Northeastern University
CS5500 – Managing Software Development
Fall 2018, Professors Jump, Katz, and Weintraub

Sprint 1 Expectations

1 Overview

The project is now underway. We will be following a modified version of agile using scrum. While usually this means daily stand-ups in person, that's not feasible in a class like this. Instead, you should conduct daily *slack-ups* when you aren't meeting in person.[1] Even if you don't plan to do anything on the project that day, you should say so on slack. You should agree as a team that everyone will post on your project channel three things (you can add to this list if your team agrees more is needed):

1. What you did the prior day.
2. What you're planning to do that day.
3. If you are stuck on something or blocked by someone else on the team.

For example, it would be acceptable to post "On the beach today." The point is that everyone is accountable to the team and to get in the habit of keeping a log that evidences each person's participation.

We will be adding a session that meets every other week (during a sprint's end) to discuss issues, review progress, and plan the future. We are using two week sprints; you and your teammates will meet with a TA at the end of each sprint for a 30-minute Sprint Review to review and reflect on the past sprint and plan for the next one. You may want to have in-person stand-ups in-between lab sessions; that decision is up to you.

Attendance at the sprint reviews is required. Only excused absences will be given opportunity to make up the review with the TA, which is a one-on-one meeting at the first possible chance. Missing a sprint review means your sprint score may be 0. See the Late Work policy on the website if you have doubts about what will be excused (`https://course.ccs.neu.edu/cs5500/pol.html`).

Here are the technical requirements:

1. You must use Jira (`http://jira.ccs.neu.edu`) to manage your backlogs[2] and store system artifacts on your team's repository on CCIS github (`http://github.ccs.neu.edu`).
2. You must use `Jenkins` for CI [Continuous Integration] unless you receive permission to use another tool. You **must not** alter the `Jenkinsfile` without Alex Grob's consent.
3. You must develop code using `Java`. You may pick the version so long as it is at least JavaSE 1.8.
4. You must use `junit` for unit testing unless you receive permission to use another tool.
5. You must use `Apache Maven` to manage the builds for your project.

[1] Ideally this should be a daily thing. Every other day is acceptable, but every day is better. Less often will lead to issues.
[2] Each project is named *CS5500-Team-XXX-F18* with the key being *MSDXXX*. This is important only if you integrate an external system with Jira.

6. You must use the course provided `Slack` channel for team chat unless you receive permissions to use another tool.[3]

2 Grading

The project will be graded as the accumulation of grades over each sprint and the final presentation.

Your grade in a sprint is an assessment of the team's overall progress (40%) and your individual contribution (60%). Remember, you will be graded on functionality achieved, quality, style, and documentation. The night before your sprint review, you must complete a peer review, through which you will evaluate the work of your teammates. The peer review is for the teaching team and private to the teaching team. Ultimately, your grade will be based on the evidence we find in your Github repo, your Jira project, what we can observe of your system, the feedback from your peers, and the discussion in the sprint review. *Completing the peer review is required.* Failure to complete the review will mean a 0 for your individual score.

Meeting the minimal expectations defines the minimal passing grade.[4] If your team does this work—the functionality works, demonstrates quality, and ensures maintainability, and the team is working well—you earn a B. Hit on more than these points, your grade goes up. Miss on these points and your grade goes down. You cannot lose on stretch work; but if you miss on it somehow (doesn't quite work, didn't do a good job of the work), it won't be as valuable to your grade.

3 Expectations

3.1 Guidance

This will involve extending or creating additional classes and methods as needed. You will need to make many design decisions as well as decisions about what external libraries. You may be allowed to use external libraries for specific tasks, but you should to expect that the vast bulk of the system will be written by the team. This code should be written in Java. Please contact the instructors if you have any questions.

We've added some ideas for stretch goals. You can add your own. A stretch must advance the project either by adding useful functionality or improving the process. If the stretch work isn't helping the project or the team, it will not be counted.

You are in charge of the development path. In Agile once the client provides the priorities, the dev team can re-prioritize tasks or introduce new ones (needed to support the sprint goal). So, if you want to change the expectations, that is fine. But, you must review the changes with your TA or a professor first to judge whether the change is an equivalent amount of work and makes sense in context of current progress. Assuming the change is approved, if it's deemed smaller, you will be asked to add work to even out the balance. If it's bigger, the teaching team will decide what balances the expectations and what is a stretch. If you change things without approval, you run the risk of having misaligned expectations with the client and for an unhappy grading conversation. Don't wait until the last minute to discuss these changes with your TA's and professors. Reach out when you realize you want to change up the priorities.

Please ensure that your code is fully self-contained (e.g., all libraries on which your system depends must be included in the distribution), documented properly, and accompanied by tests and installation

[3] Until this project works sufficiently, of course.

[4] As defined by the College's policy of a minimum 3.0.

instructions so that the TA's can install your system and run your tests without problems. You may expect that your TA will run your system on a personal machine or use some cloud service such as AWS.

3.2 Functionality

1. Getting the legacy code base running.
2. Adding a notion of user and groups of users to the system, including CRUD[5] functions.
3. Stretch (medium): Directing messages to individuals.
4. Stretch (large): Directing messages to groups, including a reply-all function.

3.3 Environment

1. The team pushes only branches to `origin` and uses `pull-requests` for all merges to `master`. `master` on `origin` has been protected so you must use pull requests. Jenkins runs on all pull-request submissions. It runs the test stage, which has Junit tests. Jenkins catches all failures in the test stage and blocks the merge to master.
2. The project uses `maven` for builds. item The project uses `junit5` for unit tests.
3. The project is using at least version 1.8 of Java.
4. The system is packaged as a standalone system (it should not require being run in an IDE).
5. The system is deployed to a cloud environment such as Amazon AWS or Mass OpenCloud. You are encouraged to find a free service. You must provide administrator access to the project executives.[6]
6. Work is being managed in Jira.
7. Stretch (small): team is using smart commits in `git`.
8. Stretch (small): Jenkins should inform the team of failure either in Slack or email.
9. Stretch (small): Github should inform the team of PRs via slack or email.

[5] **C**reate, **R**ead, **U**pdate, and **D**elete
[6] This means your TA.