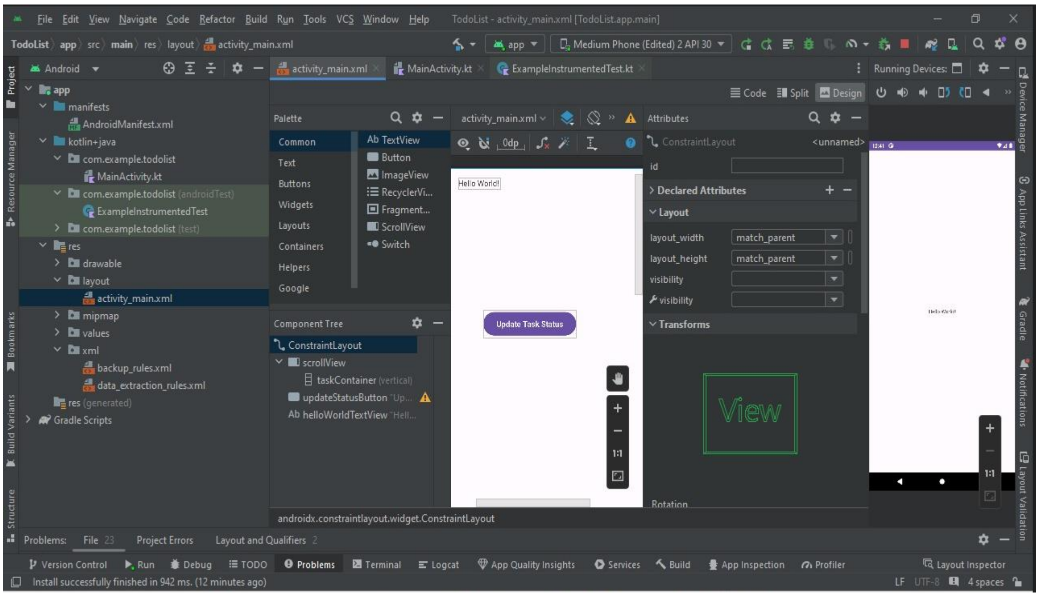
ConstraintLayout <unnamed>

id	
layout_width	match_parent
layout_height	match_parent
visibility	
visibility	

Rotation x 0

Multiple Gradle daemons might be spawned because the Gradle JDK and JAVA_HOME locations are different. // Project 'TodoList' is using the following JDK location when ru... (8 minutes ago) 1:1 LF UTF-8 4 spaces

Step 3: Define the Data Model



layout_width match_parent
layout_height match_parent
visibility
visibility

Install successfully finished in 942 ms. (12 minutes ago)

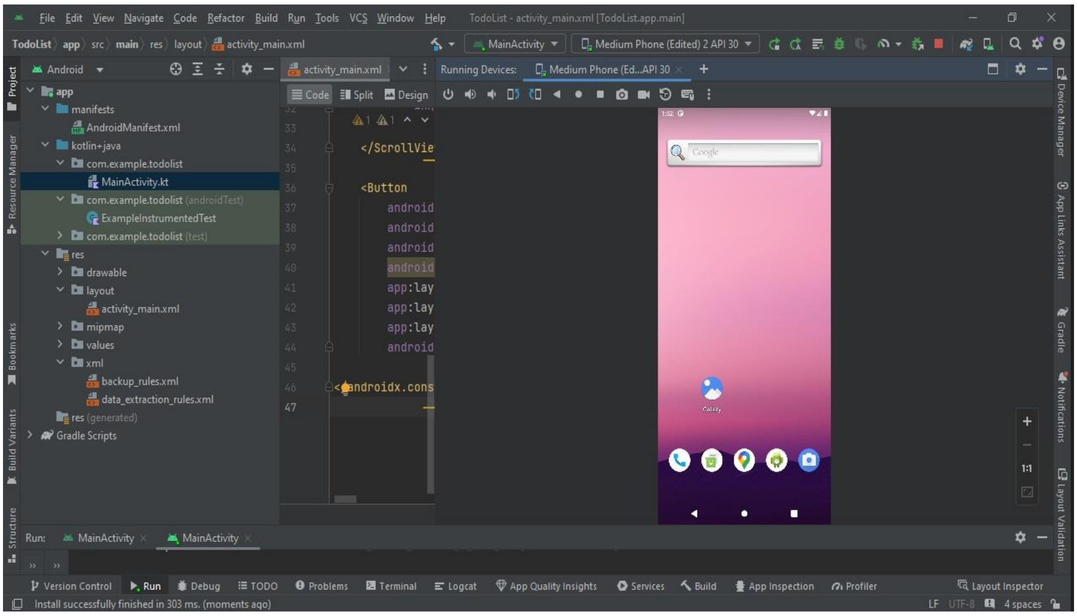
- Create a Java or Kotlin class to represent a Task.
- Define properties such as task name, description, due date, priority, etc.

Step 4: Implement TaskAdapter (for RecyclerView)

- Create an adapter class (TaskAdapter) to bind the data to the RecyclerView.
- Implement the ViewHolder pattern to efficiently manage views.

Step 5: Implement TaskViewModel (Optional)

- Create a ViewModel class to handle data operations, such as adding, editing, and deleting tasks.
- Use the ViewModel to manage the UI-related data.



```
    </ScrollVie

    <Button
        android
        android
        android
        android
        app:lay
        app:lay
        app:lay
        android

<androidx.cons
```

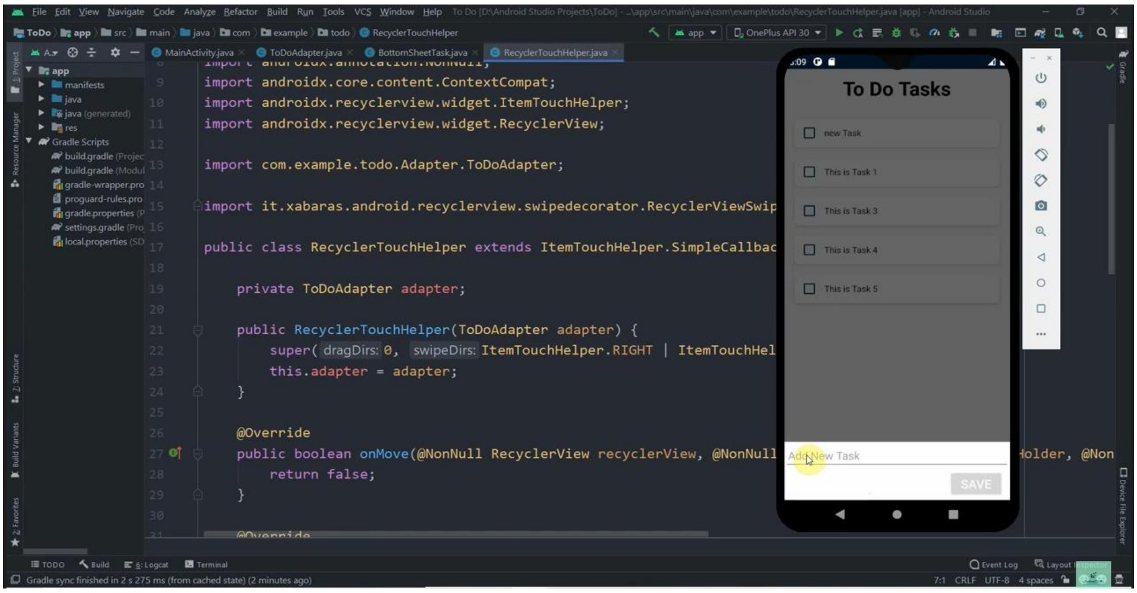
Install successfully finished in 303 ms. (moments ago)

Step 6: Connect UI with Data and ViewModel

- In the MainActivity, instantiate the TaskViewModel.
- Set up observers to update the UI when data changes

Step 7: Implement To-Do Task Functionality

- Capture user Input from the UI.
- Use the Task View Model to add a new task to the list.

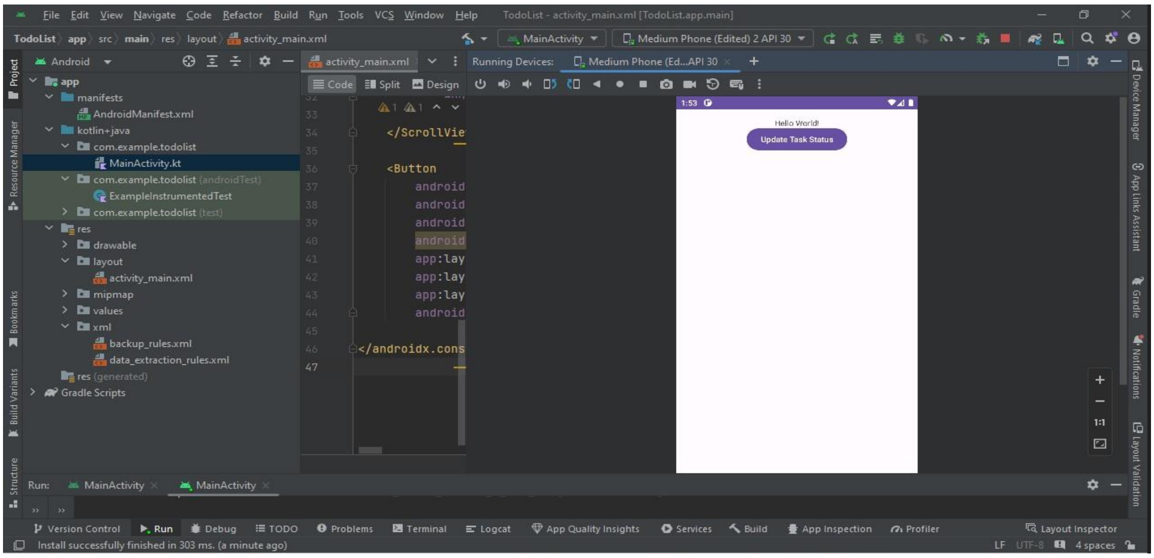


```
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.RecyclerView;

import com.example.todo.Adapter.ToDoAdapter;

import it.xabaras.android.recyclerview.swipedecorator.RecyclerViewSwip

public class RecyclerTouchHelper extends ItemTouchHelper.SimpleCallbac

    private ToDoAdapter adapter;

    public RecyclerTouchHelper(ToDoAdapter adapter) {
        super( dragDirs: 0,  swipeDirs: ItemTouchHelper.RIGHT | ItemTouchHel
        this.adapter = adapter;
    }

    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull
        return false;
    }
```

Gradle sync finished in 2 s 275 ms (from cached state) (2 minutes ago)

Step 8: Implement Edit and Delete Functionality

- Allow users to edit, update and delete tasks.
- Update the UI and data accordingly.



```
    </ScrollVie

    <Button
        android
        android
        android
        android
        app:lay
        app:lay
        app:lay
        android

</androidx.cons
```

Install successfully finished in 303 ms. (a minute ago)

Step 10: Test Your Application

- Run your application on an emulator or a physical device.
- Test adding, editing, and deleting tasks to ensure everything works as expected.