SENG440: Topics in Mobile Computing Assignment 1

1 Overview

Your responsibility in this project is to get acquainted with the Android user interface system and its event-driven programming model. You will do this by writing an app that makes use of the features that we discuss in Term 1 of the semester. 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 first Wednesday of the term break (**31 August**). If these requirements are met, the grade you receive will be calculated based on

1. how many of the grade-bearing requirements you meet, and
2. a holistic evaluation of the creativity and design of your app. This will, in part, be based on your report of your development process, as well as the final app and its suitability for its intended purpose and audience (see *mandatory requirements* next).

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 program logic, not Java. Create your user interface declaratively with XML or Jetpack Compose, not imperatively through Kotlin (non-Compose) code.
- Maintain and submit your project in a Git repository on the departmental GitLab server.
- Document your app with a 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:

1. Compose your app out of at least three screens, where a screen is either an `Activity` spawned via an explicit `Intent`, or a `Fragment` with a fullscreen layout. You can use multiple activities, a single-activity multiple fragment architecture, or Jetpack Compose navigation.
2. Invoke at least one other app on the system via an implicit `Intent`.

3. Include a list view, preferably using `RecyclerView` (XML) or `LazyColumn / LazyRow` (Compose). If you are using `RecyclerView`, compose your list view using a custom adapter whose view creation method uses a custom layout, otherwise use composables to create a custom view for items.
4. Include at least five different kinds of widgets besides a list view (buttons, textboxes, checkboxes, labels, and so on) in the user interface, and handle their interactions with event listeners.
5. If using XML for your layout use at least two different layout groups (e.g., `ConstraintLayout` and `LinearLayout`) to organize your widgets. You can convert other layouts to a `ConstraintLayout` but include documentation of the process in your report. If using Jetpack Compose, for your layout use at least 2 of `Column`, `Row`, `Box` (or `BoxWithContraints`) and they should use `Modifier` to align/decorate.
6. Support both landscape and portrait orientations in all views. 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. For Jetpack Compose you will need to create an adaptive layout.
7. Use string resources for all static text on the user interface.
8. Provide default definitions for your string resources in English. Provide definitions for one other language. (Use your favorite online translator if necessary.)
9. Use a `Toast` message or dialog to alert or interact with the user.
10. Use coroutines to trigger some computation (e.g. network call) without blocking the user interface.
11. Incorporate an animation into your UI, preferably one specified in XML or using Jetpack Compose's Animation API. We will not discuss these in lecture. You should be able to find out more information on the Android developer website.
12. Incorporate two other Android API features not mentioned above into your app.

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 (project report) on Slack in the `#project1` channel.