Assignment #02

IT University of Copenhagen (ITU)
Mobile App Development, BSc (MOAPD)
(Spring 2021)

Deadline: May 07, 2021 at 13:59

Purpose The mandatory assignment (Assignment #02) aims to make a complete version of the GoCaching app, including some of the functionality you would find in a "real" Android application for a treasure hunting game.

GoCaching app works in an environment where players look for treasures (*geocaches*) using geolocation data. The geocaches are around the city and the player must find them in a hidden place and read the QR Code under the geocache. Every time the player finds a new geocache, the application will give her/him a score based on the geocache category (easy/medium/difficult).

For Assignment #02, you should design and implement a user interface, and functionality to register geocaches and manage an account associated with each player. You are free to design the user interface as you like. It may be completely different from the user interface, we have used in the first weeks of the course.

Functionality and User Interface The cornerstone of the GoCaching app is a database containing information about some geocaches and players.

- The (*minimal*) information related to a geocache is (i) ID, (ii) geocache owner, (iii) name of geocache, (iv) longitude and latitude, (v) geocache category, and (vi) picture of the geocache. You may extend this.
- The (*minimal*) information related to the player is (i) ID, (ii) full name, (iii) e-mail, (iv) list of found geocaches. You may extend this.

Your solution must add geocaches, read the geocache QR Code, and calculate the player's score. You also must implement an administration area where you can approve the registration of new geocaches (*mandatory*), block a specific user (*optional*), activate and deactivate a geocache (*optional*), among others. You can implement any new idea in the administration area that you will have during the development of your project.

Your app(s) must have some screens (activities/fragments) to enter and store data related to geocaches and users:

- Register a geocache: register an ID, name, location, category, picture (and possibly more).
- Game maps: show the location of all active geocaches in a Google Maps.
- Find a geocache: allow the user to read the geocache QR code and give her/his a score.
- User profile: allow the user to edit information about her/his profile.
- Administration area: show this area to at least one user. She/he will be the game administrator and responsible for managing the game rules.

Minimal Requirements You are encouraged to make a beautiful, user-friendly, powerful, reliable, and robust app implementing the functionality described above. However, be realistic about your programming experience and the amount of time you have available when deciding what to include and how to implement it. For example, it is better to have a modest functionality that does not crash and does the job rather than adding a lot of unstable functionality. As a minimum, your app must use at least:

- Activities, Fragments, App Resources (strings, layouts, menus, colors), RecyclerView, Adapters, and Singletons (*optional*).
- Handle bugs regarding the application life cycle (e.g., when the user rotates the device) and resource permissions (e.g., camera, GPS data, content providers).
- Save the current status of user interface components in the *Shared Preferences*.
- A local database on the Android device (*SQLite*, *Realm*, or any other database). The remote database (*MongoDB Realm Sync*) is *optional*.
- Use Android Location (GPS) to find the current position of your Android device.
- A map to show all geocaches near the current user location (using *Google Maps*).
- Use the camera to take a picture (of a geocache) and to read the QR codes.
- A fake (*mock*) login system to manage the game users.
- Develop the entire project using the MVC design pattern.
- Add new functionality to your GoCaching app based on any topic presented during the course.

Example of Functionalities These are some examples of functionalities for your GoCaching app, namely: (i) register the geocache ID using Bluetooth or Beacon; (ii) implement a "*Find to Find*" module to find active geocaches near to the user current location; (iii) review/score the difficult to find a geocache; (iv) a real user authentication based on Google Sign-In, Facebook, MongoDB Authentication, AWS Cognito, your implementation, among others; (v) image classifier to recognize geocaches in the stored picture; (vi) using MongoDB Realm sync to synchronize the database between multiple devices; (vii) using AWS S3 to store application resources (e.g., high-resolution images), among others.

Programming Languages By default, you must develop your project using **Kotlin** or **Java**. However, you can be allowed to implement your GoCaching App using a cross-platform tool compatible with the Android platform, such as Flutter, React Native, among others. You must have a formal agreement with Fabricio (by e-mail) before starting to implement your project.

Due date The mandatory assignment must be handed in before Friday, 07 May 2021 (13:59) via learnIT.

Grading Your assignment will be evaluated by the teacher/TA's, and you will get some feedback, including APPROVED/NOT APPROVED grade. You must have an *approved* version of the mandatory assignment to take the course's final examination.

Team Work You may collaborate with other students (or anybody else), but each student MUST submit a solution **individually**. There can be similarities between the code you submit and the code provided by other students. However, the documentation (see below) **must** be done individually.

Submission Information Your submission must consist of two parts: (i) *Code*; and (ii) *Documentation*.

Code You MUST submit a complete Android Studio project directory with a working app (i.e., NO *syntax errors* or *runtime exceptions*). Your `gradle` file must look like the standard Gradle file for the course (see an example in item 6 in Work Plan #01). The entire project directory must be packed as **a single zip file** with the name `WXYZ_solution.zip` (which `WXYZ` is your ITU's account, e.g., `fabn_solution.zip`).

Documentation You MUST submit a report (in PDF format) explaining your solution. The document MUST be maximum 10 (ten) pages and contain these sections:

- Most important *design choices*, including functionality and user interface;
- Short *user guide*;
- *Test* of the app – see below;
- *Problems* with your app (if there are any), e.g., if something does not work completely as you want;
- Detailed information (theoretical and technical) about the additional functionalities you have implemented in your GoCaching app.

Design Choices It would be best to describe your design choices, both related to the user interface and the functionality. For example, what alternatives did you consider, and why do you prefer the solution you ended up using. Here is an incomplete list of design choices:

- The layout of the user interface and how it is implemented in activities/fragments;
- Structure of your database;
- Diagrams to show the overall structure of your app;
- How does primary functions work (register a geocache, show available geocaches, among others).

Testing You should make several test sequences that cover all the functionality/use cases of your app (including error handling) and all user interface parts. In the following, you can see an example of a test sequence to evaluate the "register a geocache" module:

1) Start GoCaching app with *Bluetooth*, *WiFi*, and *GPS* enabled;
2) Detect the current longitude and latitude;
3) Generate the QR Code (e.g., `https://www.qr-code-generator.com`);
4) Take a photo of the geocache and fill out all mandatory information;
5) Ensure that the geocache is registered correctly.

A complete test of your GoCaching app will require many such test sequences. As you develop the app, write down relevant test sequences. Use these in the debugging and the final test before submission.

Try to make a list of test sequences that cover all the use cases of the GoCaching app. You can imagine cases where the user makes mistakes and events outside your control, like network failure.

The final documentation of your GoCaching app MUST contain a sample of five test sequences and the test results. You can decide which five to submit, but try to pick some that cover the primary functions.