Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-004-S2024/it114-sockets-part-1-3-checkpoint/grade/grg

IT114-004-S2024 - [IT114] Sockets Part 1-3-Checkpoint

Submissions:

Submission Selection

1 Submission [active] 2/19/2024 7:04:37 AM

Instructions

COLLAPSE

Create a new branch for this assignment
Go through the socket lessons and get each part implemented (parts 1-3)
    You'll probably want to put them into their own separate folders/packages (i.e., Part1, Part2, Part3) These are for your reference
    Part 3, below, is what's necessary for this HW
    https://github.com/MattToegel/IT114/tree/Module4/Module4/Part3
Create a new folder called Part3HW (copy of Part3)
Make sure you have all the necessary files from Part3 copied here and fix the package references at the top of each file
    Add/commit/push the branch
    Create a pull request to main and keep it open
Implement **two** of the following **server-side** activities for all connected clients (majority of the logic should be processed server-side and broadcasted/sent to all clients if/when applicable)
    Simple number guesser where all clients can attempt to guess while the game is active
        Have a /start command that activates the game allowing guesses to be interpreted
        Have a /stop command that deactivates the game, guesses will be treated as regular messages (i.e., guess messages are ignored)
        Have a guess command that include a value that is processed to see if it matches the hidden number (i.e., */guess 5*)
            Guess should only be considered when the game is active
            The response should include who guessed, what they guessed, and whether or not it was correct (i.e., Bob guessed 5 but it was not correct)
        No need to implement complexities like strikes
    Coin toss command (random heads or tails)
        Command should be something logical like /flip or /toss or /coin or similar
        The result should mention *who* did *what* and got what *result* (i.e., Bob Flipped a coin and got heads)
    Dice roller given a command and text format of "/roll #d#" (i.e., roll 2d6)
        Command should be in the format of /roll #d# (i.e., roll 1d10)
        The result should mention *who* did *what* and got what *result* (i.e., Bob rolled 1d10 and got 7)
    Math game (server outputs a basic equation, first person to guess it correctly gets congratulated and a new equation is given)
        Have a /start command that activates the game allowing equaiton to be answered
        Have a /stop command that deactivates the game, answers will be treated as regular messages (i.e., any game related commands when stopped will be ignored)
        Have an answer command that include a value that is processed to see if it matches the hidden number (i.e., */answer 15*)

the hidden number (i.e., */answer 15*)
The response should include who answered, what they answered, and whether or not it was correct (i.e., Bob answered 5 but it was not correct)
Private message (a client can send a message targetting another client where only the two can see the messages)
Command can be /pm, /dm followed by the user's name or an @ preceding the users name (clearly note which)
The server should properly check the target audience and send the response to the original sender and to the receiver (no one else should get the message)
Alternatively (make note if you do this and show evidence) you can add support to private message multiple people at once. Evidence should show a larger number of clients than the target list of the private message to show it works. Note to grader: if this is accomplished add 0.5 to total final grade on Canvas
Message shuffler (randomizes the order of the characters of the given message)
Command should be /shuffle or /randomize (clearly mention what you chose) followed by the message to shuffle (i.e., /shuffle hello everybody)
The message should be sent to all clients showing it's from the user but randomized
Example: Bob types */command* hello and everyone recevies Bob: lleho
Fill in the below deliverables
Save the submission and generated output PDF
Add the PDF to the Part3HW folder (local)
Add/commit/push your changes
Merge the pull request
Upload the same PDF to Canvas

Branch name: M4-Sockets3-Homework

Tasks: 7 Points: 10.00

Baseline (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Demonstrate Baseline Code Working

Details:
This can be a single screenshot if everything fits, or can be multiple screenshots

Checklist

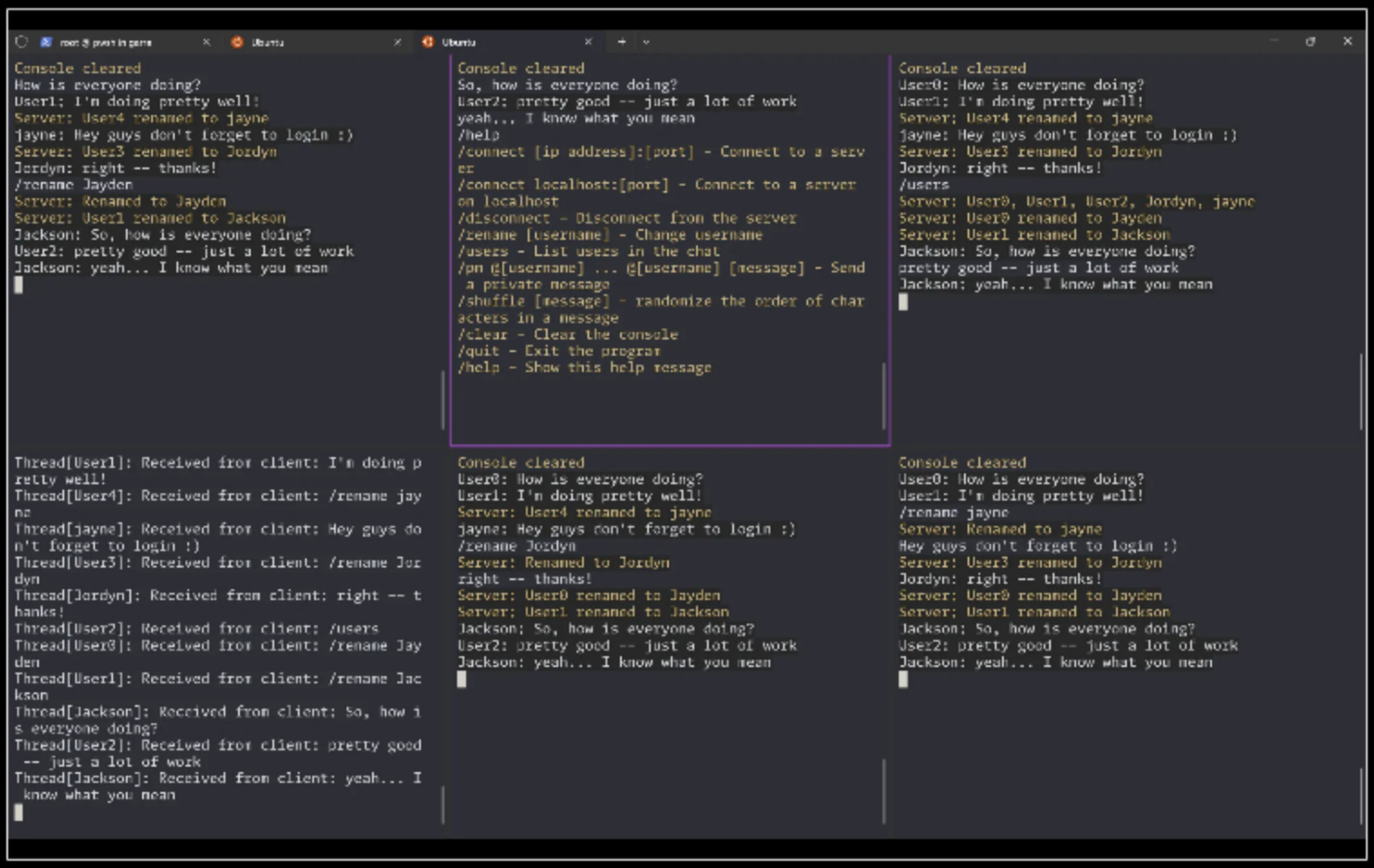
*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Server terminal/instance is clearly shown/noted
☐ #2	1	At least 3 client terminals should be visible and noted
☐ #3	1	Each client should correctly receive all broadcasted/shared messages
☐ #4	1	Captions clearly explain what each screenshot is showing
☐ #5	1	Include a screenshot showing you grabbed Parts 1-3 correctly and have them in your repository alongside Part3HW

Task Screenshots:

Gallery Style: Large View

Small Medium Large



each of the users are able to send broadcasts messages, receive the shared broadcast messages – and of course have multiple clients connecting to the same server

Checklist Items (0)

Feature 1 (3 pts.)

^COLLAPSE^

^COLLAPSE^

Task #1 - Points: 1

Text: What feature did you pick? Briefly explain how you implemented it

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Feature is clearly stated (best to copy/paste it from above)
☐ #2	1	Explanation sufficiently and concisely describes implementation (should be aligned with code snippets in related task)

Response:

Feature:

Private message (a client can send a message targetting another client where only the two can see the messages)

Command can be /pm, /dm followed by the user's name or an @ preceding the users name (clearly note which)
The server should properly check the target audience and send the response to the original sender and to the receiver (no one else should get the message)
Alternatively (make note if you do this and show evidence) you can add support to private message multiple people at once. Evidence should show a larger number of clients than the target list of the private message to show it works. Note to grader: if this is accomplished add 0.5 to total final grade on Canvas

Implementation:

I used the command /pm to indicate a private message as well as the format of @[username] to target other user(s) following the use of the /pm command.

To accommodate the targeting of multiple users, I first removed the '/pm ' from the input string, and then split the string on every " ". I also made a list of targets (to be populated later) and a stringbuilder for the private message. I then looped through the splat'd string and added any string that began with a "@" to the targets arraylist (without the '@') and appended anything that didnt begin with a '@' to the private message string builder. (in retrospect, a user might want to include a @ in their message, so should have it auto-detect the change and then break from the loop or otherwise add everything else from that point to the string builder).

I then perform a series of relatively trivial checks on the data to ensure that it is valid, and then I loop through all the targets. I then call a helper method to get the ServerThread for each client (just loops through and compares username) -- and if it is not null I perform another series of checks on the resulting serverThread. Then, if its going to just one person, it sends the message directly to that ServerThread with some additional formatting to let them know it was a private message. If it was sent to multiple people, it includes a newline, a header listing all the people, and then the message on the preceding line.

COLLAPSE

Task #2 - Points: 1

Text: Add screenshot(s) showing the implemented feature working (code and output)

Details:
Add screenshots of the relevant code changes AND relevant output during runtime

Checklist

*The checkboxes are for your own tracking

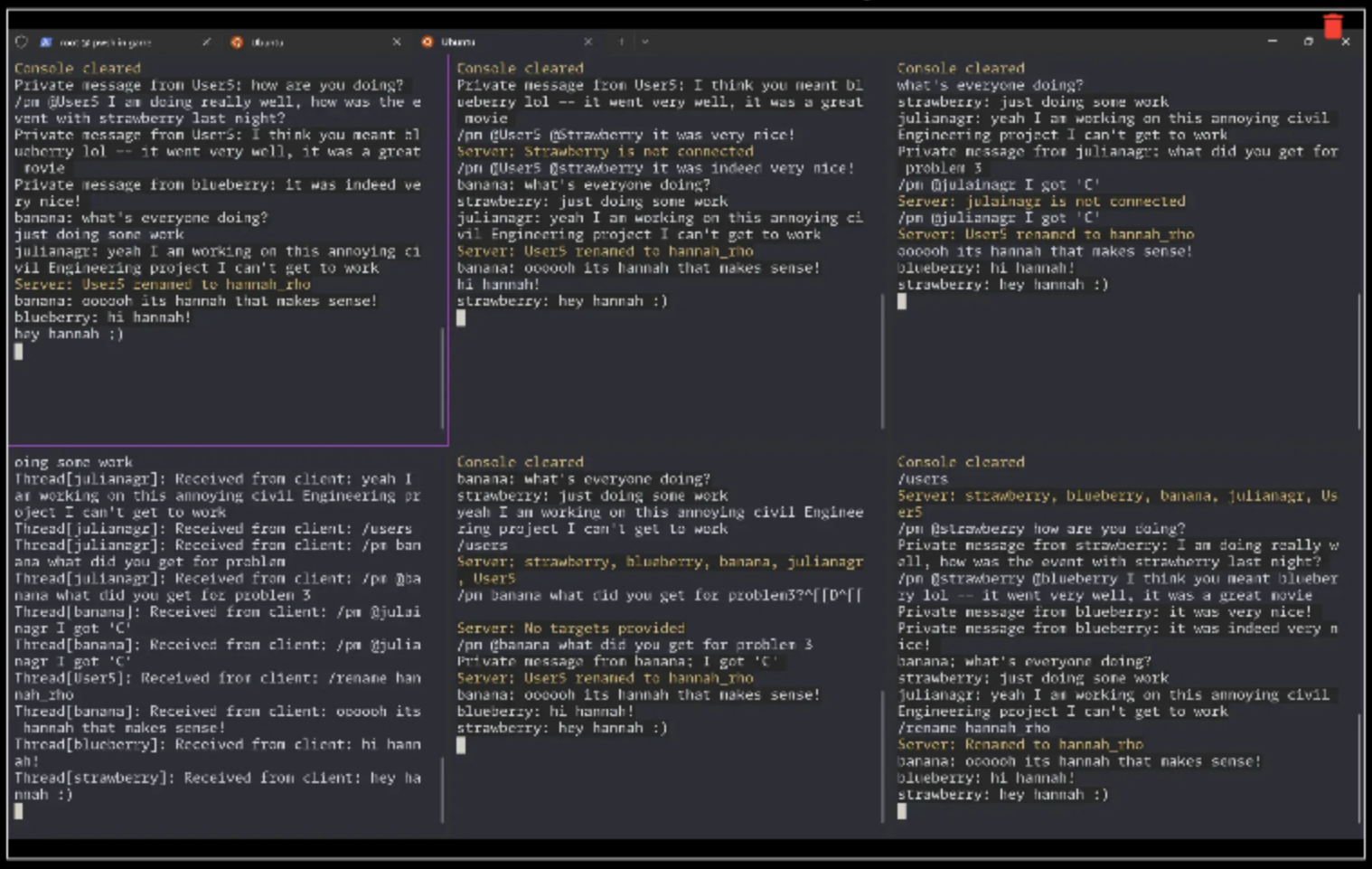
#	Points	Details
☐ #1	1	Output is clearly shown and captioned
☐ #2	1	Code shows relevant snippets that accomplish feature, UCID and date are present in all

code screenshots. Relevant captions are included for each screenshot of the code.

Task Screenshots:

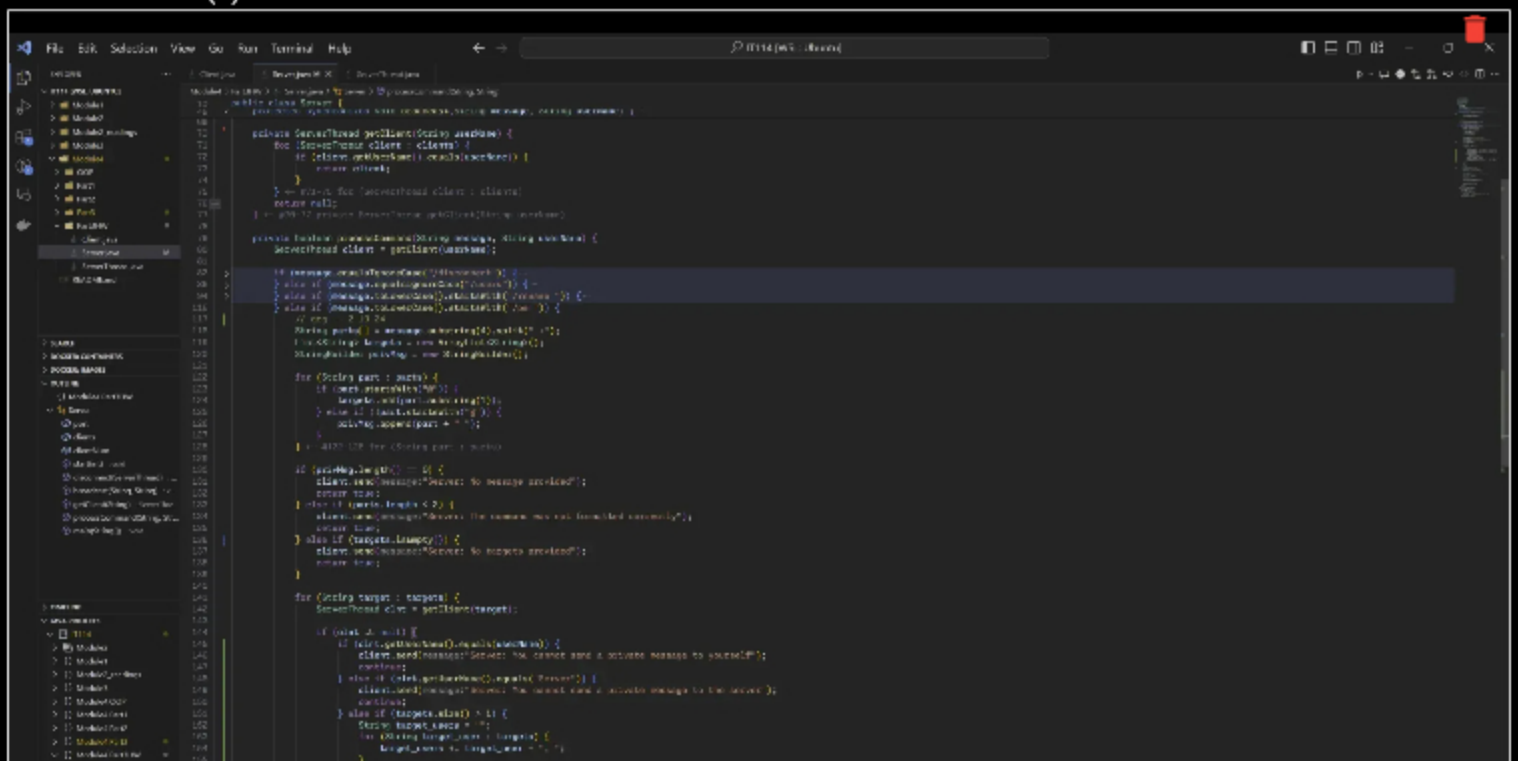
Gallery Style: Large View

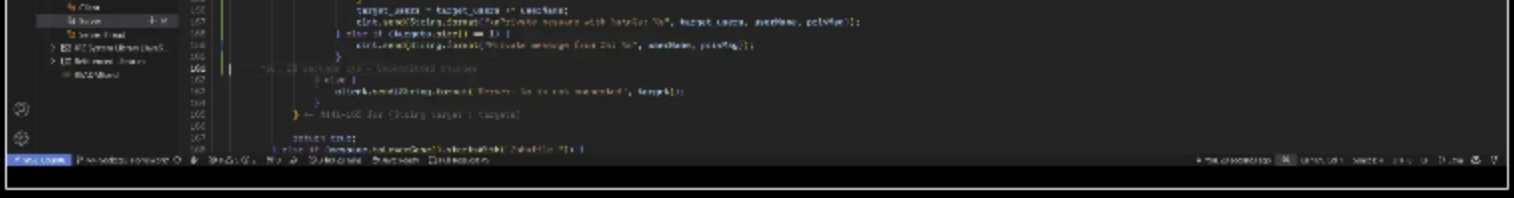
Small Medium Large



private messaging -- both to an individual/user/client or to a group of individuals/users/clients -- this still allows the use of broadcast messaging of course

Checklist Items (0)





implementation of private messaging for handling multiple users

Checklist Items (0)

Feature 2 (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: What feature did you pick? Briefly explain how you implemented it

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Feature is clearly stated (best to copy/paste it from above)
☐ #2	1	Explanation sufficiently and concisely describes implementation (should be aligned with code snippets in related task)

Response:

Feature:

Message shuffler (randomizes the order of the characters of the given message)

Command should be /shuffle or /randomize (clearly mention what you chose) followed by the message to shuffle (i.e., /shuffle hello everybody)
The message should be sent to all clients showing it's from the user but randomized
Example: Bob types /command hello and everyone recevies Bob: lleho

Implementation:

takes the passed string, removes the initial command section (the '/shuffle ' part) as well as creating an arraylist and string builder. It then has a short circuit logic check to ensure that it actually can be shuffled (should also have checked that there are enough *unique* characters but oh well). It then splits all the characters present in the input string into the arraylist. The code then appends a random character from that arraylist to the string builder and removes it from the arraylist. This new, shuffled, string is then returned both to the client as an alert as well as broadcast to the rest of the users.

^COLLAPSE^

Task #2 - Points: 1

Text: Add screenshot(s) showing the implemented feature working (code and output)

Details:
Add screenshots of the relevant code changes AND relevant output during runtime

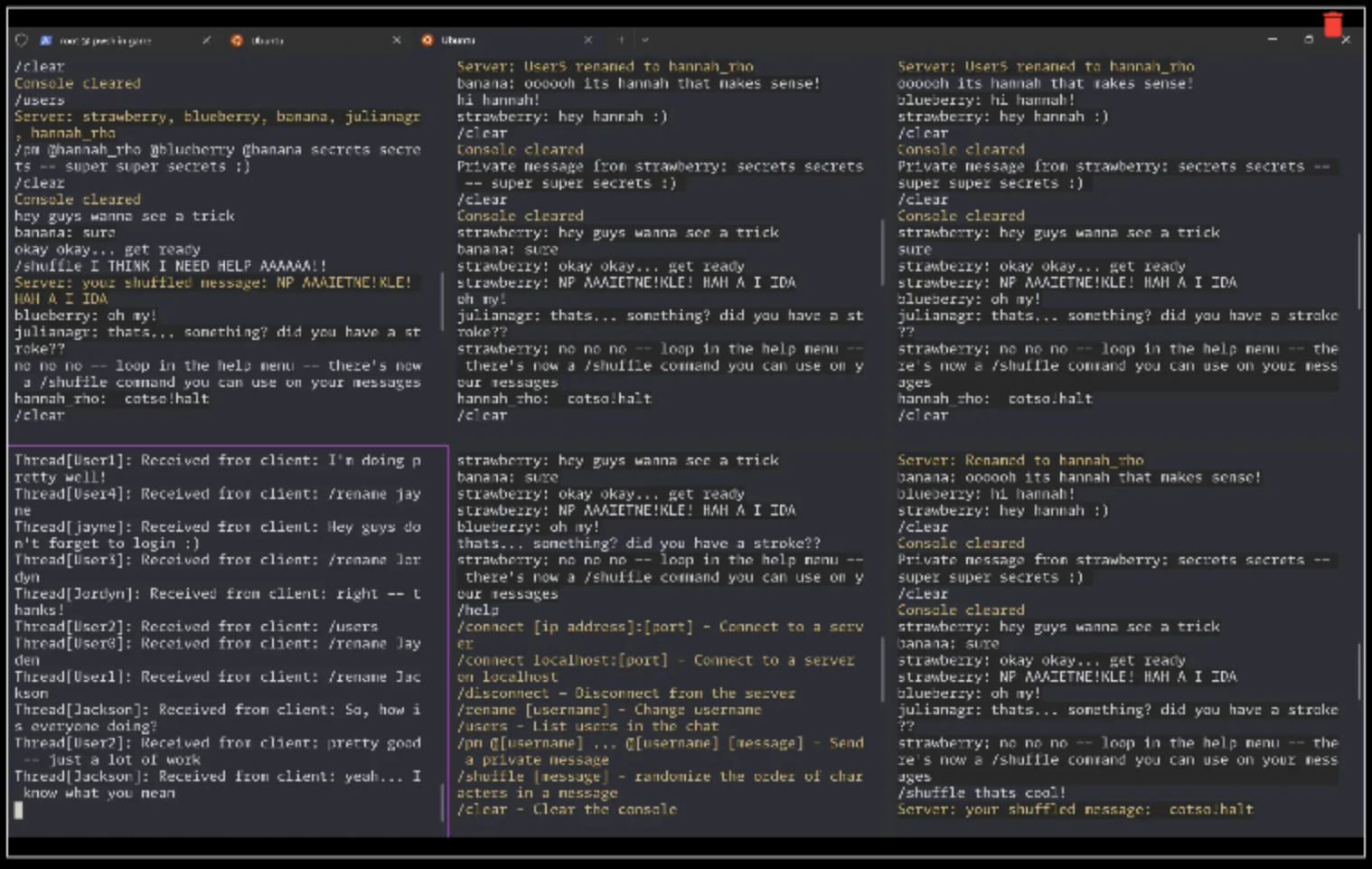
Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Output is clearly shown and captioned
☐ #2	1	Code shows relevant snippets that accomplish feature, UCID and date are present in all code screenshots. Relevant captions are included for each screenshot of the code.

Task Screenshots:

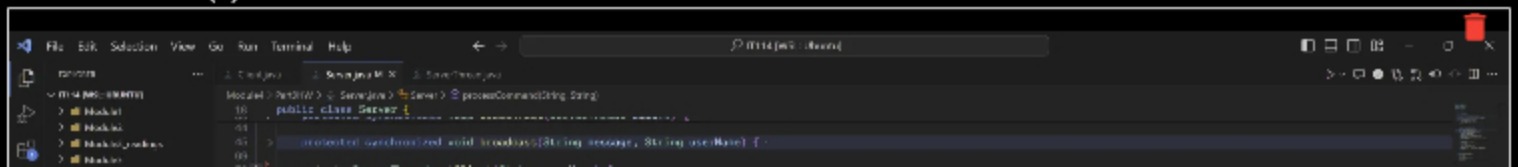
Gallery Style: Large View

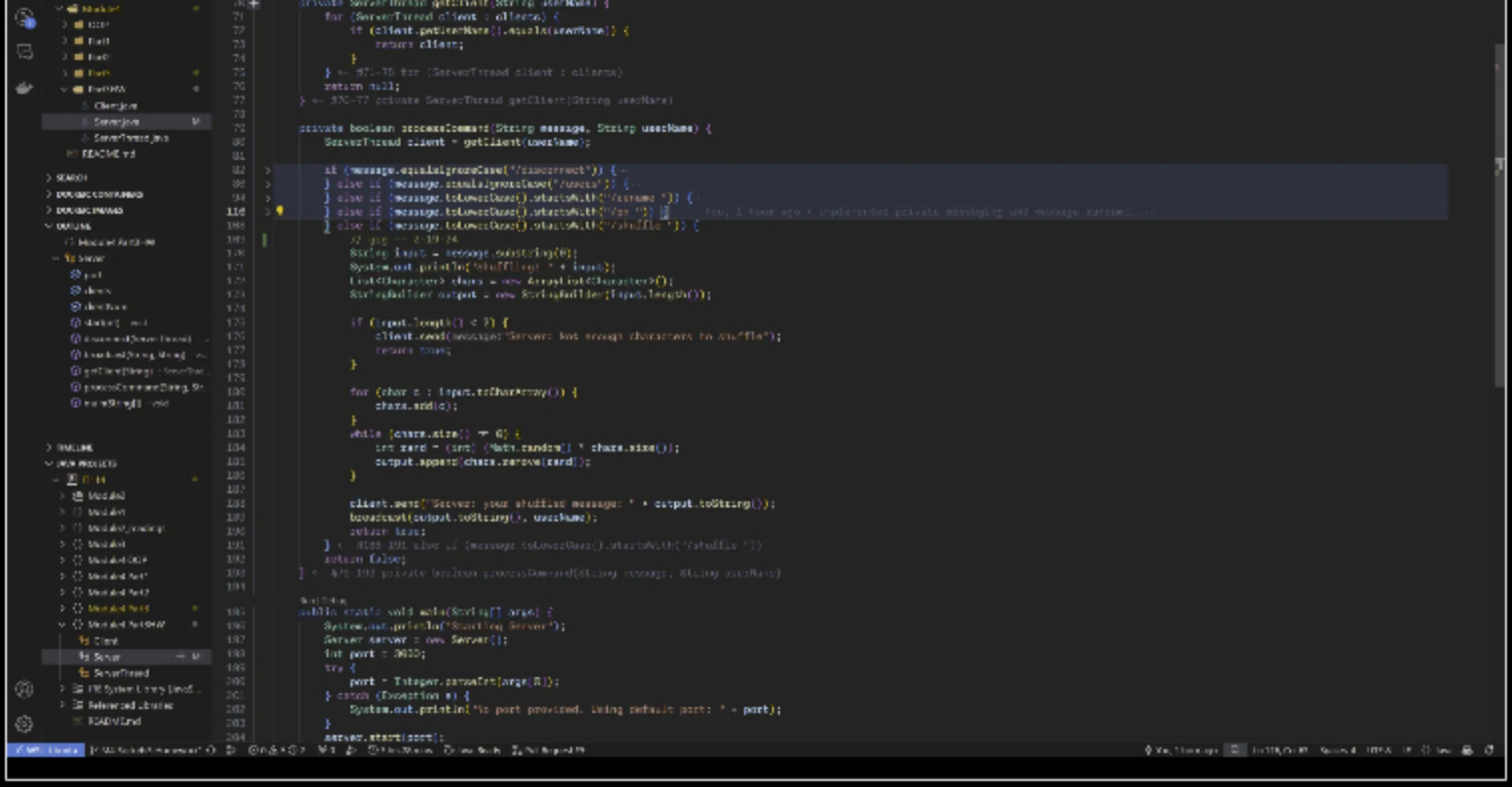
Small Medium Large



Shows multiple users using the /shuffle command to randomize the order of characters in the strings of the messages they send

Checklist Items (0)





implementation of /shuffle

Checklist Items (0)

Misc (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Reflection: Did you have an issues and how did you resolve them? If no issues, what did you learn during this assignment that you found interesting?

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	An issue or learning is clearly stated
☐ #2	1	Response is a few reasonable sentences

Response:

I didn't really encounter any error or issues -- everything implemented was rather trivial and while there may have been a couple things that didn't work immediately, I never had more than a simple type or bounds error that was just overlooking a line and was solved relatively immediately.

COLLAPSE

Task #2 - Points: 1

Text: Pull request link

Details:
URL should end with /pull/# and be related to this assignment

URL #1
https://github.com/GarrettGR/IT114/pull/5

End of Assignment