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

Sprint 3 Expectations

1 Overview

The last sprint was about getting some features going, principally around routing messages to individuals or groups. This sprint builds on that work and continues to emphasize features.

The next big feature step with message routing is to enable time-shifting, delivering messages to individuals who weren't on-line when a message was sent. Also, Legal has informed us that our work is subject to CALEA[1]. We'll take this new requirement in steps.

2 Expectations

2.1 Functionality

1. Adding a notion of user and groups of users to the system, if that's still an issue.

2. Directing messages to individuals and groups if that's still an issue.

3. Queue messages to users who are not on-line when the message was sent. When this user comes on-line, deliver the messages in order in which they were sent. Messages that are sent while this backlog is being unloaded are held until the backlog is empty. In other words, delivery should strictly follow a queue based on the time the message arrived at the server (Prattle).

4. Duplicating (dup'ing) a message targeting a specific user or group. Dup's get forwarded to an agency (who have a subpoena for the communications of particular users) and this must be done without anyone involved in the original communications being able to determine their communications are being sent to an agency. Each subpoena creates a distinct duplication of messages. In other words, you cannot create a single channel where you will send all dup's. This channel is read-only, meaning the agency may only receive messages during this "wiretap." The "wiretap" is only good for a specific period of time. You must not alter the messages being forwarded to the agency in any way other than perhaps wrapping them with additional communications data such as what's listed in item 6.

5. Stretch (small) Creating a message to recall messages sent. Recalling a message may only be done by the message's sender. Recall messages should be delivered to clients. Recalled messages are not delivered as part of the message backlog delivery (from item 3. Recalled messages are not deleted from the server.

6. Stretch (small) Wrap message stored in the system with the sender's and receiver's IP address. This is for CALEA compliance.

[1] see `https://www.eff.org/pages/calea-faq#2`

7. Stretch (medium) Add a "parental control" feature, watching and flagging messages for inappropriate content. Examples of inappropriate content could be vulgarities, phrases that suggest violence, sexual imposition, bigotries of many sorts. This is not intended to make Prattle be *THE* moral authority, but instead create the capability where message content (start with text) can be flagged as violating some criteria. The flagging may be done off-line.

8. Stretch (large) Do item 7 in line with message delivery without significant delay.

9. Stretch (small) Search for a message based on some attribute: e.g. sender, receiver, timestamp, ... This is for both end users and for CALEA compliance.

10. Stretch (medium-large) Use `Apache JMeter`™ to run system tests to determine any or all of the following:

    - Stress testing to identify the maximum number of concurrent messages that may be received or sent by your Prattle server.
    - Performance testing to identify the system's response time (measured as time of message receipt to time of message send) under multiple levels of load. Options for load include:
      - Number of concurrent users (e.g. 10, 100, 1K, 100K, 1M)
      - Number of concurrent messages

2.2 Environment

1. Dynamically turn on or off logging in the system. (This presumes you have logging embedded in the system.)

2. Stretch (small): Send failure notices to your Slack. The notice should include the level of criticality, what failed, and a synopsis of the failure. The entire text of a typical Java stack dump is not useful.

3. Stretch (small): Send security concerns to your Slack (as a placeholder for informing Security). This could be failed log-ins, messages to non-users, etc.

3 Reminders

3.1 Guidance

You own the system and the path. So achieving the goals will involve extending or creating additional classes and methods as needed. You will need to make many design decisions as well as decisions about which external libraries to use. 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.

You can add your own stretch goals. 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.

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 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.

Stay in touch. 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.[2] 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.

Don't forget to do `Team-mates`.

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 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` using 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.

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. 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.

[2] Ideally this should be a daily thing. Every other day is acceptable, but every day is better. Less often will lead to issues.