Sprint 2 Team Report

Team: 108

Repo URL: http://github.ccs.neu.edu/cs5500/Team-108-SP19
Reviewer: Robert Bronstein

Score: 94

Comments

Latest commit is after 7 PM deadline.

You all accomplished a great deal in this sprint and I was very impressed (private messaging still needs a bit of work but otherwise, well done).

Since you met your commitments, you get an A, as you should. But I want briefly stop and reflect on time management. We are only requiring you to work 15 hrs/person/week in order to get an A. Being that you set your own commitments, if you are substantially over this amount, it is up to you to adjust. If it is a particular person on the team, consider offering them assistance or re-allocating their tasks so they can catch a break. If the team as a whole is putting in well over the required amount, then you are overcommitting. As much as I love seeing progress, this is costly in the long run. Its not good for your personal well-being to be giving up substantial amounts of sleep to keep up and this will damage yours and the team's productivity as well. Lastly, if you overcommit, you put artificial and undue pressure on yourselves and this can sometimes spill over into increased tensions in the team (which if they were present, you seemed to have managed well enough), which hurts team morale and cooperation. So its a really bad thing to do and part of the course experience is learning to manage your own expectations and those of the team. You should carefully analyze as best you can what it will take to complete a task- and maybe consider setting scope for certain goals narrowly so that you can be more confident that you will deliver. You can always do additional work if needed (it never runs out does it??). If you do overcommit, you are welcome to opt not to complete certain tasks in sprint. You must notify me of what you will not be doing and why (This is part of managing expectations). I maintain that a perfect sprint score requires perfect planning and the professor has said that commitment modifications should be penalized (so you won't get 100 if this happens for sure) but the magnitude of the penalty will be determined after discussion between you and I (and maybe the professor too). I think that this is ultimately better than overworking though for the aforementioned reasons.

As far as code, mvn clean compile test runs from the command line without issue, but I am seeing unmanaged POM, missing SDK and compile errors in IDE that I did not see in other projects that might be my fault, but I wanted to mention just in case. Some code quality points (really minor things):

1. Why does Message have both a public and private constructor? The empty public constructor appears to be useless.
2. ServerConstants class itself probably should be static. It only holds constants and there is no visible reason to instantiate an object of that type.
3. Good use of singleton for the dispatcher.

4. Private helpers lacking javadoc

5. The private handler methods in request dispatcher are all very similar. Do you think they might be refactorable into a smaller number of methods (thought exercise for the reader)?

6. User/IUserGroup interfaces lack javadoc. What is IUserGroup for?

7.Setters and constructors for the user and group classes should have exception checks/validation.

8. In Group's addUser method, moderator will always be instanceof user

9. Please try to avoid unused methods. I know that we have the quality gate as the official bar, but when you think of it from the standpoint of checking in code that you've never used at all to master, there is a lower degree of confidence that these methods work correctly and they then end up reaching the client potentially at your peril. I would like to claim that these should remain in the branches until they are used but I am speaking from a point of inexperience in saying this.

10. Double initialization of GroupJPAService in GroupService

11. In homeworks we had a deduction that at least everyone I graded managed to avoid, but Professor seems to want people to keep methods under 30 lines long (see JSONBufferedReader). See if you can break it up a bit.

12. IController has no documentation.

13. Good use of factory for NetworkRequests/Responses.

14. No need for default constructor in NetworkRequest/ResponseImpl