CSCI 5708
Assignment 3

Instructor: Michael Hackett

Due: 11:30pm, Monday, July 15, 2019

The purpose of this assignment is to build on your experience writing Android applications and add network access, maps, and GPS/location services to your repertoire.

1 Background

Maps are an important tool for helping us find our way in the world, and because so many people carry a GPS-capable mobile device these days, maps have also become a key part of many mobile apps. In addition, the availability of nearly-ubiquitous networking and the instrumentation of everything and everyone provides the means to enhance these maps with *live data*, to enhance the experience and make possible decision-making that was not previously available to the average citizen.

Recently, Halifax Transit rolled out a live-tracking system for all of its buses and ferries, the raw data for which is available as part of HRM's Open Data initiative.[1]

2 Your Task

Implement an Android application that will help the user determine when their bus is coming by displaying it on a map, in real-time. The application should be a **native application**, implemented in Java or Kotlin. The application should meet the following **minimum** requirements:

- On launch, a map is displayed, overlaid with indicators showing the current positions of all or selected buses.
- The marker for each bus includes (at least) the route number.
- The positions of the buses are updated approximately every 15 seconds.
- The user can choose to centre the map on his or her current location, e.g., with a standard button.
- The user can zoom and pan the map freely.
- When returning to the app, the map shows the same region it was showing when the app was paused or closed (even if the app is terminated and relaunched).

User settings can be implemented in a second Activity, or in a "drawer" menu, that slides in from the left side of the screen.

[1] https://www.halifax.ca/home/open-data/halifax-transit-open-data

Exceptional functionality might include one or more of the following:

- An option to have the map follow the user's movements, keeping the map centred on their position (including on relaunch).
- Allowing the user to filter the buses displayed, e.g., through a multi-select list in a dialog or separate Activity.
- Additional indicators for the bus markers to show, for example, the direction of the route (inbound/outbound), whether the bus is stopped, how "fresh" the location is (i.e., how long since the location was reported), etc.
- A label over the bus icon that displays the full "sign display" for the route.
- Providing alerts for service interruptions. (Look for the "service-alerts" endpoint.)
- Other ideas that help the user filter the information to make it easier to focus on what he or she is interested in.

Of course, there is a lot of room for customizing and improving the basic interface. *Tasteful* use of colour, sound effects and animations add polish to the application. *Marks will be awarded partly on how polished the application is.*

Tip: The Android Emulator allows you to specify a virtual location by going to Settings, choosing "Location", and entering a Latitude and Longitude. (You can obtain suitable values from your own device, or via Google Maps on your computer.) You can even generate a sequence of positions and play back the sequence to simulate movement. (See the lower half of the Location screen.)

Important: You must **never** make network calls or perform other time-consuming tasks on the main thread of an Android application. Doing so will freeze the UI for the duration of the task, and can result in the system popping up a dialog asking the user if the application should be terminated. Always perform expensive operations on background threads. Android has *AsyncTasks* for this purpose, or you may use standard Java Threads, or some other mechanism. (There are many options to choose from.)

2.1 Transit Data

The format of the Halifax Transit data follows the General Transit Feed Specification (GTFS).[2] The static data is provided as a set of comma-separated (CSV) text files, collected together in a single ZIP archive. The real-time data is provided using a format called Protocol Buffers;[3]; the easiest way to decode this data is to use the *gtfs-realtime-bindings* library.[4] See the Halifax Transit Open Data site for the URLs.

For the purposes of this assignment, you can assume that the information in the static data is fixed and won't change, and can therefore be incorporated into your code, in whatever way you find

[2] `https://gtfs.org/reference/static/` and `https://gtfs.org/reference/realtime/v2/`

[3] `https://developers.google.com/protocol-buffers/`

[4] `https://github.com/MobilityData/gtfs-realtime-bindings`; follow the link to the Java bindings

convenient (if it is needed at all).

You may also find the information in Google Transit's "GTFS Realtime" GitHub repo[5] helpful, although it mostly duplicates the information at the GTFS.org site, referenced above.

2.2 Maps

You can use any map SDK, including (but not limited to):

- Google Maps (`https://developers.google.com/maps/documentation/android-sdk/`)
- Open Street Maps (`http://osmdroid.github.io/osmdroid/`)
- ARCGIS (`https://developers.arcgis.com/android/`

All of the above have free account options; Google will try to ask for billing details, but this can be circumvented. (This will be shown in the lab or lecture.) Nonetheless, you may choose one of the other options if you are uncomfortable giving Google more information on yourself.

3 What to Hand In

Submit a zip file (via `https://dal.brightspace.com`) containing the Android Studio project directory (including all source code files) for your assignment; and a *separate* APK file. (This can be used to test functionality if the marker cannot build the project.) Please indicate in a `README` file (in the root of the project archive) what functionality is present as well as any known bugs.

[5] `https://github.com/google/transit/tree/master/gtfs-realtime/spec/en`

4 Grading

The assignment will be graded using the rubric given in Table 1.

	Exceptional: A	Acceptable: B	Substandard: C-D	Unacceptable: F
Functionality (40%)	Application meets all functional requirements. Application does not crash. Application has significant and useful additional functionality.	Application meets all functional requirements. Application crashes rarely or not at all.	Application meets some but not all functional requirements. Application crash often.	Application does not work.
Usability (20%) screen layout, intuitive, consistent interface, etc.	The application is intuitive and easy to use. The screen structure and layout are well organized and most use cases have been anticipated. The interface is consistent.	The interface is consistent. Most common use cases have been anticipated.	Some common use cases have been anticipated and the screen layout includes all controls necessary for the application. The application is usable.	The interface is disorganized and does not correspond to the application's function. The application is not usable.
Polish (20%) background, icons, button images, transitions, themes, special effects, etc.	The application is very polished and engaging, with icons, images and special effects to engage the user. The application looks professional.	The application is somewhat polished. Examples where polish has been applied are evident, e.g., images and icons. Additional polish is necessary before the application can be released.	A minimal amount of polish has been applied. The application uses standard buttons and text fields. Basic polish such as application icon are present.	No application polish is evident.
Code (20%) organization, commenting, readability, etc.	Code is very well organized, commented, readable, and maintainable. Code looks professional.	Code is competently organized, commented and understandable. Requires minor tuning to bring it up to professional looking code.	Code is poorly organized, uncommented, or confusing. Requires significant effort to bring it up to professional looking code.	Code is unintelligible.

Table 1: Rubric for Assignment 3.