Team 403 (J. Borzillieri, A. Flores, Y. Liu, M. Sun)
CS 5500 - Silicon Valley
December 9, 2018

Prattle/Chatter

Team 403

Summary:

The goal of this project was to extend the functionality of the Prattle and Chatter code bases. As a team, we spent the first sprint learning the legacy code and identifying where most of the modifications would be taking place. It was also required for us to familiarize ourselves with the tools required to adhere to project guidelines, which included the use of Jenkins, SonarQube, Maven, and Slack.

We achieved this by sectioning large portions of the project to one another. Responsibilities were sectioned off into AWS and environment responsibilities, messaging functionality, database management, and user login/authentication. This eased the workload and made it so not every member needed a full understanding of what other members were working on. Our strategy also made it so we could work on our branches without needing excessive check-ins, and allowed us to specialize on more quality, and comprehensive functionality in our assigned domains.

We utilized an Agile form of development by building day-to-day and conducting regular correspondence between team members. Our communication was one of our strongest assets, and it allowed us to effectively decide which goals were feasible to achieve during the sprints, which allowed us to delegate those responsibilities fairly and accordingly.

Result:

The result of our work has culminated in a functional instant messaging server with reliable individual as well as group messaging functionalities. We have also implemented secure login functionality that uses encryption to store and retrieve user passwords. Our CRUD functions utilize the (.csv) format to organize and store user, group, and subpoena data. The database is held in the project directory under the title "Prattle_DB" along with the following 4 subdirectories, each containing their own arrangements of subdirectories:

- The "User_Data" directory contains the "All_Users.txt" file which includes descriptives for all users currently in the system. This file contained pertinent user data which helped us to achieve the encrypted login functionality outlined in sprint 2, as well as authentication. User_Data also contains the subdirectories labeled by usernames which contain all sent, received, and missed messages in files respectively names "Sent_Messages.txt," "Received_Messages.txt," and "Missed_Messages.txt." These directories were used to store information pertinent to private, public, and group messaging functionalities. These directories are also read in the case of a subpoena for one of our users (discussed further in the point 4).
- The "Group_Data" directory contains the subdirectory "All_Groups" which contains all subdirectories which are labeled by their specified group names. Each group directory contains the files "Group_Members.txt," as well as "Group_Messages.txt." The "Group_Members.txt" file contains no headers, and all users who belong to the group, while the "Group_Messages.txt" contains information pertaining to the time, sender, and message contents.
- The "Flagged_Messages" directory contains all information pertaining to messages that go against the community standards outlined in Naughty.java, which was included in the sprint 3

requirements. The database is structured to create a new directory for every user who is sending inappropriate content. That folder contains a text file labeled "Naughty_messages.txt" which contains the date sent, message sender, message contents, the receiver of the message, and the sender's IP address.

- The "Subpoenas" directory contains information pertaining to all requested subpoenas. Within this base directory are subdirectories of requesting agencies. Those requesting agency directories further branch into directories labeled with the username of the individual being subpoenaed. In the event that there are multiple subpoenas for a single user, that directory branches into subdirectories labeled with the proper subpoena ID tag. The subpoena ID file then contains all messages sent and received by the specified user within the period of time specified by the subpoena. That information is stored in the files "Sent_Messages.txt," and "Received_Messages.txt."

There are also a number of classes that were added to achieve the functionality associated with receiving user data before writing the pertinent info into storage. That work can be viewed in the "utils" package, and all crud functionality is located in the "im" package under the name "CRUD.java."

Discussion:

Our team engaged in regular communication on Slack to stay up to date with each other's activities. We also held meetings at least once per week to demonstrate each other's progress, while we also engaged in one-on-one meetups both on and off campus throughout each sprint.

One person was in charge of creating and maintaining a coherent file structure as the project moved on, and more functionality was weaved into what was already existing. This allowed other group members to focus on their corners of the project and allowed them to plug in the functions that allowed for persistent data storage (i.e. groups existing while logged off, user records, and an empirical database of flagged messages).

Another team member was tasked with encryption, login, and authentication processes, as well as defining a base and set of methods which provide the functionality of flagging inappropriate messages, and inappropriate user activity.

The next team member oversaw the upkeep of our server, and integrated much of the developed code into the master branch, including the wiretap functionality . this member also was responsible for a great deal of the environmental requirements and stretches outlined in sprints 1, 2, and 3. This involved lots of testing and bug fixes.

Another team member was in charge of the private messaging between users online and offline, as well as obtaining IP addresses from the client sockets so that information can be stored with user and messaging info. This member also implemented the recall functionality which allows users to recall messages which have not yet been delivered to offline users.

This could have been improved by more in-depth communication between the database manager and group members one-on-one to understand their specific needs. The specifications were mainly discussed while the entire group was meeting, as well as on Slack during the implementation process. Pertinent communications could have been reduced by better managing this process, which would have saved everyone a lot of time. Another shortcoming was inadequate documentation throughout the development of the code. This clearly could have provided for more time to extend the functionality rather than deciphering one another's code.

Overall, though, we agree that we had one of the stronger teams relative to the rest of the class. Feedback was open and all group members were receptive and accommodating in terms of one another's needs. Everyone worked hard to accomplish their tasks, and that work has culminated in a solid extension of the legacy code.

We could improve on our goal setting and tracking of responsibilities. For this particular project, we did not utilize Jira to the fullest, but we did maintain day-to-day communications when coding began for each sprint, and maintained a good understanding for what one another worked on.

We did, however, use the majority of our time wisely, and are all pleased with the final outcome of our work. We hit every basic requirement along the way, as well as most of the stretch goals. We enjoyed working with one another and feel fortunate to have had the support we have provided to each other. Our development process was smooth and our actions were transparent, and overall well-documented.

Retrospective:

The team gained a lot of experience working with server/client communications. This was an area that we all had little knowledge of going into this project. While it was daunting at first, our group gained confidence with each passing week. We learned to extend code that was not fully understood in the beginning phases, and we feel as though that process is a valuable this to experience before going out into the industry where such a thing is commonplace when working on massive, distributed systems with unfamiliar team members.

The team definitely felt added stress by working with concepts that were not covered in the lead-up to this project, and it is a commonly held notion that this project would have been much more feasible if the homework assignments leading up to this were related to the final project. This was a big jump from what had been covered in both class as well as the assignments.

For future cohorts, it would be worth considering having a more adequate lead-up to this project in terms of the tools being used, as well as the concepts being expressed directly in the code. Fro example, programming with sockets in the assignments would better prepare us to understand the code, and thus allow us to be more creative while extending the software rather than spending so much time learning the basics. Jenkins and SonarQube were also very confusing at first; a more hands-on and fundamental approach to these tools also would have allowed for time better spent by the group. Upon seeing the outcome, these tools were not as difficult to use as they were to figure out. We feel as though this should be addressed for future group work associated with these technologies.

Many of us also feel as though the grading is not representative of our ability in the field, or our efforts in this class.