CSE 5306 **Lab # 1** **Spring – 2018**

A Chat Room System

Objectives:

1. Understanding two communication protocols for distributed processing, HyperText Transport Protocol - HTTP, and Sockets.
2. Using an Integrated Development Environment (IDE)

Due: 4 March, 11:59:59 pm

Project Specification:

These will be individual projects. You may write the program in any language that is supported under any Integrated Development Environment (IDE). Keep in mind that more help may be available to you in some languages than in others. Furthermore, available controls, objects, libraries etc. may make some of these tasks easier in one language than in another. Finally, because of the lack of restrictions on IDEs, you will have to have that IDE available to demo to the TA. For example, you might bring a laptop to demo the program. Socket programming is so universal that you can probably find major portions of this part of the program with searching on Google. Using code you find on the Internet is fine, but be sure to document the source in the writeup and in the program source code!

You will write a chat room system consisting of a server and **four** client processes. Each client process will connect to the server over a socket connection and register a user name at the server. The server should display the names of the connected clients.

From that point, any messages typed at one client should be sent to the server which will relay it to the other connected clients. The operation should be similar to any number of chat room applications. The chat interface should be a **very simple** GUI: a box to display messages and enter text will suffice.

When a client logs onto the server, every client in the system should instantiate a timer to track the interval between messages received from that client. The interval between messages received from that client should be printed along with the message itself.

For example: clients A, B, C, and D are connected to a server. If client A send a message “Hello” at time 1:00, then sends message “All you” at time 1:04, and message “listeners” at time 1:05, the output at clients B, C, and D should read something like:

A (0:00) – Hello
A (0:04) – All you
A (0:01) – listeners

If a client logs off the other clients should be notified. The server should also display the messages that come through. The messages should use HTTP formats and commands.

The HTTP tags must use, at minimum, `Host`, `User-Agent`, `Content-Type`, `Content-Length`, and `Date`. If you are polling the server, use `GET`. If you are sending data to the server, use `POST`.

Your program must operate independently of a browser. Time on the messages should be encoded according to HTTP.

A Chat Room System

References:

1. http://www.w3.org/Protocols/ This is the standard.
2. http://www.jmarshall.com/easy/http/ HTTP Made Really Easy.
3. http://tangentsoft.net/wskfaq/ Winsock Programmer's FAQ
4. http://www.eggheadcafe.com/articles/20020323.asp VB.net Single thread Telnet client & server.

Submission Guidelines:

FAILURE TO FOLLOW ANY OF THESE DIRECTIONS WILL RESULT IN DEDUCTION OF SCORES.

Submit your assignment via the submission link on Blackboard. You should zip your source files and other necessary items like project definitions, classes, special controls, DLLs, etc. and your writeup into a single zip file. No other format other than zip will be accepted. The name of this file should be your lastname_loginID.zip. Example: If your name is John Doe and your login ID is jxd1234, your submission file name must be "Doe_jxd1234.zip".

Be sure that you include everything necessary to unzip this file on another machine and compile and run it. This might include forms, modules, classes, config. files, etc. DO NOT INCLUDE ANY RUNNABLE EXECUTABLE (binary) program. The first two lines of any file you submit must contain your name and student ID. Include it as comments if it is a code file.

You may resubmit the project at any time. Late submissions will be accepted at a penalty of 10 points per day. This penalty will apply regardless of whether you have other excuses. In other words, it may pay you to submit this project early. If the TA can not run your program based on the information in your writeup then he will email you to schedule a demo. The TA may optionally decide to require all students to demonstrate their labs. In that case we will announce it to the class. It is your responsibility to keep checking blackboard for announcements, if any.

If your program is not working by the deadline, send it anyway and review it with the TA for partial credit. Do not take a zero or excessive late penalties just because it isn't working yet. We will make an effort to grade you on the work you have done.

Writeup:

Your write-up should include instructions on how to compile and run your program. Ideally it should be complete enough that the TA can test your program without your being there. Your writeup should include any known bugs and limitations in your programs. If you made any assumptions you should document what you decided and

why. This writeup can be in a docx or pdf format and should be submitted along with your code.

Grading:

Points – element

10 - Client Process works correctly
15 – Server Process works correctly
10 – Server shows HTTP message format
10 – HTTP message formats are valid
10 – Client logs in
10 – Clients correctly handle time intervals between messages
10 – Client correctly handles rejection of a bad name
10 – Client & server handle logoff correctly
10 – Broadcast is received correctly at all clients
05 – Comments in code

Deductions for failing to follow directions:

-10 Late submission per day.
-5 Including absolute/ binary/ executable module in submission
-2 Submitted file doesn't have student name and student Id in the first two lines.
-5 Submitted file has a name other than student's lastname_loginID.zip.
-5 Submission is not in zip format.
-5 Submitting a complete installation of the java virtual machine.

To receive full credit for comments in the code you should have **brief** headers at the start of every module/ subroutine/ function explaining the inputs, outputs and function of the module. You should have a comment on every data item explaining what it is about. (Almost) every line of code should have a comment explaining what is going on. A comment such as /* Add 1 to counter */ will not be sufficient. The comment should explain what is being counted.

Bonus:

10 - Make the server multithreaded. The program should show what is happening so that the TA does not have to dig into the code. Somehow show which thread is associated with which connection. The Task Manager can show how many threads a process is running.

10 - Have the server maintain a database of messages submitted to it that will be reloaded if the server shuts down and then restarts. Display the saved text only at the server – there is no need to retransmit to the client.

Deductions for failing to follow directions:

-10 Late submission per day.

-5 Including absolute/ binary/ executable module in submission
-2 Submitted file doesn't have student name and student Id in the first two lines.
-5 Submitted file has a name other than student's lastname_loginID.zip.
-5 Submission is not in zip format.
-5 Submitting a complete installation of the java virtual machine.

Important Note:

You may discuss the problem definition and tools with other students. You may discuss the lab requirements. You may discuss or share project designs. All coding work must be your own. You may use any book, WWW reference or other people's programs (but not those of other students in the class) as a reference as long as you cite that reference in the comments. If you use parts of other programs or code from web sites or books YOU MUST CITE THOSE REFERENCES. If we detect that portions of your program match portions of any other student's program it will be presumed that you have collaborated unless you both cite some other source for the code. You must not violate UTA, state of Texas or US laws or professional ethics. Any violations, however small, will not be tolerated.