InstaClient - Instagram Client Assignment

Summary of the Application

This application is an Instagram client that allows a user to login using their Instagram credentials, and then view a stream of popular pictures from Instagram. There is a menu available to the user using which they may opt to log out from the client and reload/refresh the popular pictures from Instagram.

Component-Level Design

The application has only activity, MainActivity, which is the launch Activity. Apart from that there are two fragments – LoginFragment and PopularPicturesFragment. As the name suggested, LoginFragment is used for the user to login using Instagram credentials, and PopularPicturesFragment contains a ListView that is used to display popular pictures along with a caption and a Like button.
There are three AsyncTasks in the application that are used for completing the user authentication, fetching the popular pictures and posting a like for a picture.

Note: Per Instagram Developer portal, applications registered with Instagram after April 2015 do not have "write" access, and as such cannot post likes, comments etc. However, I have kept the code for "Like" in the app (for demonstration purposes), and I handle the error gracefully.

The ListView that shows popular pictures does so using a custom adapter – PopularPicturesAdapter. This custom adapter utilizes a ViewHolder pattern to ensure smooth scrolling. Also, Lazy Loading of images is implemented to make the ListView/PopularPicturesAdapter more efficient and smooth scrolling.

This application uses SharedPreferences to store user's Instagram AccessToken. The network communication in this app uses Android's HttpUrlConnection class and WebView is used for initial phase of authentication.

Instagram Authentication

Instagram uses OAuth for user authentication. The first phase of authentication entails getting an authentication code from Instagram after providing the client ID and the redirect URI that is configured for the client. Since this phase involves a redirect URI, generally this step of authentication is performed using a WebView, and instead allowing the WbView to get redirected to the redirect URI, the redirect URL is intercepted, and the authentication code is obtained.

The next step of authentication involves fetching an AccessToken for the user by providing the authentication code, client ID, and client secret to Instagram. Once we get the AccessToken, the user is authenticated to Instagram. This step of authentication is performed using an AsyncTask and HttpUrlConnection.

Fetching Popular Pictures

Popular pictures are retrieved from Instagram using /media/popular API (this hint was provided by Paul when the assignment details were sent). The response data is in JSON format, and this response is parsed using standard JSON library, and converted to a data model. The custom adapter to display the data in a ListView uses this data model.

Documentation

The code is well commented (I think), so you can review the code comments to see more details on the implementation.

Open Source Libraries

- Universal Image Loader - https://github.com/nostra13/Android-Universal-Image-Loader
- Picaso by Square - https://github.com/square/picasso

The app currently uses UIL for lazy loading of images, but that can be easily changed to use Picaso by changing a flag in the file Constants.java.

Design Constraints

I have tried to support configuration changes, such as rotation in this app. However, the LoginFragment is set to a fixed (Portrait) mode. The reason for that is that the LoginFragment uses a WebView for authentication, and we do not have much control over WebView once it starts loading a URL. Instead of having the app crash, I decided to pin the orientation of LoginFragments to Portrait. A lot of applications that use WebView also have a similar design when it comes to configuration changes.

PopularPicturesFragment supports rotation and configuration changes as expected by any Android application.

Technical Discussion

Lazy Loading of Images

As I have already mentioned, this application uses Universal Image Loader for lazy loading of images but also has an option of using Picaso instead. The reason I decided to go with UIL is that I have previously worked with UIL and had more confidence in UIL. I also tested my application with Picaso and didn't find any issues. I have read that UIL used to have some memory leaks, but the developer claims that these have been fixed.

JSON Parsing

For JSON parsing, I have used standard JSON library. In some of my other code, I have used Google's GSON library to parse JSON. It is widely recommended to use GSON for JSON parsing, but typically that is used when data set is relatively larger. In this case, the data wasn't really huge and the standard JSON parser works just fine. It's been a couple of years since I have used GSON, so to make things quick, I decided to go with standard JSON parser because it would have taken me some time to refresh on GSON.

Network Communication

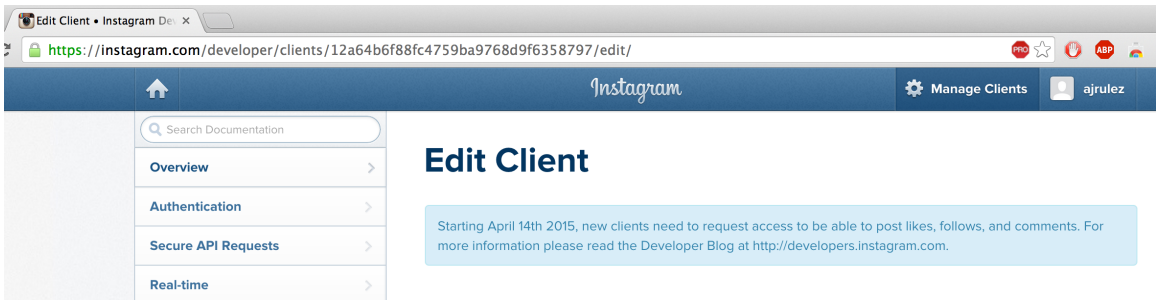
I have used HttpUrlConnection for HTTP communication in this app. I could have used open source libraries for Network communication such as Volley by Google or OkHttp by Square. OkHttp and Volley are a good choice when the app has to make multiple network connections at the same time or in parallel. These libraries do a lot of heavy lifting that is required for network communication. However, in the case of this app, we had no need to make multiple parallel network requests so the default Android-provided option, i.e. HttpUrlConnection was my choice.

Another thing to note here is that I could have checked network availability before making any network calls to bail out right away if I find that network is not available. I have not added that code to the submission, so in this case if the Network is not there, the network calls will time out with an error.

"Like" Feature

I have already discussed that the app is unable to post a like for the pictures. Even though I provide the correct scope (i.e. scope=likes) during authentication, I am unable to post likes for pictures. The reason for this is that Instagram does not provide "write" access to any application registered after April 2015. So likes, comments etc. do not work. To make likes work, the app needs to be approved by Instagram, and I have read this process can take weeks, if not months. The code

contains a couple of links to illustrate this. Also, please see the screenshot below to read about this limitation.



Over-Engineering/Over-Design

You may find some of my code over-engineered. That's done on purpose because I wanted to demonstrate how I would handle a similar project but one that's much larger in scale and more customizable. A lot of design decisions I made for this project were not necessary for a project as small as this. So if you find anything over-engineered, please be aware that it's done for demonstration purposes.

Log Off

I have added a menu option that allows a user to logout from this app. When user opts to log off, their stored AccessToken is cleared, which requires user to login if they want to use the app any more. There is one issue with the Log Off / Logout operation – even though I clear the AccessToken on the client, Instagram server is not notified of a logoff. This is because I couldn't find any logout/logoff API in Instagram's documentation. In my past when I have worked with OAuth clients, there was always an option to send a logoff/logout message to the server.

Authentication Related Discussion

This application assumes that a user is authenticated and logged in as long as an AccessToken is there. This does not hold true for OAuth based authentication. I have worked on OAuth before, and generally there is a Refresh Token provided along with Access Token, and this Refresh Token is used to get new Access Token in case the Access Token is no longer valid. Instagram does not provide any mechanism to refresh the Access Token.

Also, Instagram does not provide any documentation on error codes or error messages. It is likely that when a request is made to fetch popular pictures, the Access Token is no longer valid. In such a case, the request to fetch popular pictures would fail with an error. If proper documentation on error codes/error messages were available on Instagram, I would have captured those error codes/messages

during authentication or while fetching popular pictures and provided the user an option to re-authenticate. I have done similar implementation for other OAuth related work.

I guess one reason why Instagram does not provide Refresh Token or does not provide documentation on Access Token validity is that they could be having a really long validity for the Access Token. In such a case, once you have the Access Token, it won't expire until you logout. As mentioned, I was unable to find any details on token validity or error codes related to token validity.

IDE

This app was developed using Android Studio with support of Android API version 9 and above.

Android Permissions

android.permission.INTERNET – Required for network communication
android.permission.WRITE_EXTERNAL_STORAGE – Required by Universal Image Loader to keep a cache of popular pictures on external storage.