Assignment #3 - Mobile Software

Group # 1

1 Learning the basics

In this part, we implement 4 features in our application, namely "Hello World" display, Network usage for a file hosted on a server, online image display with its URL and a sample weather check with Yahoo Weather API.
The layout for this application is based on the sample app provided in [1].

1.1 "Hello World" Display

We use the fragment activity class to display multiple tabs of an application [2]. The first tab shows the "Hello World". The text format of TextView format is defined in .xml file in the layout folder. For example, *android:textColor* can set the text color and *android:textSize* can set the text size. Figure 1a depicts the Hello World tab.

1.2 Online Text Display

By using a HttpURLConnection, the online text can be saved in *InputStream* and then transferred to byte for reading and writing. By reading the HTTP header of the connection, we can get the information of last modified time [3] [4] [5]. Figure 1b shows the content of a file hosted in a server.

1.3 Image Display

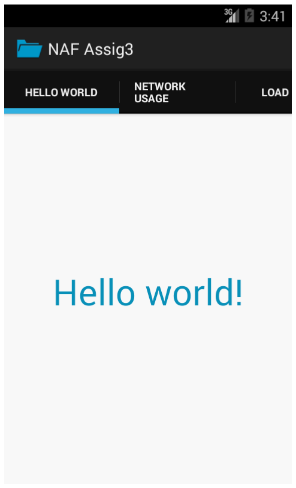
Since it costs time for online image loading especially in different network conditions, we use AsyncTask class to split one task to *pre-execute*, *do-in-background* and *post-execute*[6]. By using AsyncTask, the application can easily update the UI without manipulating the main threads.
By implementing an instance of *Bitmap* class, online image data can be transferred and displayed [7] [8]. Figure 1c presents a picture hosted in a server.

1.4 Yahoo Weather

In this part, was also required the use of AsyncTask to load the data received by Yahoo API. Basically, the user enter the WOEID that belongs to the city and this value is concatenated with the url

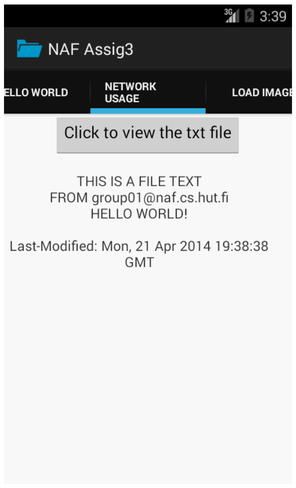
[1] `http://developer.android.com/training/implementing-navigation/index.html`
[2] `http://developer.android.com/reference/android/support/v4/app/FragmentActivity.html`
[3] `http://stackoverflow.com/questions/2416872/how-do-you-obtain-modified-date-from-a-remote-file-java`
[4] `http://www.mkyong.com/java/how-to-get-http-response-header-in-java/`
[5] `http://java2s.com/Tutorials/Java/URL_Connection_Address/How_to_read_HTTP_header_from_URL_using_URLConnection_in_Java.htm`
[6] `http://developer.android.com/reference/android/os/AsyncTask.html`
[7] `http://android-developers.blogspot.de/2010/07/multithreading-for-performance.html`
[8] `http://eclipsesource.com/blogs/2012/07/31/loading-caching-and-displaying-images-in-android-part-1/`

http://weather.yahooapis.com/forecastrss?w= for the request of data [9] The result is an XML response that after parsing is presented in a TextView. Figure 1d illustrates the results for the weather in Los Angeles.



Hello world!

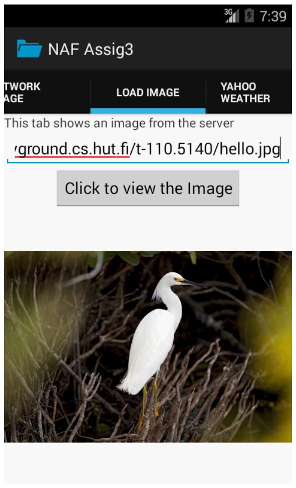
(a) Hello World



THIS IS A FILE TEXT
FROM group01@naf.cs.hut.fi
HELLO WORLD!

Last-Modified: Mon, 21 Apr 2014 19:38:38 GMT

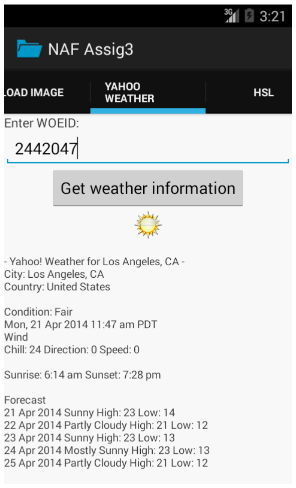
(b) Network Usage



This tab shows an image from the server

'ground.cs.hut.fi/t-110.5140/hello.jpg

(c) Load Image



Enter WOEID:

2442047

- Yahoo! Weather for Los Angeles, CA -
City: Los Angeles, CA
Country: United States

Condition: Fair
Mon, 21 Apr 2014 11:47 am PDT
Wind
Chill: 24 Direction: 0 Speed: 0

Sunrise: 6:14 am Sunset: 7:28 pm

Forecast
21 Apr 2014 Sunny High: 23 Low: 14
22 Apr 2014 Partly Cloudy High: 21 Low: 12
23 Apr 2014 Sunny High: 23 Low: 13
24 Apr 2014 Mostly Sunny High: 23 Low: 13
25 Apr 2014 Partly Cloudy High: 21 Low: 12

(d) Yahoo Weather

Figure 1: Screenshots of the application

2 Helsinki Region Transport

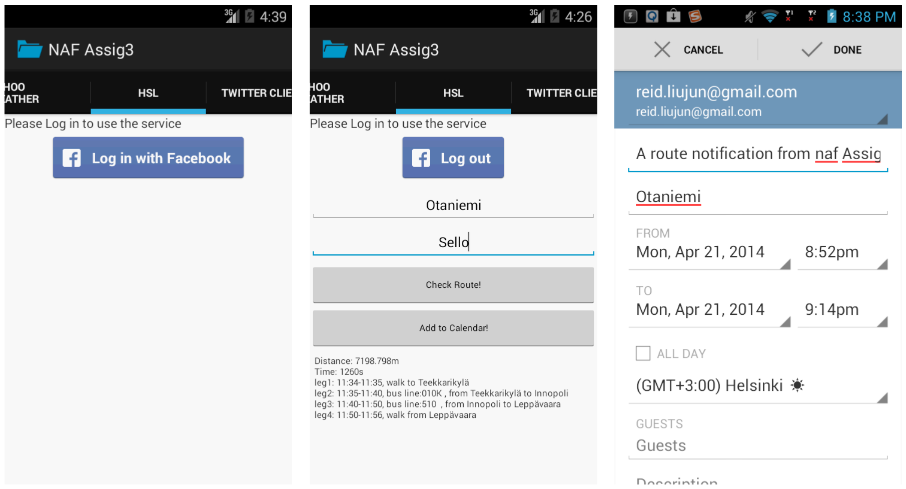
To use the HSL api, firstly a developer has to provide the information to the website to get the username and token, and then the example of api usage can be visited in the url `http://api.reittiopas.fi/hsl/prod/?user=<token>&pass=<tokenpassphrase>`. There are different types of data to choose, such as json, xml and txt. We use JSON in our application.

The feature is described as follows, first user need to login with their Facebook account on the application, and then user can set the start location and the destination location, and then our application will show the route from the HSL API, and then user can add the route to the calendar. To simplify the development, we make the start time by default the current time, and both the locations will be the first location get from the api.

The authentication is done with facebook accesstoken. After user login with their Facebook account on our application, the Facebook SDK [10] will generate an accesstoken for the session, and all the requests with the parameters such as start, destination locations and the generated accesstoken will send to our web server with HTTP GET request (e.g., `http://group01.naf.cs.hut.fi/hsl.php?token=accesstoken&loc1=Otaniemi&loc2=Sello`). When the web server get the parameter, it will use the accesstoken to validate the user [11]. For Facebook, the validation is using the url `https://graph.facebook.com/me?access_token=<accesstoken>`. If the user is verified, the web server will send request by using HSL API. For our web server, it will inquiry the coordinate codes of the locations (e.g., `http://api.reittiopas.fi/hsl/prod/?request=geocode&user=naf&pass=naf&format=json&key=Otaniemi`), and then send the received json route information to the application. We will parse the json route on the application.

[9] `http://android-er.blogspot.fi/2013/03/example-to-start-asynctask-when-button.html`
[10] `https://github.com/TechIsFun/android-facebook-login-example`
[11] `http://www.technowise.in/2013/05/validate-facebook-auth-token-in-php.html`

The process of the json format data is the most import part in this section. By using JSONArray and JSONObject in the java-json.jar library, JSON data can be easily parsed. For example, the data with the key ”legs” can be easily got by using the JSONObject get method. After the queries of the route, the result of our application is shown in Figure 2a. Total distances, duration and different legs information are shown separately on the application.
To add the route to the calendar, first we need to retrieve the start time of leg1 and the end time of the final leg. Then by using using the intent type *vnd.android.cursor.item/event*, we can start the calendar event intent. The parameter can be set by using the *putExtra* method of intent. The result is shown in Figure 2b.



Please Log in to use the service

Please Log in to use the service

Otaniemi

Sello

```
Distance: 7198.798m
Time: 1260s
leg1: 11:34-11:35, walk to Teekkarikylä
leg2: 11:35-11:40, bus line:010K , from Teekkarikylä to Innopoli
leg3: 11:40-11:50, bus line:510  , from Innopoli to Leppävaara
leg4: 11:50-11:56, walk from Leppävaara
```

reid.liujun@gmail.com
reid.liujun@gmail.com

A route notification from naf Assig

Otaniemi

FROM
Mon, Apr 21, 2014 8:52pm

TO
Mon, Apr 21, 2014 9:14pm

ALL DAY

(GMT+3:00) Helsinki

GUESTS
Guests

(a) Authentication (b) Results of HSL route (c) Add of calendar event

Figure 2: HSL Route Search

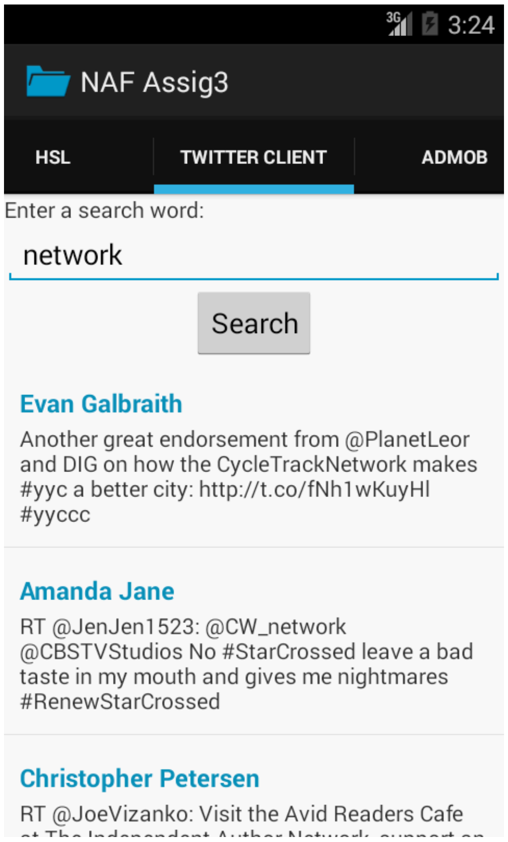
3 Additional exercises

In this part, we implement a Twitter Client to present the tweets related with a search word provided by a user. Also, a brief comparison between some mobile advertisers is described and a PoC(Proof of Concept) example is presented by using Admob as advertiser within our application.

3.1 Twitter Client

Using Twitter API required the registration of our application in Twitter applications, then using an *API Key* and *API Secret* provided by Twitter, we could perform queries without any user authentication. In order to parse the response obtained from Twitter API, we use JSONObject and JSONArray to present the fields in a TextView. Basically, the user should enter a search word and after performing search, a ListView presents all the tweets that are related with the search word. Figure 3 presents the result for *network* as a search word. [12] [13]

[12] http://www.androidhive.info/2012/01/android-json-parsing-tutorial/
[13] http://java.dzone.com/articles/how-build-android-twitter/



Enter a search word:

network

Evan Galbraith
Another great endorsement from @PlanetLeor and DIG on how the CycleTrackNetwork makes #yyc a better city: http://t.co/fNh1wKuyHl #yyccc

Amanda Jane
RT @JenJen1523: @CW_network @CBSTVStudios No #StarCrossed leave a bad taste in my mouth and gives me nightmares #RenewStarCrossed

Christopher Petersen
RT @JoeVizanko: Visit the Avid Readers Cafe

Figure 3: Twitter Client

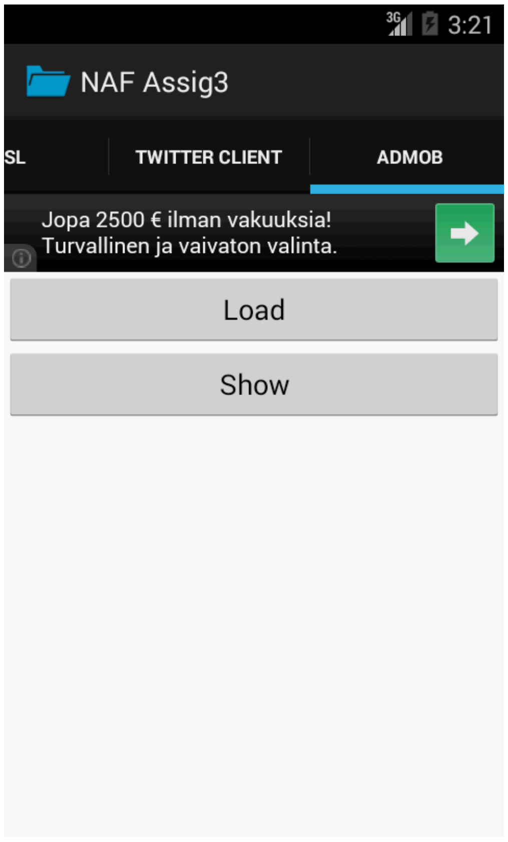
3.2 Advertisements

There are multiple mobile advertising services available in the market. The most popular of them could be Google AdMob, Google AdWhirl and Smaato.

- Google AdMob - probably is the most widely used mobile ad platform, it is simple to deployment and developers can easily check the earnings, analytic data and report on the website. The most advantage of using AdMob is that many other ads networks develop their adapter for AdMob, which means it is very simple and easy for user to add extra ads networks into AdMob. For publisher, Google has very strict verification procedure which avoids illegal and non effective click on the ads.
- Google AdWhirl - support multiple ad networks but with less documentation, and the advantage of AdWhirl is that it supports the customized home banner compared to AdMob.
- Smaato - is also support multiple ad networks and it requires less workforce to maintain impression compare to AdMob, and one more advantage of Smaato is that it supports multiple banner sytles such as toaster sytle, alert style and etc.

All the above services support multiple mobile operation system and ad networks. In this section, we choose the AdMob to display the basic usage of a mobile ad service.
We choose two typical ad forms, banner and interstitial. Interstitials, compared to banners, provide full-screen ad for users and support HTML5 and video display.

To use Admob, we can use Google account and get an application "AD UNIT ID". All the ads need the usage of *AdRequest* class to build the the ads request. More advanced applications include target group setting. For example, the group gender and birthday range can be set for the instance of *AdRequest*. Banner makes usage of the *AdView* class [14], while interstitials use the *InterstitialAd* class [15]. Admob can be tested by add test devices. Moreover, Admob also supports other ad networks by import multiple adapter libraries. The screenshots of the ads are shown in Figure 4a and Figure 4b.


3:21
NAF Assig3
SL
TWITTER CLIENT
ADMOB
Jopa 2500 € ilman vakuuksia!
Turvallinen ja vaivaton valinta.
Load
Show

(a) Banner of Admob

(b) Interstitial of Admob

Figure 4: Admob Adverstisements

4 Learning Experience

The implementation of the Android application requires the knowledge of network communication, JSON format parser, usage of API provided by different service providers and UI manipulation of Android to display the information. The level of difficulty was normal because we have programming skills and we are familiar with Java, JSON and XML. The most easy part for this assignment was the basic features that only required the implementation of static texts and images. On the other hand, the most difficult part of the assignment was the integration of different services in one application and learning how to use multiple APIs.
Each of us have worked around 20 hours on writing the code, testing the application and writing this report.

[14] https://developer.android.com/reference/com/google/android/gms/ads/AdView.html
[15] https://developer.android.com/reference/com/google/android/gms/ads/InterstitialAd.html