Project 1 Description

2023-02-01 Edits: Clarified behavior for multiple instances of the same user. Added option to to exit the client program.

CS 2510, Spring 2023: Project 1

Project Overview

In this project, you will implement a basic group chat system. The system includes a server program that is responsible for storing messages, and a client program that can be used to join and leave groups, and to send and receive messages in those groups.

For this project, you should assume:

- You have an extremely reliable server that never crashes
- You may have many clients; you should not assume any specific bound on the number of clients
- Clients may crash
- The network may arbitrarily drop, delay, or re-order messages

Over the course of the semester, we will learn how to build more reliable systems that weaken the assumption that the server never crashes.

Chat Service Specification

You should implement a chat service that allows clients to perform the actions below, along with a simple text-based client program that allows users to access those actions interactively as follows:

1. Connect to a chat server. `c localhost:12000` will connect to a server running locally on port `12000`. Note that either a hostname or an IP address may be provided. A client must be connected to a server to perform any of the operations below.

2. Login with a user name. `u Amy`. The user can change their identity at any time using the same option (the client will be removed from the chat group the old user was in, if any). Note that a **user** may be simultaneously logged in through multiple **clients**. Different instances of the same user (i.e. different client programs) may join the same group or different groups. If one user instance (client program) logs out or changes groups, that should NOT cause other instances of the same user to change groups or be removed from their current group.

3. Join a chat. `j group1`

If the specified group does not exist, it is created.

Upon joining a chat, the current state of the group should be presented, such that all the current (logged in) group participants are listed, and the most recent 10 messages in this group (lines) are presented, oldest to newest. The number of unique 'likes' for each line (see below) should be presented to the right of the line.

Example:

```
Group: CS2510 Staff
Participants: Amy, Maddie, Shriharsha
1. Amy: Hello!
2. Amy: Has anyone started the project?                     Likes: 2
3. Maddie: Not sure yet...
```

After joining a chat, upon an update to the chat, the screen should automatically update to include the new message, or to include or remove attendees (the screen should only refresh upon a relevant update).

A connected client can switch to a different chat room at any time by joining the new chat room. Joining a new chat removes the client from its previous chat room. The client needs to be in a chat to perform any of the operations below.

4. Append to a chat. `a` and then type a line.

   The relevant chat should be updated so that the new line is added at the end of the chat. You can assume that the maximum number of characters in a line is 80 and that each line is, in fact, a single line (i.e. no newline characters other than the final one at the end).

5. Mark line as 'liked'. `l 5`. (Note, this is a lowercase L, not the number one)

   The relevant line on the screen will be marked as liked by this user, but only if the user is not the creator of the line. Each user can only be counted once for each line. Note that as a consequence of this update, the screen may need to be refreshed. Line numbers are only relevant to the current display.

6. Remove your 'like' from a line (if you 'liked' it). `r 5`.

   The user's 'like' will be removed from that line if it is currently liked by that user.

7. Print the full message history of the chat (not only the latest 10 lines). `p`. Note that it is ok to go back to showing only the latest 10 lines the next time the screen refreshes.

8. Quit the program. `q`

Submission Details

You are required to submit an **initial design document (https://canvas.pitt.edu/courses/186401/assignments/1151064)** by **9:30 am Tuesday January 31**.

Your final submission is due by **11:59 pm Wednesday February 15.**

For your final submission, you must include:

1. An updated design document that describes your final project design

2. Your project code

3. A Dockerfile that creates an image we can use to run your code (installs dependencies and compiles code if needed) and a README with the exact commands to run your code.