Assignment 3: Android App

Pre-Assignment Notes

The **late penalty** is -10% per day late. After two days, late submissions won't be accepted.

This assignment can be completed **individually OR in pairs**, so do not share your code or solution with others who are not in your group, and don't copy code you found online. There's still the class's Piazza if you want to ask questions to the group, or you can ask during lecture, send an email to the class's help account, or attend an office hour on Discord.

Note you can use any code shared by the instructor or anything from class or the tutorial videos linked on the course website. You *can* get outside help from guides and other people, just try to make sure they're teaching you how to solve the assignment, not writing the code for you. You won't learn anything that way, and they can't write your exam or midterm for you. If in doubt about whether something is plagiarism or not, don't be afraid to reach out and ask!

One tip: if you copy more than 4-5 lines of code from a guide or tutorial, try citing it in your code with a comment, like //Code found at [url goes here].

Whether or not you're working in pairs, you will have to use the group submission feature in Coursys to submit! There will be more details about this in the deliverables section of this document, so don't ignore this.

1. Application Requirements

In this assignment you will be implementing an Android application to play a simple game.

1. Read the software requirements document on the course website. You must meet all requirements marked as **"must"**, while requirements marked as "**may**" are optional, but may be worth credit.

2. Some options for learning more Android programming needed for this assignment include:
A) Watching the video demos listed on the course website on the assignment page.
B) Reading chapters 1-6 of the Big Nerd Ranch Android Programming Book.

3. Use Git and GitLab to develop your code. You must create at least three GitLab issues. These could be bugs or features to implement. Close the issues when they're completed. You must also create three feature branches to implement the changes/features for some GitLab issues. Merge these feature branches back into the master, as per the process covered in lecture. For using the GitLab Workflow. You need not have any merge conflicts to resolve: you just need to have the feature branches created and then eventually merged.

A good rule of thumb is you should do your work on feature branches and commit at least once each day when you are done coding. Merge to master when you have finished a feature. If you are working with a partner, merge more often when you have some little increment of functionality working, and make sure you don't break the build!

4. You must create a separate (or sub) Java package for your model classes. For example, if your application is ca.cmpt276.as3, your model package may be ca.cmpt276.as3.model. Your game logic must be in a class in your model package. You must also have a **Singleton** class for storing the options data (Hint: have your game activity use this singleton to get the configuration data for each new game.

Have the options activity get/set the values stored in the singleton. Use get/set methods, not public fields!).

5. Each Java class must have a class-level comment (above the class) briefly describing the purpose.

6. Some graphics are provided on the course website that you may use in your application. You may also make your own, or find resources elsewhere. If you use resources or images, you must list where you got each of them in your application's help screen.

2. Deliverables

Submission is done through CourSys. You must create a group in the system before submitting, even if you are working individually. If you are in a group, only submit one solution for your group. Each group member should accept the group invitation before the group submission is made.

Create a docs/ folder in the project which contains:
readme.txt: State which optional features you want to be marked on. If you don't have this file, the TA may fairly assume you didn't attempt any optional features.
playing.jpg: A screenshot of the game when the user is playing it. Take the screenshot after more than half the mines have been discovered (your choice of board configuration).
won.jpg: A screenshot of the app's screen when the user has won the game.
git_graph.jpg: A screenshot of GitLab -> Repository -> Graph, which should show 3+ feature branches being merged back in.
git_issues.jpg: A screenshot of GitLab -> Issues -> List -> All, showing 3+ issues.

Commit these files into your Git repository and push them to GitLab.

Submit the following to CourSys:

1. **A .zip file of your project, as generated by Android Studio**
See the course website for details on how to generate this file. Ensure your ZIP file contains the docs/ folder mentioned above.

2. Git Tag
1. Add the TAs for the course as **"Developer"** members of the repo. Do this by going to csil-git1.cs.surrey.sfu.ca and selecting your project. On the left hand side, click the Settings (cog-wheel) menu option. Select "members" and then add myself (**jackt)** and the TAs (**kpathani** and **tirthp**).
2. Create a tag for your submission in Android Studio by going to VCS -> Git -> Tag... and entering a name like final_submission. Leave Commit and Message blank and click Create Tag. Push the changes to the remote repo, and on the "Push Commits" window be sure to check the box for "Push Tags: all". You can check the tag was pushed correctly on the GitLab website. If you resubmit, be sure to create a new tag following these steps and then submit the new tag using CourSys.
3. Submit the Git tag (the git repository URL and tag name) to CourSys. The .git url is the same one used to clone the repository and ends in .git. The tag is just the name you used in step 2.