The iLogger Application

1. Introduction

A lot of data can be collected via mobile devices today. In particular, while at one spot (or Point-of-Interest, or POI, in short), one can collect data about surrounding devices (via Bluetooh scans), data about surrounding WiFi Hotspots (via WiFi scans), take photos and perhaps even record audio or video. One can also manually enter data about a place one is at (e.g., saying this is a beautiful spot). Such data can be geotagged and then stored as a log or record (or diary) about locations one has visited. Such data can also be aggregated and analysed, e.g., to find spots with the most number of WiFi Hotspots. The data can also be visualized, for example, on a map showing where certain WiFi Hotspots are found, a map showing where the noisy points are at a given time and so on. While an individual can collect such data for personal use. The data can also be used by other mobile apps on the same device. One can also share such data on a Web site which can be aggregated and visualised as information maps that can be used by friends or other people.

2. Aim

The aim of this assignment is to design, document and develop an Android mobile app, which we call **iLogger**, which enables its user to collect information and to create logs about spots/places s/he has been to. A range of information should be collected.

3. Your Tasks and Marking Scheme

Your overall task is to design and implement the iLogger mobile app which can be used to collect information about Points of Interest.

When a user is standing at a certain spot or at a certain point (as identified by a pair of GPS coordinates), the user can use the app to initiate collection of a range of information at that place.

For example, the following types of information about a place can be collected via the app:

- a record of the current location/point (i.e., its GPS coordinates),
- some information about a place can be collected using the Google Places API (https://developers.google.com/places/); given a pair of GPS coordinates, possible places a person can be add can be given via this API,
- a set of WiFi access points that can be seen from that point (via a WiFi scan); this can vary since it depends on what access points are available at that point at the time of scanning,
- a set of Bluetooth devices that can be seen from that point (via a Bluetooth scan); this can vary over time since it depends on what devices are actually nearby at the time of scanning,
- pictures taken from that point,
- sound of the environment at that point (i.e., recording a short audio clip), and
- textual comments on the place, entered manually by the user.

The app should also be able to make use of screen estate if viewed on a tablet, compared to viewing on a phone, i.e., it should automatically adapt its layout according to the orientation of the screen and screen size.

Some data may already have been collected a short time ago and the user should be given the choice to reuse cached data or to collect data.

The user might want to view the data collected in the past, within certain periods or at certain points, or a particular type of data, visualised on a map.

An overview of the tasks are as follows:

PART 1 - ALL THESE Coding TASKS MUST BE ATTEMPTED FIRST BEFORE PART 2 (for a maximum of 30 marks):

TASK 1 (SKELETON APP WITH LAYOUT ADAPTABLE TO MULTIPLE SCREEN SIZES AND ORIENTATION) [10 marks]: [Coding]
TASK 2 (RECORD AND VIEW LOCATIONS) [10 marks]: [Coding]
TASK 3 (RECORD AND VIEW PLACE INFORMATION) [10 marks]: [Coding]

PART 2 - CHOOSE 6 OUT OF THE 7 Coding TASKS BELOW, BUT NOTE THAT TASK 8 DEPENDS ON TASK 3 (for a maximum of 60 marks):

TASK 4 (RECORD AND VIEW WIFI SCANS) [10 marks]: [Coding]
TASK 5 (RECORD AND VIEW BLUETOOTH SCANS) [10 marks]: [Coding]
TASK 6 (ENABLE PICTURE TAKING AND VIEWING OF GEO-TAGGED PICTURES) [10 marks]: [Coding]
TASK 7 (ENABLE AUDIO RECORDING AND LISTENING TO PAST GEO-TAGGED RECORDINGS) [10 marks]: [Coding]
TASK 8 (CACHING) [10 marks]: [Coding]
TASK 9 (RECORD INDOOR POSITIONS) [10 marks]: [Coding]
TASK 10 (INFORMATION SHARING) [10 marks]: [Coding]

PART 3 - REQUIRED

DOCUMENTATION TASK [10 marks]: [Documentation: 4 to 6 pages]

BONUS PART - TO BE DONE ONLY AFTER PART 1 AND PART 2 OF THE ABOVE HAVE BEEN COMPLETED

BONUS TASK [7 marks]

Note that the above adds up to 107 marks so that successfully doing all tasks (INCLUDING BONUS) can lead to 107/100 marks!

NOTE: You are encouraged to use multithreading (in one or more of the forms such as the Thread class, AsyncTask and/or IntentService) and the Android component types of Activity, Service, BroadcastReceiver and ContentProvider as much as possible in implementing the functionalities as appropriate.

STATE ANY ASSUMPTIONS YOU MAKE - WHERE DETAILS ARE NOT SPECIFIED, OR THE ASSIGNMENT IS UNDERSPECIFIED, YOU ARE FREE TO DESIGN THAT ASPECT OF THE APP AS YOU SEE FIT.

You must attempt TASK 1, TASK 2, and TASK 3, and the DOCUMENTATION TASK. You are free to attempt the other tasks in any

order or implement any subset of the tasks, BUT note that TASK 8 is dependent on TASK 3.

Note that all the tasks must be integrated into the same app - there should only be one Android app.

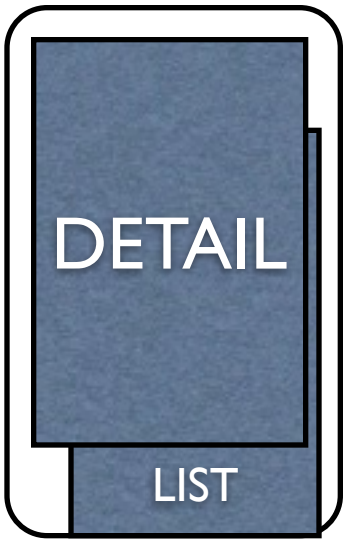
The details of the tasks are as follows.

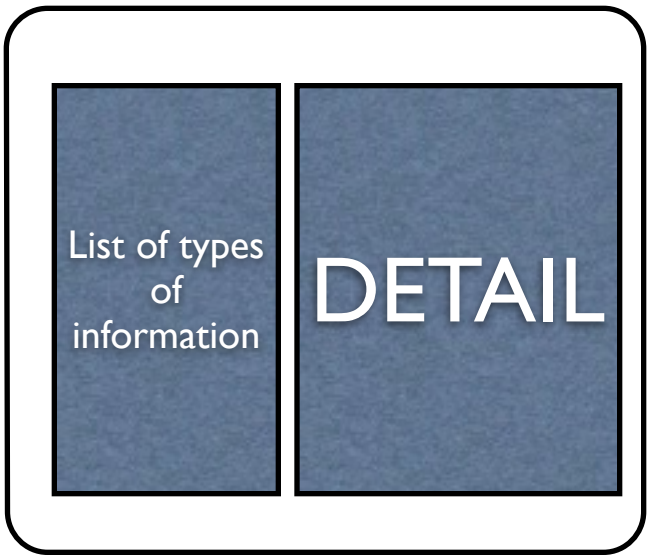
PART 1 (attempt all 3 tasks)

TASK 1 (SKELETON APP WITH LAYOUT ADAPTABLE TO MULTIPLE SCREEN SIZES AND ORIENTATION) [10 marks]: [Coding]

iLogger should have a user interface that supports multiple device sizes including a smartphone (e.g., Nexus 6 size) and a tablet (e.g., Nexus 9 size) (landscape and portrait orientations). Your app should have (at least) two layouts, one layout for the smartphone and portrait orientation of a tablet, and another layout for the landscape orientation on a tablet. You can use the the Master/Detail Flow template (available in Android Studio when you create a New Project).

The list view contains a list of types of information the app can collect and the detail view could contain the interface related to specifying parameters for collecting information and/or contain the interface for displaying or visualising the information on a map.





An example of what the detail might contain when it is used to visualise geo-tagged photos taken is shown here.

Note that in this task, it is enough to set up the list of types of information to be collected, and the adaptive layout, leaving the DETAIL blank.

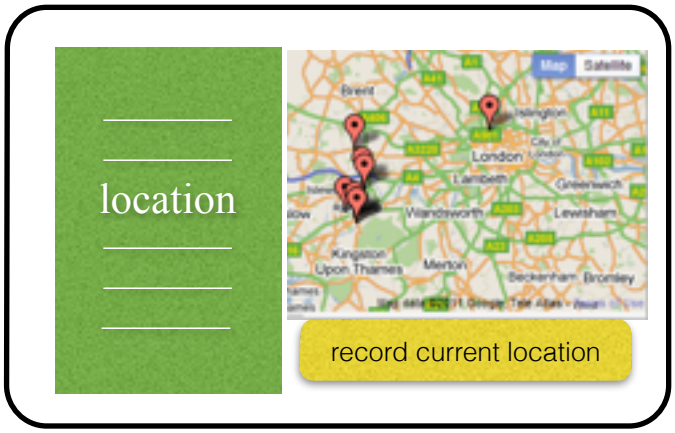
TASK 2 (RECORD AND VIEW LOCATIONS) [10 marks]: [Coding]

This task implements functionality to collect one type of information regarding the point where the user is, i.e., to collect the GPS location of the user and store it in a database. The user should be able to select this type of information from the list of types of information, and then

(i)click a button to record the current location, or

(ii)view previously recorded locations on a map.

The current or last recorded location should have a marker with the label ("last recorded location"). The colour of the last recorded location should be different from the other locations on the map. A sketch is given below but you are free to design the layout differently and have other features as you see fit.



TASK 3 (VIEW PLACE INFORMATION) [10 marks]: [Coding]

You should have a functionality to get place information using the Google API. When at a point/location (with given GPS coordinates) and reverse geocoding, it is possible to have several places that match a given set of GPS coordinates. The task is to retrieve one or more places (or place names/descriptions) corresponding to the current point/location of the user and show that to the user.

[Hint:
You can use APIS such as
https://developer.android.com/training/location/display-address.html
and/or use
https://developers.google.com/places/android/current-place
]

PART 2 (attempt any 6 tasks)

TASK 4 (RECORD AND VIEW WIFI SCANS) [10 marks]: [Coding]
This task is to allow the user to do a scan of the WiFi hotspots at the current location where the user is and to store results of the scans (i.e., the findings), and to retrieve and view that information later. Note that you also need to store the location where you do the scan, i.e. each scan being done at a location will be geotagged and timestamped.

TASK 5 (RECORD AND VIEW BLUETOOTH SCANS) [10 marks]: [Coding]
This task is to allow the user to do a scan of the Bluetooth devices in the surroundings at the current location where the user is and to store results of the scans (i.e., the findings), and to retrieve and view that information later. Note that you also need to store the location where you do the scan, i.e. each scan being done at a location will be geotagged and timestamped.

TASK 6 (ENABLE PICTURE TAKING AND VIEWING OF PICTURES) [10 marks]: [Coding]
This task will enable the user to take a photo and to store it, together with a textual description entered by the user and the location where the photo was taken, and later to retrieve that location or view them on a map. Note that the user might choose to take several photos at the same location. Since the photo and its location are stored, the photo is geotagged and timestamped.

TASK 7 (ENABLE AUDIO RECORDING AND LISTENING TO PAST RECORDINGS) [10 marks]: [Coding]
This task will enable the user to record a short audio clip and to store it, together with a textual description entered by the user and the location where the recording was done, and later to retrieve that audio clip for listening. The audio clip should be geotagged and timestamped.

TASK 8 (CACHING) [10 marks]: [Coding]
Note that TASK 3 involves retrieving place information over the Internet. Whenever there is no network connection or to save bandwidth/time, the app should be able to retrieve place information from a cache. Design and implement a cache that will store place information for a given location (i.e., for a given set of GPS coordinates), so that the user can choose to use the place information from the cache first (if it exists) rather than send a query over the Internet.

[Hint: since GPS coordinates are precise up to a range of decimal points, you can define a boundary (with a fix radius) around a given point beyond which you will query the Internet, that is, suppose the history contains the following two points:
Location: (x1,y1); Place: Melbourne Museum
Loation: (x2,y2); Place: La Trobe University
Then, for some fixed r you set, if the user is currently at (a,b), then
distance((a,b),(x1,y1)) < r means the user is at Melbourne Museum, or if
distance((a,b),(x2,y2)) < r means the user is at La Trobe University.
You can use Euclidean distance as your distance measure.
]

TASK 9 (RECORD INDOOR POSITIONS) [10 marks]: [Coding]
GPS currently may not work well indoors. Add a feature on iLogger to record indoor locations, instead of using GPS.
[Note: this feature relies on having indoor maps and an indoor positioning technology; for this assignment, you can draw up your own maps and need only show three locations that your application can distinguish (e.g., PW218, BG139, BG117, or any choice of three indoor locations on the LTU Bundoora campus you choose) - use either triangulation, signal strength measurements, and/or fingerprinting using WiFi signals to determine indoor locations.]

TASK 10 (INFORMATION SHARING) [10 marks]: [Coding]
Add a feature to allow the user to share selected information about its history of recorded places and associated information to a server or to friends via Bluetooth or WiFiDirect.
[Note: You need only to implement one way of sharing, ONE of (i) uploading via the Internet to a remote server, (ii) via Bluetooth to another mobile device OR (iii) via WiFi-Direct to another mobile device, not all three.
If you choose the server option, this involves uploading data to a Web server - you are free to choose any Web hosting for the Webserver or to use the university's server but you need to write PHP scripts to receive and store the uploaded data; only choose this option if you are familiar with PHP.]

PART 3

DOCUMENTATION TASK [10 = 4(i)+3(ii)+3(iii) marks]: [Documentation: 4 to 6 pages] The documentation describing the system should include:
(i) a detailed wireframe diagram with explanation describing the structure of the mobile client GUI (showing the main screenshots and how they are related), for the iLogger mobile application, and
(ii) describe how multithreading is used in your application, including for which functionalities and explain why it was used,
(iii) a description of how context-awareness (beyond only location-awareness) might be useful in extending this application. For example, what other types of context information can be used? How can this context information be acquired and processed? (Sketch this and mention what other sensors can be used, etc) [Relate to material from lectures as much as possible.]
As far as possible, provide rationale for your design – that is, provide reasons for why it was designed that way.

BONUS PART

BONUS TASK, to be attempted only after all the above parts have been completed [7 marks] (for a possible total of 107/100!)

(i) [4 marks] Use the Activity Recognition API [http://developer.android.com/training/location/activity-recognition.html] to determine if the user is really on foot, and disallow tracking at the points where the app detects that the user is no longer on foot, and allow when the user is back on foot (e.g., the user walks, takes a bus and then gets off the bus and continues walking) – the app only allows recording when the user is walking or on foot.

(ii) [3 marks] Add a feature where the user can mark a given point as "hot". Then use the geofencing API [http://developer.android.com/training/location/geofencing.html] to add geofencing around the hot points so that the user is notified when s/he is within 100m of any one of the hot points.

S.W. Loke, 2015, La Trobe University.

END OF DOCUMENT