Mobile App Engineering and User Experience - Fall 2017

3rd Assignment – Localization and the Daily Path

Published: Wednesday, Nov. 1st, 2017

Deadline: Sunday Nov. 19th 11:55pm, 2017

In this assignment, you will build an app to acquire and map device locations. The purpose is to understand what an app with location permissions could learn about your daily life , to further explore callback and concurrency concepts as well as become familiar with the location and the Google Maps APIs. The required API level for this assignment is 23 (Android 6.0). This assignment requires using Google Play services location APIs and Google Maps Android API.

All projects are required to be submitted with a one-page report that specifies what was implemented (until what point range) and provides detailed explanations of the code. Justify your design decisions. Student in the graduate course, please see the additional implementation and report requirements at the end.

You need to design the whole app and database (on the phone). Please read all of the requirements before you start programming. The later requirements may change the design from the prior ones. You only need to submit the app version at the highest point level you have implemented.

1) 45 points maximum

Build an app to display and remember locations in terms of latitude and longitude coordinates as well as postal address. Review the location and maps section in the online Android Developer Guide (https://developer.android.com/training/location/index.html) before getting started.

- Create an app that obtains the phone's current location and displays the latitude longitude values in a TextView. (5 points)
- Use Android Geocoder class to get your current ADDRESS, and display next to the longitude and latitude. (5 points)
- Use callbacks so that when a new location becomes available, the coordinates in the TextView are updated. Find out whether the callback is called from the UI thread or a different thread (hint: compare thread ids) and if race conditions exist, modify your code to address them (explain in a comment why your solution is safe). (5 points)
- Add a button to "check in" to your current location. This should store the current location and time in an SQLite database. Also, all stored locations (longitude, latitude, time, address) should be shown in a list in the app. (10 points)

- Allow users to add a custom name for a location when checking in. Store the names in SQLite database. Check in at five positions of a few Rutgers locations and name them. The marked positions should along the red route on the map below. Add the custom name to the check-in list from the previous bullet (leave blank if no name assigned) (5 points)

- Any check-in within a radius of 30m of an existing check-in should be automatically associated with the prior check-in name, if any, and postal address. Design your database so that one location can be associated with several check-ins. Store the exact latitude, longitude and time of each check-in. Normalize your tables so that no redundant information is stored. Design a single SQL query to retrieve all information for updating the check-in list (the list from the previous bullets). Explain your query and how you designed the database in your report. (15 points)

2) 75 points maximum

Add another activity to the same app to show a separate map view of the locations using the Google Maps Android API. Follow the directions at (https://developers.google.com/maps/documentation/android-api/) to install the Google Play services SDK, acquire a Google Maps API key, and display it within your app.

- The map should be centered on your own location. It should include standard UI controls: zoom in/ out buttons, Compass, "My location" button. (10 points)
- Show all of the check-in locations on the map with markers (such as small dots). (5 points)
- When a user clicks on the map, enable the user to add a new named location with a marker on the map. The marker should be draggable. The user should be able to name it. The new named location should be added to the database (without recording a check-in at this location) (5 points)
- If a user approaches any of the named locations (within 30m), an info window should popup and show the name of the location, and the time of last check-in (find the time of last checking using an SQL Query), if any. If the user is more than 30 meters away from the marked locations, the popup window should disappear. (10 points)

3) 90 points maximum

Automatic check-in in background.

- Add a mode where the app automatically checks in the whenever 5 mins have passed or when the user moved more than 100m (5 points)
- Enable such automatic check-ins even when the app is running in the background and not visible to the user. (10 points)

4) 100 points maximum
- Quality of the report and UI design. (10 points)

Graduate course additional requirements (up to -30 for graduate students who do not complete this):

1) Install your app on a smartphone. Modify it to log the delay and accuracy of the location updates from GPS and network location. You can do this, for example, by manually enabling either GPS or network localization. Experiment with the app in at least 5 indoors and 5 outdoors locations. Estimate the accuracy in meters based on a map of the area and plot the results in a graph. Prepare a report about your findings (this is in addition to the 1 page report required for all students). The report should include the accuracy reported by the location API, your estimated accuracy, and the measured delay. Think about the delay and accuracy with each localization methods and explain your findings. (20 points)

2) Build a basic wifi positioning system as described in the "RADAR" paper as an alternative to the Android location API. Add a toggle switch to allow the user of your app to switch between using your implementation and the Android API (as in the previous questions). Expand your database to store signal strength measurements and access point IDs from a wifi scan, whenever a user checks in. To find a position in your implementation, start a new wifi scan and find the closest fit in signal space in the database. Report the location of this closest fit. Test around the DSV lab. (10 points)

Upload your solution source code, apk package and report in the same zip or tar.gz file named as netid_yourname. The apk named as the name defined in manifest for the main activity. The apk file should be copied to the root directory of the submission folder, also keep the original apk in the bin folder.