TESTING STRATEGY AND FINAL TESTING REPORT

Our chat program (NyanChat) works using a M-V-C Model. The Client connects with a Server to pass messages back and forth. Messages received by the Client are then passed to the Controller, and changes are made to the GUI. Similarly, changes in the GUI trigger ActionListeners that send messages to the Server via the Client.

We tested the Server independently (treating it as a Black box. We also made sure the Client was sending messages to and from the server correctly. We also tested the Controller by checking if it made changes to the GUI. Finally had to test if ActionListeners in the GUI sent the correct messages to the client.

1. Testing that messages to GUI update the GUI correctly

These tests involve JUnit and Manual tests that pass commands to the Controller's handleServerMessage method. We then checked manually if the expected result matched the actual result.

1.1 ServerLogin tests

This was tested through the GUI.

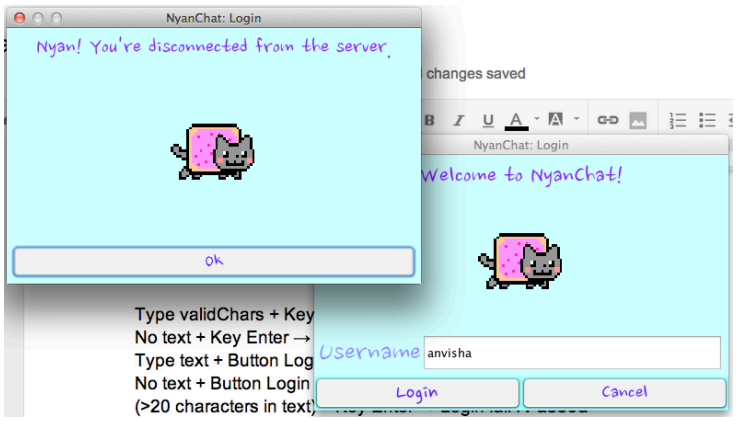
- Enter wrong IP address -> Gives error message **Passed** (shown below)
- Enter correct IP address, after entering wrong IP address -> Server login screen closes and username login opens **Passed**

1.2 LoginAttempt tests
This was tested through the GUI.

Validity tests: these test if the LoginAttempt form correctly validates data before sending to server
Type validChars + Key Enter → Login success. **Passed**
No text + Key Enter → Login fail. **Passed**
Type text + Button Login → Login success. **Passed**
No text + Button Login → Login fail . **Passed**
(>20 characters in text) + Key Enter → Login fail . **Passed**
(>20 characters in text) + Button Login → Login fail. **Passed**
Invalid chars (non alphanumeric/spaces) + Key Enter → Login fail. **Passed**
Invalid chars + Button Login → Login fail. **Passed**

Connectivity Tests: these test if the GUI responds correctly to server messages
Login a user with name “xyz”. Then login another user with the name “xyz” -> Login screen says “Username in Use” **Passed**

Server disconnected while login window open, and then attempt made to login -> Server disconnected dialog box displayed. (see diagram below) **Passed**



1.3 Controller Tests (what happens during chat)

These are tests to validate how the Controller method handleServerMessage(String message) treats various Server messages according to the protocol. JUnit tests were written (in TestControllerGUI.java) to pass commands to the client and the response in the GUI was checked visually.

1.3.1 testOnlineUsers()

This is anvisha's chat. It checks for online users changes.
"`ONLINEUSERS anvisha will jesika nyan`" -> online users table shows 3 names (username of client not included) **Passed**
"`ONLINEUSERS anvisha will`" -> online users table only shows 1 name. **Passed**

1.3.2 testBasicChanges()

This is anvisha's chat.
Starting a 2-person conversation. "`CSTART CID 99990 USER will`" passed to Controller -> New Tab Opened with name of other user in conversation. **Passed**

Storing tabs of conversation in HashTable. In the JUnit test, we asserted that a new field in the HashMap cidToTabs was created with this conversation ID. **Passed**

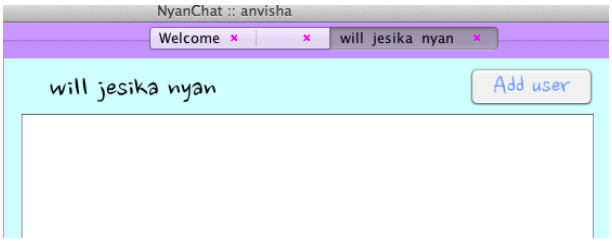
Chat Message. "`CHATSENDER will CID 9990 DATA abcd`"-> shows up in conversation. **Passed**

Chat Message sent after conversation closes. "`CHATSENDER will CID 1 DATA abcd`" -> nothing should happen, since conversation doesn't exist. **Passed**

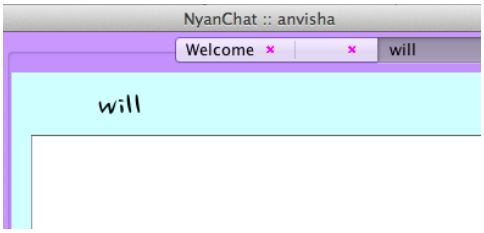
Only current client left in conversation. "`CCHANGE CID 99990 anvisha`"-> Will is removed from conversation. Since anvisha is only user in conversation, label says "You are the only user in the conversation". Tab name becomes empty. **Passed**

Multiperson chat started. "`CCHANGE CID 0 anvisha will jesika nyan`" -> Multiperson

chat started. **Passed**



Label and tab name changes when users leave multi-person chat. "`CCHANGE CID 0 anvisha will`" -> 2 users leave conversation. lblConversationWith changes to reflect this, tab name changes. **Passed**

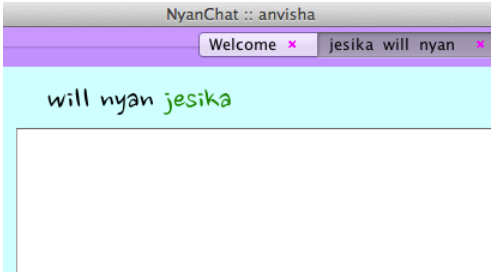


1.3.3. testTypingStatus()

This tests formatting changes due to typing status.
Typing message updates relevant user color to red. Create a conversation with 4 users ("`CCHANGE CID 5 anvisha will jesika nyan`"), and then send a typing message (eg: "`TYPING will CID 5`") -> will turns red. **Passed**

Change in users preserves a user's typing status. Send a conversation change message -> Will's color remains red. **Passed**

Do the same for Typed message (green) *and Cleared message (green)* **Passed**



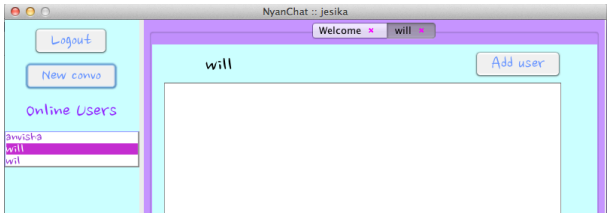
What happens if a typing message is received after a user leaves a conversation "`TYPING will CID 4`" -> Nothing should happen. **Passed**

2. GUI-to-Client communication:
To test how the system responded to user input, and to verify that it was indeed sending the right messages to the chat client once the right server IP had been entered , we used a double tiered testing strategy. We added `System.out.println()` commands to indicate

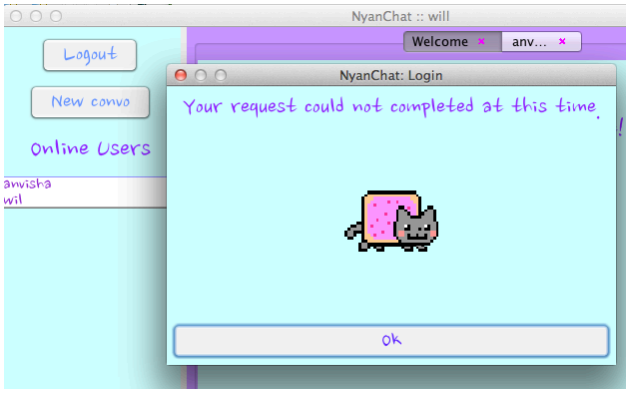
that the user input (key pressed, mouse click or focus event) had actually entered the right action handler, and another layer of `System.out.println()` messages in the chat client's `addInputMessage` method, whenever something was successfully added to the queue.

Testing Procedures:
This is a chat from Client with Username jesika.

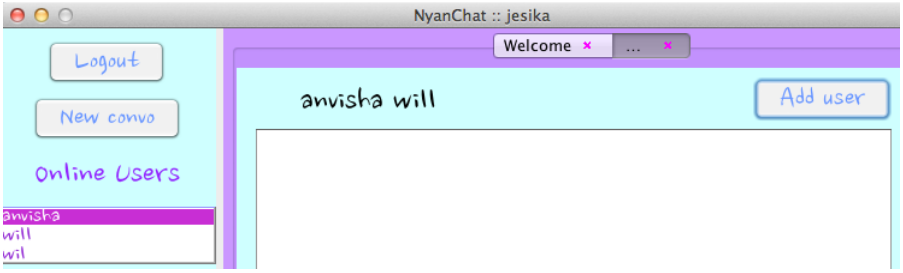
2.1 *Starting Conversation*
Select an online user, click on New Convo -> Sends message "`CREQ SENDER jesika RCPT will`" **Passed**



Do not select online user, click on New Convo -> Error Message pops up **Passed**



2.2 Adding users to conversation
Select an online user, and within a conversation click on Add User -> Sends message "`ADDUSER anvisha CID x`"



There is no case in the GUI that a user will not be selected in the list of online users, after initially making a conversation with one user. Hence, we did not need to write a test case for no user being selected and then Add User being pressed.
If someone selects a user that was previously in the conversation and then clicks Add User, nothing happens.

2.3 Leaving a conversation

Press cancel on the tab -> Sends "`REMOVEUSER jesika CID x`" to the chat client.
Press logout on the main window -> Server just removes them, and handles the rest of messages to be sent to chat client.

2.4 Status messages

Initial default status of user is CLEARED, thread run at 500 ms intervals
User starts typing initially -> Sends "`TYPING username CID x`"
User stops typing but textbox contains text -> Sends "`TYPED username CID x`"
User stops typing and clears the textbox -> Sends "`CLEARED username CID x`"
There is no way of getting from CLEARED to TYPED without first going through TYPING.

3. Server-Side:

To test the server, we wrote a simple testClient that can connect to the server and send string messages. It will output whatever it receives from the server. The test results were verified by confirming that the expected messages were sent to each client.

Test Methods:

`3.1 SignInTest():` This method created 4 clients; 3 with unique username's and 1 with a repeated username. The server was started and each client sent a `SIGNIN` message simultaneously. As expected 3 of the clients successfully signed in and 1 received a "`Username is in use`" message. I then created 100 clients who signed in. Then all clientHandlers were destroyed on the server simultaneously. This demonstrates the servers ability to serve and disconnect a large number of clients which may or may not be registered with the server.

`3.2 StartConversationTest():` This method tested starting conversations and passing messages between users. To test this, the test method created 100 clients and signed them in. The method then started 100 conversations between different pairs of users. The test method then destroyed each user simultaneously, showing that the server can handle removing clients and conversations simultaneously. The correct clients received the CSTART message indicating that the server responded properly.

`3.3 SendMessageTest():` This method tested message passing in conversations. To do this 100 clients were made, and conversations were created between the users. In each conversation, a message was sent. The message was sent correctly to each user indicating that the server was working properly.

Other tests were done via telnet. The server was ran and 3 connections were made via telnet.

Here is a transcription of the terminal input and output of the test:

```
T1 in >> SIGNIN USER will
T1 out << SIGNEDIN
T1 out << ONLINEUSERS will

T2 in >> SIGNIN USER anvisha
T1 out << ONLINEUSERS anvisha will
T2 out << ONLINEUSERS anvisha will

T3 in >> SIGNIN USER jesika
T1 out << ONLINEUSERS anvisha will jesika
T2 out << ONLINEUSERS anvisha will jesika
T3 out << ONLINEUSERS anvisha will jesika
```

(indicates that the ONLINEUSERS message is being sent correctly)

```
T1 in >> CREQ SENDER anvisha RCPT will
```

(there was no response in any terminals, indicating that the server only responds to conversation requests when the client matches the SENDER.)

```
T1 in >> CREQ SENDER will RCPT anvisha
T1 out << CSTART CID 105421439 USER anvisha
T2 out << CSTART CID 105421439 USER will
```

(this indicates that a conversation was successfully created between will and anvisha.)

```
T2 in >> ADDUSER jesika CID 105421439
T1 out << CCHANGE CID 105421439 will anvisha jesika
T2 out << CCHANGE CID 105421439 will anvisha jesika
T3 out << CCHANGE CID 105421439 will anvisha jesika
```

(This indicates that the user “jesika” has been added to the conversation and the conversation now contains “will” “anvisha” and “jesika”.)

```
T3 in >> CHATSENDER will CID 105421439 DATA abcdefg
T3 in >> TYPING will CID 105421439
T3 in >> TYPED will CID 105421439
T3 in >> CLEARED will CID 105421439
```

(There was no response in any terminal indicating that chat and typing messages will only be sent if the sender matches the client’s username.)

```
T1 in >> CHATSENDER will CID 105421439 DATA abcdefg
T1 out << CHATSENDER will CID 105421439 DATA abcdefg
T2 out << CHATSENDER will CID 105421439 DATA abcdefg
T3 out << CHATSENDER will CID 105421439 DATA abcdefg
```

```
T1 in >> TYPING will CID 105421439
T1 out << TYPING will CID 105421439
T2 out << TYPING will CID 105421439
T3 out << TYPING will CID 105421439


T1 in >> TYPED will CID 105421439
T1 out << TYPED will CID 105421439
T2 out << TYPED will CID 105421439
T3 out << TYPED will CID 105421439

T1 in >> CLEARED will CID 105421439
T1 out << CLEARED will CID 105421439
T2 out << CLEARED will CID 105421439
T3 out << CLEARED will CID 105421439
```

(We can see that all users in the conversation receive the messages back.)

The first terminal was then closed

```
T2 out << CCHANGE CID 105421439 anvisha jesika
T2 out << ONLINEUSERS anvisha jesika
T3 out << CCHANGE CID 105421439 anvisha jesika
T3 out << ONLINEUSERS anvisha jesika
```

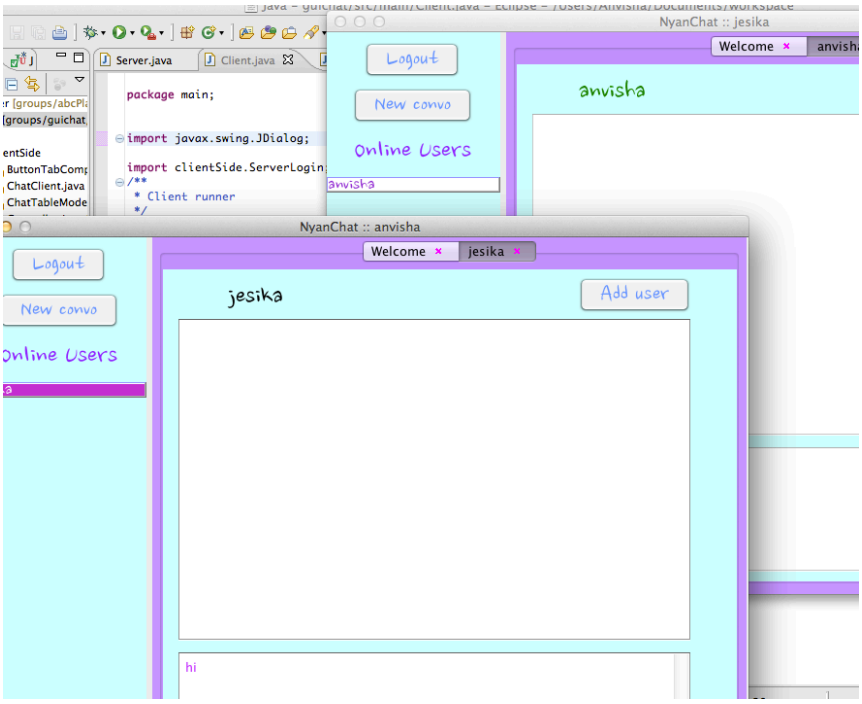
(Showing that the server handles sudden disconnections and that it sends the correct messages to the clients.)

4. INTEGRATED TESTS

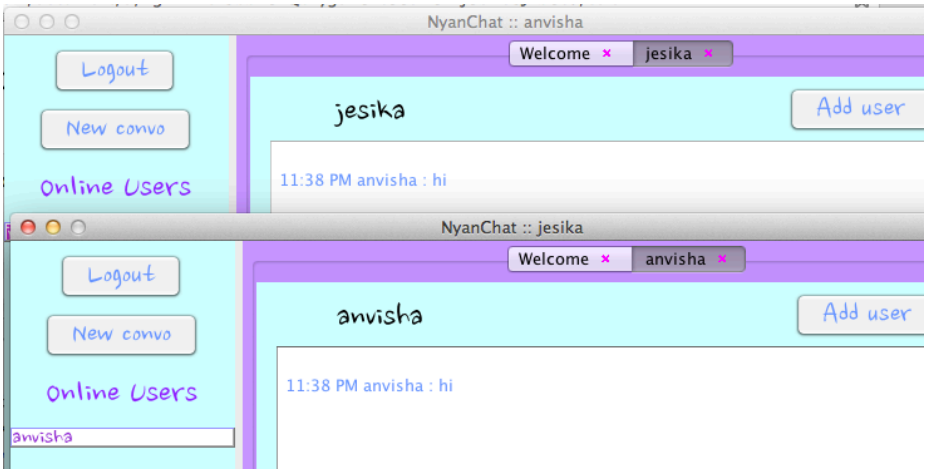
This was mainly to check the speed of some of the responses between users. We conducted adjacent tests with the 3 of us to check that typing messages, cleared text messages and entered text messages (as well as other chat messages, online users etc) were communicated in a timely manner.

Here are some examples of two chat clients on the same computer:

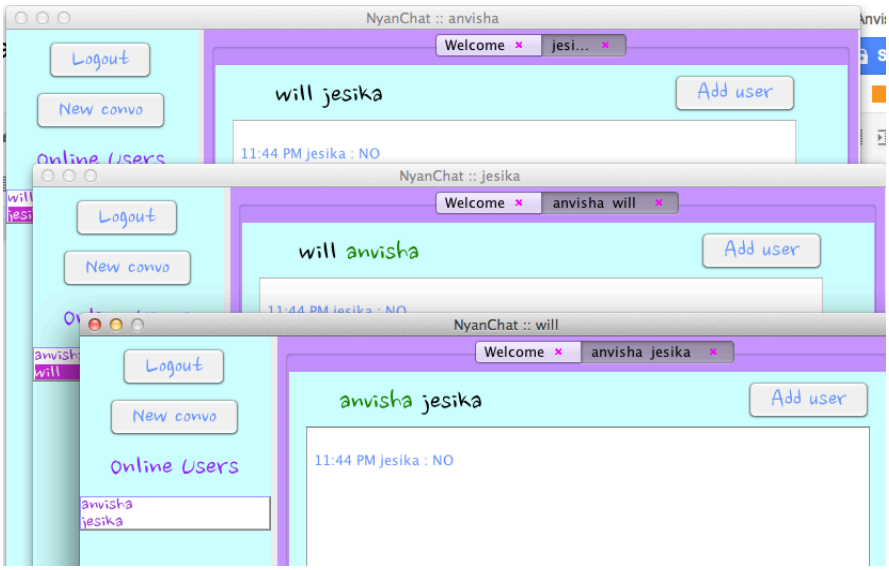
Chat with Typing Status



Message sending capabilities



Multi-person chat


NyanChat :: anvisha
Logout
New convo
Online Users
Welcome
jesi...
will jesika
Add user
11:44 PM jesika : NO
NyanChat :: jesika
Logout
New convo
Welcome
anvisha will
will anvisha
Add user
NyanChat :: will
Logout
New convo
Online Users
anvisha
jesika
Welcome
anvisha jesika
anvisha jesika
Add user
11:44 PM jesika : NO