SENG440: Topics in Mobile Computing Project 2

1 Overview

Your responsibility in this project is to make an app that fully embraces mobility by interacting with the surrounding environment through sensors or proximity-based networks. The design and purpose of the app is mostly unspecified. You are encouraged to make something that interests you or for a community that you care about.

2 Requirements

The requirements for this project fall into two categories: mandatory requirements and grade-bearing requirements. For your project to be eligible for marking, you must satisfy all of the mandatory requirements by the end of the due date—which is the last lecture day of term 2 (29 May). If these requirements are met, the grade you receive will be calculated based on how many of the grade-bearing requirements you meet.

Complete the following **mandatory requirements** by the due date:

- Create an app that has a particular and meaningful purpose for some audience. It should not just be a sandbox app in which you cobble together disconnected features.
- Use Kotlin for the program logic, not Java. Create your user interface declaratively with XML, not imperatively through Kotlin.
- Maintain and submit your project in a Git repository on the departmental GitLab server.
- Document your app with a post mortem post on Slack. In your post, tell us a story about the app's purpose and development process. Include screenshots. At the end of your post, enumerate which of the grade-bearing requirements you believe you have met and how. Be brief but specific in your enumeration.

The more **grade-bearing requirements** you complete, the better your grade. There are 20 points available in this project. Each of the following requirements is worth 1 point:

1. Interact with the nearby physical world in some way. This interaction might rely on sensors, GPS, the camera, or Bluetooth to create a local area network.
2. Interact with the nearby physical world in some additional way.
3. Provide a facility for “openness”—for a user to interoperate with entities beyond their phone or beyond your app. This could take many forms. You could allow the user to export a backup of your app's data that could be imported by someone else or on another device. You could provide support for deep-linking, such that certain patterns of URLs would trigger your app. You could allow the user to share achievements via texting or social media. If your idea for this feature situates your app within the larger community, it probably qualifies.
4. Gracefully handle configuration changes, not losing any of the user's data.

5. Use a local database to persist data, preferably using Room instead of raw SQLite.
6. Send the user notifications related to your app in some way.
7. Integrate an action bar in at least one activity.
8. Provide a preference screen using the modern AndroidX Preference Library.
9. Add a multi-resolution launcher icon.
10. Support both landscape and portrait orientations in all view—unless your content demands a fixed orientation, as in a game. In other words, all widgets should be able to be made fully visible in either orientation. This may happen automatically given your layout manager, or you may use a `ScrollView`, or you may specify two separate layouts.
11. Use string resources for all static text on the user interface.
12. Provide default definitions for your string resources in English. Provide definitions for one other language. (Use your favorite online translator if necessary.)
13. Write a set of *user stories* for your app and include them in your post mortem. In addition, if you have someone available in your "bubble", have them test your app and record their feedback.
14. Incorporate an animation into your UI, preferably one specified in XML. This time around, animated GIFs and fragment transitions don't count.
15. Incorporate some other Android feature not mentioned above into your app. (something different from the extra feature in project 1)
16. Incorporate some other Android feature not mentioned above into your app. (something different from the extra feature in project 1)
17. Share a plan for your app by end of Friday (1 May), week 8 in a post on `#project2` in Slack—before you've written any code or created any layouts. Include hand-drawn sketches or wireframes.
18. Share an update of your work by the end of Friday, week 9 in a post on `#project2` in Slack. Include screenshots.
19. Share an update of your work by the end of Friday, week 10 in a post on `#project2` in Slack. Include screenshots.
20. Share an update of your work by the end of Friday, week 11 in a post on `#project2` in Slack. Include screenshots.

Your instructor would consider substituting three of the requirements above for publishing your app on the Play Store. (Publishing will not count as extra credit, only as a substitution.) Be forewarned that preparing an app for publication is a lot of work and costs a wee bit of money—by some definitions of *wee*. If you are interested in this substitution, declare your intentions in your first Slack update.

If you have questions about any of these, communicate with your instructor. Do so early and often. Please share your questions in `#general`.

3 Submission

To submit your project for grading, complete the following before the due date:

- Commit and push your app to your Git repository.

- Verify that the push succeeded by visiting your repository via your provider's web interface.
- Publish your post mortem on Slack in the `#project2` channel.