G53MDP Coursework 2 – Running Tracker

Summary

In this exercise you are required to build an Android running tracking application, and document its design and architecture in a report. This is an assessed exercise and will account for **40% of your final module mark**. This is an individual coursework, and your submission must be entirely your own work – please pay particular attention to the section of this document regarding plagiarism. This is a sizeable and open-ended coursework compared to the previous assessed exercises. This document sets out general requirements that your application should meet rather than specific instructions.

Your application and report should be submitted no later than:

- **3pm on Monday the 14th of January 2019**

Submissions should be made electronically via Moodle. Standard penalties of 5% per working day will be applied to late submissions.

Your coursework should be submitted as a .zip or .tar.gz file containing your report and application, including all relevant source code, configuration and related files, and a compiled .apk file – i.e. the contents of the directory containing your Android Studio project. Do not submit RAR files.

Specification

The *Quantified Self* or *life-logging* movement has been around for a number of years, but advances in mobile and wearable computing have increased the ability of people to collect data about their physical activities. The most common of these track activity as it happens for fitness, health or gamification purposes, for example displaying comparisons with previous activities, keeping track of best time or longest distances etc.

Application

The goal of this coursework is to design and implement a mobile application that functions as a basic *Running Tracker*, in that it should allow the user to track their movement when they decide to walk, run or jog, principally by logging the change in physical location using GPS. The application should allow the user to inspect this data in a useful manner. The user might expect to be able to ask simple questions of the data such as “how far have I run so far today?”, “how far have I run this month?” or “have I run faster than my best time today?”.

At the minimum, your application should support:

- Logging the movement of a user when they go running
- Saving the movement data in an appropriate manner
- Allowing the user to inspect their progress in some way
- Allowing the user to control how and when the logging occurs

How you approach building this application is up to you, however in principle appropriate use of all four major Android application components is expected:

- Activity
- Service
- Content Provider
- Broadcast Receiver

For this reason, it is important to consider how the task can be broken down into multiple atomic components, how they communicate with one another, and how their various lifecycles should interact. There is no requirement that your components will be accessed by components outside of the application, however it is good practice to consider how your components might be made available to other processes for subsequent reuse.

Some hints and tips regarding getting started with location services / GPS monitoring are provided below.

Your application must be written in Java and make use of the Android SDK. There are no requirements to target a specific Android API version, however you can assume that your application will be tested on an emulated device (1080 x 1920 420dpi) running Android API version 28 (Android 9.0 Pie). If you make use of functionality only available in a different version this should be clearly identified in your report.

Your application should have appropriate comments and variable / class names, so that a reader can easily understand how it works at the code level.

Adding further additional functionality to the application is encouraged, as are, for example, different interpretations of what it means to log *running* – you could consider walking – however as always your application should meet the above specification primarily. Indeed, an appropriate interpretation of the app's required functionality is an implicit part of this assessment.

Report

You should provide a report alongside your application that documents its design and technical architecture, in particular providing a rationale for the components that you have implemented and their communication, and the behaviour of the application from the user's point of view.

The report should be at minimum 1000 words long, with a maximum length of **1500 words**.

There is no set structure for the report, however you may wish to include a diagram showing the components and their relationships, and a short explanation of each one, for example how the task is broken down into discrete Activity components, how and when Services are started, how data is abstracted from underlying storage etc.

Plagiarism

N.B. Use of third party assets (tutorials, images, example code, libraries etc.) MUST be credited and referenced, and you MUST be able to demonstrate that they are available under a license that allows their reuse.

Making significant use of tutorial code while referencing it is poor academic practice, and will result in a lower mark that reflects the significance of your own original contribution.

Copying code from other students, from previous students, from any other source, or soliciting code from online sources and submitting it as your own is plagiarism and will be penalized as such. FAILING TO ATTRIBUTE a source will result in a mark of zero – and can potentially result in failure of coursework, module or degree.

All submissions are checked using both plagiarism detection software and manually for signs of cheating. If you have any doubts, then please ask.

Assessment Criteria

	Marks Available
Application Functionality	
The application meets the Activity Tracker specification, including novelty and appropriateness	40
Application Structure and Implementation	
Implementation and appropriate use of Android components	30
Programming style	
The application is easy to understand, with comments explaining each part of the code, correct formatting, and meaningful variable names	10
Report	
Description of the design and architecture	20
Total	100

Each element of your coursework will be assessed against the standard criteria:

https://workspace.nottingham.ac.uk/display/CompSci/Marking+Criteria

The following areas will be taken into account for each part of the assessment:

- Demonstrating knowledge of the area
- Quality of the concept, including appropriateness and novelty
- Quality of the technological design, including appropriate use of software design concepts, and appropriate good coding practice (abstraction, commenting, naming)

- Quality of the realization, including how well it works and elaborations over and above the basic requirements

- Including all of the above aspects, clarity of structure, quality of argument / evidence, and insight / novelty

Hints and tips

Using Location / GPS tracking

There are different mechanisms for obtaining the location of the device, including GPS, Wi-Fi or cell-tower signal triangulation, and different mechanisms for how this data can be accessed by the device.

Increasingly Android is attempting to push this functionality into Google Play services (giving Google more control over parts of the Android stack), and this provides a unified approach that fuses multiple location systems into one to provide an abstraction over multiple pieces of hardware and to reduce battery usage. This requires making use of an emulator with the Google APIs installed – generally this will be a different emulator system image.

https://developer.android.com/training/building-location.html

There is, however, a simpler approach that is perfectly adequate for this coursework, and that is to use the LocationManager system service to provide GPS (global positioning system) updates that reveal the user's location.

https://developer.android.com/reference/android/location/package-summary.html

Accessing location requires permission from the user:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

```
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
```

The LocationManager is a system service, and so needs to be retrieved from the service manager via *getSystemService*. Then it can be passed an instance of a *LocationListener* that will receive updates from the GPS provider. The two other parameters specify the minimum frequency of updates (i.e. we can say that we want at most 1 update every 5 seconds), and distance between updates (i.e. we can say that we only want to be told when the device has moved at least 5 metres). The fastest update frequency for GPS is around 1 second, and accuracy varies from a few metres upwards depending on environmental conditions.

```
LocationManager locationManager =
        (LocationManager)getSystemService(Context.LOCATION_SERVICE);
MyLocationListener locationListener = new MyLocationListener();

try {
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
```

```
            5, // minimum time interval between updates
            5, // minimum distance between updates, in metres
            locationListener);
} catch(SecurityException e) {
    Log.d("g53mdp", e.toString());
}
```

The MyLocationListener class receives these location events by implementing the LocationListener interface as follows:

```
public class MyLocationListener implements LocationListener {
    @Override
    public void onLocationChanged(Location location) {
        Log.d("g53mdp", location.getLatitude() + " " + location.getLongitude());
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        // information about the signal, i.e. number of satellites
        Log.d("g53mdp", "onStatusChanged: " + provider + " " + status);
    }

    @Override
    public void onProviderEnabled(String provider) {
        // the user enabled (for example) the GPS
        Log.d("g53mdp", "onProviderEnabled: " + provider);

    }

    @Override
    public void onProviderDisabled(String provider) {
        // the user disabled (for example) the GPS
        Log.d("g53mdp", "onProviderDisabled: " + provider);

    }
}
```

onProviderEnabled and onProviderDisabled methods are called when the user enables or disables the GPS, and onStatusChanged gives information about the status of the GPS signal:

https://developer.android.com/reference/android/location/LocationListener.html

The important method call is onLocationChanged, which reports the current location as it is measured, and provides a Location object that can be inspected to obtain WGS 84 latitude, longitude, altitude (elevation), reported accuracy of the signal etc.

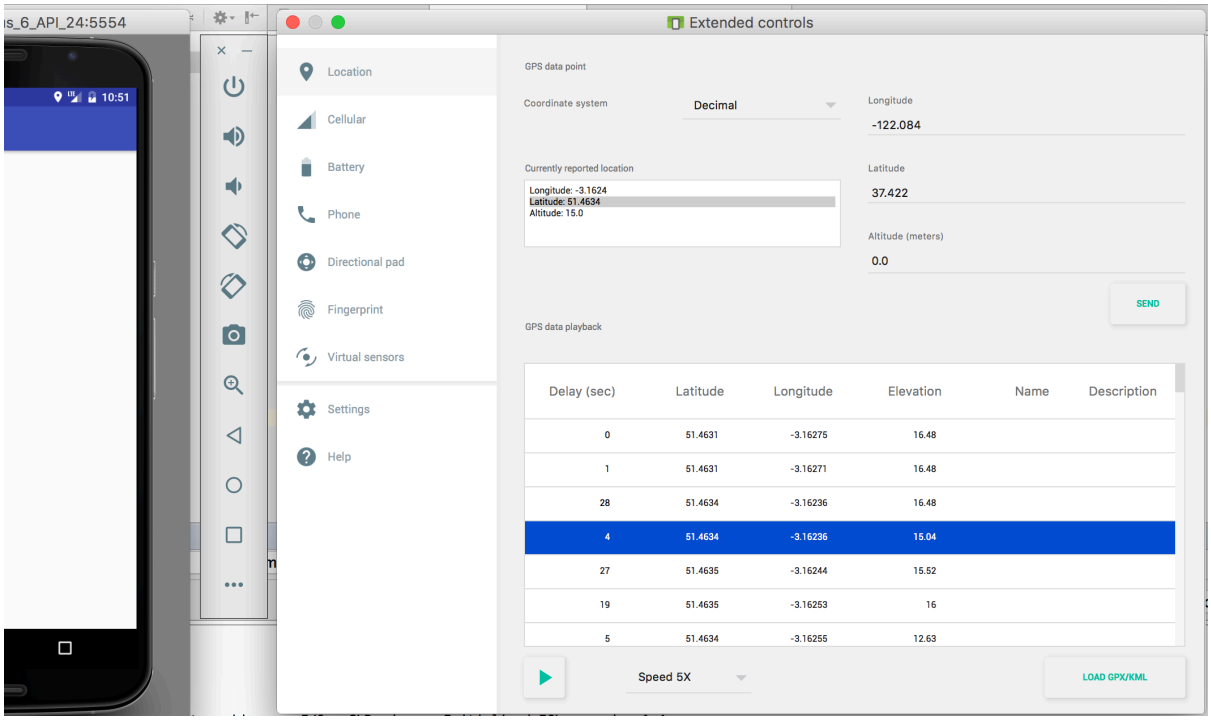
https://developer.android.com/reference/android/location/Location.html

Note that geodesy and global positioning are incredibly complicated subjects in their own right - the Earth is in no way perfectly spherical, and we like to think of linear distances on a locally flat surface as opposed to degrees around the world – however the Location class hides most of this from us. In particular the distanceTo method will calculate the distance between two points given as latitude and longitude:

```
    float distance = myLocation.distanceTo(someOtherLocation);
```

Emulating GPS

It is possible to complete this coursework entirely using the emulator – there is no advantage to or necessity of having a physical Android phone. There is also no expectation that you handle the everyday practical details of GPS – losing signal, inaccurate signals etc. You can assume that it will be tested on an emulated device with "perfect" GPS.



The emulator provides a mock GPS device that feeds NMEA (latitude and longitude position updates) to the phone where they will be handled by the LocationManager as if they were real updates, via the *extended controls* menu. This can be found by clicking "…" on the emulator side bar.

Furthermore, the emulator can replay a series of GPS events from a GPX file (a standard log format for many GPS devices and applications). It is also possible to export from Google Maps to GPX.

Three example GPX files have been uploaded to Moodle for use as "real" latitude and longitude positions that can be played out. These include two of runs around Wollaton park, and also a log of Alan Dix (a famous HCI professor) walking around part of Wales.