Project Details

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

1. Iteration

1.1 New and Updated Use Cases

Each iteration, Dr. Jack Thomas 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. Thomas) 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 members 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 the other 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 the team who can contact the customer (Dr. Jack Thomas) 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 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 team has code review policy then the repo manager ensures that code reviews are done (often by other team mebers). Merge requests should be reviewed and accepted promptly if everything is okay.

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 work with other team mebers to help figure things out.

Team Member

Like everyone else, team members contribute to delivering working software by 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 (where N is the iteration number, such as roles1.txt) and list which team members filled which role 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 be further detailed during class at the end of each iteration. Each team member must participate in the retrospective to receive their own marks!

1.4 Submission and Marking

Each iteration, your team must tag your Git repository and submit it to Coursys.

Each iteration's user stories include information on how many marks a feature is worth, so you know what is expected and how much it is worth. All iterations will be marked by a TA for your official grade.

Questions about the marking of your work should be directed to the course's help email address (listed on the course website) so that the relevant TA and I will both see your question.

Additional requirements apply to the 3rd iteration's delivery, which will be posted to the course website when they are relevant.

2. Application Constraints

*All development must be in the GitLab repository created for your group.

*Use the GitLab workflow, as discussed in class:

-Start a feature with a GitLab issue.

- -Create a feature branch.
- -Commit code to the feature branch often.
- -Merge the Master to the feature branch once completed.
- -Submit a Merge Request.
- -Code review, fix, and merge back into the Master.

*The application must run under at least Android OS version 10.0 (API 29, Q) and newer.

*Well-built UI, including:

- -Well-laid-out, usable screens.
- -Works on different-sized screens. Run tests on 4”, 6”, and 10”.
- -Use of icons and images where possible.
- -Strings that the user can see must be in string.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 Object-Oriented Design, supporting not much more (if anything) than the current set of requirements.

*Got commit comments on almost all commits.