CSI32/E01 Project 4

May 9, 2016 Due: May 25, 2016

On p. 557, there are a series of suggested projects for improving the chatroom.  
For project 4, pick one of 16.8 - 16.13  
Or...think of a more ambitious improvement...just clear it with me first.  
Extra credit will be given for documentation, such as giving sequence diagrams for any new behavior.

**Exercise 16.8:** The chat room server of Figure 16.10 takes care to terminate a client’s

handler thread if it detects that the connection to that client has failed. However, the

implementation of the chat room client of Figure 16.13 does not gracefully handle

the case of a dropped connection to the server. If the server fails, the client continues

to run but without ever receiving any activity from the server. Although there is no

way to recover the connection, rewrite the client software so that it explicitly informs

the user that the connection has been lost.

**Exercise 16.9:** Consider the following behavior of our chat room software. When a user

broadcasts a message to the group (such as that on line 5 of Figure 16.14), that

message is then echoed on that user’s client as it is received from the server (as with

line 6 of that figure). Rewrite the *client* software so that it does not display an echo

of a message sent by this uers. Accomplishing this task does not require any change

to the server’s code nor to the underlying protocol being used.

**Exercise 16.10:** If a person sends a private message to an unrecognized screen name, our

server ignores the request. However, the user who sent the message is not informed

of this failure. A better protocol is to have the server transmit an UNKNOWNUSER

message back to the client in this scenario. Rewrite both the server and client software

based on such a protocol, ensuring that the user is appropriately informed.

**Exercise 16.11:** Our chat room software has the following flaw. If a new person connects

to the server using the same screen name as an existing person, the new person’s

connection wipes out the first person’s connection in the \_socketLookup dictionary.

A better approach is to adapt the protocol as follows. When a new person sends

an ADD request to the server, the server should reject the request by transmitting

an UNAVAILABLE message back to the client. Rewrite both the server and client

software based on such a protocol, ensuring that the user is given an appropriate

explanation and an opportunity to select a new screen name.

**Exercise 16.12:** When a new person comes into the chat room, they have no initial idea

who else is in the room. Redesign the network protocol and reimplement the client

and server, so that a person is informed about the existing members when joining.

**Exercise 16.13:** Extend the chat room software so that private messages can be sent simultaneously

to more than one person.

**Exercise 16.14:** The software we have written for the chat room server and client work

well together. However a third party could crash our server by writing client-like

software that abuses our protocol. Consider the following interactive session with an

active server s:

>>> s.connect( ('localhost', 9000) )

>>> s.send('ADD hacker')

10

>>> s.send('\n')

1

>>>

The final transmission causes our server to crash. First, explain the underlying cause

of the crash, and then rewrite the server software to avoid such a scenario.