CS 522—Spring 2016
Mobile Systems and Applications
Assignment Ten—Location Mapping

This assignment continues the previous assignment. At the conclusion of that assignment, you should have a chat client that attaches location information to every message that it posts, and attaches the current location of the client to every Web service request it makes to the chat service. In this assignment, you will add a second app, that will use Google Maps to display the location of the clients with whom this device is communicating. You can find documentation about the Google Maps API for Android here:

```
https://developers.google.com/maps/documentation/android/
https://developers.google.com/maps/documentation/android/start
```

This API requires that you use the Google Play Services SDK, which mainly involves adding the Google Play Services library to your application. You should have already done this for the previous assignment. As with that assignment, you can ignore checking the version of Google Play Services that is installed for its currency, as long as you are running the current version of the library:

```
http://developer.android.com/google/play-services/setup.html
```

To use the Google maps API, you will need to generate a SHA1 hash of the key used to sign your application from your keystore. *Do not use the debug certificate generated by Android Studio for debugging purposes.* Instead generate your own "production" keystore[1], generate a signing key in this keystore, and use a hash of this signing key to obtain a Google Maps API key for your assignment:

```
https://developers.google.com/maps/documentation/android-api/signup
```

Create a Maps API project in your Google APIs console, use your signing key hash to obtain an API key, and include this API key in your application to enable you, and the grader, to access the Google Maps API. Include the API key in your application by adding the following element to your application manifest:

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="YOUR_API_KEY"/>'
```

Make sure that your app has enough permissions: `INTERNET` and `ACCESS_NETWORK_STATE` (to enable downloading of map tiles) are automatically merged

[1] If you build a release version of the app without specifying the release certificate, then Studio will leave the release apk file unsigned. See here for more information about signing your app: `https://developer.android.com/tools/publishing/app-signing.html`.

by Gradle into your application manifest. You will still need to specify the `ACCESS_FINE_LOCATION` permission required for the location service.

Since the Maps API requires OpenGL ES version 2 to render maps, you should provide this metadata in your application manifest:

```
<uses-feature
        android:glEsVersion="0x00020000"
        android:required="true"/>
```

Google Play Services will refuse to provide services to the application if the device does not support the required version of OpenGL.

To use maps, define an XML layout that defines a single fragment element, specify the MapFragment class for the fragment, and use the maps attributes to customize the map, e.g.,

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:map="http://schemas.android.com/apk/res-auto"
  android:id="@+id/map"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  class="com.google.android.gms.maps.MapFragment"
  map:cameraZoom="13"
  map:mapType="hybrid"
  map:uiCompass="false"
  map:uiRotateGestures="false"
  map:uiScrollGestures="false"
  map:uiTiltGestures="false"
  map:uiZoomControls="true"
  map:uiZoomGestures="false"/>
```

The map should be centered on the current location of the device, which your app will have to obtain from the chat app which is maintaining current position. Define a URI for the current chat client in the content provider in the chat app, and the map app can obtain the position for the current device by specifying that URI in a query to the content provider.

If you decide to experiment with camera tilt and bearing, you will need to add your controls for changing the tilt and bearing, unless you are running the app on a physical device, in which case you should enable rotate and tilt gestures.

You should populate the map with the location information of the other chat clients whose coordinates you are provided by the chat service. Place a marker on the map for each client, at their last known location. For each marker, set its title to be the user name of the corresponding client, and set the snippet of text underlying that to be the address that you computed with geocoding when you downloaded the location information from the chat Web service.

You do not have to interact with the user in your map application. The default behavior when a user clicks on a marker is to center the map on that point and to display its info window. The default info window is constructed from the marker's title and snippet, so just setting these properties as specified above is sufficient.

In your chat app, provide a button for displaying a map of current chat peers. Pressing that button will cause the map app to be started (with an intent). You should use an implicit intent, with your own application-specific action, to start the map app. The map app will need to query the content provider in your chat app to obtain information about the current chat peers, but this is the only other application that should be able to access this location information stored in the chat app. We will discuss in class how to ensure this.

Submitting Your Assignment

Once you have your code working, please follow these instructions for submitting your assignment:

1. Create a zip archive file, named after you, containing a directory with your name. E.g. if your name is Humphrey Bogart, then name the directory Humphrey_Bogart.
2. In that directory you should provide the Android Studio projects for your Android apps.
3. Also include in the directory a report of your submission. This report should be in PDF format. Do not provide a Word document.

In addition, record short flash, mpeg, avi or Quicktime videos of a demonstration of your assignment working. Make sure that your name appears at the beginning of the video. For example, put your name in the title of the app. *Do not provide private information such as your email or cwid in the video*.

Your solution should be uploaded via the Canvas classroom. Your solution should consist of a zip archive with one folder, identified by your name. Within that folder, you should have a single Eclipse project, for the app you have built. You should also provide a report in the root folder, called README.pdf, that contains a report on your solution, as well as videos demonstrating the working of your assignment.