Distributed Systems Assignment

CPD, 2024/2025

Deadline: 23rd May

1 Introduction

You have recently been hired by a computing company to design and develop in Java a new client-server chat system using TCP. The company has users that must authenticate with the system in order to exchange text messages among them in different chat rooms. When chatting in a room their name/id will precede the message and messages appear as a timeline. Example:

```
Room: Library
[Alice enters the room]
Alice: Hi people! Anyone?
[Eve enters the room]
Eve: I am giving free apples and Eve memecoin.
Alice: Give me an apple please.
...
```

Users connect to the system using some Java client code that allows users to first authenticate. After successful authentication, the client lists available rooms for the user to choose from. As soon as a user enters a room s/he should immediately see any new messages that are written to the room, even if the user does not type anything.

Users can also create new rooms by entering a room name not yet in use. They can also create special AI rooms that include a *prompt*. In the AI rooms a connection is opened to a local LLM, e.g. Ollama[1], and every time a new user message is added to the room, the whole context of messages and prompt are given to the AI and the response is written back to the room under the special user name `Bot`. As an example, the room "AI doodle" would summarize all user availability suggestions and propose a common meeting time. Think of the `Bot` as an extra user that is keeping track of the conversation and giving back some help.

The server needs to accept and authenticate user connections, and add the users to rooms of their choice. In AI rooms a local connection must also be made to the LLM process.

[1] **Check the notes on Ollama in CPD's Moodle.**

2 Concurrency

This is one of the most important grading criteria of your assignment. Your design should strive for the following goals:

No race conditions in the access to shared data structures or in synchronization between threads. You cannot use thread safe implementations of the Java collection classes that belong to `java.utils.concurrent`. Instead, you should design your own implementations using the classes of `java.utils.concurrent.locks`

Minimize thread overheads i.e. use the recently introduced Java virtual threads. (You need to use Java SE 21 or more recent.)

Avoid slow clients from bringing the system to a crawl.

3 User registration and authentication

You may provide a (sub) protocol for user registration. Furthermore, you may also persist the registration data in a file. Alternatively, you can provide a file with registration data.

You may want to decouple authentication from chatting in a room. I.e. allow for a user to send its username and password once, and then move around and chat in different rooms.

You may also want to use secure channels **(i.e. package `javax.net.ssl`)** in the communication between the clients and the servers, so as to ensure good practices in secure communications.

4 Fault Tolerance

Your implementation should tolerate broken TCP connections, i.e. the user state, both in the client and in the server, should not be lost in that event. For example, if the connection is broken while the user is in a room, the client (not the user) should reconnect and the user will resume its session in the same chat room, without having to authenticate itself again, or even rejoining the room: the server will start relaying the messages received on the chat room using the new connection.

Note that the client should not cache user credentials (this is not secure). Instead, it should use a token (similar to an HTTP cookie) sent by the server, that the client shall send upon reconnecting, allowing the server binding the new TCP connection with the user.

If tokens do not expire, they can be used to replace user credentials in every user authentication after the first. Thus, you may wish to associate an expiration time to every token.

5 Implementation and Submission

You should implement this client-server system in Java SE 21 (or more recent). You cannot use packages that do not belong to Java SE.

You should use your CPD group repository at Gitlab@UP to submit the code as well as a README file with instructions about how to run the server and the clients. Submission deadline is the **23rd of May 2025, but we will accept submissions without penalty until the 25th of May (no submissions with a later date will be accepted)**.