Northeastern University
CS 5500 – Managing Software Development
Fall, 2018, Jump, Katz, Weintraub

Sprint 1 Review

Team: **212**

TA: **Srinithi Ramesh**

Review Date: **11/03/2018**

Base Expectations

1. Getting the legacy code base running.
   Comments: YES

2. 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.
   Comments: YES.

3. The project uses maven for builds. The project uses junit5 for unit tests.
   Comments: YES.

4. The project is using at least version 1.8 of Java.
   Comments: YES.

5. The system is packaged as a standalone system (it should not require being run in an IDE).
   Comments: YES.

6. The system is deployed to a cloud environment such as Amazon AWS, Azure or Mass OpenCloud. You are encouraged to use a free service. You must provide administrator access to the project executives.
   Comments: YES.

7. Work is being managed in Jira.
   Comments: YES.

Stretches

8. (medium) Adding a notion of user and groups of users to the system, including CRUD functions.
   Comments: YES (+8)

9. (medium): Directing messages to individuals.
   Comments: NO.

10. (large): Directing messages to groups, including a reply-all function.
    Comments: NO.

11. (small): team is using smart commits in git.
    Comments: YES. (+2)

12. (small): Jenkins should inform the team of failure either in Slack or email.
    Comments: YES. Include continuous Deployment to deploy to AWS instance automatically (+2)

13. (small): Github should inform the team of PRs via slack or email.
    Comments: YES (+2)

Additional Comments:

1. Include exclusion rules in .gitignore for IDE specific files

2. Include a menu in Chatter to enable user and group creation from the console.

3. Great job incorporating LDAP to persist.

4. Few too commits and branches. Create a branch for each ticket in Jira (preferable). Make more frequent commits so that it is easier to maintain the code

5. Follow a standard naming convention for git branches. Including the ticket number with a small description is usually the convention followed.

6. You can create a pseudo-master and create branches out of it and work on features. After the features are done you can merge to the pseudo-master and test if the components work well together before pushing it to master. This provides an opportunity for you to test your code and make sure that the components integrate well before you push it to master.

7. If you create a branch, A, out of another branch, B, then push code into B and not any other branch. This will help reduce the merge conflicts you will encounter and help to maintain the project in git.

8. Overall good job.

Team Score: 99