CS5500 PROJECT REPORT

TALKWORKHERE

Team-201

Kunal Patil

Ketan Kale

Rahul Bhat

Sachin Haldavanekar

A. Project Goals

The summary of project goals can be classified into multiple categories as follows:

Goals based on Requirements

We were provided a legacy code base of a chat server and a requirement backlog which made it clear that our goal was to enhance the current code to deliver a simple but effective communication application which would help users to communicate with other users' individually but also in groups. A user was supposed to be able to register and login to the system. A logged in user would have the facility to view all his past conversations, be able to search for other users using their name or username or look for other online users, perform CRUD operations on a group after becoming the moderator of that group and also being able to add or remove users from that group. User was expected to be able to invite other users to join a group which they are a member of. Users should also be able to follow or unfollow another user. The system was expected to support the facility of tapping a user's activity allowing a user to privately reply to a group message, recall or delete any sent messages. Parental control, support for different languages and time zones along with the ability to transfer multimedia content, alternate authentication methods were some of the other goals that the system was expected to deliver to the end user.

Goals based on Code Quality

Although getting the functionalities added was our utmost priority, one of the major goals were to ensure that the code quality was always kept in check. Ensuring a minimum of 85% test coverage (which included each of statement, method and branch test coverage) was one of the necessary requirements as that was the minimum threshold to merge the project code to master. The source code was expected to pass the quality checks configured in the SonarQube server where each pull request would get reviewed before making it eligible to merge with the master branch. Other code quality goals included ensuring minimum or no code duplications, usage of design patterns wherever possible, having complete and meaningful Javadoc for all the major code components like classes, methods, constants etc. Sonarlint checks on the local systems was expected to serve as guidelines to eventually pass all the Sonarqube checks during code deployment. The code was expected to be modularized into relevant packages and follow the software development best practices.

Goals based on Processes

The third category of goals were related to process which included following the Agile Methodology during the four sprints. Jenkins was the tool responsible for Continuous integration which would leverage the maven dependency management system and Sonar quality checks to ensure that every piece of source code which was built into master passed the four stages of Build, Test, SonarQube checks and Master branch tasks. The team was also expected to follow the scrum methodology and have daily standups, regular in person meetings, sprint planning, backlog grooming, sprint retrospective and a demo to the product team at the end of every sprint. The development process needed the team to create a pull request every time a new feature, bug fix or a minor change was added to the code base. Thus, constant code review and feedback was expected to maintain code quality and bring different ideas to the table while all the functionalities were being developed. Git and GitHub were tools to be used for versioning in a distributed manner while keeping a history of all the code changes. Developers were expected to user Jira – a sprint management tool, to attach git commits to task tickets which was also known as smart commits. Jira was also expected to be leveraged for analyzing the sprint progress using the different reports provided by the tool.

B. Result

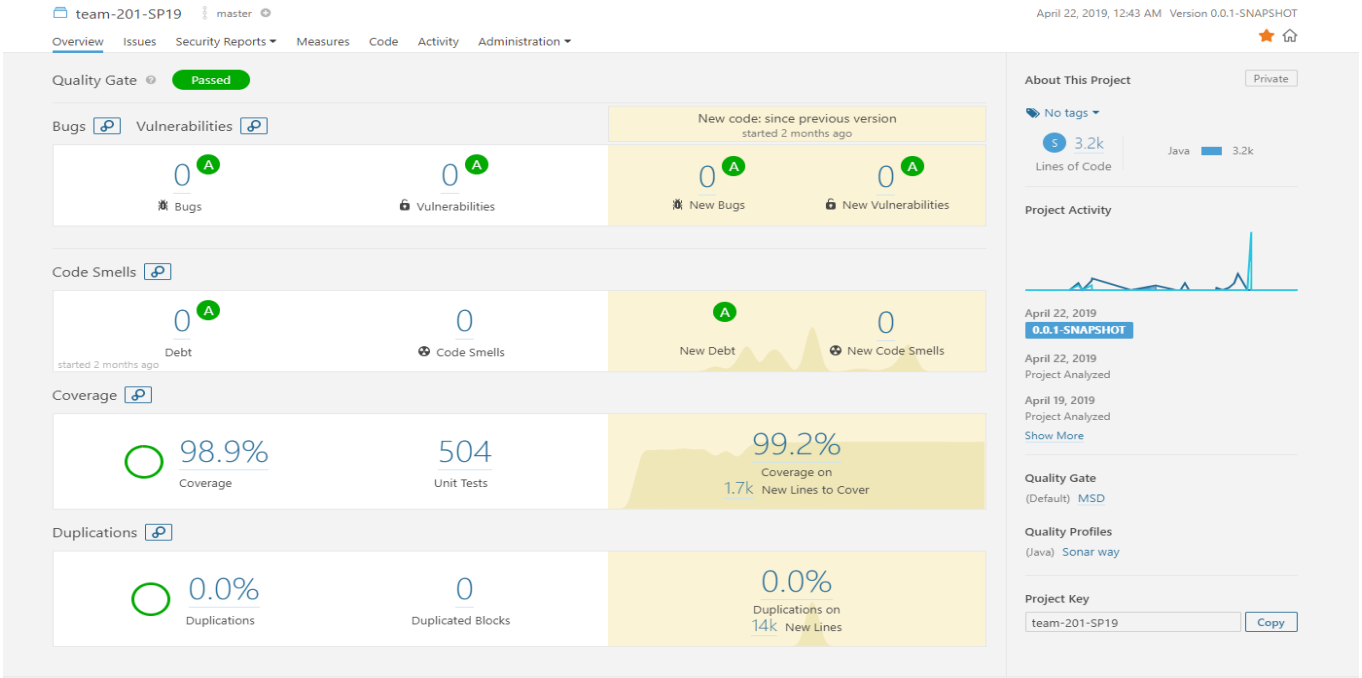
After four sprints we were able to achieve a vast majority of the project goals which we had prioritized keeping the user as the focal point of all decisions. We can present the results in line with the project goals starting with the functionalities achieved.

Requirements Goals Achieved

Users can register to the system and login using their credentials, once logged on they receive any messages that were queued up while they were offline. Users can update their profile attributes, like searchability. A user can search for other searchable users and follow or unfollow them. They can also see which users are online. A user can create a group, leave it, invite other users to the current group, also search for groups using the group name. The moderator of the group can update the group settings like group searchability, they can also add /remove users from the group and approve invitations initiated by other group members. The moderator can also delete a group. Our system allows a user to send messages to another user or to groups that the user is a part of. In our system we have keys that identify each message uniquely. This unique key allows us to provide the next three functionalities which are replying to a private message, replying to a group message privately and deleting / recalling a message. The system also allows a user to fetch his past messages. Our system stores all the messages permanently. So, when a message is deleted, we just mark it deleted in our database and not actually deleted from the database. We, thus, allow retrieval of past messages anytime. Our system allows a 3rd party to do surveillance on a user of our system. So, for example, if Government wants to do surveillance on a system user it can do so after providing a subpoena. In that, the 3rd party can tap a user, get conversation history for the tapped user and get notified when this tapped user logs in.

Quality Goals Achieved

We were able to cross the 85% threshold of code coverage in the very first sprint and were able to merge the code to master. This was possible since the team entirely focused on adding as many unit tests to the legacy code base as we could. We also followed a practice of removing any existing code smells in the existing code base while ensuring none of the new code introduces any. Eventually we maintained a code coverage of well over 98% with a total of 504 unit-tests.



There were 0 bugs and vulnerabilities and the same number of debt or code smells. The Maintainability rating was A on all code quality attributes. We have also ensured that the code is modularized into the separate packages based on the different types of classes we have used.

Process Goals Achieved

Since we were good on the quality and test coverage in the very first sprint this was a perfect scenario to kick of sprint 2 as we could focus on following all necessary process like creating pull requests having multiple code reviews, only using smart commits while working on new features since we were in control of the speed of the sprint. We were able to deploy and run our chat server to a remote AWS EC2 instance while the database was hosted on Amazon RDS. We followed the CI/CD process while delivering every feature right from sprint 1. Setting up the Jenkins config file was a challenge when we added the MySQL connector jar as a dependency to the project and the dependencies were not getting built on the Docker container in the Maven build. We also ensured that we followed all Agile scrum processes by having regular meetings, standups and the weekly demo, retrospective and sprint planning to have our tasks prioritized for the upcoming sprint.

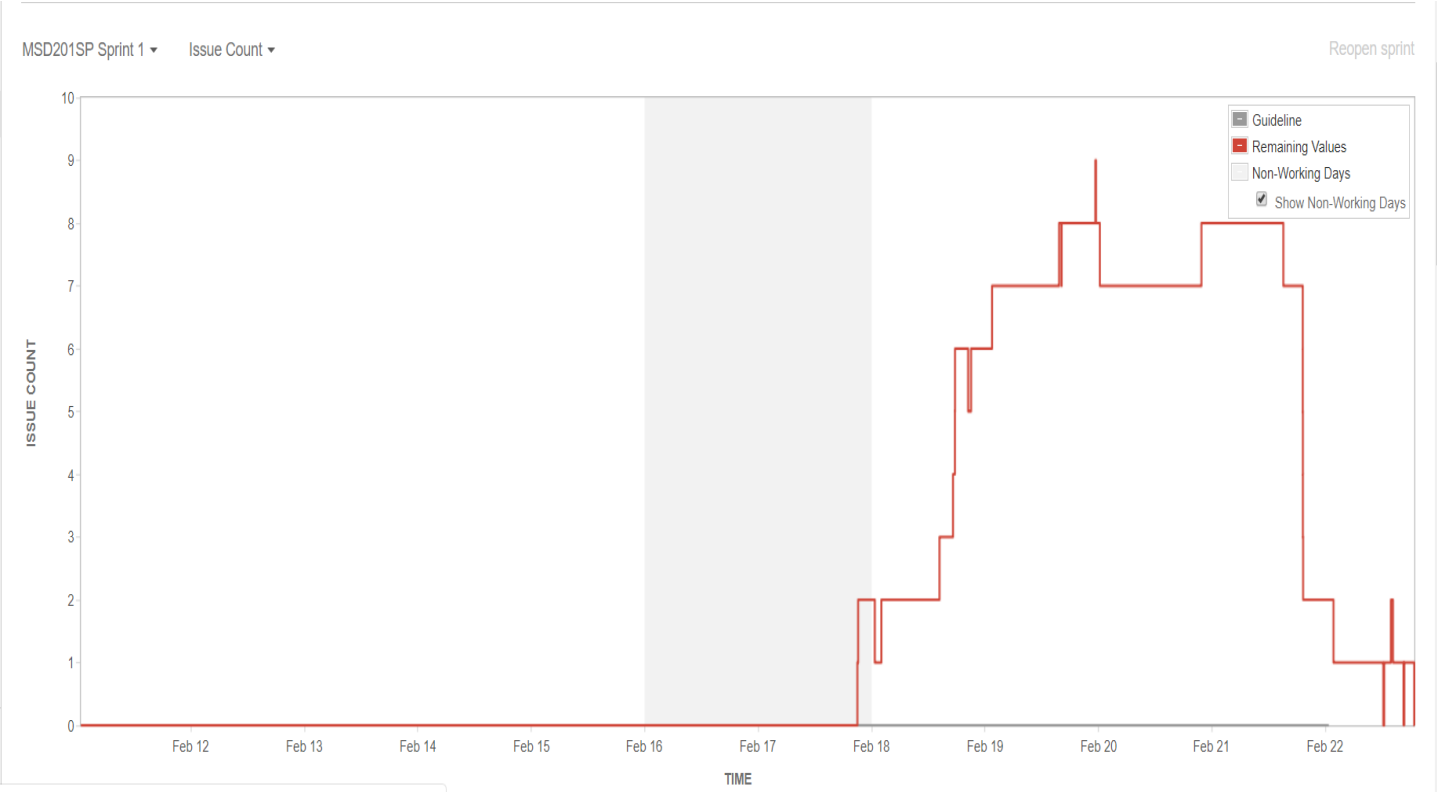
C. Development Process

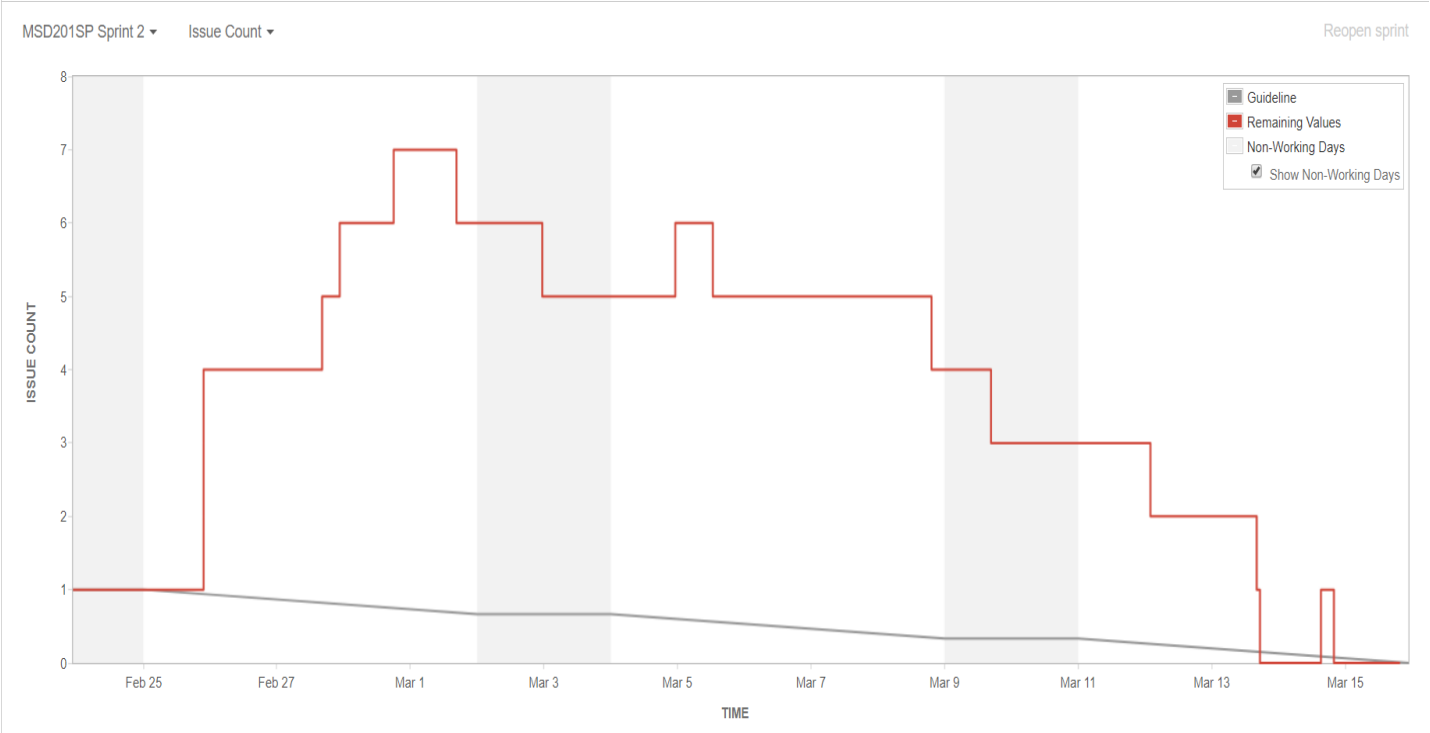
Overall, we, as a team, followed the agile scrum development process which helped us plan and execute our goals in a well-structed manner and to accommodate all the feedbacks without having to add extra tasks. To walk through our team development process initially we were given a product backlog out of which we prioritized and selected specific tasks which we would work on in every sprint. These tasks would then become our sprint backlog for that sprint. Sprint goals also included all the feedback given to us by the product manager who is our TA in this scenario. We worked on these sprint goals for next two weeks. And at the end of the sprint we had to give a detailed demo of the sprint goals achieved in that sprint on which we would receive feedbacks Which helped us understand our mistakes and scope of improvements. This gives a brief description of the overall sprint process.

During each sprint we had stand ups every day. We also met in person for a minimum of 4 times during every sprint which helped us understand each other's capabilities very well. We followed a fix pattern for stand up where a team member first talks on the tasks which he has been working on and/or discuss any impedance he had been facing, if any, during tackling that issue .This helped him to get inputs from other teammates on solving the issue. The scrum master could also keep track of these obstacles until they are resolved. After that he would assign himself the next task that he will be working on. Based on all these the other team members could raise any concerns if they had. We also ensured that no code would be merged into master without being reviewed by other teammates and that every member in the team reviews the code before merging it into master and before marking the task as done on Jira. We, as a team, insured that each team member is making smart commits which help us track the progress and the history of a specific task. As we had stand ups daily, everyone was very much updated about each other's progress and if there were any blockings due to someone else's task.

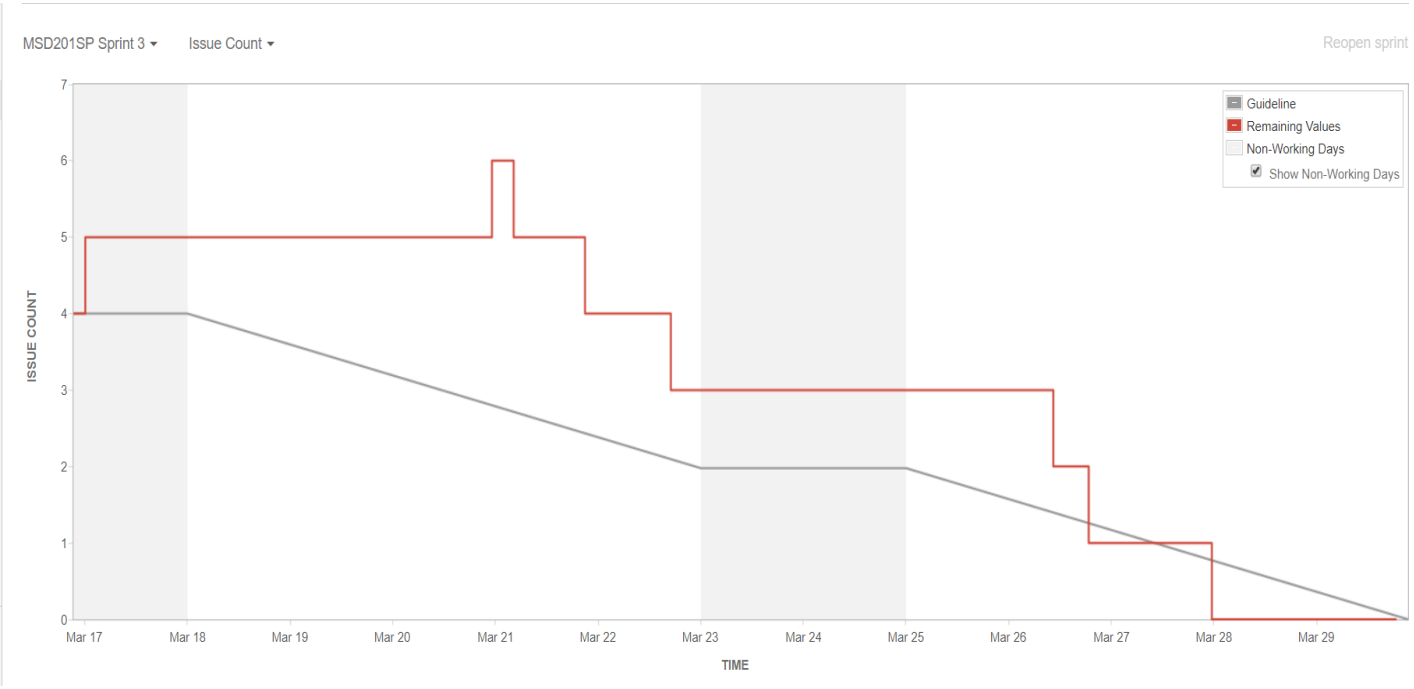
Another key aspect of the development process was to work efficiently as a team. **T**ogether **E**veryone **A**chieves **M**ore can be very well abbreviated to the word TEAM. This project taught us a lot about teamwork and challenges faced while working with people from different technological backgrounds. We had trust in each other's capabilities of achieving our respective goals both qualitatively and quantitatively. We could easily approach any of the teammates regarding any issues or troubles for help. Working in team required a lot collaborative work which also included pair programming so there was constant review for the code. Daily reviews and constant feedback from teammates over the code allowed us to maintain consistency in coding patterns and to ensure that everybody was engaged in development process. Further,

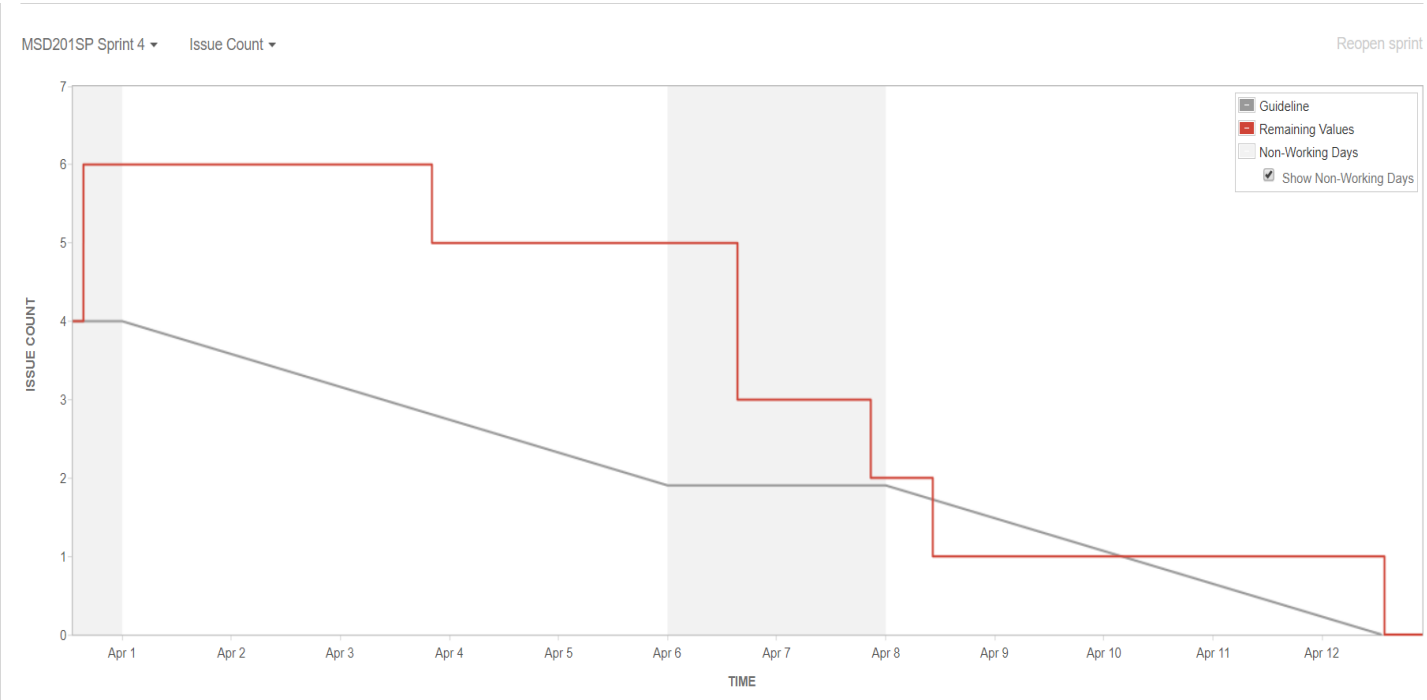
to have insights about different tasks in the development process we used an Issue Management system – Jira which made everyone aware about the ongoing and completed tasks. Using Jira helped us manage and achieve our sprint goals well in time. It provided us with guidelines for every sprint based on our estimates for the tasks in that sprint. For all the sprints we tried following the guild line curve in order to complete the development process.





As it is clear from the above charts that in the first sprint, we had not estimated the task very well as we did not have a good idea about JIRA. After some reviews and understanding on working of JIRA estimates we did a good progress in second sprint.





Going ahead we started using JIRA more efficiently and tried estimating the issues more accurately. As a result, we followed the guidelines as much as possible. This helped us achieve the sprint goals on time.

D. Retrospective

As someone has correctly said, “the journey is the reward”, we believe that the process that we adopted for the development for this product was the best part. In that, we learned a lot of things. We learned the various aspects of software development life cycle like requirements elicitation where we had to translate the product requirement backlog into system specific requirements that we would then discuss within the team and then convert to Jira backlog items. This proved a useful exercise as we learned to understand a product team’s language and translate them into good system specific requirements. What also went well was we learned the importance of team work as amount of work that we had to achieve wasn’t something an individual can take up and work on as there were various challenges on the way which were resolved quickly because of the different perspectives that the team members brought from the their personal experiences. We collaborated well because we made in person meetings a priority and there would be around 3-4 instances in a sprint where the team would meet up and work together for a stretch of five to six hours all the while coordinating and planning what the next steps were going to be and how the current work would make it easy or difficult to get it done. All the above activities meant we had to manage our time well and as all of us had other course commitments and this project demanded a lot of our time and involvement. We ensured that if any of the team member was unable to devote a lot of time to the project during a sprint, then others who had the time would make up for the time lost and this was done by everyone at various stages in the four sprints.

We also learnt the importance of honest communication where people would own up to any tasks that seemed difficult or they weren’t unable to get done in estimated time. Our daily standup would act as a platform for someone to put their hand up and say that they needed any help or would take more time to do some task. This ensured that we did not have any surprises at the very end of a sprint and thus we were always able to deliver the committed goals. The sprint end meeting with TA was an important part of the two-week sprint as we were learning an important job skill I.e. presenting our work to the product team. Planning for the next sprint involved us knowing our time limitations and making a calculated guess on deciding the goals we could accomplish; estimation was another skill that we failed and learned heavily on as we went wrong on multiple occasions. We believe we are in a better position in terms of being able to estimate tasks of different complexities.

Testing was another major aspect of our two-month experience of working on this project. We learnt how to mock dependencies, use Java reflection to manipulate behavior of a private variables and mock them to get the expected values. Testing was challenging since all of us had never done code testing at such an extent before and since we had learnt a lot of it during the assignments and were eager to implement them while also keeping a tab on the time we planned to allocate to this project. Working with network channel and understanding what can be tested and what cannot be was another challenge that we faced and were happy to have accomplished it well.

Feedback on the course:

- We specifically liked the course structure of this course as all the homework were completed before starting the project. The homework gave us good amount of practice of building test suites for any code module and writing clean and modular code. This gave us enough time to focus on the development process involved in the project.
- Another thing that we liked about the course was Professor’s real-world examples that he gave to make the concepts easier to understand.

- The session by the Law school students was an informative one. It enlightened us about a less known aspect of software development.
- The session on Refactoring helped us a lot. We took it as an opportunity to get our code reviewed by other teams in the class. This proved to be very beneficial for us as we received valuable suggestions to be incorporated in our code. We went on to refactoring our code and thus made it more efficient and We would especially like to thank professor and other team members for contributing to the betterment of our project.
- TAs in this course were resourceful. They helped us complete this course successfully.
- Time span of the project could be increased with respect to the number of features stated in the backlog