**Reflection for the Developer Rule**

In this sprint review session, we have completed most of the items listed in our sprint commitments. Our team works much better together, we talk about our ideas at the team meeting, split the workload to make sure that every member does contribute and grow. For me, I contributed and have learned and achieved a lot.

About the features, we did have a deep talk with our product at the end of our last sprint review. We briefly list several goals and discuss one by one, all of which did sound good the product. Then we added several backlogs on Jira. Now if we look back at this sprint, we managed to complete most of them.

Firstly, we have generated a full UML for 1 on 1 messaging, which does show the dependency relationship of each class existed. Meanwhile, we added several use cases about the chat server.

Secondly, here comes some work we've done on the database, we successfully set up a database instance on AWS, which we might use later.

Thirdly, the most important part is about the implementation of 1 on 1 messaging feature. Now the basic communication feature has been implemented. Now any client who gets access to the server could send a private message to anyone who is connected to the server at the same time.

Initially, we are planning to implement the features by simply add a new message type called '1-1' and add some if-else statements at all the relative clauses. While after second thought, we think this is not a good idea, while this means every time we have a new feature, we will have to modify all our former implementations, this is not a good design pattern, which goes against the 'SOLID' design principle.

Then, we struggled to come up with a better design. Basically, we complete the 1 on 1 messaging functionality by adopting the broadcast messaging, In this case, we are able to take advantages of the current functionality, this goes along with the 'Open to extension, Close to modification' rule. Thus, we could easily add following features like group chat with a massive modification.

To get it done, we do create a parser to help us to parse the message that clients message received. At the same time, we have a class for 1 on 1 message, the trick about the class is that it holds a hash map of a pair of clients to help record those 1 on 1 pair. More importantly, we adopt an '@' identifier help recognize the receiver, if the text right after the identifier is 'all', then mark and send the message as a broadcast message, while if it is some other text, the parser will see it as the name of a client, then the parser will go through the logic and deal with message differently. The difference lays on that if the parser marks a message as 1 on 1, then the prattle controller will only broadcast the message to the receiver himself. More importantly, the parser will be a great help to us in future work.

After the implementation, we do have some code smells after running the Sonarqube checks while we successfully remove most of them. Along the development process, we have done a lot of git commits. W are trying to make sure that every commit does have a singular focus, thus it will be much easier for other team members to understand, which is kind of good practice. In the end, we manage to raise the code test coverage up to more than 85%.

In this sprint session, our team holds two team meetings. Because of the spring break flight, I missed the first. While I managed to go through the discussion by review the team meeting record. Fortunately, I was able to attend the second meeting. During the meeting, we mainly talked about the current implementation of 1 on 1 messaging feature, brainstormed the future opportunities for refactoring and assigned the remaining work of this sprint.

Personally, I do contribute to this sprint from two aspects. Firstly, I actively communicated with other team members, discussed how should the implementation be done and how to better follow good OOD pattern. Secondly, I did generate the UML diagram to help us better understand the architecture of our current implementation, also I managed to remove most of the code smells that Sonarqube complains.

**Reflection for the Team**

| **Name** | **Rating** | **Comments** |
|---|---|---|
| Yitian Zhu | 10 | Successfully covered the tests for prattle server and network connection, helps to raise the code test coverage up to more than 85%. Also records the team meetings. |
| Nan Liu | 10 | Excellent job. Contribute a big part of the implementation of the 1 on 1 messaging feature. Add two use cases. |
| Jiyang Zheng | 9 | Good job. Mainly play with the database research and set up the db instance on AWS. |
| Yuanjie Yue | 9 | Good job. Create UML Diagram and go through the Sonarqube tests, and remove the code smells. |