Recycler View 😄 :

1. activity_mail.xml → In this we import the “Recycler View” layout on the mobile screen and set it.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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:id="@+id/main"
    android:layout_width="match_parent"
    android:orientation="horizontal"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2. item_layout.main → In this there is the prototype like how is our card is view for all of the enteries in the mobile app.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    app:cardElevation="8dp"
    app:cardBackgroundColor="@color/white"
    android:layout_margin="4dp"
    app:cardCornerRadius="15dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">
    <TextView
        android:id="@+id/textView"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello" />

  <TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="World" />
  </LinearLayout>
</androidx.cardview.widget.CardView>
```

3. adapterClass → Go to kotlin+java and make a file with name “adapterClass” this is the normal class and also we create itemDetails and this is the data class (means what are the enteries we are shown on the recycler view list !!).

```
package com.example.recyclerview

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class adapterClass(private val items:List<itemDetails>) :
RecyclerView.Adapter<adapterClass.ViewHolder>()
{
  class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
    val title : TextView = itemView.findViewById(R.id.textView)
    val description : TextView = itemView.findViewById(R.id.textView2)
  }
  override fun onCreateViewHolder(
    parent: ViewGroup,
    viewType: Int
  ): ViewHolder {
    val view = LayoutInflater.from(parent.context).inflate(R.layout.item_layout, parent, false)
    return ViewHolder(view)
  }

  override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    val item:itemDetails = items[position]
    holder.title.text = item.title
    holder.description.text = item.description
  }

  override fun getItemCount() = items.size
```

```
}
```

3. itemDetails.kt → It is the data class where we initialize that what is the entries we take in the recycler view list.

```
package com.example.recyclerview
data class itemDetails(
    val title : String,
    val description : String
)
```

4. MainActivity.kt → It is the kotlin class. Here, we not use the APIs, we use predefined list and show it on the app.

```
package com.example.recyclerview

import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        val itemList = listOf(
            itemDetails("Item 1", "Description 1"),
            itemDetails("Item 2", "Description 2"),
            itemDetails("Item 3", "Description 3"),
            itemDetails("Item 1", "Description 1"),
            itemDetails("Item 2", "Description 2"),
            itemDetails("Item 3", "Description 3"),
            itemDetails("Item 1", "Description 1"),
            itemDetails("Item 2", "Description 2"),
            itemDetails("Item 3", "Description 3"),
            itemDetails("Item 1", "Description 1"),
            itemDetails("Item 2", "Description 2"),
            itemDetails("Item 1", "Description 1"),
            itemDetails("Item 2", "Description 2"),
            itemDetails("Item 3", "Description 3"),
            itemDetails("Item 1", "Description 1"),
            itemDetails("Item 2", "Description 2"),
```

```
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 3", "Description 3"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 3", "Description 3"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 3", "Description 3"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 3", "Description 3"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 3", "Description 3"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 3", "Description 3"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 3", "Description 3"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 3", "Description 3"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 3", "Description 3"),
    itemDetails("Item 1", "Description 1"),
    itemDetails("Item 2", "Description 2"),
    itemDetails("Item 1", "Description 1"),
```

```
        itemDetails("Item 2", "Description 2"),
        itemDetails("Item 3", "Description 3"),
        itemDetails("Item 1", "Description 1"),
        itemDetails("Item 2", "Description 2")
    )
    val recyclerview = findViewById<RecyclerView>(R.id.recyclerview)
    recyclerview.layoutManager = LinearLayoutManager(this)
    recyclerview.adapter = adapterClass(itemList)
  }
}
```

---

adapterClass

This is your main class. It **manages the entire process**. It takes your list of data (items) and adapts it so the RecyclerView can understand how to show it. It's the bridge between your data and the UI.

ViewHolder

This is a small helper class that acts like a **container for a single row's view**. It holds the actual UI components for one item (in this case, the title and description TextViews). By holding them, it makes updating the row super fast, which is why RecyclerView is so efficient.

onCreateViewHolder

This method is the **"View Factory."** 🏭 It's called by the RecyclerView only when it needs to create a brand new row from scratch. It inflates your item_layout.xml file into a View object and then puts it inside a ViewHolder container. This is an expensive operation, so Android tries to call it as few times as possible.

onBindViewHolder

This method is the **"Data Binder."** binder. It's called every time a row needs to display data. It takes a ViewHolder (which might be brand new or recycled) and a position. Its job is to find the correct data item from your list at that position and "bind" its contents to the views inside the ViewHolder (e.g., setting the text for the title and description). This method is called frequently as you scroll.

getItemCount

This is the simplest one. It just **tells the RecyclerView the total number of items** in your data list. The RecyclerView uses this count to know how many rows it needs to prepare to show.