Anusha Nelluri – anhx2@umsystem.edu

GitHub Link - https://github.com/NelluriAnusha/Demo_Remote/tree/main/Mobilepart/ICP10

Achyuth Kumar Valeti – avgh3@umsystem.edu

GitHub Link - https://github.com/AchyuthValeti/Demo_Remote/tree/main/Mobilepart/ICP10

ICP10 (RESTful Services, Retrofit, List View, Adapter & Recycling)

A. Introduction:

Android Studio is an open-source platform that allows us to create apps for Android OS devices such as smartphones, tablets, and televisions. In this task, we are fetching information from a remote server using RESTful APIs and printing the same information on a mobile device.

In this ICP, we used **Android Studio, Java & XML** languages for mobile application development.

B. Task Description:

In this task, we were asked to print the GitHub users 'ID' and 'User Name' details from website **'https://api.github.com'** using android studio.

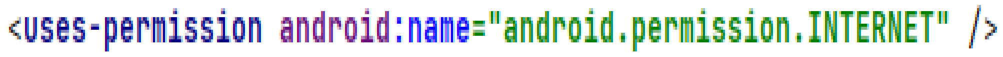
I. GitHub users' information will be printed on Android phone when clicked on 'Run' button.
II. The information will be scrolled to see all GitHub users information.

C. Implementation Process:

- We have created a new project in Android studio application and selected "Empty activity".
- Empty activity provides .java and .xml files (MainActivity.java, activity_main.xml) and it is used for printing the GitHub user's information.

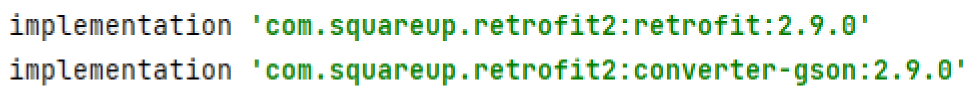
1. **<u>Adding internet permission in Andriod studio (AndroidManifest.xml):</u>**
   We need to add extra permission in androidManifest.xml to get the access from internet else we cannot retrieve the information from internet.

   So, we put the below code AndroidManifest.xml file to access internet.



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

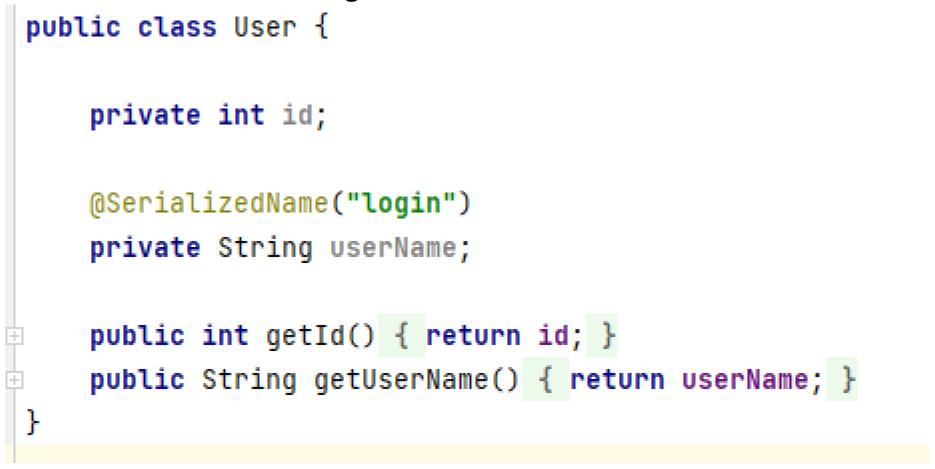
2. **Adding Retrofit configuration in “build.gradle” file:**
   To get the retrofit configuration in our project, we have added the following dependencies in “build.gradle” file.



```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

   Retrofit is the class that converts API interfaces into callable objects.

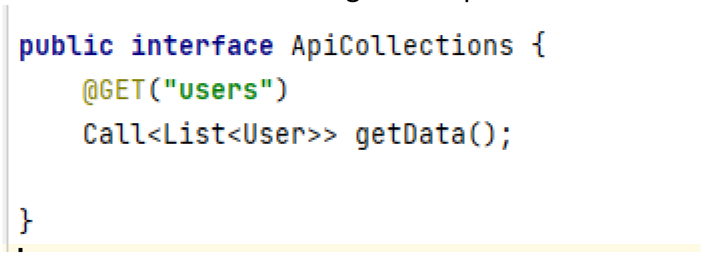
3. **Creating a modal class for storing the data (“User.java”):**
   a. We have created a new Java class i.e., “User” by right clicking on the package.
   b. In this file, we created two variables “ID” and “userName”.
   c. Here, we were retrieving the data from JSON. In general, we should use the exact key names.
   d. But, by using @SerializedName annotation we can use any alternative name instead of using the original name.
   e. We can provide the expected serialized name as an annotation attribute, Gson can make sure to read or write a field with the provided name.
   f. Here is the code for storing data into ID & userName variables.



```
public class User {

    private int id;

    @SerializedName("login")
    private String userName;

    public int getId() { return id; }
    public String getUserName() { return userName; }
}
```

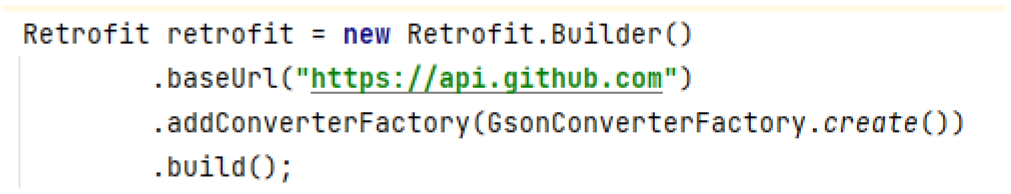
4. **Creating an Interface for API call (“ApiCollections.java”):**
   a. We have created a new java class and selected it as Interface i.e., “ApiCollections” by right clicking on the package.
   b. Since we are making GET request, we are displaying “GET” as an annotation and inside that we are sending last parameter of GitHub url.
   c. Here, we were also using Recycling and List functionalities as we are calling data from an array.
   d. Since we are calling data from an array & List, we are calling it with JSON object and named that method as “getData()”.

e. Here is the code for making GET request call and storing the data:



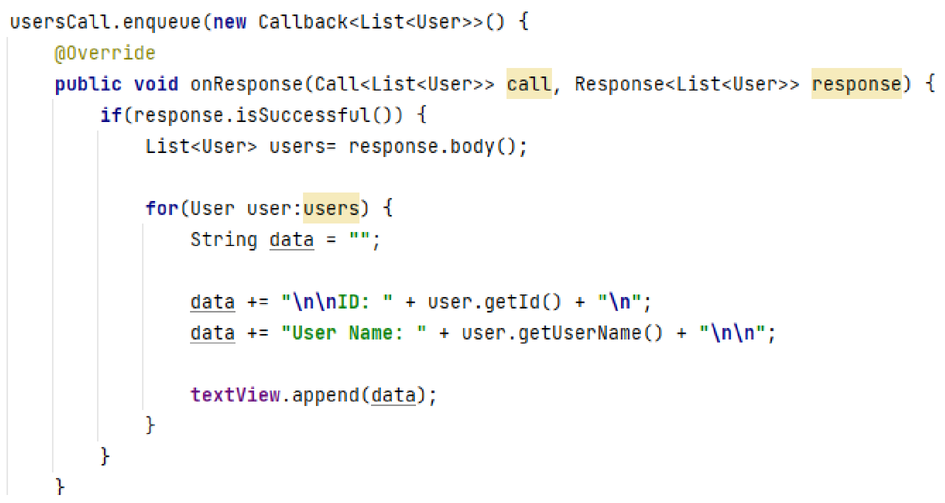
```
public interface ApiCollections {
    @GET("users")
    Call<List<User>> getData();

}
```

5. **<u>Displaying the fetched information from internet using RESTful API and other services (“MainActivity.java”):</u>**
   a. I have created the variable for textview field.
   b. Initialized that variable using findViewById() method.
   c. Then, we have created retrofit builder and passing GitHub URL to it and added Converter factory as GSON converter factory for deserializing the response using the Gson library and building the retrofit builder as below.



```
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://api.github.com")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
```

   d. And, we have created an instance for ApiCollections class.
   e. Using this newly created instance object, we stored the data into a new variable “usersCall”.
   f. Since we want to consume the API asynchronously, we will call the data as mentioned in the below code.



```
usersCall.enqueue(new Callback<List<User>>() {
    @Override
    public void onResponse(Call<List<User>> call, Response<List<User>> response) {
        if(response.isSuccessful()) {
            List<User> users= response.body();

            for(User user:users) {
                String data = "";

                data += "\n\nID: " + user.getId() + "\n";
                data += "User Name: " + user.getUserName() + "\n\n";

                textView.append(data);
            }
        }
    }
```

   g. As shown in the above code, Retrofit will use a background thread to download and process the API data, then provide the results to the UI thread through the “onResponse()” or “onFailure()” methods.

h. In onResponse() method, we have extracted the ID & username details from JSON using GET request call to GitHub url and stored into the data variable.

i. And then , I have appended the ID and userName details to the textView variable to display it on the Android Mobile.

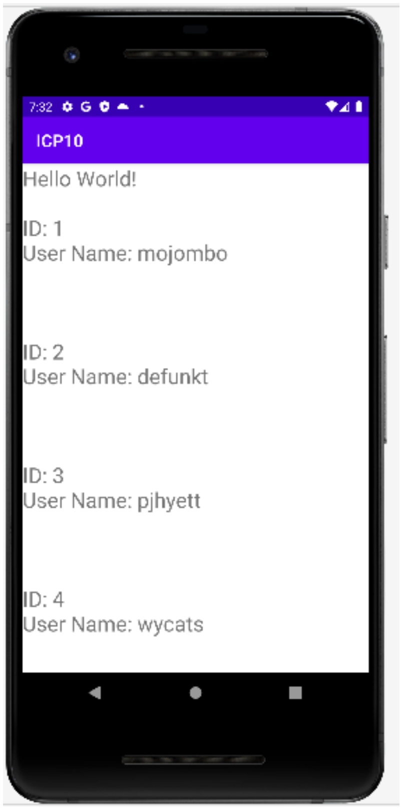
j. To scroll down the details in Android Mobile, we used the following code in "activity_main.xml" file.

```
<androidx.core.widget.NestedScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    tools:ignore = "MissingConstraints">
```

k. In onFailure() method, We have displayed the error message in toast incase if we get any failure cases as below.

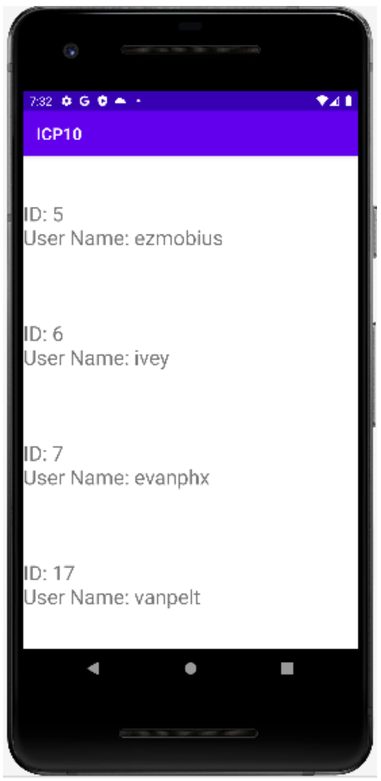
```
public void onFailure(Call<List<User>> call, Throwable t) {
    Toast.makeText( context: MainActivity.this, text: "Data Failed",Toast.LENGTH_SHORT);
}
```

D. Output Page in Android Mobile:



```
7:32
ICP10
Hello World!
ID: 1
User Name: mojombo

ID: 2
User Name: defunkt

ID: 3
User Name: pjhyett

ID: 4
User Name: wycats
```

When we click on run, this will be displayed on android phone.



```
7:32
ICP10
ID: 5
User Name: ezmobius

ID: 6
User Name: ivey

ID: 7
User Name: evanphx

ID: 17
User Name: vanpelt
```

To check the more users details information, the page Can be scrolled down/up.

E. Contribution:

We have contributed equally.

F. Conclusion:

In this ICP, we have learned basics of android & how to use Android studio to develop an application and developed a mobile android application using the same.

G. Challenges:

We have not faced any major challenges while doing the assignment.