Final Assignment: Chat Room

Group Members:
Taylor McGough
James Riddle
Matthew Bollinger
Martin Anilane

We will be creating a chat room application. Users will be able to register with the application, login, choose a chat room to enter, and chat with other users who are online. The server is expected to support up to 10 separate client connections at once.

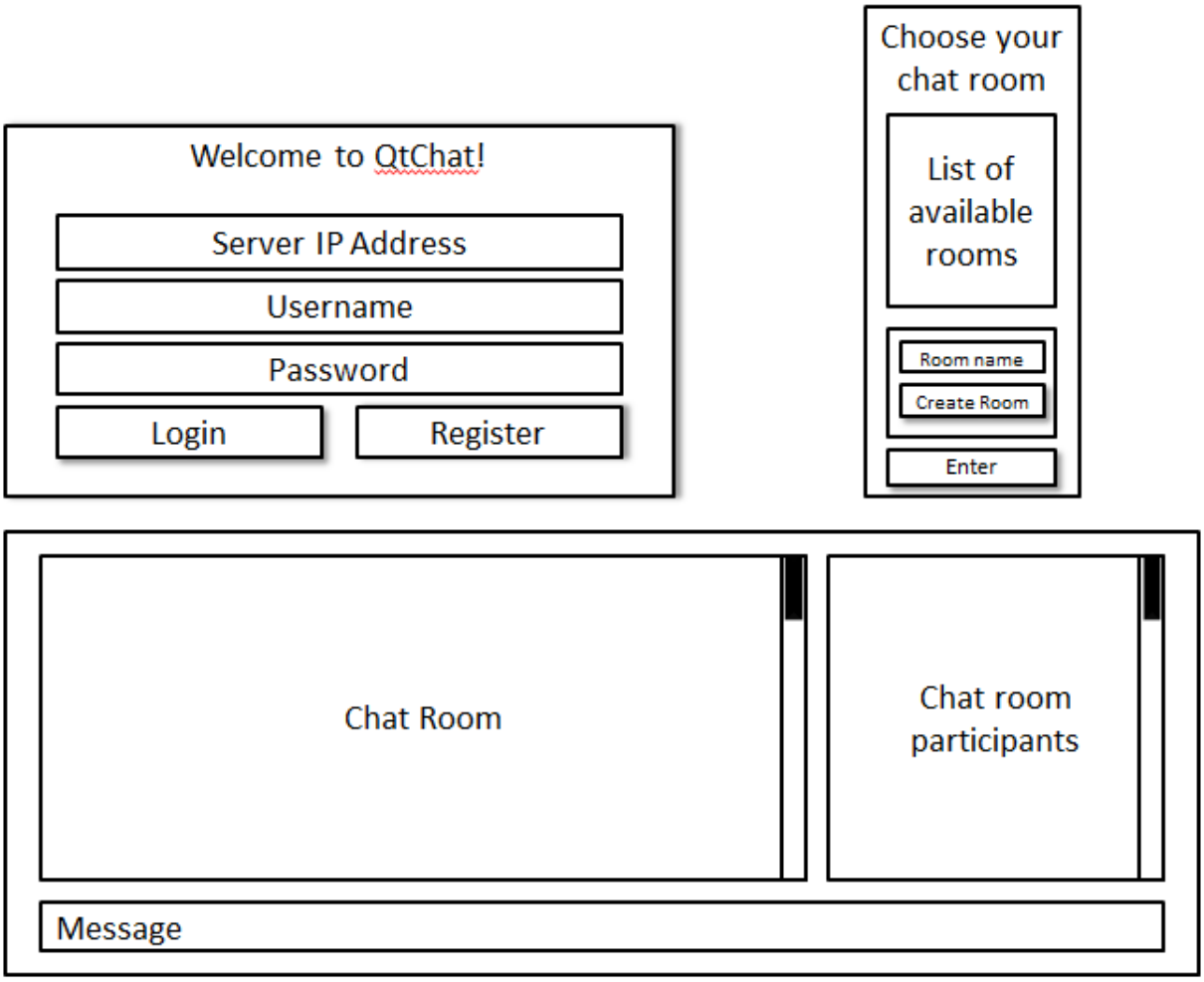
Our Qt Chat room will consist of three GUIs. The first GUI will be the login screen, which will be comprised of two input text fields and two buttons. The first input field is the IP address of the server. The second input text field will store the “Username” and the third input text field will store the “Password.” The “Login” button will check if the username/password combination is a valid entry in our database of users, while the “Register” button will generate a new database entry for the username/password combination. If the user inputs an invalid username/password combination and attempts to login, the GUI will prompt them with an error message. If a user logs in with a valid username/password combination they will continue to the chatroom selection GUI.

In this GUI, the user will be able to select from a list of pre built chat rooms, and hit the “Enter” button to join the chat room. Or, if they want to make a new room, the user will be able to enter the name of the new room and click the “Create Room” button to have the server create a new chat room. Once a user has joined a chat room, they will use our third and finally GUI. This GUI, will have a main read only text field in the center of the screen displaying messages sent to the server by the chat room users. This field will scroll to display all of the messages sent in the chat room for the time the user has been in the room. As messages are sent in the chat room, they are displayed in this field. Each message is displayed on its own line with the following format: <name of sender>: [message text]

To the right of the main text field, a list of current chat room participants will be displayed. This field will scroll if the number of current users in the room exceeds the space of the window. The purpose of this list is to inform the chat room participants what other participants are available to talk to.

Below the main text field, the user may input their message into the “Message” text field and press the “**Enter**” key on the keyboard to send their message.

To leave a chat room, the user simply exits the chat window. The user remains logged in as long as they keep the chat room list GUI running.



One method of validation that we will be using to test will be simulating a client and server on a single local machine. We will use QEvents to create a new user, and join a chat room through the client. We will then query the database stored in the server and test to make sure that correct corresponding entries were stored in the database. Similarly, we will test to make sure that valid messages were sent from client to client through the server on a local machine using QEvents.

To test the login GUI, we will have test cases for valid login, invalid login (incorrect username/password combination), valid new user registration, and invalid new user registration (the username already exists). These tests will send off arbitrary username/password combinations and test on predetermined responses. We will also have tests to check if the correct information is properly sent to the server. The valid login test will receive a “login okay” message and verify that the chat room selection GUI is launched. The invalid login test will receive a “login bad” message and verify that an error is shown. The valid new user registration test will receive a “registration okay” message and verify that a message notifying successful registration is shown. The invalid new user registration test will receive a “registration bad”

message and verify that an error is shown.

To test the server's login functionality, we will have test cases for valid login, invalid login (incorrect username/password combination), valid new user registration, and invalid new user registration (the username already exists). To test valid logins, the server will receive a username/password combination that exists in its database. It will query the database with this username/password combination and verify that it exists. To test invalid logins, the server will receive a username/password combination that does not exist in its database. It will query the database with this combination and verify that they do not exist. To test new user registration, the server will receive a username/password combination that does not exist in it database. It will verify that the combination does not exist, add it to the database, and then verify that it was properly added. To test invalid new user registration, the server will receive a username/password combination that exists in its database. It will verify that the combination exists in the database and send a message saying that the combination already exists.

The chat room list GUI will be tested by selecting a chat room and validating that the chat window was spawned. This will be tested with an input of QEvent to click on a corresponding chatroom. To verify that the client is in a chatroom, we will call a method called getstate(), and it will return whether the client is in login, chat select, or chat room. We will then compare this to the expected state.

The next test is to to verify method transmission, a test message will be sent using QEvents to a local client. The input for this test will be a QEvent and a test QString. We will verify that the slot for receiving messages in the local client was called by the server socket and the message was received. The output will be to call a method in the receiving client called getMessage() which will return and output the most recently received message, and compare against the tested sent message.

To test for multiple chat rooms, we will follow all of the procedures above to create 2 new and separate clients in a separate chatroom than the above. We will then verify a message sent in this new chat room with a test message will be sent using QEvents to a local client. The input for this test will be a QEvent and a test QString. We will verify that the slot for receiving messages in the local client was called by the server socket and the message was received. The output will be to call a method in the receiving client called getMessage() which will return and output the most recently received message, and compare against the tested sent message.

Finally we will test simultaneous clients attempting to send a message. To do this, we will create a connection from a signal in the test class to 2 separate clients which will cause them both to send a message. We will emit the signal, and then verify that the slot for receiving messages in the local client was called by the server socket and the message was received. The output will be to call a method in the receiving client called getMessage() which will return and output the most recently received message, and compare against the tested sent message.

This chatroom server is designed and expected to handle 10 clients connected to it. We will be verifying this in the testing by creating 10 individual clients on a local machine, and connecting all to a single chatroom. We will do this through the same process as described above to connect 2 clients to a chatroom. Once all 10 of these clients are successfully created and connected, we will verify that they are able to successfully receive message through the same process as described above.