U08971 Advanced Mobile Software Development   SEP 2017

Coursework 1

Revision History

2017-10-10: Preview version released to enable students to request clarifications.

2017-10-17: Clarifications added (in bold) and specification is now definitive

Value

50%

Mode

Individual work

Deadline

See Moodle

Learning outcomes

This coursework is designed to test your attainment of the following learning outcomes:

- Create apps to take advantage of the relationship between the web and mobile software.
- Design and implement software to take advantage of the range of communication opportunities available on a mobile device.
- Research and exploit API classes that enable mobile devices to use network protocols.

Introduction

For this assignment you will write an Android app that helps people meet up in London. The app will use SMS messages to allow users to communicate their location to friends. The app will provide information to the user graphically through the use of the Google Maps API. It will also use the Transport for London Unified API to obtain data about the London public transport network. The appendix at the end of this document explains where to find information about the TfL Unified API.

Functional requirements

FR1	The application should be capable of displaying the user's (real or simulated) location on a map. There is a note later in the document describing acceptable ways of simulating location. **The application must be capable of working with a simulated location, but the use of a real location is optional.**
FR2	The application should be capable adding markers to the map that indicate the locations of tube stations, bus stops, and bike points that meet criteria set by the user in the settings screen (see FR4).
FR3	Clicking on one of the markers indicating a tube station or bus stop should result in a list of departures from that stop being displayed. Information about departures should include the times of trains/buses and their destinations. Clicking on a marker for a bike point should display the number of available bicycles.
FR4	The app should have settings screen that allows the user to specify the criteria used to decide which of the locations mentioned in FR2 are displayed. The settings screen should allow the user to set the following options: • Whether stop points of each type (e.g. bus stop, tube station, bike point) get displayed or not. • A limiting distance, such that markers are only added if the distance between the stop point and the user's current location is less than or equal to this distance.
FR5	The app should allow the user to maintain and add to a list of "friends" by exchanging SMS messages. You are free to decide for yourself the exact mechanism by which this happens. However you must make sure that: • The friend relationship is symmetric in the sense that, if some person *P* is in the list of friends maintained by the copy of the app on your device, then you are in the list of friends maintained by *P* on his/her device. • Two users can only be friends if one user has made a request to be added as a friend, and the other user has accepted this request. • The mechanism by which friends get added relies solely on SMS messaging. For example the app might interpret SMS messages of a particular form as friend requests, and others as acceptance of these requests. • The app should send the SMS messages itself, rather than relying on any other telephony app installed on the device.
FR6	The app should allow the user to send an SMS message that indicates his/her current location (real or simulated), and which will be sent to all the numbers on his/her

	friend list. On receipt of such a request from a friend the app should add a marker to the map indicating the specified location, and should remove any previous marker indicating the sender's location. Clicking on this marker should allow the user to see which friend sent the location update, and when. **The application must be capable of working with a simulated locations, but the use of a real location is optional.**
FR7	The app should allow the user to send an SMS message that indicates the location of a meeting place, and which will be sent to all the numbers on his/her friend list. On receipt of such a request from a friend the app should add a marker to the map indicating the specified location, and should remove any previous marker indicating any previous meeting place. You can assume that everyone who receives this message will be happy to have any previous destination overwritten, there is no need for the app to handle negotiations about where people might want to meet up!
FR8	The list of friends should be persistent. In other words you can switch the device on and on again without losing it.
FR9	**There must be some means by which the user can enter a simulated location, as a latitude and longitude (to be used for the purposes of FR1 and FR6). The app should not rely on being able to use the emulator controls to simulate location.**

Non-functional Requirements

NFR1	Your Java code should adhere to the following, standard, Java naming conventions. Method and variable names (except for static final variables) should begin with a lower case letter. Class names should begin with an upper case letter. Neither class names, method names, nor variable names (except for constants) should contain underscores. If a variable is being used to indicate the value of a constant, then its name should be entirely in upper case and can include underscores. You can choose what conventions to use when writing XML code, but your choice should be consistent. For example, if some identifiers are written using all lower case words separated by underscores, then all identifiers should follow this format.
NFR2	Maps should be displayed using the Google Maps Android API.
NFR3	All the data about the London transport network that is used by the application should be acquired by the application itself using the Transport for London Unified API. For example you must ***not*** write a separate application that acquires this information and then stores it in a database to be used by the game application.
NFR4	All of the code you write should run on the user's mobile device, and should not present information as a web page.
NFR5	**You must not make use of "third party" libraries. In this instance "third party" means anything that is not described in the Android reference at https://developer.android.com/reference/classes.html** **Contact the module leader if you are in any doubt about the acceptability of any class.**

Simulation of Location

If the application being created was intended for "real world" use, then the markers displayed would represent actual locations of physical devices. However this would present problems when you need to be demonstrate your app, and when we need to assess it, because we won't be in London and we won't be able to move around very much. Unfortunately the simulation of location on emulators can be a bit fiddly, so **your app must provide the user with a simple way to simulate the by entering a latitude and longitude**. For example you could allow the user to specify a location using the settings screen, and have the app take this as the location to use when sending updates. **The app itself must provide a means for the user to enter a simulated location. You must not rely on the emulator controls.**

What you have to submit (*.doc* and *.zip*)

You will make two submissions. The first will be a progress report, and the second will be the finished coursework. You will also be required to give a demonstration.

Progress Report

You should submit this by the deadline set on Moodle. It should contain whatever executable code you have written; and a short report explaining what you have done so far, why you have done that (rather than something else), and what you plan to do in order to complete the work. The report does not have to be long (1,000 words should be sufficient) and it does not have to be formal, but it does have to be clear and readable.

If we feel it is necessary, we will ask you to give us a face to face demonstration of your work at a mutually convenient time. If we haven't asked for a face to face demonstration, but you feel you would like to give one, then we can arrange that as well.

We do not expect that your executable code will be a complete solution to the problem. You should, however, have attempted to solve parts of the problem, and it would make sense to start with the parts that you think will be most difficult. It is unlikely that a submission that does not include any executable code will get good marks.

You will be given a mark out of 100% for the progress report. This mark will act as a cap on the mark you get for your final submission. For example, if we award you 60% at this stage then you can get a maximum of 60% of the marks for the final report (i.e. a maximum of 36 marks). Anyone who gets a mark of less than 100% will get the opportunity to do further work and resubmit their progress report. Please note that a mark of 100% does not mean that your submission was perfect. It just means that we cannot see any reason why you would not be able to successfully complete the coursework, given the progress that you have made to date.

Final Submission

For your final submission you should submit the following **three** files

1. A Word, PDF, Open Office, or Libre Office document containing:
   a) A screenshot of your user interface in its initial state, before any user interaction has taken place, including the location of the user.
   b) Screenshots indicating the mechanism by which the friend list is maintained.
   c) Screenshots indicating the mechanism by which the user sends updates of his location, or of the meeting point, to the people on friend list. These should demonstrate the appearance of a marker on the device(s) to which the location update is sent
   d) A *reflection* on the degree of success you have achieved in this work. In this section you **must** specify which of the functional requirements you have met, and the ones which you have **not** met.
2. A zip file containing your complete Android Studio project. Note that this must include all of the Java and XML sources for your application, and the .apk file for the application. Please ensure that the file is in .zip format (for example .rar and .7zip files should not be submitted).
3. The .apk file as a separate file. This should be the same as the .apk file in the zipped file mentioned above (the reason why we want you to submit it again as a separate file is so that we can be absolutely sure that it has been submitted).

Demonstration

You will be required to demonstrate your application in the week following the final submission.

Mark Scheme

Note that all marks given for meeting functional requirements are dependent on the application also meeting the non-functional requirements**, and on meeting FR9.**

For meeting FR1	6 marks
For meeting FR2	4 marks
For meeting FR3	3 marks
For meeting FR4	1 mark
For meeting FR5	6 marks
For meeting FR6	3 marks
For meeting FR7	2 marks
For meeting FR8	2 marks
For the quality of your code. Note that it is not possible to completely define what "good code" consists of. However the programing style guidelines given to you took Further Object Oriented Programming (U08026) will for the most part still apply.	15 marks
For the usability and attractiveness of the UI. Note that it is not possible to give a complete description of the features that make a UI usable or attractive. However you should take care to make good use of screen "real estate", for example avoiding the use of buttons if they would reduce the amount of screen space available for other information, and you should try to design interfaces that offer "affordances" that is to say that their elements allow the user to guess how they should be used.	4 marks
For the reflection on how successful you were. Note that you must specify which functional requirements you have met in this section. You must also be clear about which requirements you have **not** met.	4 marks
Total	50

Appendix: The Transport for London Unified API

The TfL Unified API gives access to web services that provide information about all aspects of London's transport infrastructure. Documentation can be found on the TfL website. In order to familiarise yourselves with the API, I would suggest that you start by reading the following blog posts.

https://blog.tfl.gov.uk/2015/10/01/tfl-unified-api-part-1-introduction/

https://blog.tfl.gov.uk/2015/10/08/unified-api-part-2-lot-location-of-things/

https://blog.tfl.gov.uk/2015/10/19/unified-api-part-3-rot-routes-of-things/

https://blog.tfl.gov.uk/2015/11/18/unified-api-part-4-roads-data/

https://blog.tfl.gov.uk/2015/12/07/unified-api-part-5-aot-arrivals-of-things/

and then have a look at the documentation available at

https://api-portal.tfl.gov.uk/docs

I very strongly suggest that you explore the API using a tool such as Poster, Postman, or the Chrome Advanced REST Client.