Sprint 3: Personal Sprint Reflection - Developer Role

The goal of this sprint was of developing more features on the server side, developing a frontend application and connecting both of them.

The above goals required to write Java code to support user management features, group management features, conversation and message exchange features, developing React frontend application and connecting backend and the frontend. To achieve this, I specifically worked on defining architecture of the frontend and the backend and wrote a specific code to enable REST like MVC architecture.

Major issues assigned to me:

1) Develop robust and scalable architecture MSD208SP19-76
The current MessageType contains three main type of messages HLO, BCT and BYE. I added a message type called API. API type is used for communicating between client and the server once the user is signed in. The message text for API is defined to be in following manner.
<route>::<method_type>::<data>
Here route refers to a specific api endpoint, method_type can be POST or GET and data contains a JSON string containing data to be passed for a specific endpoint.

The client runnable class calls relevant method in the Route class depending on the method_type. This route class returns a JSON response to the client runnable after processing a request which is returned to the user.

Here route works as a view and it calls relevant controller. Controller interacts with the model class and model communicates with the database.

Thus, now onwards, developing any new feature is as simple as adding a Route in Route.java and adding controller for that feature. The architecture is highly modular now.

2) Refactor the code and use relevant design patterns MSD208SP19-81
I used factory method pattern for controller and model classes.

3) Enable push notification MSD208SP19-86
Defined a new MessageType called NOTIFICATION which is used to send push notifications to the user. Whenever React application receives a message with type NOTIFICATION, it creates a push notification from a browser. This uses an observer pattern. Whenever a client thread is available on Prattle server, it sends a message to that thread.

4) Define architecture for react application, integrate flux MSD208SP19-81
This helps to write a scalable react application by separating out actions, data store and the UI. I also wrote a module to enable the communication between Prattle and the frontend. Browser application can not directly communicate with a TCP server. A proxy sever is created to convert a WS request to TCP message. The react module sends a WS request for each API call to our proxy server which redirects it to Prattle from there.

5) Develop API endpoints MSD208SP19-87
Developed API endpoints for various features like get available users, signin, signup, sendMessage, get conversations, get groups etc.

6) Develop data access restrictions for un authorized users MSD208SP19-88

Wrote methods to send 403 response for methods like get group user or change group settings or any other relevant feature which requires user to be verified. For an example, only admin change group name, only admin can add other user to a group, a user should not be able to access conversation not related to himself.

7) Implemented checks for parameters in an API calls
Prattle returns 500 error if parameter is not provided by user. This was done as a part of refactoring the code

8) Create AWS ec2 instance for hosting java application, frontend application and a proxy server. MSD208SP19-44

9) I worked with everyone in the team to solve their queries specific to architecture (Where to put which module) or database queries.

For the above mentioned issues, the code I wrote meets sonarQube quality gates. Teammates were able to understand and develop upon the code written by me.

During this sprint, we as a team communicated almost every day with major meeting on every second day. Apart from that we used, slack and telephonic conversation to solve each others road blocks.

Name	Rating	Comment
Himanshu	7	Himanshu worked as expected for this sprint. He could write code for all the features assigned to him. He worked more on testing after finishing the tasks assigned to him. Himanshu managed to communicate with other team and quickly develop backend APIs for required features. He contributed to write testis for features. He managed to boost line coverage to 80%+ from 47%.
John (Scrum Master)	7.5	John worked more than ever this sprint. From very beginning of the sprint, he created a list of features that we need to deliver this sprint. He invested a lot of time to create a working frontend application. He made sure that everyone in the team is working and making progress.
Ram	7	Ram worked as per the expectation. He wrote code for features. Ram worked on dependency injection to insert database connection as a parameter to a model.
Shashwat	7.5	As a developer, I worked on defining and developing architecture for the frontend and the backend. I invested a lot of time reorganizing the code to make in scalable. I incorporated MVC architecture for the backend and Flux architecture for the frontend. I developed factory method for models and controllers. I developed a code to enable push notification for any new message. Apart from this, I worked with everyone else to solve their architectural queries and debugging their code.