CSCI-P 538 Fall 2016 Project 3
Hoosier Chat Room
(Engineering Track A)

A Deadline

December 2 2016 23:59:59 EST. This is a hard deadline and no extension will be given. Any clarification queries should be sent to `p538fall16-l@list.indiana.edu`.

B Teamwork

This project is an individual assignment, and no collaboration among students is allowed. In other words, a team consists only one student.

C Problem Description

In this project, you will practice socket programming by developing a chat room application called HoosierChat. It allows multiple users on different hosts to join a virtual chat room where a user can send messages to other user(s). HoosierChat uses the client-server architecture. A message from a client will be first sent to the server, which will then distribute the message to the corresponding recipient(s). You will need to implement both the client and server applications on Linux. Your program must use the TCP socket interface provided by C/C++/Java. You are not allowed to use any third-party library or any operating system other than Linux.

C.1 Required Features

Your HoosierChat application must support all following features.

1. *Account registration.* A new user connects to the server and registers a new account. The user will need to provide a username and password which will be stored on the server side. The lengths of both a valid username and password are between 4 and 8 characters including only letters (case-sensitive) and digits.
2. *Login and logout.* A user with an existing account can log onto the chat room by providing her username and password. The server admits the user if the credential is correct. A user in the chat room can log out at any time. We say a user is *online* if she is currently in the chat room. On one client, at most one user can be online at the same time. Your server, on the other hand, needs to support at most 16 online users.

3. *Sending and receiving public messages.* An online user can send a public message, which will be delivered to all online users except the sender herself. By default, all recipients will see the sender's username and the message content. Optionally, the sender can choose to make the message *anonymous*. In that case the sender's username will not be sent to or shown to the recipients. The maximum length of a message is 4,096 bytes. A message must be delivered and displayed in real time.

4. *Sending and receiving private messages.* An online user can also send a private message that is delivered to only one recipient specified by the sender (by username). The sender can also choose to make a private message anonymous. One cannot send a private message to herself. A message must be delivered and displayed in real time.

5. *List all online users.* An online user can ask the server to provide a list of all online users.

6. *Server-side monitoring.* The HoosierChat server program should display in real time all messages received and sent by the server, as well as the events of (both successful and unsuccessful) user registration, logging in, and logging out.

C.2 Optional Features

Besides realizing all required features described in §C.1, you also need to implement at least one optional feature. There is no restriction on what kind of optional feature(s) you implement, as long as they are relevant to HoosierChat from an end user's perspective. Some examples are listed below. You are free to choose any one (or more) of them. Also feel free to think of other interesting features.

1. *Message Encryption.* Messages sent in plain text are vulnerable to packet sniffing attack. Consider encrypting the messages and commands to prevent this.

2. *File transfer.* Allow an online user to transfer files (*e.g.,* an image) to another online user.

3. *Group chat.* Allow a subset of online users to form a group where a group member can send messages to all other online users in the same group. You will need to work out the details such as how to create, join, and leave a group.

4. Other cool features you can think of.

You need to document your implemented optional feature(s) in the documentation (§C.6).

C.3 Protocol Design

You need to design the HoosierChat protocol by yourself. There are several design decisions you need to make, such as deciding to use a text-based or binary-based protocol, defining the message syntax, defining the semantics of message fields *etc.* Usually there is no "correct" or "incorrect" solution here. You may want to make your protocol easy to parse and to debug.

C.4 User Interface Design

Since our focus here is networking, you just need to make a console application (having a fancy graphical user interface will not bring you extra points). HoosierChat uses command-based UI *i.e.,* users issue a set of simple commands to interact with the application. Below shows an example of the set of commands. Feel free to design your own commands as long as they can realize all required features of HoosierChat. To support optional features (§C.2), you need to further expand the command set.

Command	Description
`REGISTER [username] [password]`	Register a new account
`LOGIN [username] [password]`	Log in with an existing account and enter the chat room
`LOGOUT`	Log out and leave the chat room
`SEND [msg]`	Send a public message
`SEND [username] [msg]`	Send a private message to a user
`SENDA [msg]`	Send an anonymous public message
`SENDA [username] [msg]`	Send an anonymous private message to a user
`LIST`	List all online users

C.5 Error Handling

Your program should be able to handle to various error situations such as the following.

1. When your client cannot reach the server (*e.g.,* due to loss of the Internet connectivity or unexpected termination of the server), the client application should print out some error message and exit (instead of crashing directly).
2. Your server should never crash even when there is no network connectivity. When a client is lost unexpectedly, the server should continue serving other clients if possible.
3. There are other errors or corner cases you need to consider, such as a client attempting to send a message to an offline or non-existing user, providing wrong password when logging in, or issuing an unknown command or a command with incorrect arguments.

C.6 Documentation

You need to prepare a document describing the following.

1. Provide a short summary of the features (both required and optional) you have implemented. Describe any known bugs/problems with your implementation.
2. Provide detailed instructions on how to compile and run your code.

3. Provide a user manual describing how to use your software (so we can test it ourselves). In the user manual, list all commands and their formats.

4. Describe specifications of the application protocol you designed for HoosierChat.

D Submit Your Program

Compress all source code files (.c/.cpp/.java) and the documentation (.pdf/.txt/.doc/.docx) into a single file called `project3A.zip`, `project3A.bz2`, or `project3A.tar.gz` (depending on the compression program you use). Submit it to Canvas. Do not include any executable in your submission.

If you use C/C++, we strongly recommend you work with an update-to-date GNU C/C++ compiler (gcc/g++ version 4.3 or above). Do not use other C/C++ compilers that might be incompatible to gcc/g++. For Java, please use JDK version 7.0 or above.

E Test and Grading

We will test your software over real Internet. The server will be running on an Amazon EC2 instance, a commercial cloud platform. You are also strongly recommended to use EC2 to test your server (detailed instructions will be given). We will launch multiple client instances to test your application by letting clients to "chat" with others. Note you do not need to use multiple physical machines to test multiple clients – they can run on the same end host.

You will also be asked to give a demo of your code to the AIs (detailed logistics will be announced soon). The following table describes how your final grade will be calculated.

Test Cases	Points
User registration and login/logout	8
Sending and receiving messages	28
List all users	4
Server-side monitoring	4
Optional feature(s) (§C.2)	8
Error handling	12
Usability	8
Documentation	8
Total	80 Points

F Honor Code

Students must follow the IU honor code (`http://www.iu.edu/~code/code/responsibilities/academic/index.shtml`). This project is an individual assignment, and no collaboration among

students is allowed. In no case may your code be copied from another student or a third-party source on Internet. We will use an anti-plagiarism software to detect code "shared" among students. Any violations of the honor code will be dealt with strictly, including but not limited to receiving no credit for the entire project.