Project Details

This document provides details about the project such as how iterations will work, and general application constraints.

1. Iteration

1.1 New & Updated Use Cases

The project kickoff presented a long-term road-map for the app. These are feature ideas that may or may not be required for some iteration.

Each iteration Dr. Fraser will post new and/or updated use cases which describe the new functionality that must be implemented. The use cases will be written from the user's point of view with plenty of room for interpretation by your team. This not only simulates how real users will give requirements, but also allows your team some flexibility in implementing features to add your own ideas. If you would like to implement a feature in a way which is contrary to how it is described in the use case, please talk to me (Dr. Fraser) for clarification.

1.2 Scrum Roles

Each iteration, the group assigns roles to different team members. This must rotate between group members; nobody can be the same role twice. All members of the team must contribute to delivering features during the iteration. Some roles have additional duties and hence will contribute slightly less to the project’s code.

Scrum Master

- Organizing group meetings and helping to keep meetings on topic.
- Organizing how activities during the iteration are completed, such as team retrospective and project submission.
- Being the scrum master does not give anyone management powers over team members.
- Should help team members resolve conflicts, work with members who are falling behind (though it’s not their job to do the work, nor teach the others what to do, but rather to help keep everyone on the same page and communicating).

Product Owner

- The only person on a team who can contact the customer (Dr. Fraser, or the real customer via Dr. Fraser) and ask questions related to features for that iteration. Anyone may ask about other aspects of the project (such as marking), or raise concerns about the project or the team. Basically, if it's a question the customer should answer, the product owner must ask it.
- Creates initial issues in GitLab at the very start of each iteration. Maintains these as needed with new understandings. All team members are invited to add/edit as well.

Repository Manager

- Helps everyone work with Git and GitLab.
- Responsible for accepting merge requests in GitLab. If code looks good, should be

accepted promptly. If code needs improvement, should give feedback to the team member and have them make additional commits to cleanup the branch before merging.

- Helps others do code reviews. Might ask someone to review a merge request. May step in to help encourage a positive discussion between two team members with respect to a code review.
- Need not be an expert in Git or GitLab, but willing to learn and help figure things out.

Team Member

- Like everyone else, teams members contributes to delivering working software implementing the required features.
- All members of the team contribute to discussions, communicate with each other, and share in making decisions.

Create a `docs/` folder in your project. In it, create a `rolesN.txt` file (where N is the iteration number, such as `roles1.txt`) and list which team members filled which roles for this iteration.

1.3 Team Retrospective

Each iteration ends with some reflection on how the iteration went. This is key to Agile methodologies because the team owns the process, so the team evolves the process by analyzing how things are going.

Retrospectives will likely be done during class. Must be present to earn marks for the retrospective.

1.4 Submission and Marking

Each iteration your team must tag your repository and submit to CourSys the Git tag.

Each iteration a marking guide is also posted so you know exactly what is expected and how much it is worth. All iterations will be marked by the TA for your official grade. Iteration 3 will also be evaluated by the customer for feedback about meeting their needs. Possible bonus and/or prizes for outstanding work meeting the customer's needs.

Questions about the marking of your work should be directed to the TA (via the cmpt-276-d2-help@sfu.ca email list). If you are unable to resolve your question with the TA, then please bring your question to the instructor.

Additional requirements apply to the 3rd iteration's delivery. See course website for more.

2. Application Constraints

- All development must be in the GitLab repository creating for your group.
- Use the GitLab workflow, as discussed in class:
  - Start feature with a GitLab issue
  - Create a feature branch
  - Commit code to feature branch often
  - Merge Master to feature branch once feature completed.
  - Submit Merge Request
  - Code review, fix, and merge.
- Apps must run on at least Android API 19 or newer (i.e., min API set to 19 or lower). This

is KitKat, Android 4.4.

- Well built UI, including:
  - Well laid out usable screens.
  - Works on different sized screens. Example tests on 4", 6", 10".
  - Use of icons and images where possible.
  - Strings that the user can see must be in `strings.xml` to support internationalization.
- Clean Code
  - Good class, method, and variable names.
  - Perfectly formatted code.
  - Comments on all classes (not needed *inside* classes, just on the class level)
- Simple OOD supporting not much more (if anything) than the current set of requirements.
- Good commit comments on (almost) all commits.