Course Project

Grouping: The groups are formed using computer algorithms and the groups cannot be changed (unless you have medical proof that you are allergic to certain group member). You may be stuck with someone who you don't appreciate. Take that as a test of your people skill. Talk to us if there are issues that you can't deal with. Kindly find your group number.

Project: Each group is expected to developed an app in the Android platform. My recommendation is that it should be a game which supports multiple users playing concurrently. The scenario is that a group of you, each has an Android device which has the APP installed, can start a game whenever multiple players are ready to have some fun. You may use one of the devices as the server or you can leverage on Google Play Game Service. It is perhaps not a good design if each player must wait for his/her turn to play. You, as a group, will (A) identify the system requirements (e.g., what are the features of the game, how users interact, etc.); (B) come up with the design of the system (e.g., what are the components in the system implementation and how do they communicate, etc.); (C) implement the game; (D) and apply proper techniques so that the system functions properly and reliably. It is possible if you, as a group, decide to develop something other than a game, you are required to communicate early with me on your project though.

Final Presentation: In week 13, you will present the final product to the instructors and the classmates. You are expected to present.

- the software development process employed in your project;
- a live demo of your app;
- and the hiccups and lessons learned through your project.

Final Deliverable: You will deliver a self-contained package of the app, including: (A) the code itself; (B) installation and user manuals; (C) a report detailing what are the system requirements, how are the system designed, how are the system evaluated/tested. In particular, we would like to know how you ensure your implementation is thread/safe. (D) a separate report, delivered by individual group member on contribution of every member in the group.

Scoring

- Presentation and Demonstration of the APP (40%)

- Report/Code (60%):
  Clear documentation of the system requirements (10%);
  Clear documentation of the system design (15%);
  Code structure (15%);
  Clear documentation on how the system is tested (10%);
  Clear explanation on how concurrency issues are handled or avoided (10%)

Sample Project

The following gives some details on a sample project. Notice that your proposal should be more interesting. The game is a multiplayer variant of the classic computer game "Minesweeper". You can review the traditional/single-player minesweeper concept / rules here: http://en.wikipedia.org/wiki/Minesweeper_(video_game).

We will refer to the board the board as an NxN grid where each square has a state which can be 'flagged', 'dug', or 'untouched', and each square either has a bomb or does not have a bomb. Our variant works very similarly to real minesweeper but with multiple players for one board. the main functional difference is that when one player blows up a bomb in single player, they just lose. when one player blows up a bomb in our version, they still lose, [i.e. server ends their connection] but the other players may continue playing. The square where the bomb was blown up is now a dug square with no bomb. (The player who lost may also reconnect to the same game.)

Note that there are some tricky cases of user-level concurrency: as a notable example, user A has just modified the game state (i.e. by digging in one or more squares) such that square i,j obviously has a bomb. Meanwhile, user B has not observed the board state since this update has taken place, so user B goes ahead and digs in square i,j. Your program should allow the user to dig in that square– a user of Multiplayer Minesweeper must accept this kind of risk.

The specific components for a game like this include, at the least,

- setting up a server to deal with multiple clients; you should decide what is the best way to handle client requests.
- implementing a data structure for Minesweeper.
- making sure the data structure, which will be updated by multiple clients concurrently perhaps, thread safe.
- putting the server and the clients by defining the protocol for communication so that the system runs robustly.

One particular implementation of the same project is here.

Online Resources/Tutorials on Game Development

- https://unity3d.com/
- https://marvelapp.com/
- https://libgdx.badlogicgames.com/
- http://www.kilobolt.com/introduction.html