Iteration 5

At the end of each iteration, every team needs to have a tagged version of their code up to that point (Iteration1, Iteration2, etc.). The course staff will only grade the tagged version. How to tag depends on your version control, so that is something to discuss with your TA.

Meetings for this iteration should be scheduled from April 6 to April 11 (Wed-Monday).

This is an appropriate time to let your TA know if you need to reduce the scope of your project (with convincing reasons) and cut off some of your remaining user stories. Of course, the initial goal of your project should still be present – after all, that’s why your project was accepted in the first place. So please discuss with your TA an appropriate scope for the project in order to avoid problems when your grade is assigned.

Deliverables

- You need to convince us that you are following the process that your team agreed on. Remember that in cs428 we grade the process not only the product.
- You should have addressed any issues that were discussed during the previous iteration meeting. This includes making sure that your wiki page has been formatted properly, ensuring that your user stories are well described, setting up any resources necessary for your project with TSG, having automated tests, and any other issues that were brought up during the meeting.
- The wiki should be updated with your progress for the current iteration:
  1. You should fill in the actual time you’ve spent on the user stories in this iteration. Remember to document any bumping of (partial) user stories from one iteration to another.
  2. You should revise the user stories and estimates for the next iterations.
  3. You should consider your new estimates while deciding how to divide the user stories among iterations (which user story is going to be done in which iteration).
  4. You should reevaluate project risks, etc.
- You should have new code that your team has written checked in your repository, therefore, there should be several new commits in your repository of new code that your team has written – not just code from the web or from the examples/sample folder of your framework. Please select a portion of the code that you find most interesting to present during the meeting. We are not looking for a perfect design but you should be able to convince the TA that you are making progress in your project. Also, we will be checking to see that your code adheres to your team's coding standards.
- You should have made significant progress with your application, shown by working and well tested user stories. You should have at least 3 more implemented user stories compared to the previous iteration and at least 5 unit tests for each user story. We will ask you to show how you implemented and tested each user story.
- You need to have some advanced testing for your projects. It could be one of the following (You can discuss this with your TA and you can refer to lecture 5 and 13 slides for more information):
  1. Parameterized unit tests that are not specific to only one value
  2. Behavioral-Driven test (Cucumber)
  3.Mocks
  4. GUI advanced testing
- You should have a polished UI for the user stories that you have implemented so far.
- You should be doing some code-cleanup and refactoring on your code for this iteration. Each team member has to have at least one refactoring that they can show in their version control (code before and code after the refactoring).
- You should start thinking about packaging your project so that someone else can install and run it.
- You should tag your code in the repository.

Iteration Meetings

- It is expected that teams keep notes about their meetings (especially, but not limited to, the iteration meetings). What did the team accomplish? What are the issues causing trouble to the team? What needs to be done? So for each meeting, someone in the team will be the scribe. The notes can be in the form of minutes or anything else you agree on with your TA. Look to your peers to see which method works for them and for you. The bottom line is to create a transparent project that the TA can follow and more importantly that your teammates can follow to ensure excellent team communication. Leverage the wiki and the tools you have chosen to their fullest extent. These notes need not be excessively long (longer is not always better), but they should contain enough detail so that your team members and the TA know what is going on.
- There should be a different leader and scribe for each iteration meeting. Everyone should be able to and is expected to contribute to the discussion.
- All team members should attend all meetings. Any absences should be arranged ahead of time with the team and with your TA.
Documentation

As part of your final project submission you will be expected to compile a documentation document for your project and the software you developed. We will provide more information about the different types of information that will be expected in the near future. It should not be a surprise, however, that this documentation will include information on the requirements and design of your system, as this is an important aspect of the course. As you watch the lectures, read the required readings, and do the homeworks on requirements and design, reflect on how this material should be applied to your project. For instance, you should expect the requirements and design portion of your project documentation to include use cases, class diagrams, and sequence diagrams as discussed in the lectures.

You could use the following outline to start on your documentation:

- A brief description of your project
- The process followed by your team
  - Make sure to address the issues of iterative development, refactoring, testing and collaborative development (even if you are not using XP you have to address these issues in your documentation).
- Requirements & Specifications of your project
  - This should include the user stories and use cases which you implemented in your project. You do not need to mention the requirements which were dropped.
- Architecture & Design
  - Include multiple UML diagrams to show the important parts of your system (you must have UML diagrams).
  - Describe in a top-down manner, the big picture of your system.
  - Include enough details about the design of your system such that anyone who refers to your documentation can understand the major components of your system and how they are related.
  - Describe, if your system design was influenced by your choice of a particular framework? Mention how did the choice of framework influence the design.

We won't be grading you on the final documentation for now, but we provided you with this information so that you can start preparing it.