Assignment 3: Android App

1. See the Deliverables section for what functionality is required, and how to submit.
2. Late penalty is 10% per calendar day (each 0 to 24 hour period past due). Maximum 2 days late (will not be accepted after that).
3. This assignment is to be done individually or in pairs. Do not share your code or solution, do not copy code found online, and do not post questions about the assignment online (other than the course’s forums). Please post all questions to the online discussion form, or email the instructor or TA. You may (of course) follow online tutorials, look-up suggestions on how to code things, but you may not submit significant sections of someone else’s code or your code you have previously submitted.

1. Application Requirements

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

1. Read the software requirements document on the course website.
   - You must meet all requirements which say "must".
   - Requirements which state “may” are optional; see marking guide for ones that are worth credit.
   - I recommend you read a chapter of the book first, and then implement the associated functionality required for the assignment. Jumping straight to coding the assignment will likely take longer than if you read the book first.
   - If nothing else, watch and try the video-demos listed for this assignment.
3. Use Git and GitLab to develop your code.
   - You must create at least three (3) GitLab issues. These could be bugs, or features to be implemented like “Implement help screen”, etc. Close these issues when the work is done.
   - You must create at least three (3) feature branches to implement the changes/features for some GitLab issues. Merge these feature branches back into 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 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.
get/set methods (not public fields!).

**Hint:** The singleton’s private fields will likely be the number of row, column and mines.

5. 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 images or resources, you must list where you got each of them in your application’s help screen.

---

### 2. Deliverables

Submission is done through CourSys ([www.courses.cs.sfu.ca](http://www.courses.cs.sfu.ca)). 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 what “Optional” features you want to be marked on. If you don’t have this file, the TA will assume you did not attempt any optional features.
- **playing.jpg:** A screenshot of the game when the user playing it. Take the screenshot after more than half the mines have been discovered (you choose board configuration).

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

**Submit Git Tag**

1. Add the TA for the course as a **Developer** member of your repo:
   - Goto csil-git1.cs.surrey.sfu.ca and select your project
   - On the left hand side, click the Settings (cog-wheel) menu option
   - Select “Members”
   - Add the TA to your repo as a Developer. You can find the TA’s SFU ID (email address before the @sfu.ca) on the course website’s Course Info page.

2. Create a tag for your submission as follows:
   - In Android Studio, go to VCS --> Git --> Tag...
   - Enter a name for your tag, such as: **final_submission**
   - Leave Commit and Message blank.
   - Click Create Tag
   - Push changes to remote repo. On “Push Commits” dialog, select “Push Tags: All”.
   - You can check the tag was pushed correctly in GitLab online.
   - (If you resubmit, create a new tag as above and submit the new tag via CourSys).

3. Submit the GIT tag (git@... URL and tag name) to CourSys
   - Find git@... URL on csil-git1.cs.surrey.sfu.ca/. Should be similar to:
     git@csil-git1.cs.surrey.sfu.ca:yourid/myProjName.git
   - The “tag” is the name you used above, such as “final_submission”

Please remember that all submissions will be compared for unexplainable similar submissions.