Testing Report

hnassif-jiarui-sresht

This testing report will first declare the testing strategy and then go into depth for each class and then discuss the specific tests that we wrote for each one.

The overall test strategy of the project is to look at each class's role independent of the whole project, and then to do some integrated tests to test the whole system. We will write JUnits along the way to test methods that are easily tested using automated testing.

ConversationTest

The best way to test the Conversation class is to make sure that both constructors work for it and then also to make sure that every method executes the expected function. We will test adding a user, going offline, checking if a user is offline, checking if a user is active, checking if a user is contained, getting the history, adding to the history, and getting the conversation ID. Each of these methods will be tested in a JUnit test in the ConversationTest Suite.

TimeTest

Our Time method returns a timestamp for messages, so we will write a JUnit test case to test times. We can take the current time using this system, and then wait five seconds and then take the time again. This way, we should be able to infer what the time should be for the second call. We set up a series of rules regarding time changes (for example, increment the minute counter if the second counter is at 55 seconds or greater) that make it so that the test could run at any given time. Since the Time class depends on the current system time, we can't pass in absolute values – the only thing that we could do to test it is to make sure that the test works at many different times. Therefore, we will test it at different times for the first call, including:

- 02:28:15
- 02:28:05
- 02:59:59
- 23:59:55
- 23:59:59
- 00:00:00

We expect that the time should pass at any of these times.

Client and Server Testing

For client testing, we will primarily use telnet instead of JUnits. We outline the testing strategy in detail in the Design Document. We will run through these tests in telnet and also run through the allowed tests through the GUI. For example, we cannot attempt to send a message using another username in the GUI, so we will skip that in the GUI testing. Below are the results of our tests (black checks mark success in telnet testing, while red checks mark success in GUI testing).

Our tests are divided into three different parts:

a) valid commands,
b) invalid commands that should return an error message but re-prompt the user, and
c) invalid commands that kill the socket.

Type A Tests

1. Login with valid username ✓ ✓
2. Login with uppercase username ✓ ✓
3. Login, then logout, then login again ✓ ✓
4. Create chatroom successfully ✓ ✓
5. Create chatroom and then leave chatroom ✓✓
6. Create chatroom and then logout ✓ ✓(destroys the chatroom on the GUI)
7. Create >1 chatrooms by the same person ✓ ✓
8. Have three users make mutual chatrooms ✓ ✓
9. Create chatroom 1, leave chatroom 1, create chatroom 2 ✓
10. Type messages to each other (sub-tests included below)
    a. Have three chatrooms, in which three users will have mutually exclusive conversations (i.e. chatroom 1: AB, chatroom 2: BC, chatroom 3: AC, and send messages back and forth) ✓✓
    b. Have three chatrooms, in which three users will all participate in all three chatrooms ✓ ✓

Type B Tests

11. Nonsense keywords (ex: die, disconnect, kill, etc) ✓

12. Login with valid username then try logging in again (ignore the second attempt to login, regardless of whether it's valid or not) ✓ ✓
13. Create chatroom with an invalid name specification (i.e., Henry tries to create chatroom under Sresht's name) ✓
14. Send message with an invalid name specification ✓
15. Leave chatroom in which the user is not already participating ✓
16. Send message to a chatroom that the user is not in ✓
17. Specify non-numerical chatroom ID (i.e., create foo Henry) ✓ ✓
18. Try to join the chatroom when the user is already in the chatroom ✓ ✓
19. Logout before logging in ✓

Type C Tests

*We decided to close the socket on these invalid commands because these would result in our login screen returning a warning and then clearing the text input boxes. In essence, the GUI would "refresh" upon running into these errors without actually creating a socket for the user. Therefore, we should not have these errors maintain a socket.

20. Login with invalid username ✓ ✓
21. Login with taken username ✓ ✓

GUITest

For GUI testing, we will use manual tests. The process is outlined in detail in the GuiTest file (as a block comment). It is replicated below. All of these tests passed through our GUI, along with the functional tests from above.

Before testing the GUI, we want to ensure that the entire server-side design is working as expected. Now, we can turn our attention to GUI testing.

Firstly, we will ensure that **all JButton objects can be clicked**, all JTextField objects have appropriate eventListeners when the user pressed the return. ✓

Secondly, we will ensure that the eventListeners **correspond to the right action**. ✓

Regarding design choices, we will allow for the window to **resize appropriately unless resize is disabled**, and make sure that there are no overlapping or conflicting problems on the interface when the window is resized. ✓

To ensure that, we will also make sure that **everything is appropriately sized and spaced**. ✓

Since we have a login screen, we will test to make sure that **appropriate logins work while inappropriate logins don't**, and instead clear the input JTextField and also return a warning message. ✓

When a user opens multiple chats, we will ensure that the newest chat window is created and does not overwrite the existing chat window, but instead creates a new window. ✓

The following tests are sequence-oriented GUI tests (which we will test manually when we build the GUI):

- Login with invalid username (should return a warning as a popup or text message) ✓
- Login and then force close out of the GUI, and then login again ✓
- Login, create a chatroom, force close out of the chatroom, then create the same chatroom (we couldn't get this to work. See future improvements)
- Login, create chatrooms, logout (which should close all existing chatrooms), then login with the same username and then create the same chatrooms ✓

Concurrency Testing

This is also outlined in the design document. While we didn't do the concurrency testing because of time constraints, we have outlined the strategy to do the concurrency testing. This is replicated below:

- 2 people try to login with the same username at the same time. We will make sure only one of them can login with that username and the other is prompted to choose a different username.

- 2 people create a chatroom at the same time. This should allow both users to join that chatroom without any problems.

- 2 people send messages to one another at the same time. We will make sure both receive the message destined to them.

- 2 people send a message to a third person at the same time. We will make sure that the third user receives both messages.

-2 people logout at the same time. We will make sure that all their conversations are cancelled.

-One user logs out and one user logs in with that same username at the same time. This will be processed in one of two orders: either everything will work as if the first user logged out first, or the second user will be prompted to try a different username and then the first user will be logged out. Then, if the second user tries the same username some short period of time later (as long as it takes to process a logout request), he will be allowed to successfully login.

Integrated Testing

In our integrated testing, we will log in to the client using two different computers and then join chatrooms. We will test a series of actions, such as starting a chatroom, leaving a chatroom, getting the history of a chatroom that the user has joined, returning an error message while trying to access the history of a chatroom that the user has not joined, joining a chatroom that the user has left, force-closing out of a chatroom, and force-closing out of the lobby (which should also close out of all open chatrooms). All of these tests will be done on the GUI side. Finally, we will all enter the same chatroom and then have a conversation about how cool this project was! :D