Demo Day 1 Review

# Reviewers: Andrew Holmes, Lauren Cooke, Patrick Thornton

# Reviewees: Cassandra Marcussen, Yeongwoo Hwang

**Instructions**

Perform a (written) code review on the chat servers you were given to run. The review should examine code clarity and correctness, completeness of the documentation (both code documentation and deployment documentation), and the coverage of the unit tests. Grade on a 25 point scale, with 5 points for each implementation on meeting the spec (10 pts. total); 5 points for code correctness and clarity; 3 points for code documentation correctness and clarity, and 2 points for deployment documentation.

**Review layout**

* This page (page 1) has the instructions, layout and review summary.
* Page 2 has feedback for the gRPC implementation.
* Page 3 has feedback for the custom wire protocol implementation.
* Page 4 has feedback for the documentation.

**Summary**

Overall we thought your project was great – your code is extremely well documented, such that even as non-Java users, we can mostly work out how all the parts linked together. It is nicely structured with folders to help with that, and both versions very clearly meet and exceed the specification for the assignment. In particular, we liked the REGEX wildcard and two-way stub system as beyond-the-requirement implementations. The deployment documentation is good, and everything worked without a hitch during the live demo session. In summary, it seems like an obvious **25/25**, Great job!

# gRPC notes

1. **gRPC — meeting the specification (5/5 points)**

Just going off of the basic spec we were provided, the requirements were:

1. **Create** an account. You must supply a unique user name.
2. **List** accounts (or a subset of the accounts, by text wildcard)
3. **Send a message** to a recipient. If the recipient is logged in, deliver immediately; otherwise queue the message and deliver on demand. If the message is sent to someone who isn't a user, return an error message
4. **Deliver** **undelivered messages** to a particular user
5. **Delete an account**. You will need to specify the semantics of what happens if you attempt to delete an account that contains undelivered message.

This is an easy one, your code very clearly implements all of these. It checks whether users already exist before creation or login and denies access to a second client attempting to login as the same user. Messages send and display nicely, and your command to check for undelivered messages implements (4) well. Your list functionality using REGEX wildcards is certainly beyond the requirement for the list functionality, and your delete function works well too.

* **gRPC — code correctness and clarity (5/5 points)**
* *Disclaimer: none of us have any serious experience with Java, but from our understanding this is our feedback.*
* The organization of your repo makes it very clear how the project works and is super organized and well thought out. I particularly like the *objects* folder and your implementation of objects overall, with high level general Response/Reply/Status objects.
* Your two-stub approach is also a different take (and one we even considered) to pull messages to users which is super interesting and seems to work well.
* Your code is extremely well commented, and all object/function/variable names are intuitive. We also liked your ServerCore/ClientCore separation from the Server/Client files which are actually what’s run – it helps keep the actual driver files very readable.
* Using regular expressions for the wildcard is also a nice implementation of the requirements for the wildcard and is certainly beyond the expectations of the spec.
* Your testing framework is well documented and seems to cover all the functionality on the specification, even if this is supposedly no longer a factor in grading :)

# Custom wire

* **Custom wire — meeting the specification (5/5 points)**
* This is largely going to be a repeat of the above feedback, since your implementations are deliberately aligned to behave as similarly as possible.
* All the functionality works essentially identically for both versions, just the actual message objects being passed between the server and clients are encoded differently.
* For example, if you compare the *ClientGRPC.java* and *Client.java* files, the main function within both classes is essentially identical logic-wise, the only real differences come in the encoding (generating generic response objects and handling those in your custom version or generating gRPC objects through other methods in your gRPC implementation).
* **Custom wire — code correctness and clarity (5/5 points)**
* Again, your code is very clean and well documented.
* I’m slightly curious why the ClientCore is so small in the gRPC version and whether there were constraints stopping you from putting some of the Client functionality in the ClientCore file, but without writing Java it’s impossible for me to know whether there’s some constraint there making it hard to do. Either way, we just had everything in one file and did our best to keep it clean, your way is very clean as is, that’s more a question out of interest than critique.
* Your proto file is also well documented, and I especially like your Status and StatusReply objects to reduce redundancy.
* As previously, your testing framework is nicely implemented and documented for this version too.

# Documentation

* **Code documentation correctness and clarity (3 points)**
* Your documentation is very thorough (and interesting), split between *README.md*, *design\_notebook.md* and *wire\_protocol.md*.
* *README.md* stores the installation and running instructions (see below), the design overview and an example to try with the application. The instructions are clear and I like the succinct design overview, especially for highlighting some of the interesting features of your implementation with two messenger.
* *Wire\_protocol.md* explains your wire protocol clearly and succinctly and was useful in understanding your code.
* *Design\_notebook.md* does a good job of explaining your process in designing the project and decisions you made throughout (e.g. the *decision*: sections make it very clear what decisions you made throughout)
* **Deployment documentation (2/2 points)**
* The *readme.md* file has very clear instructions to setup and run the chat application. We watched you guys follow the steps and get it up and running so this is certainly adequate. I’m having a few issues with gradle but I think that’s me and/or Windows being dumb, and anyway I believe we were told that it was fair to assume that they have the appropriate language and compiler set up already - you guys demonstrated that your instructions work on that basis already, so full points.