

Building a server response system using Python 3 (Flask) & SQLite for processing JSON responses (Part 2)

1. Introduction

In this tutorial, we will continue to use Python & SQLite to build a web server response system for processing responses in JSON format such that the Android apps will be able to get data from the SQL database in an easy manner.

2. Communicate with Android

This application contains two activities. The first activity contains 1 TextView, 1 EditText and 1 Button. The user can enter a name into the EditText field. Upon the button being pressed, a connection will be made to “/tutorial”. The name will be added to the table “student” on the server. The app will then receive the updated name list from the server.

Next, the user will be guided to the second activity which contains a ListView (a dynamic layout). The received JSON object will then be interpreted and will be inserted into the ListView for displaying. An extra layout file serves as a template to define how each item is displayed.

Note: **My package name is “com.example.myapplication”** and you should update lines concerned into your package name.

Let's create a new Android Studio project with “Empty Activity”. In **activity_main.xml**, add a LinearLayout together with a TextView, an EditText and a Button.

```
<LinearLayout
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

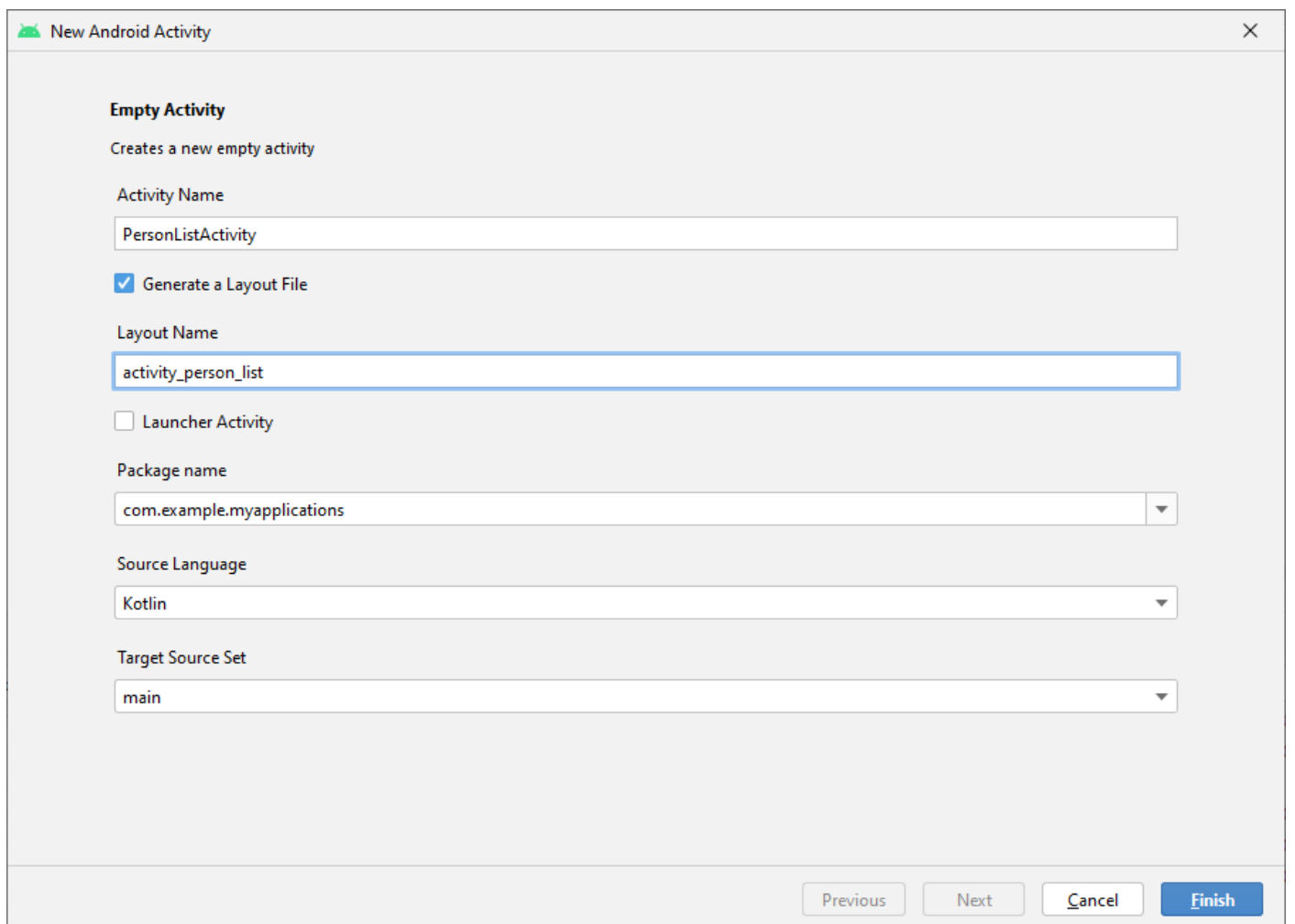
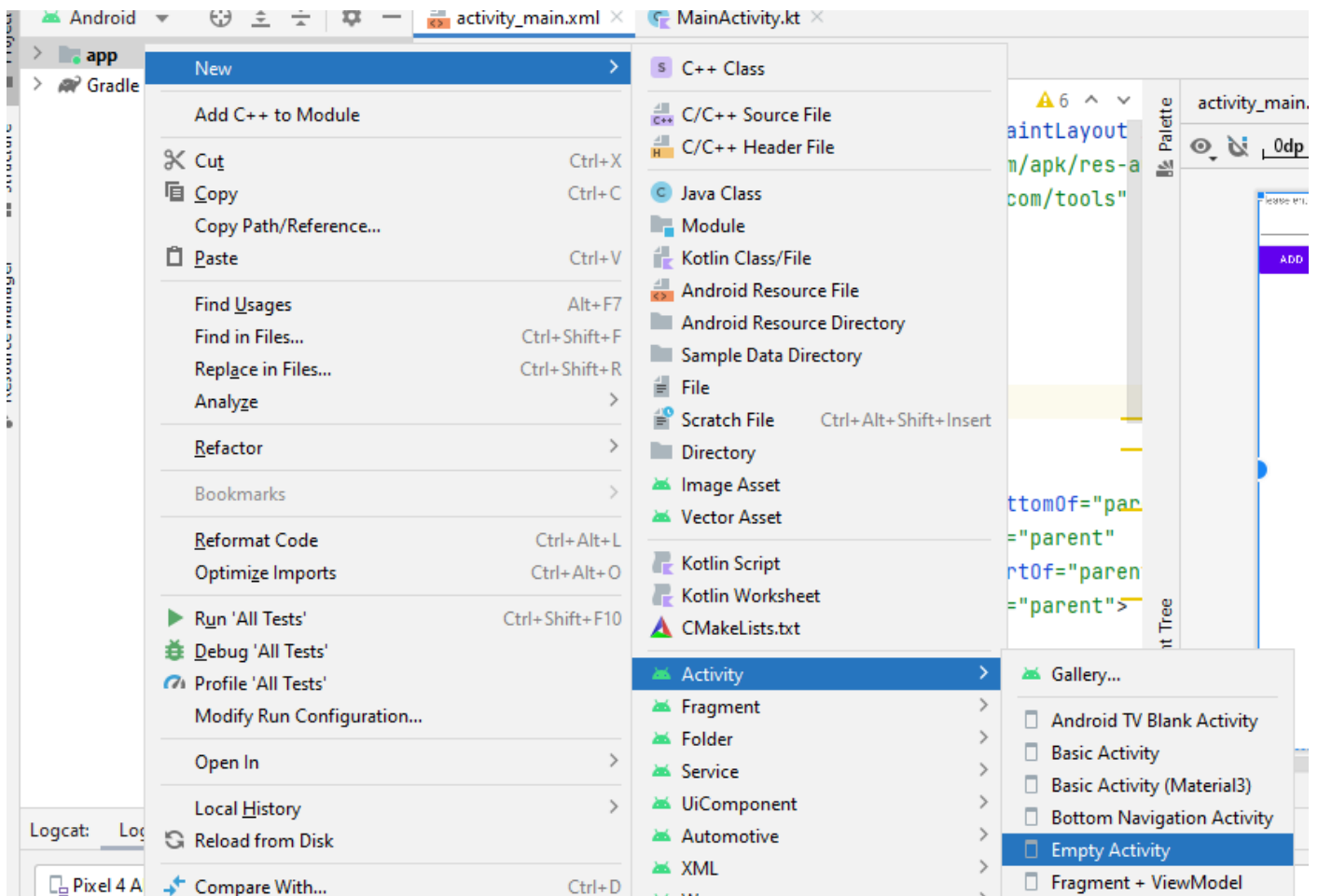
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Please enter a name:" />

    <EditText
        android:id="@+id/txt_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="500px" />

    <Button
        android:id="@+id/btn_add"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ADD" />

</LinearLayout>
```

Add a new activity by right-clicking the “app” folder -> “New” -> “Activity” -> “Empty Activity”. Name the Activity Name as **PersonListActivity** and Layout Name as **activity_person_list** and then click “Finish”.



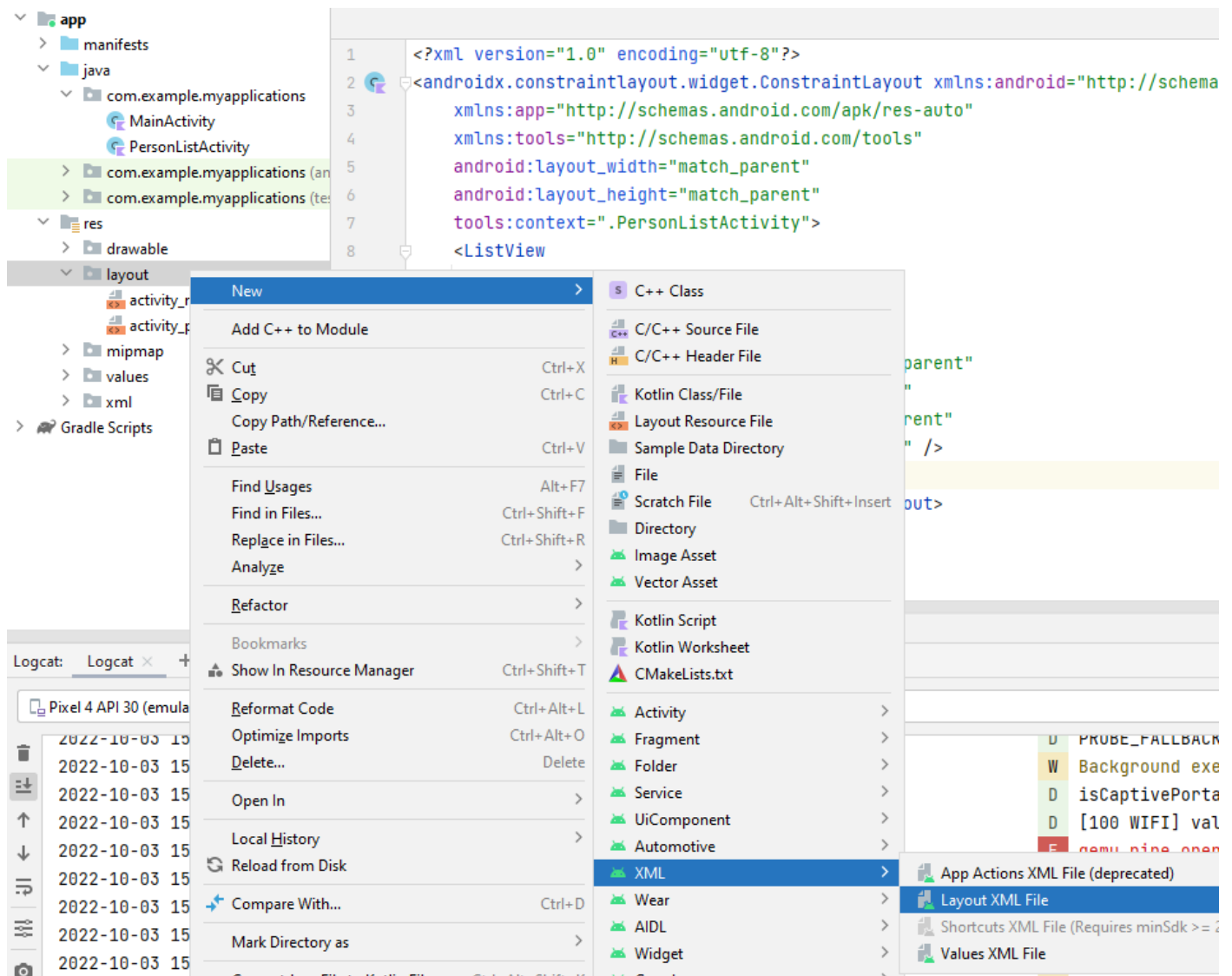
Then, Add a ListView to **activity_person_list.xml**.


```

<ListView
    android:id="@+id/list_view"
    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" />

```

Create a XML file by right-clicking the “layout” folder -> “New” -> “XML” -> “Layout XML File”. Name the Layout File Name as **person_list_item**. This XML file defines how each item in the ListView is displayed.



 New Android Component

Layout XML File

Creates a new XML layout file

Layout File Name

person_list_item

Root Tag

LinearLayout

Previous

Next

Cancel

Finish

Replace the **person_list_item.xml** with the following components. This defines how each item is displayed. Thus, each item is represented in a vertical **LinearLayout** with data fields of role and name displayed in **TextViews**.

```
<?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="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/role"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FF7300"
        android:textSize="16sp" />

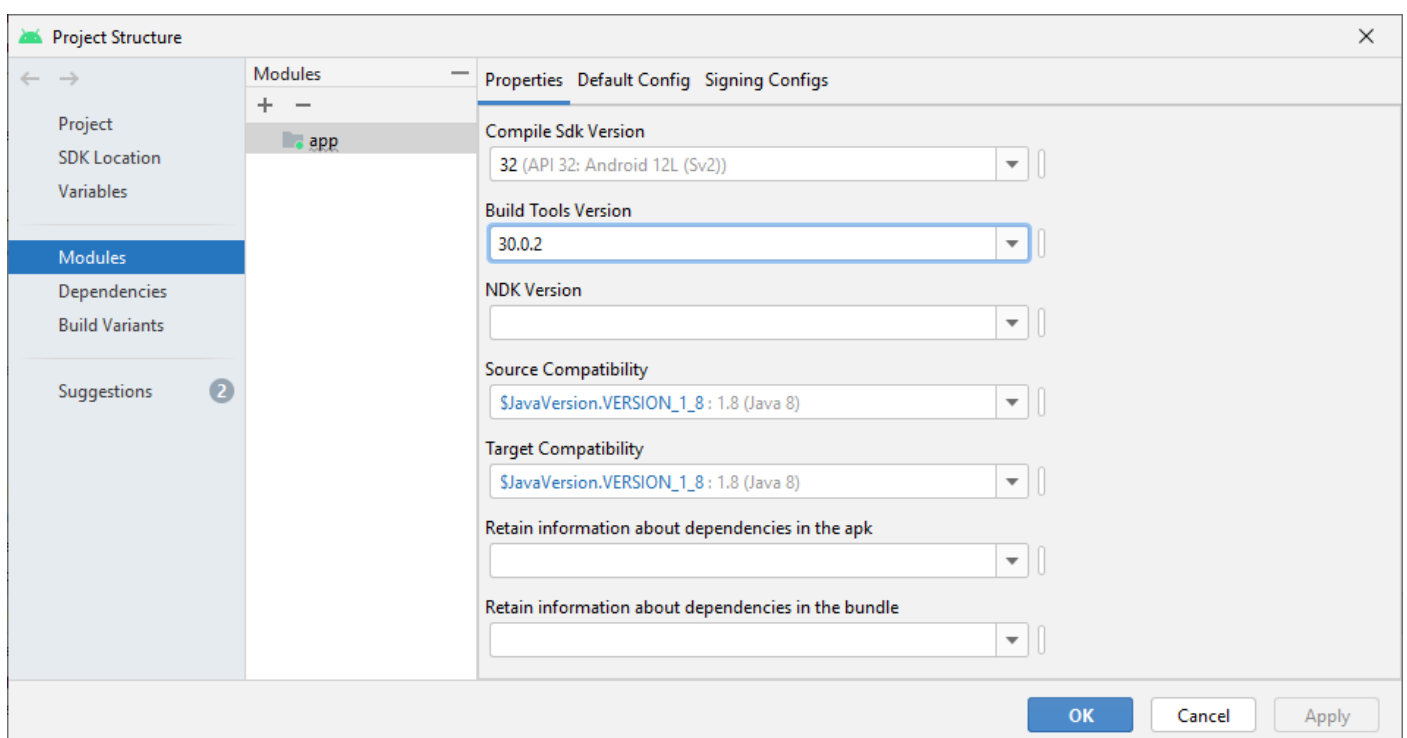
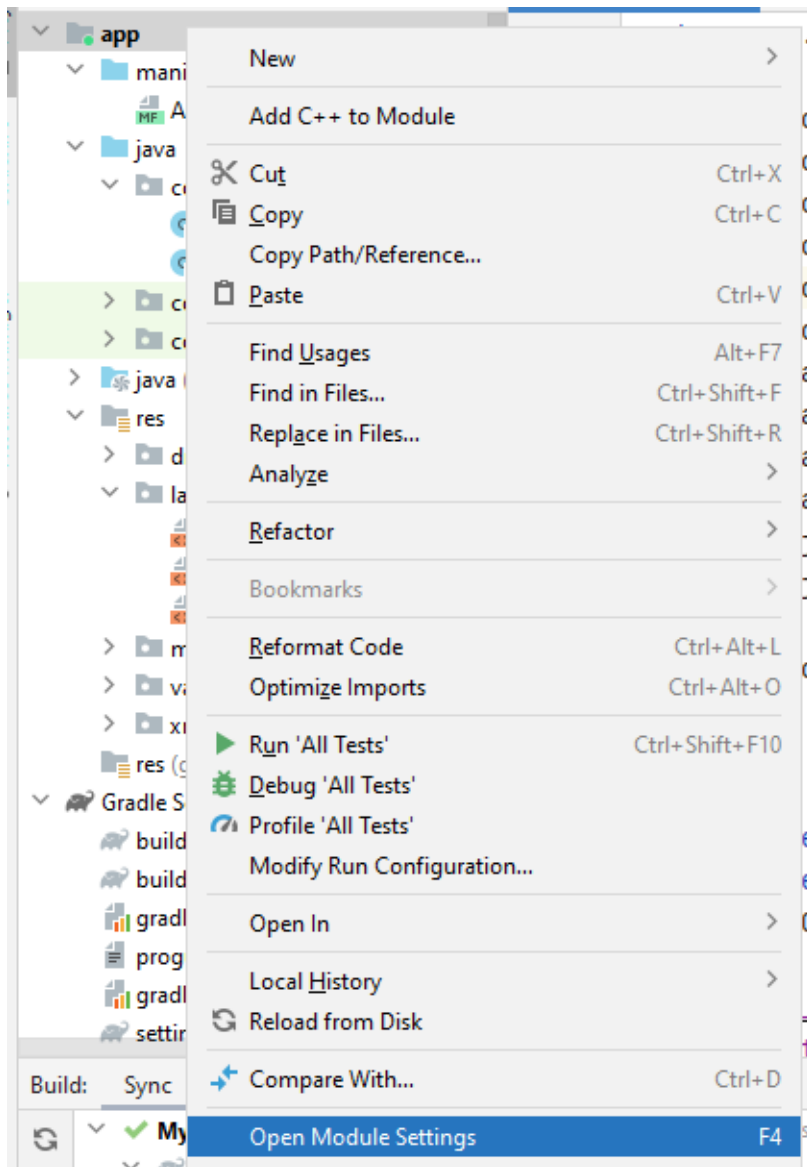
    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:textSize="13sp" />

</LinearLayout>
```

We use the Volley library to make HTTP connections. Add the following dependency to your app's **build.gradle** file.

```
dependencies {
    implementation "com.android.volley:volley:1.2.1"
}
```

This library requires a later version of Build Tools Version setting. Right-click “app” -> “Open Module Settings”. For Build Tools Version, select a version of at least 30.



MainActivity.ky

Replace the MainActivity.ky with the following code. Note that you may need to change the package name and the URL.

```
package com.example.myapplication

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.EditText
import androidx.appcompat.app.AppCompatActivity
import com.android.volley.Request
import com.android.volley.Response
import com.android.volley.toolbox.JsonObjectRequest
import com.android.volley.toolbox.Volley
import org.json.JSONArray
import org.json.JSONObject

class MainActivity : AppCompatActivity() {
    private var btn_add: Button? = null
    private var text_name: EditText? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        btn_add = findViewById(R.id.btn_add)
        text_name = findViewById(R.id.txt_name)

        btn_add!!.setOnClickListener {
            val name:String = text_name!!.text.toString()
            sendMessage(name)
        }
    }

    fun sendMessage(name:String) {
        val url:String = "http://147.8.177.239:5000/tutorial" + "?name=" + name

        val jsonObjectRequest = JsonObjectRequest(
            Request.Method.GET, url, null,
            Response.Listener { response ->
                switchActivity(response)
            },
            Response.ErrorListener { error ->
                Log.e("MyActivity",error.toString())
            }
        )
        Volley.newRequestQueue(this).add(jsonObjectRequest)
    }

    fun switchActivity(jsonObj: JSONObject){
        val jsonArray: JSONArray = jsonObj.get("students") as JSONArray
        val studentList = arrayListOf<String>()

        for (i in 0..jsonArray.length() - 1){
            studentList.add(jsonArray.get(i) as String)
        }
    }
}
```

```

        val intent = Intent(this, PersonListActivity::class.java).apply {
            putStringArrayListExtra("data", studentList)
        }
        startActivity(intent)
    }
}

```

PersonListActivity.kt

Replace the the PersonListActivity.ky with the following code.

```

package com.example.myapplication

import android.os.Bundle
import android.widget.ListView
import android.widget.SimpleAdapter
import androidx.appcompat.app.AppCompatActivity

class PersonListActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_person_list)

        val studentList = intent.getStringArrayListExtra("data")

        val list = arrayListOf<MutableMap<String, Any>>()
        for (i in studentList!!.indices) {
            val map: MutableMap<String, Any> = HashMap()
            val count = i+1
            map["Role"] = "Student $count"
            map["Name"] = studentList[i]
            list.add(map)
        }

        val adapter = SimpleAdapter(this, list, R.layout.person_list_item,
            arrayOf("Role", "Name"), intArrayOf(R.id.role, R.id.name))

        val list_view: ListView = findViewById(R.id.list_view)
        list_view.adapter = adapter
    }
}

```

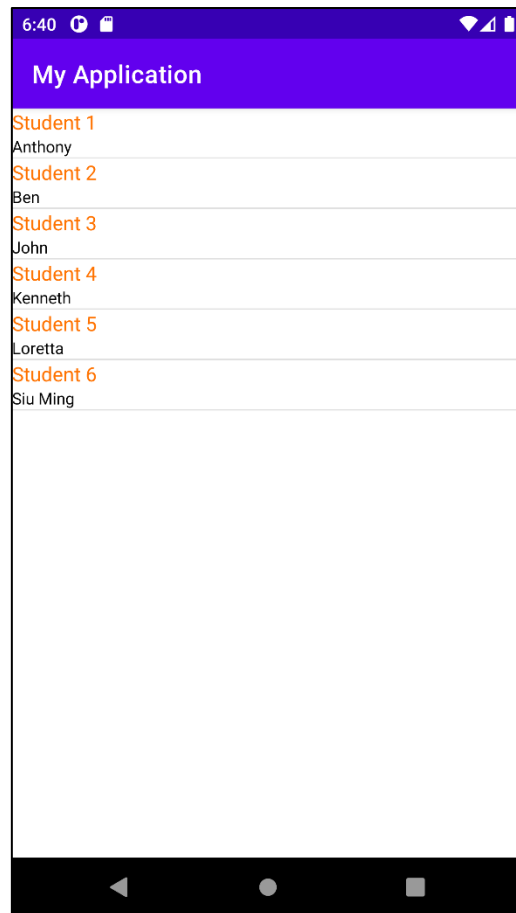
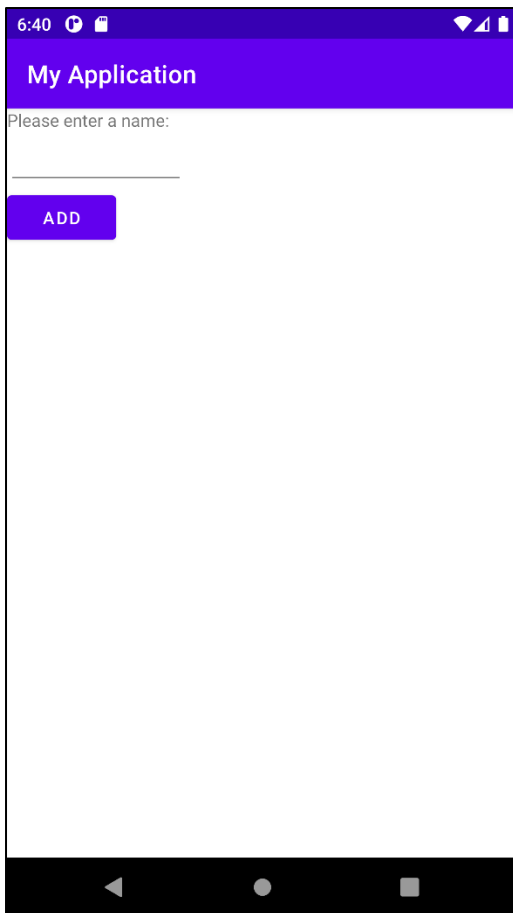
In **AndroidManifest.xml**, add the internet access permission to the app (put this outside the application tag)

```
<uses-permission android:name="android.permission.INTERNET" />
```

Since the server we built in Tutorial 4 doesn't have an SSL certificate, we can only rely on HTTP instead of HTTPS connection to access the JSON. To allow HTTP traffic in the app, add this property to the application tag in AndroidManifest.xml.

```
android:usesCleartextTraffic="true"
```

Sample output



3. Proof of tutorial participation

Add your own name to the database. Please take screen captures of the two pages (with your name in the EditText view on the first page and as the last item on the second page) and submit them to Moodle.