Project: Prattle Chatter
CS 5500 Managing Software Development
Final Report

Team 204
Kasi Karuppiah Alagappan
Neha Gundecha
Ruturaj Nene
Vishal Patel

Table of contents

1. Project Goals

The goal of the project was to build a message application on top of a legacy code. The legacy code was server application with basic setup for socket code. The project aimed at developing the server end more than the client end. Additional features added to the legacy code are.

1. User Login and Registration with unique username.
   Only an authorised user can login to the system. The system supports user registration. The system has unique usernames.
2. Data persistence.
   Every message and user/group information is persisted in a relational database.
3. Password encryption
   The password stored in the database is hashed. So the product owners do not know about user passwords
4. View and edit user profile
   A user can view his profile. Can lookup for some other users profile. A user can edit his own profile. He can change everything other than the username.
5. Follow and Unfollow
   A user can follow or unfollow some other user. If a user follows some user. He is notified of login and logout activity of all the other users he follows.
6. Private messages
   A user can send a private message to some other user.
7. Group messages
   A user can send a message onto a group that he is part of.
8. Retrieve chat
   Users can retrieve both the personal chats between two users and group chats of the group that he is part of.
9. Chat notification
   Users are given notification for each incoming messages. If the user is online.
10. Create and delete a group
    Users can create or delete a group. Deleting a group is only limited to the moderator of the group. Default moderator of the group is the group creator.
11. Add and delete a user from the group
    If the group has approval settings on, only the moderator can then add other users to group. If the moderator settings are off, then all the users who are part of the group can add other users.

12. Change moderator approval
    The moderator approval settings of a group can be changed.
13. Send group invite
    If the approval settings is on users can request moderator to add some other user. The request is in form of an invite.
14. View group profile
    Users can query for a group profile. The group profile contains all the information about the group.
15. Dashboard for systems to monitor the product.
    A deployed application that logs the number of active clients in the system. It helps operation people to monitor the product load.

2. Overview of the result

The team completed all the goals taken in the sprints and also completed few bonus goals apart from the goals of the sprint. The team delivered a full fledged product with a frontend(CLI) and a backend deployed and ready to use by the client.

2.1 Delivered features of the product:

Users:

- Users can login and register in the system with a unique username
- Password encryption with salted hash so that the user's password is not exposed to anyone
- Users can view their profiles and edit them. Users can also view profiles of other users in the system
- Users can follow and unfollow a user to the view the activities of the users in their circle
- A help is documented to preview the predefined intuitive commands for the command line interface

Messages:

- Users can send private messages to other users registered with the system
- Users can send broadcast messages to all the users registered with the system at once
- Users can send messages in a group if part of the group
- Users can seamlessly chat with other users when they are online in the system using the open chat facility
- Chat notifications are recorded when user is offline and receives messages from other users and group

Groups:

- Users can create and delete a group
- Moderators of the group can add and delete users from a group
- Group moderator approval can be changed to add and delete users from a group
- Group invites can be send to add users
- Group profiles can be viewed by all and the moderators can edit the group profile

2.2 Success measuring factors:

Functionalities implemented

We as a team have implemented all the must have functionalities stated in the backlog provided by the client. In the first sprint we spent our time on getting the legacy code to work and understanding the code provided. In the second sprint we got the code up for the client side along with sending user to user private messages. The third sprint we got the functionality of groups up and in the fourth sprint we added few additional features and refactored the existing code.

Quality of the process and the implemented features

The teams focus was more on getting things done the right way rather than implementing a lot. Through all the sprints we focused on designing the code modularly and injecting design patterns like factory design pattern for messages, command design pattern for commands on the CLI and singleton for the database handle making the code more readable and intuitive which could accommodate additional functionalities in future without much refactoring. Our team used smart commits for Github and used Jira diligently. We always went a step ahead with our features implemented and focused on completeness of the feature.

2.3 Completion claims supported by backlog statistics

Feature implementation was thought carefully by all the team members during the sprint meetings and we made use of Jira to manage and maintain the sprints issues and backlogs.
We spent the first sprint understanding the legacy code and writing tests to get the code into work which took most of our time. We picked up pace starting from the second sprint and started with building the features then. The following graph shows the amount of work completed from sprint to sprint and the number of issues created vs resolved through the sprints.

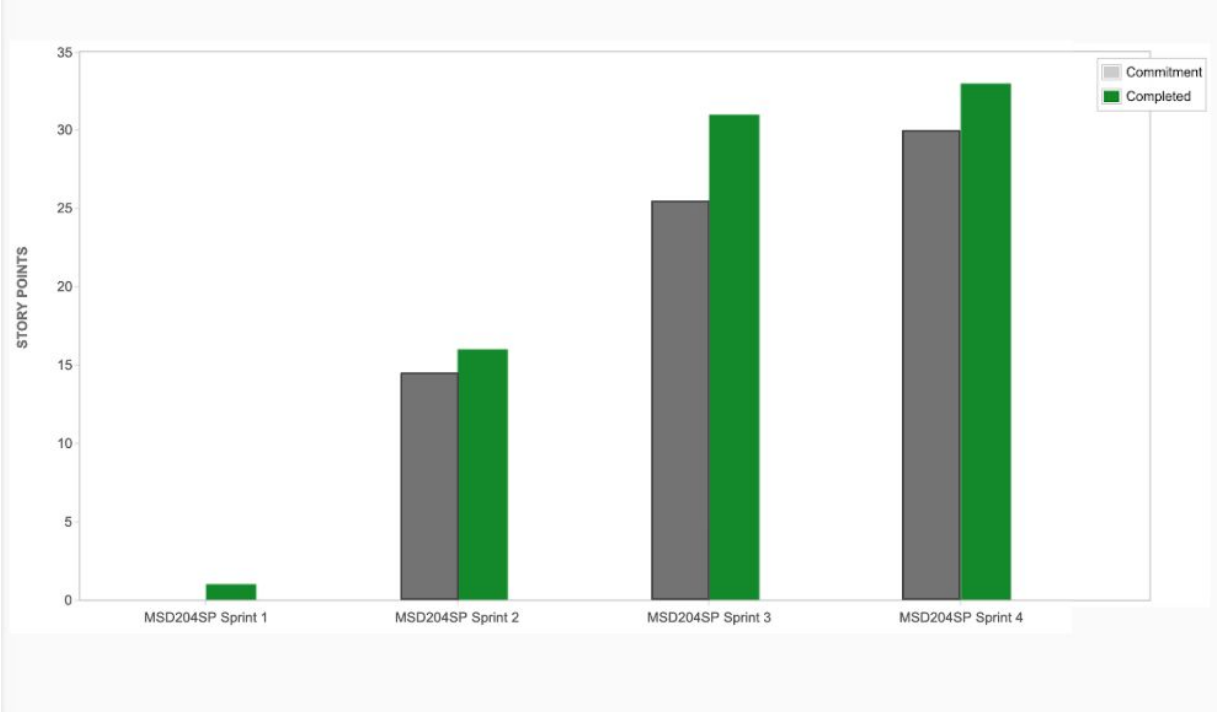

Fig 1 : Amount of work completed from sprint to sprint

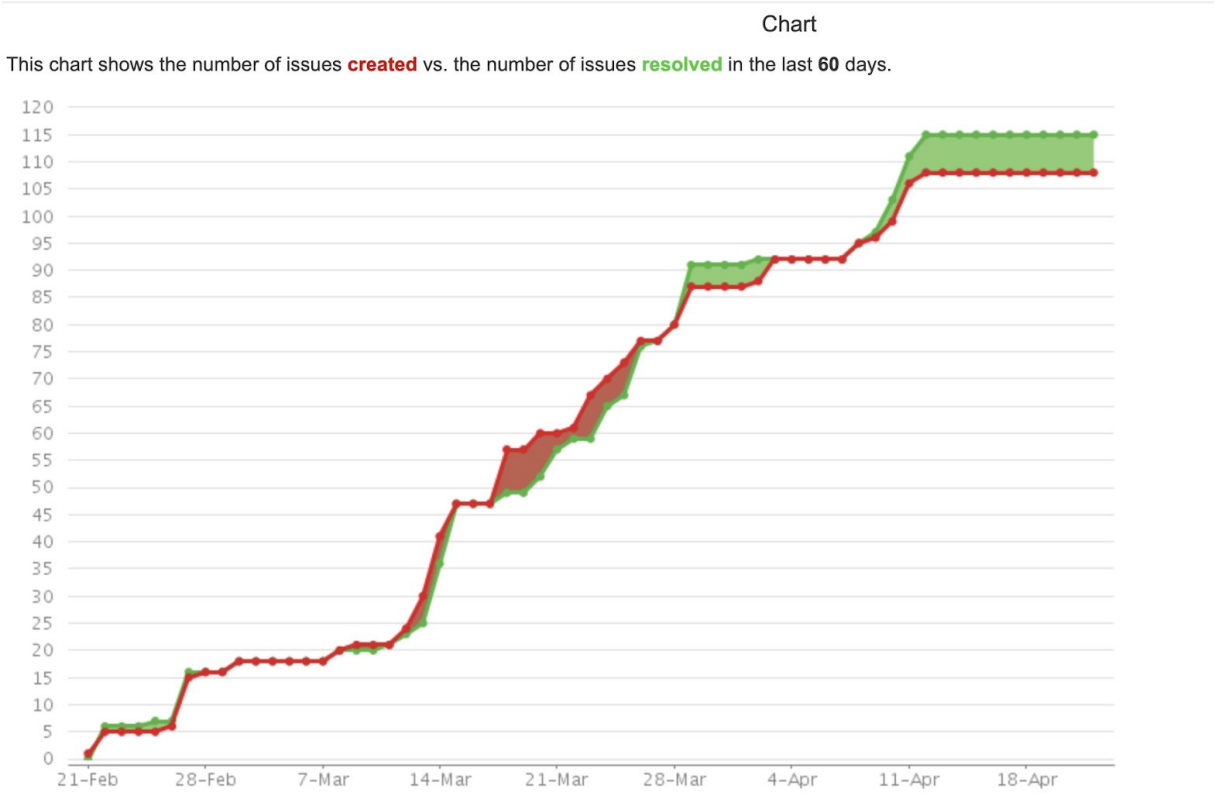

Fig 2 : Issues created vs issues solved

2.4 Quality claims supported by testing statistics

Testing was one of the most essential task of this project. We made sure to write tests as and when adding features and functionalities. JUnit tests were written for all the functionalities with a one task per unit test manner. We have written about 128 unit tests and achieved 83.3% code coverage. There is 0% code duplication with no bugs and vulnerabilities. There is just one minor code smell.

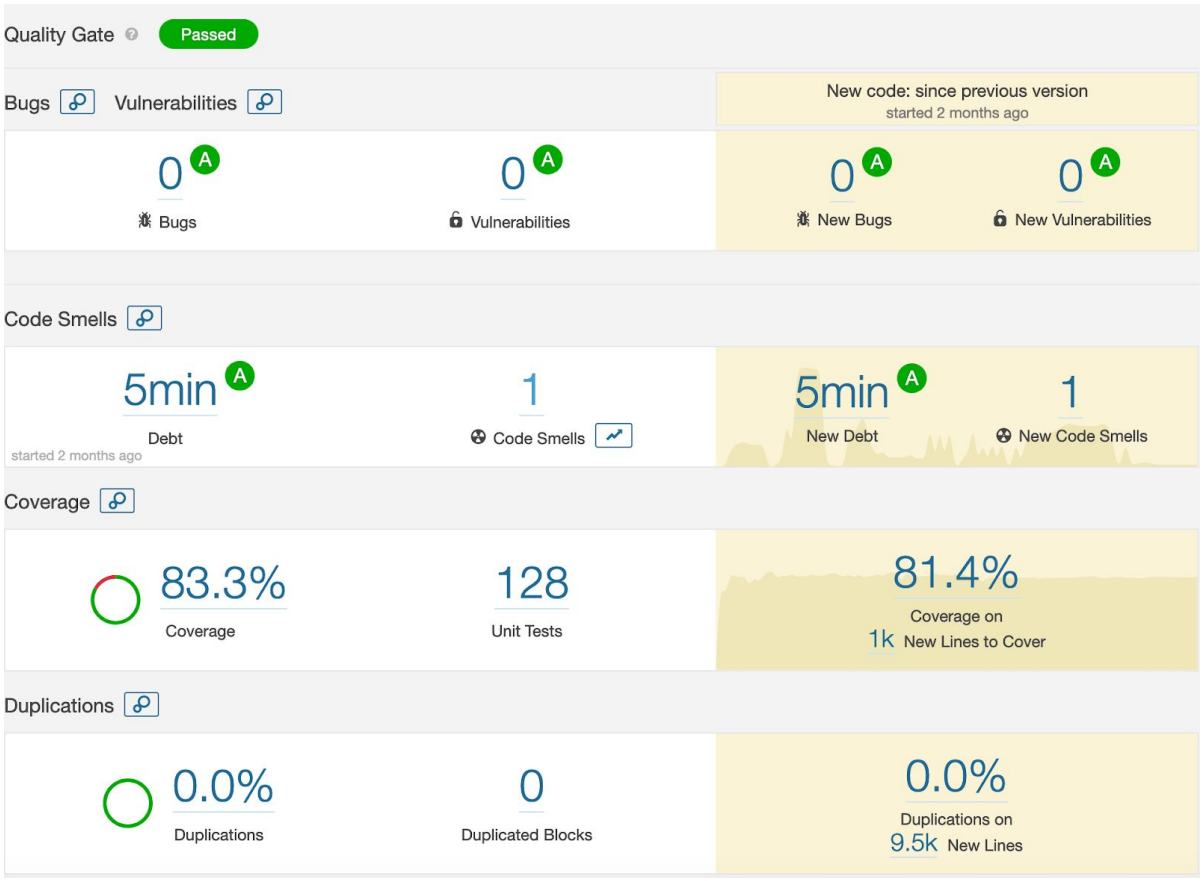

Fig 3: SonarQube report

3.Team’s Development Process:

The team worked quite efficiently throughout the course of the project. The team had a very Agile approach where we planned at the start of each Sprint, the things to be done towards a done increment at the end of each Sprint. The team’s development process wasn’t all smooth, we did face some bumps along the road but were able to solve those through a collective effort.

3.1 What works:

As a part of the development process the team successfully employed some of the Software Development techniques learnt in the course for e.g. the design patterns like Factory pattern, Command design pattern etc. The code is well documented, and the design practices used makes it easy for any new developer to start working on it in very less time.
During the development process we were able to deliver the following functionalities and implementations:

1. Data Persistence: We used a Relational Database to persist the messages and the details for the users and groups. Hibernate was used in the backend to interact with the Database.
2. In-memory Database for testing: We used a H2 in-memory Database for running the Junit Tests
3. Deployment to AWS: The server is hosted on AWS. A CI/CD pipeline is setup where each done increment is deployed to the AWS server.
4. Command Line Interface: The Command Design pattern was used to implement the User interface for a Client.
5. Sending Messages to users and groups: Messages can be sent to any user/group or broadcasted to all users. The message will be delivered to the user if online, else it is queued for the that user to be received whenever he/she comes back online.
6. Users and Groups: A User can register and login using a username and password. Passwords are encrypted. A User can create a Group and add other users to the group based on Moderator approval. A User can also invite users to a group
7. Moderators: Every group has a Moderator who can add/delete users from a Group. The moderator can add users to the group based on any invite requests. The moderator can delete the group.

8. Followers/Following: A user can follow/followed by other users. A follower who is online will know when the followed user comes online or goes offline.
9. User/Group Profile: A profile for the user which displays the public information when accessed by other users. A user profile can have details like Username, First Name, Last Name, Email, Groups, Followers etc. based on their visibility labels. A profile for the group which displays the participants in the Group amongst other details about the group.
10. User Statistics: A Statistics website for the System that logs the number of active users at timely intervals.

All the above was developed and managed using the following:

- Jenkins: Facilitated the CI/CD process
- GitHub: Facilitated version control and source code management
- JIRA: Facilitated the task of tracking issues/tasks
- SonarQube: Maintained Code quality with reports covering test coverages, code smells, bugs, etc.
- JIRA-GitHub Integration: Facilitated Smart commits
- Slack-GitHub-Jenkins Integration: Notified about commits to GitHub and jobs on Jenkins

3.2 What could be done better:

As a team we did achieve the goal we set for ourselves considering the resources and the constraints but, if given more time or another chance to work on this, these are some of the things we could do better.

1. Graphical User Interface: The team made a design choice to use the Command Design pattern for the UI due to the time constraint. But a Graphical User interface is something the team would want to implement in the future to provide a better User Experience.
2. Additional Features: Given the short span of time and we prioritized delivering a complete product, but with the base set we would want to implement more features.
3. Followers/Following: The current implementation can be improved and refactored using the Observer pattern, thus making the code more manageable and improving the code quality.
4. Operational Concerns: We started of with building a basic Live-User statistics System and have huge scope to add other functionalities in this area for e.g. Risk Mitigation, Tracking of System metrics such as Performance, Uptime etc.

4. Retrospective of the project

4.1 What the team liked

1. The team learnt about multi-threading and Java sockets which was a new domain for most of us
2. The learnt that it is important to spend enough time into planning before beginning the work
3. Brainstorming about ways to solve issues as a team and making decisions with having the entire team on the same page was an amazing experience
4. Working our heads to refactor code to perfection by incorporating design patterns and other good practices in coding definitely has given us hand ons into the real world
5. Getting the taste of working on an industry like project with a team was very enjoyable and an excellent learning experience
6. It was an amazing experience working in a fast-paced environment and embodying the Agile methodologies

4.2 What the team disliked

1. Testing the legacy code to get the code coverage was a big task which took a lot of time.
2. Things initially were quite unclear which caused us to work very less on the first sprint. We started really late really unclear about quite some things. If we had been clear like we were after the first sprint we would have achieved quite a lot of tasks.

4.3 What the team learnt

1. The team learnt how to use tools like SonarQube, Jenkins and Jira which are popular tools for production in the industry.
2. The team learnt to write efficient and reliable JUnit tests for the code written
3. The team learnt how important it is to collaborate and learn skills to work efficiently in a team as that's the way things work in the industry
4. The team learnt how to improve on the mistakes in the previous sprints to ultimately make a good product
5. Technically, the team learnt to incorporate good coding styles and various design patterns into practice

4.4 What needs to be changed in the course

1. You can let students decide the stack to work on. By that they will learn to make decisions on the programming languages and tools that are appropriate for development of the given project.
2. You can have a one on one sessions. We feel they can help a lot.
3. A discussion on approaches that people took to solve a particular problem. Learning from others mistake.