MANAGING SOFTWARE DEVELOPMENT SPRING 2019

PROJECT REPORT

Members:

Animesh Mane
Atanas Pavlov
Dharmish Shah
Karl Obermiller
Manush Patel

Instructor:
Dr. Michael Alan Weintraub

CONTENTS:

SUMMARY

As a high-level overview, the goal of the project was to reach a level of functionality that would be comparable to a mature and established communications product. This meant that the system would be designed from the ground up for future scalability and expansion. The initial stages of preparations were spent almost entirely in research and prototyping the various aspects of the system, prior to committing any substantial portions of code, as well as thorough testing of the provided legacy code portion.

The first major goal for our team was to immerse ourselves in the legacy codebase given to us, and thoroughly understand it’s strong and weak-points. This meant both manual code walkthroughs as well as the formation of a robust testing suite. The next goal we established as being crucial, was future planning in the form of UML Diagrams, which would guide our development process to come, and allow us to properly incorporate new features into the existing system.

The next major goal we had was to extend the basic in-place messaging such that it would resemble any modern communications software. This meant point to point messaging (Person to Person), as well as the ability to target multiple predefined users with a single message (Group Messaging). Extending this point, we also wanted to have all interactions with our system be recorded, and that was our next major goal. Persistence was an all-encompassing task, and it is something we focused on greatly, as we wanted to have it done in a truly reliable manner.

Once we had a plan, as well as the basics mentioned above covered, we tended to more advanced features, that we felt would differentiate our product from among the already established players. We wanted our system’s users to feel safe when using our product, and so one of our major goals was to establish end-to-end encryption. This could have been done in a very simple manner, however the result would be essentially that, once server-side, all messages could be decrypted with a master key of sorts. That would be putting the user second, and breaking the sacred seal of trust, so, instead, we decided we wanted to implement truly safe messaging. That meant that once a user sent their message, the only person who could decrypt it, was the recipient of that message. The result would be that no backdoors would be present in the system, and users could feel safe using our messaging platform.

Our final goals were in relation to user convenience when using our platform. To facilitate this, we decided on two major additions. The first one would be a message queue system, which would allow target message opening, as selected by a user. This meant that they would not be barraged by an influx of messages when first logging in and could instead pick and choose based on their preference, what they wanted to see. The other aspect we wanted to touch upon was accessibility for our non-English speaking users. To accommodate them, we made it our goal to support, on the fly message translation if requested by our users.

RESULT

While the above goals may seem lofty for such a short time span, as a team we managed to accomplish all of them and then some. We had some highly productive initial meetings, and very quickly formed a team spirit that would last us through the length of the project. In addition to a very high degree of testing, our first Sprint also included a thorough UML chart that would guide us through the rest of the project, as well as detailed UI/UX mockups that helped us concretely express our vision for the future. During the time given in 4 sprints we defined our goals in two categories: base goals and stretch goals. We planned in such a way that all of our base goals are achieved and we also get extra time to achieve stretch goals. These goals were decided on the basis of discussion between all the team members and also keeping in consideration what client suggested.

Sprint 1 :
Goals we thought were necessary and delivered were, we missed one goal which was achieving 80% test coverage and missed it by nearly 2% less coverage.

1. Construct basic UML for high level design
2. UI/UX design prototypes for various flows
3. Everyone should be able to run all tools locally, and should be able to merge a branch.
4. Write tests for existing code (Prattle server)
5. Exploring Jira, and familiarizing ourselves with the platform
6. Prioritizing features based on stakeholder input

Sprint 2:

We deprioritized UI/UX related goals and implementation of conversations and user roles as basic functionalities took time to get working. Major achievement was to merge the code to the master branch and achieve necessary coverage as well as modifying the network code in such a way that it allowed us to send and receive multicast as well as unicast messages. We could not achieve Client related goals as well but we added major persistence related classes on the server which was not initially part of our base goals but we decided to prioritize as Vaibhav suggested it should be a deliverable for the sprint.

Base Goals

1. We will close out the last sprint, merging all branches to master and getting the coverage up to quality standards.
2. We will create basic implementations of Groups and users in prattle.
3. We will add new message types to accommodate new features, including login, register,

4. chat multicast (group messages), chat unicast (private messages). These message types may change on further consideration.

Exemplary Goals

1. We will begin working with the Chatter client code and provide various functionalities.
2. We will implement basic Conversations on the Prattle server. These would not be functional beyond storing lists of messages for groups or private conversations on a per run basis with no persistence.
3. We will implement basic user roles as per our UML from sprint 1.
4. We will add the ability to add users as group admins on both the server and client side.
5. We will implement private or invite only groups on the Prattle server.
6. We will begin architecting basic database design for data persistence, including users,
7. groups, and messages.
8. We will setup a basic graphical UI that can perform some degree of basic communication with the Prattle server. This GUI need not necessarily implement all features on the Prattle server, but simply display that communication is possible.

Sprint 3:

During this sprint majority our focus was towards making the client work and provide a proper dialogue with options that let client have control over the client application. We also extensively tested the actual integration apart from writing test cases and uncovered bugs. We also merged integration of persistence to all of our domain objects and started persisting them on a remote heroku based dataset. We could not merge profanity filter in the client but we had it ready and tested at the end of the sprint. Also we did not work at all on profile section as the client part took more time than expected and we underestimated the client tasks.

Base Goals:

1. Implement console functionality thoroughly.
2. Re-examine communication initialization flow between client and server.
3. Clean up and fix multicast, unicast, and message interaction between client and server ensuring that all features work perfectly.
4. Allow for targeted message opening on Client, only open Messages that the client user
5. wants, not every message that was sent to the client all at once
6. An initial Message would contain from whom the User has messages from, and
7. options for opening them individually
8. Add more Group functionalities
9. Complete data persistence, so everything that needs to be persisted such as messages is.
10. Handle “deleted” messages by marking them as Deleted, but not actually deleting them.

Exemplary Goals

1. Add Profile section per User
2. Add Profanity Filter
3. Add Parental Control to Groups (Profanity Filter Enabled / Disabled)

Sprint 4:

This sprint was more of polishing what currently exists and refactoring for better design. We added new group functionalities every sprint after 1st sprint and continued to do so in last sprint as well. We were able to achieve all goals except for the backlog of user roles and conversations. User roles was a concept that we thought would be helpful at the start of the design phase, but as we moved on we found better ways to implement the same concept without adding new class in class hierarchy. Conversations also were handled very well by our database operations so we decided we could work on important functionalities such as responses of commands from server and encryption during this sprint. We achieved all of the following tasks except for timestamps and conversations along with multiple language translation and rest of the stretch goals achieved.

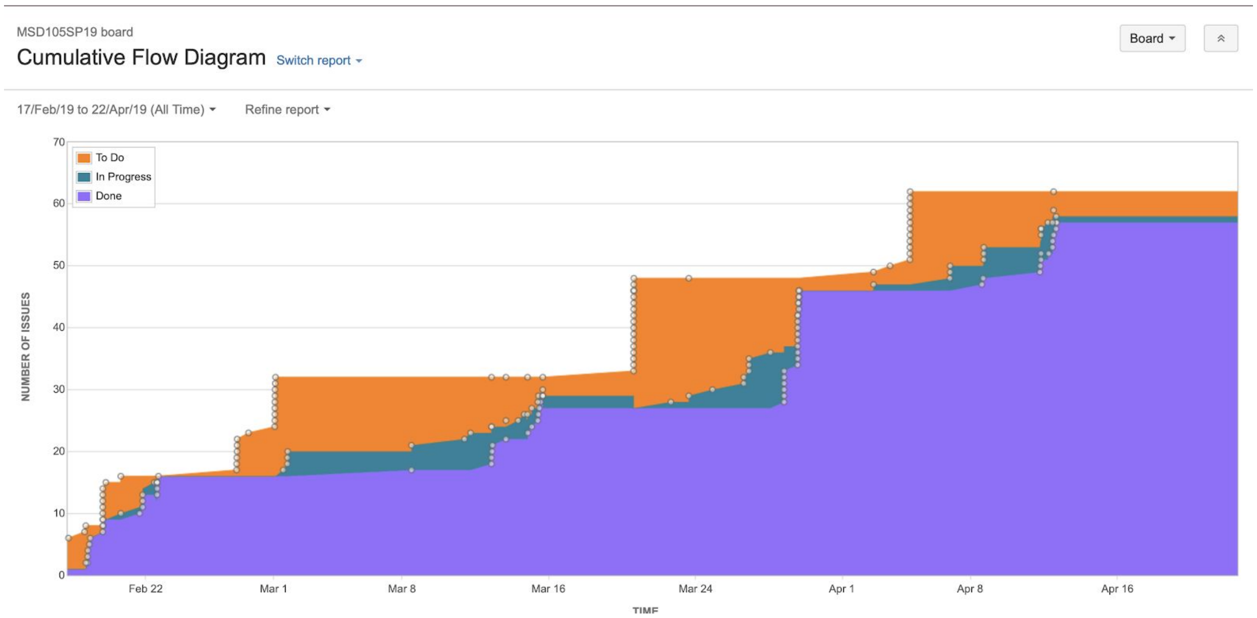
Base Goals -

1. Show all active users on the console users.
2. Provision to add multiple moderators in a group
3. Implementing the conversations
4. Redeploying backend on the aws
5. Displaying proper responses on the console for user on execution of the process.
6. Deleting private and group messages which were sent by users.
7. Encrypting and Decryption on messages from client to server and vice versa to maintain privacy of user data and confidentiality.
8. Integrating profanity filter on messages to filter out inappropriate words from the messages.

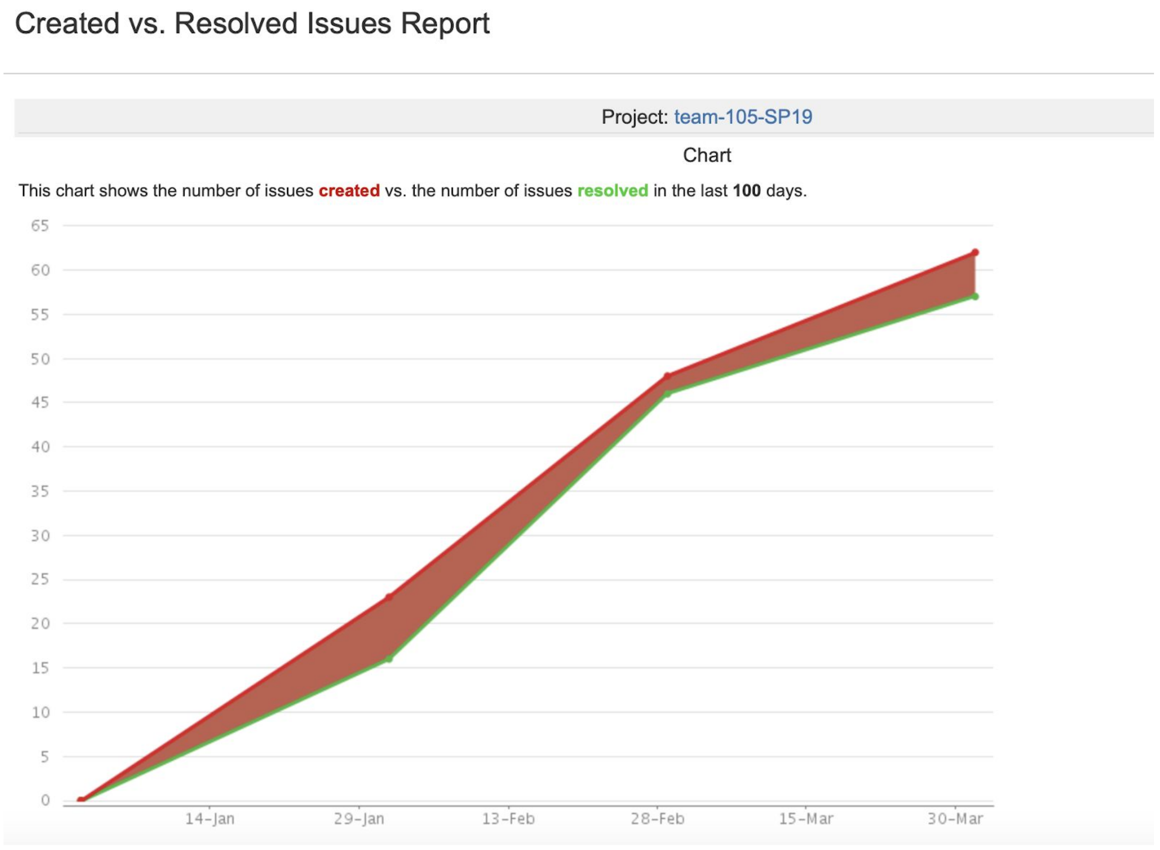
Exemplary Goals -

1. User Profile
2. Internationalization (Spanish)
3. Showing Timestamp of messages based on time zone
4. Integrating encoding and decoding mechanism for the images

Cumulative Flow Diagram - Displaying Number of Issues across Time



Issues Creation and Resolution Diagram - Displaying how tasks propagated through time


Created vs. Resolved Issues Report

Project: team-105-SP19

Chart

This chart shows the number of issues **created** vs. the number of issues **resolved** in the last **100** days.

DEVELOPMENT PROCESS

Our goal was to create a chat messaging system. We followed modified version of Agile process development using SCRUM. We had 4 sprints, each sprint for 2 weeks and every member got a chance to be a Scrum master for each sprint.

Sprint 1

First, we went through product backlog and broke it into number of sprint goals prioritizing the important features. As a first timer for jira and Jenkins , we created tickets, went to various jira functionalities and also though documentation if required. We also committed and tried pull requests in github to get familiar with continuous integration of Jenkins, Jira and GitHub. We faced a deployment issue on Jenkins and tried to solve it and discussed possible solutions which was later resolved. As legacy system was given to us, we thought to get familiar with the system and also wrote test cases for the same giving good quality coverage. As this was our first sprint, we focused on requirements analysis, design and architecture of the product and how it will look like at higher level.We had various meetings with client discussing their priority features and use cases of various scenarios. We worked on UML and class diagrams of the system.

Sprint 2

As for this sprint, we already had our infrastructure for continuous integration was ready and our focus in this sprint was on client's priority features. The major goal of this sprint was to get the basic functionality of sending private and group messages working on server. We designed user, group and messages models. We also implemented persisting the models using Hibernate. We implemented message handlers for login, registration, creating groups, adding members to group, removing members from group, sending private and group messages to users on Prattle server. We maintained the code and made sure they meet application quality standards. We planned to integrate data persistence with Prattle Legacy system, but we faced issue in setting up network communication after addition of message handlers, so integration of data persistence was pushed to next sprint. On client side, to make this product usable, we used command line UI for user operations of product.

Sprint 3

As for this sprint, we made sure that basic operations like registering, login, unicast, multicast and group related crud operations which were key in this product, were working and integration of client and server is done and also data persistence is integrated with Prattle server. We used our own designed JSON format for communicating requests and responses with client and server. It was time consuming to get the network messages on client and server channels working and it took more time than expected. We gave users option to login, register, send private and group messages. He also got option to create and delete groups, adding and removing members of the group. We gave support of sending JSON requests from client to server.

Sprint 4

We were pretty much done with basic functionalities of the system, so we focused on advance features of the product. We implemented internationalization supporting upto 5 languages giving users flexibility to read messages in his preferred language, encryption and decryption of messages for maintaining user data privacy, joining a public group and adding multiple moderators. We implemented and handled server responses on the client and showing appropriate messages to user. We also faced some stable database connections issue on remote heroku connection which was solved by deciding to use in memory H2 database. Apart from that, on client side, we implemented encoding and decoding of images and profanity filter which removes offensive and inappropriate messages.

RETROSPECTIVE

Overall, the project was good because it offered a look into how real projects in the workplace might play out. It gave each team member the chance to not only work on a significant project while having to communicate effectively with their team members, but also as the scrum master the chance to experience being in a leadership role and develop skills in that area which is not particularly emphasized in most other classes. It also emphasized deadlines in a more realistic manner, and allowed us to experience the pressure to deliver on a deadline. As we set our own goals for each sprint, we had to learn what reasonable workload for the sprint looks like for our team and how to manage and prioritize the goals that we set.

An issue with this format is that it feels as though a significant amount of work is required to achieve all the goals laid out in the project description. There were a wide variety of goals presented initially, and it seemed as though they were all expected to be completed when of course completing all of them would be unreasonable to expect in just two months. The pressure to complete such a large list of goals in such a short time frame makes it feel as though we must put in many hours each week, which was not helpful to have looming when other classes had deadlines and tests. Perhaps in the future the feature list could be reduced, elsewise laid out in a way where it is more clear that not all the goals must be met to do well on the project. Perhaps there could be more of a “must haves”, “should haves”, “nice to haves” structure to the list, as opposed to the straight list of desired features.

The license to control our stack was very welcome. The ability to choose our libraries allowed us to include helpful libraries that simplified some of the more complex tasks, and also allowed us to experience the process of investigating libraries to determine if they are feasible to use and would meet our needs. It also allowed us to work with various technologies at the same time, and the freedom to choose which technologies we used let us play with our strengths in what we already know, but also experiment with new technologies that we’re interested in.

There was, however, a significant amount of confusion around what was acceptable to use and what was not allowed. When we were initially considering creating a graphical interface, we wanted to use a spring boot server to serve a web interface, but there were previous statements that spring boot was not allowed to be used because it would trivialize the Prattle server’s functionality. It was not made terribly clear that the use of spring boot as an independent server that interfaced with Prattle was acceptable, which is what we had wanted to do. We also wanted to use the Hibernate library to facilitate database persistence, but as this was used in spring boot it was unclear whether that was acceptable for use as well. More specific guidelines about what technologies and libraries are acceptable (or which are not acceptable might be easier) would vastly improve the experience, especially early on.

Added clarity in the realm of the client would be great as well. The focus of the project was on the server, however a lot of the features requested seemed impossible to do without modifying the client in some way. The lack of documentation in the code and clarity surrounding seemingly client-dependent features left us floundering as we tried to get going. Additionally, the complication surrounding the client due to the fact that it is not tested and somewhat hasty made working with it frustrating and in some cases difficult. We were told not to merge it to the master branch even, which meant extra management required on the repo. Some clean up of the client and expectations for it would vastly improve the project experience.

In conclusion this project was a worthwhile endeavour, and definitely exposed us to something that is not typically seen in most courses. Having a taste of the real world of development, on a reasonably sized project was eye opening experience for a multitude of reasons. The biggest perhaps, compared to most other academic projects was having stakeholders that had to be satisfied. Rather than individually coming up with features, and prioritizing them based on our own preference, we had to actively accommodate the ever changing wishes of the people who tasked us with the project. Even with all the initial administrative issues we faced, this was still a great project to be a part of, and we came away from it as better developers.