Project Report

The project fully implements all required functionality, to the best of my knowledge. Automated unit tests are complete to the specifications provided in the Test Plan. One unit test fails, an anticipated failure: the server does not properly reject duplicate logins, instead leaving the system in an invalid state. This is easily fixed in a future patch; in the meantime, users should take care not to log in twice. Code review has been performed per Test Plan specifications, and manual integration testing has been completed as well.

# Building the project

The project is divided into three Modules: Client, Server, and Shared. Client and Server both have entry points (main methods); no arguments are necessary for either.

## Dependencies

The project uses the Netty, Protobuf, jUnit, and Hamcrest libraries. The project is built against netty-all-4.1.10.Final.jar, jUnit-4.12 (for unit tests) and hamcrest-core-1.3.jar (extra matchers), and protobuf-java-3.3.0.jar, using Java SE 8. Given the use of lambdas, versions of Java prior to 7 may not compile. Swing from the Java Standard Library is required.

The Client and Server modules depend on the Shared module. The Shared module depends on Server for the User class.

## Recommended build process

The project was built using IntelliJ IDEA 2017. Most IDEA-specific files are excluded from the repository, but importing the project into IDEA should be trivial. Configure module dependencies through the Project Structure dialog. Add required JAR files to /Code/lib and add them as module dependences as stated above. Create run configurations for Server (main class: ChatServer) and Client (main class: ClientController), plus compound run groups if desired.

## Other notes

The server’s User class should create/load the filename specified in User.userFileName (defaults to users.dat). If it cannot, simply create an empty file. (Note: this is a binary file.) The client, meanwhile, will create/load log/*username*.log when a user successfully logs in. This is a plaintext file containing chat history on a per-user basis. **Since git cannot store empty folders, you will need to create the log folder yourself.**

# Client-side feature tour

First, on the login screen, note that you can press Enter from any text field to attempt to log in or Shift+Enter to attempt to register. On success, the main chat screen will be shown. On failure, the cause will be displayed.

Once you are logged in, you will see your user name in the chat screen window’s title bar. The **list of online** users (excluding you) is automatically maintained in the “Online Users” list. Double-click on a user name in this list to send a **private message** to that user. With focus in the bottom text field (where you type messages), press Escape to return to **public message** mode. (Note: it is not possible to retrieve a list of offline users at this time, though the server certainly knows this information. My interpretation of the requirements did not include this feature.) Use the menu bar in the top left corner to access **Chat History**, **Log out**, or **Quit**.

# Differences from Requirements documents

The project implementation differs from the requirements document in one significant way. Upon closer inspection of the project specification documents, I did not see any justification for the Delete User use case. As a result, this feature has not been implemented. (It could be added very easily if needed, however.)

# Unit tests

The unit tests can be found in the ClientControllerTest, ServerControllerTest, and UserTest classes. My rationale for focusing my tests (given the extremely limited time for testing) is described in my Test Plan. The plans are trivially runnable through a JUnit4 run configuration in IDEA.

# Questions/issues

If you have questions or issues that may affect my grade, please feel free to contact me by email and I will happily resolve them promptly and to the best of my ability.