ITMD 455/555 *Intelligent Device Applications* Lab 4

#### Playlist App- 50 points

**Introduction**. For this lab create a Login screen and do some snazzy JSON parsing!

**STEP 1 Creating a New Android Project**

Create a new project as an Empty Views Activity and name the application **Playlist**. Choose an appropriate Minimum API level such as API 28 and click Finish when complete.

**STEP 2 Create a LoginActivity screen and work your layout.**

Add into your package within your java source folder, a new Empty Views Activity called **LoginActivity** (right click on package and choose New > Activity > Empty Views Activity). Keep the option to Generate a Layout File and make sure that the option to create a Launcher Activity is checked. Keep the default Layout Name (ex. activity\_login).

Add in necessary elements into your activity\_login layout to allow for a username and password to be accepted. Password field must be protected. Include a button as well to allow the user to Submit their credentials, and check upon the click event whether the user will be accepted or not!

**STEP 3 Coding your LoginActivity file.**

Allow the user up to ( three tries ) to input their correct credentials. Credentials to check for username and password, can be hardcoded into your activity file. Each incorrect input response to either a bad username and/or password should display a long toast message stating a message such as “Wrong Credentials”. Include also how many attempts are left in your message.

If the user has successfully input their correct credentials, show a long toast message as “**Redirecting…**” to the screen. Upon successfully logging in, finish your current LoginActivity file to allow for the **MainActivity** screen be able to start so a playlist renders and becomes visible to the user.

Be thematic with your app for this project, so include a nice subtle color/font(s), etc. for all your activity screens!

**STEP 4 Run your app and try to pass the login screen.**

Test your app at this point and see if your LoginActivity is working properly and you are allowing upon a successful login by the user a redirection to MainActivity.

As an option so you can test your app further without having to login everytime, you can allow for an alternative or ‘specific activity’ to run at start, such as your MainActivity. To try this option, Go to **Run > Edit Configurations…** from your menu and choose under Launch Options the following…

Graphical user interface, text, application

Description automatically generated

Click OK and run your app at this point and if successful, you should see your MainActivity screen.

**Snapshot your Login screen showing an incorrect attempt message via your Toast widget that will display “Wrong Credentials…” and then for a successful attempt, a Toast widget that will display the “Redirecting…” message).**

**Paste your screenshots into a Word file and label your screenshots accordingly as follows: Login 1- Bad attempt, Login-2 Successful attempt.**

For now it can be beneficial to bypass your ‘working’ Login screen so it is faster to test your MainActivity file, starting the activity right away. Of course you can always change it back to the LoginActivity for the app Launch when submitting your project.

[ As an alternative, you can of course do a few things to have your MainActivity auto start such as:

1. Place an intent in your LoginActivity, in your OnCreate method, to immediately load Main Activity.
2. Adjust your manifest file to spark up your main activity as the first launcher activity versus the login activity for testing purposes. ]

**STEP 5 Creating the logic to parse JSON data.**

You will need to add in some class files to assist you parsing your json data which is located

at this url =>[**http://www.papademas.net:81/cd\_catalog.json**](http://www.papademas.net:81/cd_catalog.json)**.** Open the file   
 within your browser such as MS Edge to **observe** the containment and layout of the data.

As you can see the data from the file (as shown partly below) contains objects inside an array that will need to be parsed, examined and presented ultimately to the user.

**{**

**"Bluesy\_Music" : [**

**{**

**"SOLD": "yes",**

**"TITLE": "Empires Burlesque",**

**"ARTIST": "Bob Dylan",**

**"COUNTRY": "USA",**

**"COMPANY": "Columbia",**

**"PRICE": "10.90",**

**"YEAR": "1985"**

**},**

**{**

**"SOLD": "yes",**

**"TITLE": "Hide your heart",**

**"ARTIST": "Bonnie Tylor",**

**"COUNTRY": "UK",**

**"COMPANY": "CBS Records",**

**"PRICE": "9.90",**

**"YEAR": "1988"**

**},**

**::**

**]**

**}**

Notice the order of each record in the file, i.e., the order data (by key) that is to be read into your java source file, starting with SOLD first, down to the YEAR.

Let’s add in our first class file. Create a Java class file called **ItemObject** (right click on package and choose New > Java Class). This file will serve to hold your source data including getters and setters.

Code your class as follows:

1. Create instance fields (String type) for each object (key) as represented from the file data, with the EXACT names shown just above for the data file example from the server. It is good to include all represented data (keys/values) from a source file in case you need them along the way.

Ex. instance field names => **sold**, **title**, **artist**, etc.

1. Create a constructor with a parameter list as follows. Assign each argument passed into the constructor to appropriate instance fields.

Ex.

**public** ItemObject(String title, String artist, String year) {

**this**.**title** = title;  
 **this**.**artist** = artist;

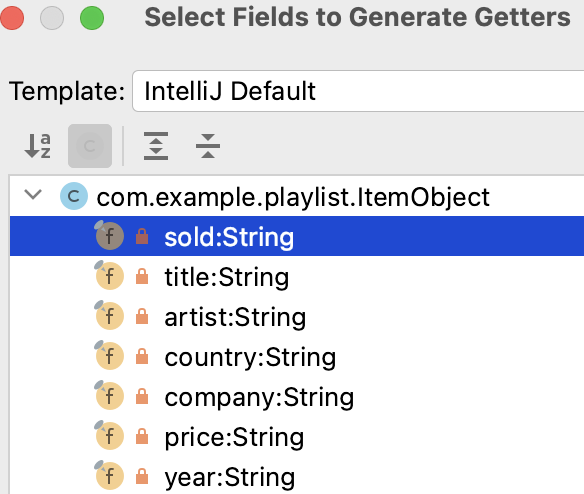
**this**.**year** = year;  
}

1. Then to finish, just after your constructor opening bracie, generate getters simply by right clicking and choosing **Generate…** from the pop up menu. Then choose **Getter**.

A blue and white text

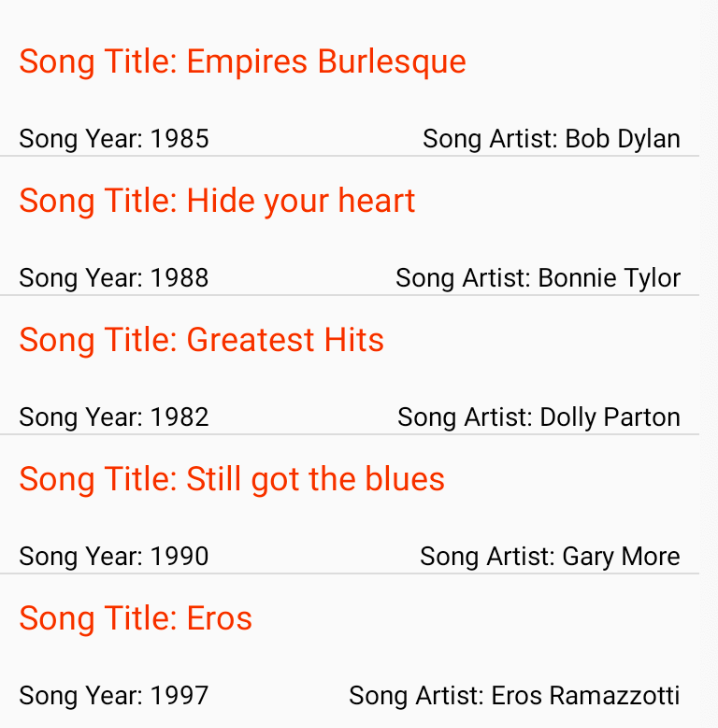
Description automatically generated

Then highlight all the fields visible as shown next and press OK. Example follows.



**STEP 6 Crafting layouts.**

Layouts will now be created and defined to display the following data to the activity.

****

For your **activity\_main** layout spec., remove your TextView code (tag) and include a **ListView** with the following constraints as follows.

<**ListView  
 android:id="@+id/listView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="10dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** />

For more on constraints check out this [link](https://constraintlayout.com/layouts/relativelayout.html).

Next create a xml layout file called **list** (right click on your res/layout folder and choose

New > XML > Layout XML File). Make it a RelativeLayout view.

Add in 3 textviews to allow for the displaying of the three data items in an order similar to the snapshot on the prior page, namely for song titles, years and artists.

Include ids for text views in order as textView, textView2 and textView3. First two views are given below as follows

<**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginRight="10dp"  
 android:layout\_marginTop="10dp"  
 android:textSize="18sp"  
 android:textColor="#df4510"  
 android:id="@+id/textView"** />

<**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/textView2"  
 android:textColor="#000"  
 android:layout\_below="@+id/textView"  
 android:layout\_alignLeft="@+id/textView"  
 android:layout\_alignStart="@+id/textView"  
 android:layout\_marginTop="40dp"** />

::

**STEP 7 Creating the adapter class.**

Create an adapter java class file (call it CustomAdapter) that will serve up views of your data (from your ItemObject class) via a list of type ItemObject!

**Include imports as follows…**

**import** android.content.Context;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.BaseAdapter;  
**import** android.widget.TextView;  
  
**import** java.util.List;

Have your class extend BaseAdapter.

**Ex.**

**public class** CustomAdapter **extends** BaseAdapter {

}

Include needed overrides for theabstract **BaseAdapter** class. Code will follow for this no matter what.

Within your class, include functionality to inflate list iems at runtime as well as declaring a list object of the generic type **ItemObject** as follows…

**private** LayoutInflater **lInflater**;  
**private** List<ItemObject> **listStorage**;  
  
**public** CustomAdapter(Context context,

List<ItemObject> customizedListView) {

**lInflater** = (LayoutInflater)context.getSystemService

(Context.***LAYOUT\_INFLATER\_SERVICE***);  
**listStorage** = customizedListView;  
}

Code in your OVERRIDES, as follows

@Override  
**public int** getCount() {  
 **return listStorage**.size();  
}  
  
@Override  
**public** Object getItem(**int** position) {  
 **return** position;  
}  
  
@Override  
**public long** getItemId(**int** position) {  
 **return** position;  
}

Implementation details for the overrides coded above are necessary to allow data to be presented, as well as the **getView** method which you can deploy as follows…

@Override  
**public** View getView(**int** position, View convertView, ViewGroup parent) {  
 TextView songTitle = **null**;  
 TextView songYear = **null**;  
 TextView songArtist = **null**;  
  
 **if** (convertView == **null**) {  
 convertView = **lInflater**.inflate(R.layout.***list***, parent,

**false**);  
 }  
 songTitle = convertView.findViewById(R.id.***textView***);  
 songYear = convertView.findViewById(R.id.***textView2***);  
 songArtist = convertView.findViewById(R.id.***textView3***);  
 songTitle.setText(**"Song Title: "** +

**listStorage**.get(position).getTitle());  
 songYear.setText(**"Song Year: "** +

**listStorage**.get(position).getYear());  
 songArtist.setText(**"Song Artist: "** +

**listStorage**.get(position).getArtist());  
  
 **return** convertView;  
}

**STEP 8 Finishing up details in your driver file.**

In your MainActivity file, include the following additional imports and code to present the playlist data to the screen via your CustomAdapter.

**Additional imports needed**

**import** android.os.AsyncTask;  
**import** android.widget.ListView;  
**import** org.json.JSONArray;  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
**import** java.io.BufferedInputStream;  
**import** java.io.BufferedReader;  
**import** java.io.InputStream;  
**import** java.io.InputStreamReader;  
**import** java.net.HttpURLConnection;  
**import** java.net.URL;  
**import** java.util.ArrayList;  
**import** java.util.List;

**public class** MainActivity **extends** AppCompatActivity {  
  
 **private** ListView **playList**;  
   
 @Override  
 **protected void** onCreate(Bundle savedInstanceState)

{  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **playList** = findViewById(R.id.***listView***);  
 **new** AsyncDataClass().execute();  
  
 }

**private class** AsyncDataClass **extends** AsyncTask<String, Void,

String> {  
 HttpURLConnection **urlConnection**;  
 @Override  
 **protected** String doInBackground(String...params ) {  
  
 StringBuilder jsonResult = **new** StringBuilder();  
  
 **try** {  
  
 URL url = **new**

URL(**"http://www.papademas.net:81/cd\_catalog.json"**);  
 **urlConnection** = (HttpURLConnection)

url.openConnection();  
 InputStream in = **new**

BufferedInputStream(**urlConnection**.getInputStream());  
  
 BufferedReader reader = **new** BufferedReader(**new**

InputStreamReader(in));  
 String line;  
 **while** ((line = reader.readLine()) != **null**) {  
 jsonResult.append(line);  
 }  
 System.***out***.println(**"Returned Json url object "** +

jsonResult.toString());  
  
 } **catch** (Exception e) {System.***out***.println(**"Err: "** + e);}  
 **finally** {  
 **urlConnection**.disconnect();  
 }  
 **return** jsonResult.toString();  
 }  
  
 @Override  
 **protected void** onPreExecute() { }  
  
 @Override  
 **protected void** onPostExecute(String result) {  
  
 System.***out***.println(**"Result on** post execute**: "** + result);  
 List<ItemObject> parsedObject =

returnParsedJsonObject(result);  
 CustomAdapter jsonCustomAdapter = **new**

CustomAdapter(MainActivity.**this**, parsedObject);  
 **playList**.setAdapter(jsonCustomAdapter);  
 }  
  
 } *//end AsyncDataClass class*

**private** List<ItemObject> returnParsedJsonObject(String result){  
  
 List<ItemObject> jsonObject = **new** ArrayList<ItemObject>();  
 JSONObject resultObject = **null**;  
 JSONArray jsonArray = **null**;  
 ItemObject newItemObject = **null**; *//interior object holder*  
  
 **try** {  
 resultObject = **new** JSONObject(result);  
 System.***out***.println(**"Preparsed JSON object "** +

resultObject.toString());

*// set up json Array to be parsed*  
 jsonArray = resultObject.optJSONArray(**"Bluesy\_Music"**);

} **catch** (JSONException e) { e.printStackTrace(); }

**for**(**int** i = 0; i < jsonArray.length(); i++){  
 JSONObject jsonChildNode = **null**;  
 **try** {   
 jsonChildNode = jsonArray.getJSONObject(i);   
 *//get all data from stream*

String songSold = jsonChildNode.getString(**"SOLD"**);  
 String songTitle = jsonChildNode.getString(**"TITLE"**);  
 String songArtist =

jsonChildNode.getString(**"ARTIST"**);  
 String songCountry =

jsonChildNode.getString(**"COUNTRY"**);  
 String songCompany =

jsonChildNode.getString(**"COMPANY"**);  
 String songPrice = jsonChildNode.getString(**"PRICE"**);  
 String songYear = jsonChildNode.getString(**"YEAR"**);  
 newItemObject = **new** ItemObject(songTitle,

songArtist, songYear);  
 jsonObject.add(newItemObject);  
  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 **return** jsonObject;  
 } *//end method*  
} *//end MainActivity class*

**STEP 9 Modify your AndroidManifest file!**

To make sure you connect to the internet for data retrieval by adding permissions to your manifest file, that is within your manifests folder. Note, you can place the line above your opening **application** tag as shown next.

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

Then within your opening application tag, add in support to read “clear text” over the web as follows.

**android:usesCleartextTraffic="true"**

**STEP 10 Time to run this app!**

Run your app and check things out. Check Logcat output for data records displayed!

Snapshot your Activity screen result at this point and paste it into your doc file. Label your snapshot appropriately.

**STEP 11 Modify your layout design.**

Include code to “programmatically” set varying background colors in your

**getView** function located in your CustomAdapter class file. To toggle colors, merely include the following code snippet in the method of choice at an appropriate place. Adjust any font/background colors you deem necessary, so display looks appropriate.

**if** (position % 2 == 1) {  
 convertView.setBackgroundColor(Color.***GRAY***);  
} **else** {  
 convertView.setBackgroundColor(Color.***LTGRAY***);  
}

Snapshot your modified Activity display and place in into your doc file, labeled appropriately.

**Grads:**

Include a button in your main layout, aligned at the bottom of your view, that when clicked, will include a pop up (using an [AlertDialog](https://stackoverflow.com/questions/2115758/how-do-i-display-an-alert-dialog-on-android) box) displaying some stats/data analytics as follows….

1. Display a count of the number of titles in the playlist.
2. Display the title and price of the most expensive CD in the collection.
3. Display countries of origin in order of the listing.

**Extra credit for everyone (choose one option below):**

- Include in your view to the user, song titles by year in ascending order.

- Optimize getView using a ViewHolder to display your textual data.

**STEP 12 Submitting your assignment:**

For **full** credit turn in the following **pdf / zip** files filled with this:

**PDF file #1** containing the following:

1. All your java files.
2. All your XML files (ex. layout files, and your AndroidManifest.xml files).

**PDF file #2** containing the following:

1. Four Snapshots of your various displays as listed along the way in the specs.
2. \*Grads include a pop up dialog box snapshot of your analytic results.
3. Any extra credit snapshot(s).

Zip file of all your project files.