Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-451-M2024/it114-module-4-sockets-part-1-3/grade/sa2796

IT114-451-M2024 - [IT114] Module 4 Sockets Part 1-3

Submissions:

Submission Selection

1 Submission [active] 6/10/2024 10:45:07 AM

Instructions

COLLAPSE

Overview Video: https://youtu.be/5a5HL0n6jek

1. Create a new branch for this assignment
2. If you haven't, go through the socket lessons and get each part implemented (parts 1-3)
    1. You'll probably want to put them into their own separate folders/packages (i.e., Part1, Part2, Part3) These are for your reference
3. Part 3, below, is what's necessary for this HW
    3. https://github.com/MattToegel/IT114/tree/M24-Sockets-Part3
4. Create a new folder called Part3HW (copy of Part3)
5. Make sure you have all the necessary files from Part3 copied here and fix the package references at the top of each file
    1. Add/commit/push the branch
    2. Create a pull request to main and keep it open
6. 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)
    1. Simple number guesser where all clients can attempt to guess while the game is active
        1. Have a /start command that activates the game allowing guesses to be interpreted
        2. Have a /stop command that deactivates the game, guesses will be treated as regular messages (i.e., guess messages are ignored)
        3. Have a /guess command that include a value that is processed to see if it matches the hidden number (i.e., */guess 5*)
            1. Guess should only be considered when the game is active
            2. 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)
            3. No need to implement complexities like strikes
    2. Coin toss command (random heads or tails)

    1. Command should be something logical like /flip or /toss or /coin or similar
    2. The result should mention *who* did *what* and got what *result* (i.e., Bob Flipped a coin and got heads)
3. Dice roller given a command and text format of "/roll #d#" (i.e., /roll 2d6)
    1. Command should be in the format of /roll #d# (i.e., /roll 1d10)
    2. The result should mention *who* did *what* and got what *result* (i.e., Bob rolled 1d10 and got 7)
4. Math game (server outputs a basic equation, first person to guess it correctly gets congratulated and a new equation is given)
    1. Have a /start command that activates the game allowing equaiton to be answered
    2. 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)
    3. Have an answer command that include a value that is processed to see if it matches the hidden number (i.e., */answer 15*)
        1. 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)
5. Private message (a client can send a message targetting another client where only the two can see the messages)
    1. Command can be /pm, /dm followed by the user's name or an @ preceding the users name (clearly note which)
    2. 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)
    3. 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
6. Message shuffler (randomizes the order of the characters of the given message)
    1. Command should be /shuffle or /randomize (clearly mention what you chose) followed by the message to shuffle (i.e., /shuffle hello everybody)
    2. The message should be sent to all clients showing it's from the user but randomized
        1. Example: Bob types */command* hello and everyone recevies Bob: lleho
7. Fill in the below deliverables
8. Save the submission and generated output PDF
9. Add the PDF to the Part3HW folder (local)
10. Add/commit/push your changes
11. Merge the pull request
12. Upload the same PDF to Canvas

Branch name: M4-Sockets3-Homework

Tasks: 6 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

#1) Show and clearly note which

Caption (required) ✓
Describe/highlight what's being shown
Showing the left most terminal in the server

#2) Show and clearly note which

Caption (required) ✓
Describe/highlight what's being shown
Showing the middle and right are the clients

#3) Show all clients receiving the

Caption (required) ✓
Describe/highlight what's being shown
Showing messages work

#4) Include a screenshot showing you

Caption (required) ✓
Describe/highlight what's being shown
Showing parts 1-3 and their files

Feature 1 (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Solution

#1) Show the code related to the feature (ucid and date must

#2) Show the feature working (i.e., all terminals and their

Caption (required) ✓
Describe/highlight what's being shown
showing solved feature 1

Explanation (required) ✓
Mention specific feature and explain sufficiently and concisely the implementation (should be aligned with code snippets)

PREVIEW RESPONSE

Feature 1 was the simple number guesser and I configured the server class so it could handle the start, guess, and stop commands. Essentially, when the game is started, guesses will be checked against a hidden number. Then, the ourpur will be shown to the clients that displays who and what they guessed alng with if it was correct or not. There is also astart and /guess command implemented. /start will start the game and a random number will be generated. /guess will process the guess and will show if it is corre

Caption (required) ✓
Describe/highlight what's being shown
displaying the output of the code in terminal

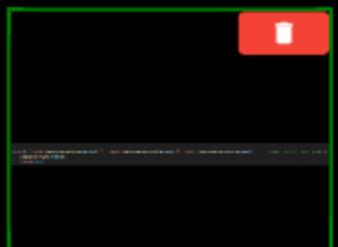
Feature 2 (3 pts.)

COLLAPSE

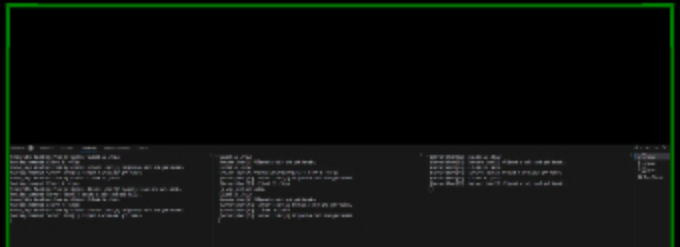
COLLAPSE

Task #1 - Points: 1
Text: Solution

#1) Show the code related to the feature (ucid and date must be present as a comment)



#2) Show the feature working (i.e., all terminals and their related output)



Caption (required) ✓
Describe/highlight what's being shown
showing solved feature 2

Explanation (required) ✓
Mention specific feature and explain sufficiently and concisely the implementation (should be aligned with code snippets)s
PREVIEW RESPONSE
The process command methos will see if the response begins with "/flip" and if it does it will call the flip coin method which essentially generates a heads/tails and displays it to the clients. It then returns true to show that the command had processed.

Caption (required) ✓
Describe/highlight what's being shown
displaying the output of the code in terminal

Misc (2 pts.)
COLLAPSE

COLLAPSE

Task #1 - Points: 1
Text: Reflection

#1) Learn anything new? Face any challenges? How did

Explanation (required) ✓
Provide at least a few logical sentences
PREVIEW RESPONSE
I have learned how to implement server side logic by doing the activities, simple number guesser, and coin toss. I did have an issue with java.net.bindexception when running the code but was able to fix it by tweaking a section of my code which had been causing the error.

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/SHUAIB2796/sa2796-IT114-451/pull/8

COLLAPSE

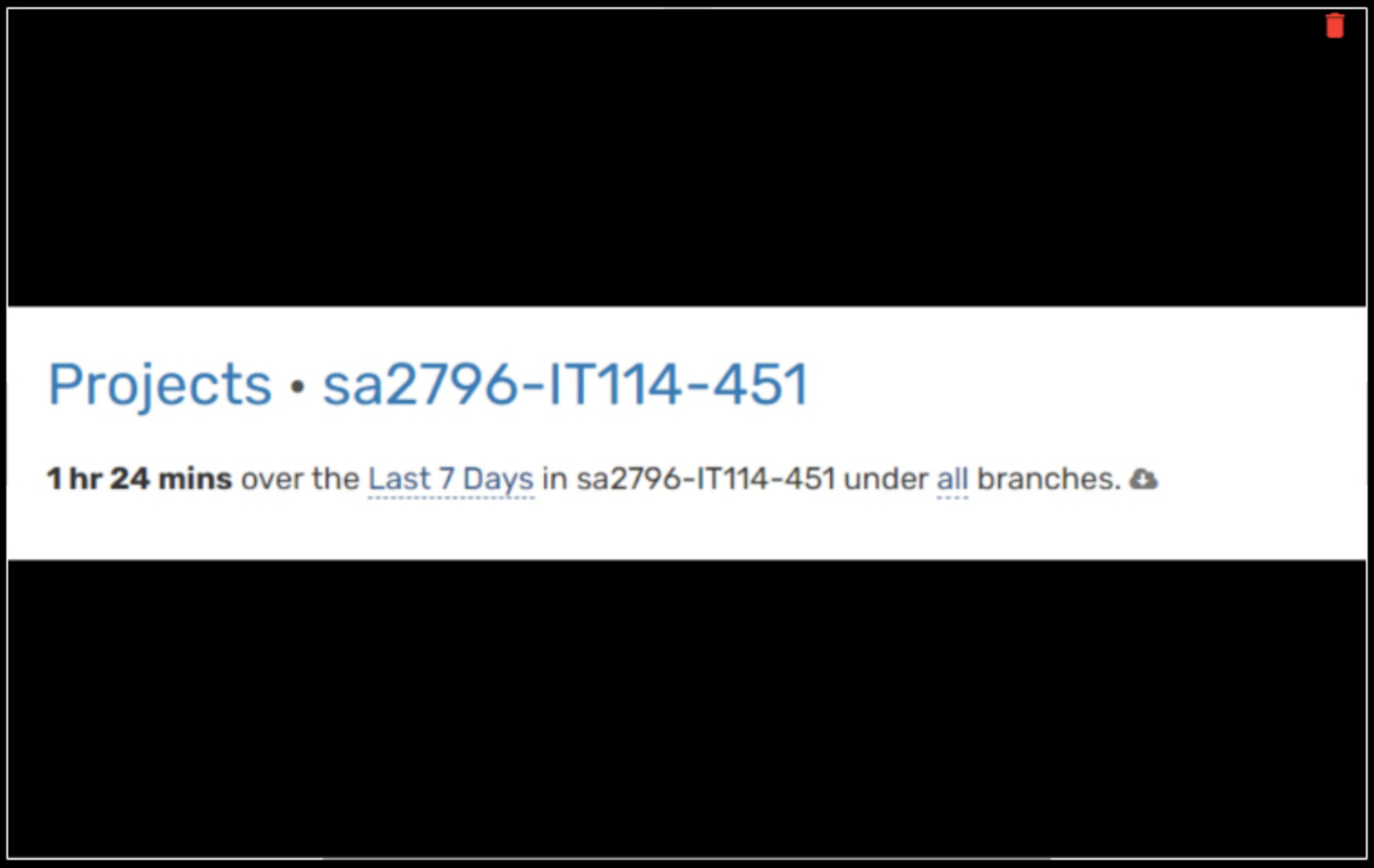
Task #3 - Points: 1
Text: Waka Time (or related) Screenshot

Details:
Screenshot clearly shows what files/project were being worked on (the duration of time doesn't correlated with the grade for this item)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



overview of wakatime

[redacted]

Files

1 hr 17 mins	Server.java
3 mins	Client.java
2 mins	ServerThread.java

Branches

1 hr 24 mins	M4-Sockets3-Homework

[redacted]

For this assignment's task time

End of Assignment