The University Of Sheffield.

COM4510/COM6510 Software Development for Mobile Devices

Assignment 2019-2020

This assignment is primarily concerned with applying the ideas that are being presented in the module on methods and technologies for Mobile Computing. Some of the basic algorithms needed for it have already been introduced during the module, and can be obtained from MOLE.

Organisation of the Assignment

The assignment will require just one submission and it will be worth 100% of the marks. See MOLE for the exact deadline. All submissions are electronic via MOLE.

Workload and Groups

The whole assignment is intended to account for between 20 and 30 hours of each person's work towards the module as a whole. Experience has demonstrated that it is not really possible for one person to do all the work for the assignment in that time, unless they are an exceptionally competent programmer with previous knowledge of mobile programming. Therefore, the assignment is organised on the basis that people should work on it in **groups**. Groups must be composed of a **maximum of 3 members.** Students are allowed to do the assignment in pairs or on their own.

Please note!
You must register your group at https://tinyurl.com/y9eagx7f

Given a group, you will normally be expected to keep with that group for the whole of the assignment. However, if any problem arises that makes this difficult, you should notify the lecturer **immediately**, so that appropriate action can be taken. See lecture 1 slides on this topic. **If the group splits after week 7**, its members are not allowed to join another group.
It is important that you make clear what contribution each group member has made in your final report. It is required that each person contributes to each stage of the assignment, but it is up to each group to decide how to divide the work up between individuals.

Deadlines

The deadline is **absolutely fixed**. You should therefore plan your work to aim at handing the report in at least a few days before the deadline – do not leave it until the deadline, just in case any minor thing goes wrong and you then find that you are late.
Note that the Computer Science department applies fairly severe penalties for handing coursework in late. If you want to look at the details you can find them on the University's web site.

Material Provided

Lecture notes, lab class examples and websites/books as detailed on the lecture notes.
NOTE: no third party code can be used in the assignment, except the code what has been **explicitly** provided in the lectures. For example you are allowed to reuse the code given in the lecture/lab slides and any library (e.g. EasyImage) used in the lab classes. However you are not

allowed to download any code from the Web or to use any other software that will perform a considerable part of the assignment. **Unauthorised re-use of third party software will be considered plagiarism.** In case of doubt ask the lecturer for permission before using any third party code. In general, we will allow only the use of generic libraries designed to improve the look and feel of any interface. However permission must be requested before use.

1. Scenario

The learning objectives of the assignment are to learn:

- to build an app with a flexible sophisticated layout
- to use separation of concerns (using MVVM)
- to cope with multimedia data
- to store data locally using abstractions of databases (Rooms)
- to use the phone's sensors (GPS, barometric pressure, etc.)
- to use background services
- to work as a group

The field chosen is management of photos.
The solution will be composed of the following parts, each covering each of the above learning objectives.

1.1. The problem

Design, build and evaluate an application for taking personal photos on a mobile and to organise them along geolocated visits.
The app will allow to:

1. Taking pictures and allowing pictures to be uploaded to the map (10% of marks)
2. Recording a trajectory (visit) using continuous geolocation and sensor data recording over a fixed period of time; during a visit it must be possible to take pictures that are then associated to the path and its sensor readings (20% of marks)
3. Visually browsing previews of photos taken or uploaded to the app (30% of implementation marks) using a number of strategies
4. Showing details of a photo including its location on a map and the details of the path the picture is part of (20% of implementation marks)
5. Saving the data to local Room database (20%)

These steps are detailed in the next sections. Examples are provided to help understand the requirements. You are not required to implement that exact solution. You can use your creativity, as long as the formal requirements are met.

Requirements for the solution are:

- It must be fully functional and robust
- It must work on multiple devices with different screen size, processing power and screen resolution
- It must work at least for Android>6.0
- It must be efficient, able to cope with a library of thousands of photos
- The interface must be pleasant to the eye
- The interface must work both in portrait and landscape mode
- The interface must follow the typical design patterns of Android and in particular **MVVM**
- The solution must be of high quality. A simplistic solution (although functional) will not attract many marks.

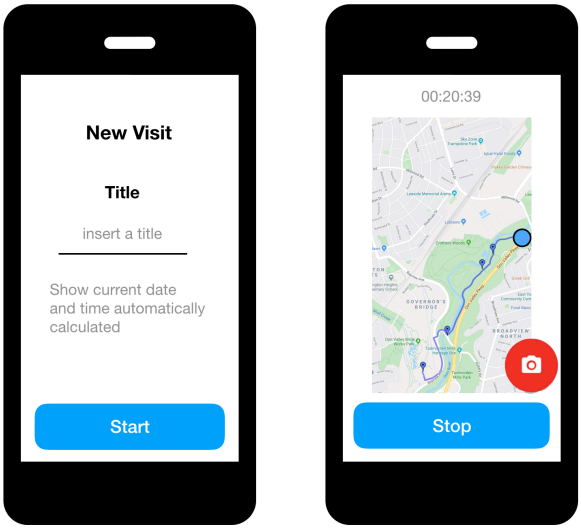
All implementation must be done either in Java (for Android phones). No other languages are allowed (e.g. Cordova, React Native, etc.). If interested in using Kotlin, let the lecturer know in advance.

1.1.1. Taking and uploading Pictures

The app must allow taking pictures using the camera. This functionality must be working both when used on a real device and on the emulator. If the phone model does not have a camera the functionality must not be available to the user. The new photo must become available for retrieval to the app.
The user must also be able to upload a new picture from the gallery.

1.1.2. Capturing a visit

The app should allow capturing a geolocated visiting path, i.e. an activity over a limited period of time when the app tracks location, temperature and barometric pressure at regular intervals (20 seconds). The activity must be started and stopped explicitly by the user. Date and time must be captured at the start of the visit (click of the start button). A title must be associated to the visit. Example of interface.



While on a track, the user can take pictures and these are associated to the path. While tracking is active, the app should show a map showing (i) the geolocated path taken so far, (ii) the location of the pictures taken so far and (iii) the current position. The map can either be updated every time a new location is received (better solution) or just be refresh at the push of a button (basic solution).

The tracking should be implemented as a service tracking geolocation (gps coordinates), temperature and barometric pressure every 20 seconds. The service will have to work even if the screen is off and or the app is swiped out from the list of recent apps.

1.1.3. Persisting data in a local database

All the data captured during the visit (title, date, description, GPS coordinates, sensor data, etc.) must be saved in a local database implemented using Room (i.e. not directly using SQLite), so that it can be retrieved at a later stage.
The database must allow searching of images and visits as explained below. Note that you must implement an aSync process retrieval, as accessing a database on the UI Thread is not allowed.

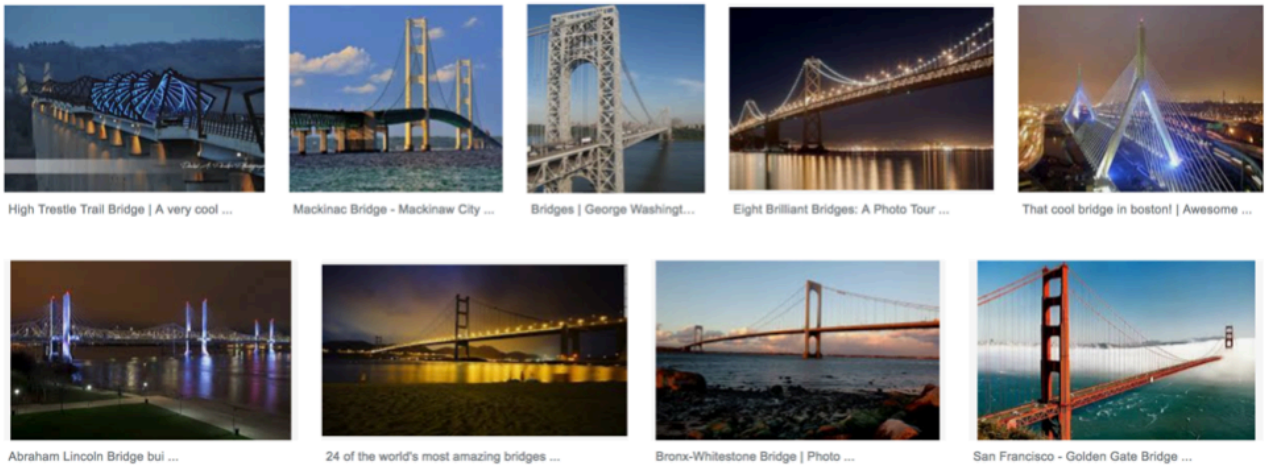
1.1.4. Visually browse previews of photos

The user must be able to browse all the pictures taken or uploaded to the app. In doing so you must design and implement an original program that:

- Allows visualising previews of the pictures
- Allows selecting a picture for further detailed inspection

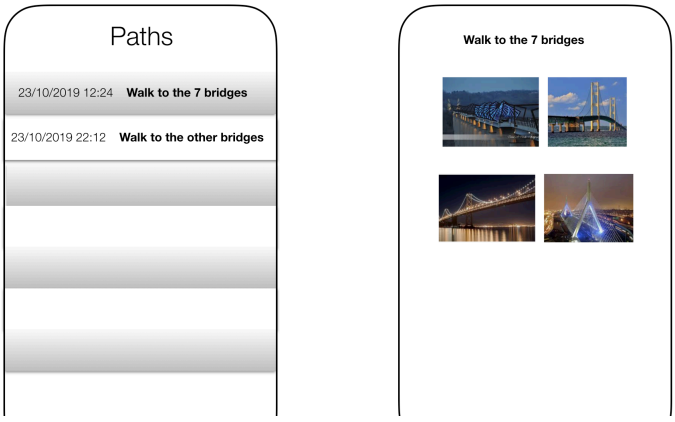
It is important that the interface is efficient and able to cope with a library of thousands of photos. We expect to be able to access in different ways, i.e.:

- A grid sorted by date in ascending order: example:



- Sorted by path, e.g.

Paths must also be browsable via a list, e.g.:



As mentioned you are not required to implement this exact solution. You can opt for a different

layout (e.g. all pictures may have the same size, etc.), as long as the same functionalities are provided by the app.

1.1.5. Inspecting the details of a photo

Tapping on a photo in the browsing interface should allow inspecting the details of that specific photo by tapping on it. In this case the photo must be shown together with the following data:

- A larger version of the photo (if clicked it should sow the full size photo)
- Path title
- Last available sensor readings at time of taking the picture (barometric pressure and temperature recorded from the phone's sensors)
- Location on map (in red in the example) with full path displaying trhe location of other pics taken on the same path (in blue in the example)

An example:

As mentioned you are free to implement as you deem fit. Make sure that the solution is intuitive for the user.

2. Marking schema

Each part described in the subsections above will carry marks divided as follows:

50% for the quality of the solution, inclusive of separation of concerns, use of async processes, quality of the user interface, etc. as well as compiling and running without an issue
35% for the quality of the documentation
15% for the correctness of results.

Please note

- the direct consequence of the marking schema is that providing a program returning the correct solution is not enough to get a pass mark. You will need to implement the correct strategies and document/discuss them properly! Quality of documentation and code are very important issues for computer scientists.
- Solutions not working on the departmental computers (e.g. working only on personal

computers) will not be considered.

3. Division of work

It is important all team members provides equally to the solution. It is also important to define in advance what each member will contribute. This will allow to provide individual marks.
We expect the following division of work to be implemented. Please note that each member must support the others. The individual mark will be computed as 30% based on group performance and 70% for individual performance. We may vary this balance in special cases.

Groups of 3 members:

- Member 1
    a. Implementation of picture taking and photo uploading (1.1.1)
    b. Room database implementation (1.1.3)
    c. MVVM and live data framework implementation
- Member 2
    a. Implementing the visit tracker (background process) and sensor tracking including the interface (1.1.2)
- Member 3
    a. Implementation of details of the photo (1.1.5)
    b. Implementation of Browsing (1.1.4)

Groups of 2 members:

- Member 1
    a. Implementation of picture taking and photo uploading (1.1.1)
    b. Room database implementation (1.1.3)
    c. MVVM and live data framework implementation
    d. Implementation of Browsing (1.1.4)
- Member 2
    a. Implementing the visit tracker (background process) and sensor tracking (1.1.2)
    b. Implementation of details of the photo (1.1.5)

Groups of 1 member

- There is no special arrangement for these groups. The group is expected to do the entire assignment as any other group.

4.Handing in

Your solution must be contained in a self-contained directory called *COM4510 or COM6510* (<***MainDirectory***> in the following) compressed into a zip file submitted through MOLE.
The directory must contain:

1. The source code of the solution (please note that we will both inspect and run the code) contained in the directory **<*MainDirectory*>/code/**.
   All the code must be in the exact format to be run via AndroidStudio
    (a) All code must be **developed in Java**. We must be able to run your solution without problems on a standard lab machine. Please note that the quality of the code carries a relevant portion of marks, so be sure to write it properly.
    (b) All the external libraries must be included in your solution. Maven or Gradle links are acceptable. Libraries are allowed only if previously agreed with the lecturers. No library doing a substantial part of the assignment are allowed. Always ask in writing before using any library.

(c) The documentation in the Java files must be of very high quality. Please note that this documentation carries a relevant portion of marks, so be sure to write it properly. More information on guidelines for Java (JavaDoc) see http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

2. Screenshots of the different app screens showing the implemented functionalities **<*MainDirectory*>/screenshots**
3. The filled self-assessment form documenting the solution. Please note in the form you are required to assess the quality of your solution using a number between 0 (not implemented) and 5 (full requirements met) AND to describe in details how you have met the requirements, e.g. how you have implemented a flexible sophisticated layout, how you have used separation of concerns (using MVVM), how you have stored data locally (using Rooms), why the Room schema is appropriate to the problem, how you have implemented the tracking of the visit (background service), why your solution is efficient, etc
4. A formal declaration of the contribution given by each group member. See below for the expected division of work that has to be followed

4.1. How to submit

Everything must be submitted electronically. Nothing is to be handed in at the reception! **Use MOLE**. Store your solution in a .ZIP file that when unzipped will generate the directory organisation described above. As emergency measure (and only in that case!), if any last minute issue should arise in handing in electronically, please send your solution by email to the lecturer (cc to demonstrators) in a self contained .ZIP file.

4.2. Anti-cheat measures

Please note that measures are in place for detecting plagiarism and in general cheating.

5. Queries about this assignment?

Should you have any queries about the assignment, feel free to contact
Fabio Ciravegna AND Po Yang: {f.ciravegna, po.yang}@ shef.ac.uk