Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-module-5-project-milestone-1/grade/cae6

Course: IT114-003-F2024
Assigment: [IT114] Module 5 Project Milestone 1
Student: Chizorom E. (cae6)

Submissions:

Submission Selection

1 Submission [submitted] 10/21/2024 7:29:12 PM

Instructions

COLLAPSE

Overview Video: https://youtu.be/A2yDMS9TS1o

1. Create a new branch called Milestone1
2. At the root of your repository create a folder called Project if one doesn't exist yet
    1. You will be updating this folder with new code as you do milestones
    2. You won't be creating separate folders for milestones; milestones are just branches
3. Copy in the code from Sockets Part 5 into the Project folder (just the files)
    2. https://github.com/MattToegel/IT114/tree/M24-Sockets-Part5
4. Fix the package references at the top of each file (these are the only edits you should do at this point)
5. Git add/commit the baseline and push it to github
6. Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
7. Ensure the sample is working and fill in the below deliverables 1. Note: Don't forget the client commands are /name and /connect
8. Generate the output file once done and add it to your local repository
9. Git add/commit/push all changes
10. Complete the pull request merge from the step in the beginning
11. Locally checkout main
12. git pull origin main

Branch name: Milestone1

Group

100%

Group: Start Up
Tasks: 2
Points: 3

COLLAPSE

Task

100%

Group: Start Up
Task #1: Start Up
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:

Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 1

Sub-Task

100%

Group: Start Up
Task #1: Start Up
Sub Task #1: Show the Server starting via command line and listening for connections

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Server starting via command line

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

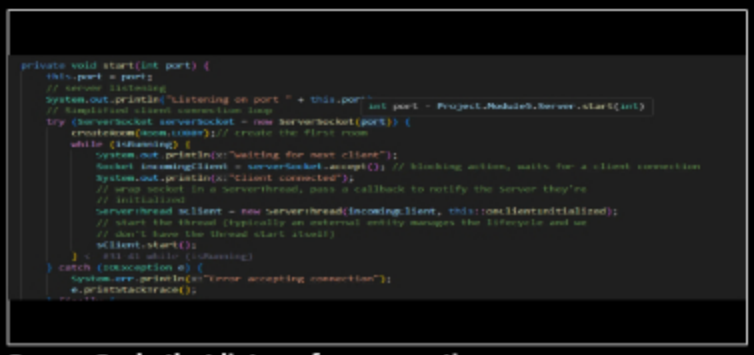
100%

Group: Start Up
Task #1: Start Up
Sub Task #2: Show the Server Code that listens for connections

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Server Code that listens for connections

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code related to starting up and waiting for connections

Response:

In this code, a new server socket is created and the socket starts on port 3000 and it listens and waits for a client connection. The blocking action blocks the server from executing anything until a client tries to connect to the server. Once they c

Sub-Task

100%

Group: Start Up

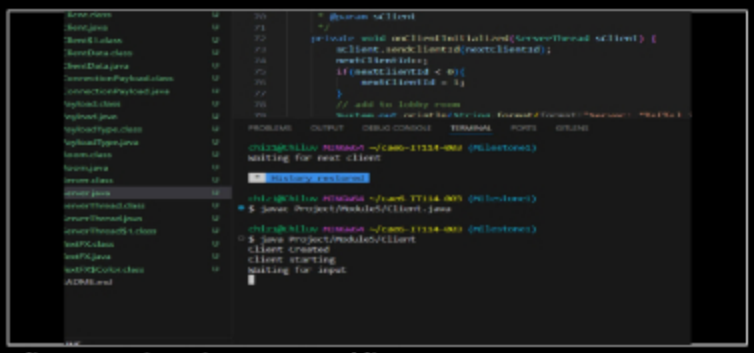
Task #1: Start Up

Sub Task #3: Show the Client starting via command line

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Client starting via command line

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Start Up

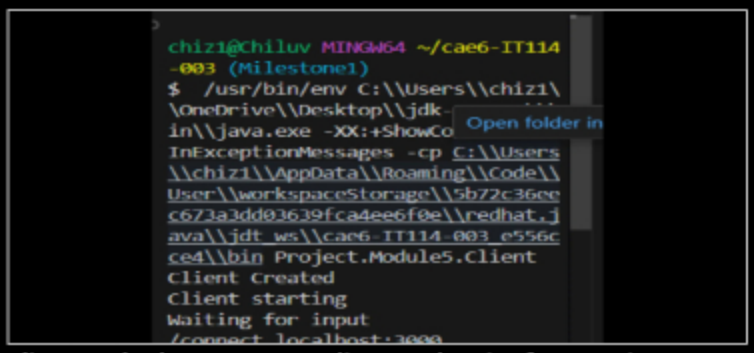
Task #1: Start Up

Sub Task #4: Show the Client Code that prepares the client and waits for user input

Task Screenshots

Gallery Style: 2 Columns

4 2 1



client code that prepares client and waits for user input

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow leading up to and including waiting for user input

Response:

First, the client is initialized and then the client is started up. Then the listen to input method is run and this allows the client to listen for any user input. In this method, a scanner is also used which allows input to be read from the user until the client is no longer running. Then the method processes the client's command if the client is connected and sends a message to the server and it sends an input as a message if the command isn't processed well.

End of Task 1

Task

100%

Group: Start Up
Task #2: Connecting
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:

Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 1

Sub-Task

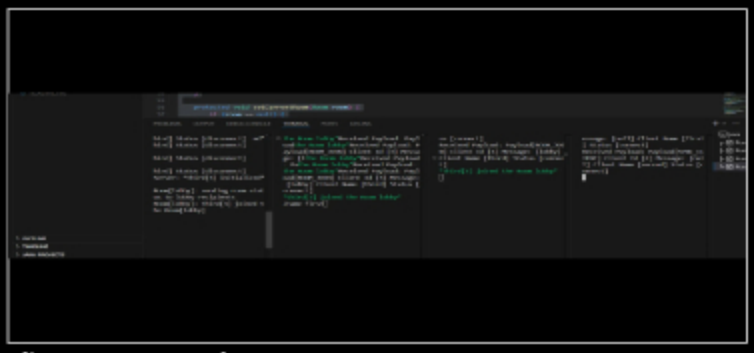
100%

Group: Start Up
Task #2: Connecting
Sub Task #1: Show 3 Clients connecting to the Server

Task Screenshots

Gallery Style: 2 Columns

4 2 1



clients connected

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Start Up

Task #2: Connecting

Sub Task #2: Show the code related to Clients connecting to the Server (including the two needed commands)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

connecting to the server

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow

Response:

The processClientCommand method processes the commands input by the user. First, the user will have to connect the local host on port 3000 and then once that is recieved and processed by the server side, it will send a message back saying that localhost is connected. The boolean connect method takes an IP address and a port to build a socket connection to the server. A new socket gets created and it will try to connect to the server. Object output and input stream will be used for sending and receiving messages over the socket.

End of Task 2

End of Group: Start Up
Task Status: 2/2

Group

100%

Group: Communication
Tasks: 2
Points: 3

COLLAPSE

Task

100%

Group: Communication
Task #1: Communication
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:

Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 1

Sub-Task

100%

Group: Communication
Task #1: Communication
Sub Task #1: Show each Client sending and receiving messages

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Client sending and recieving messages

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: Communication
Task #1: Communication

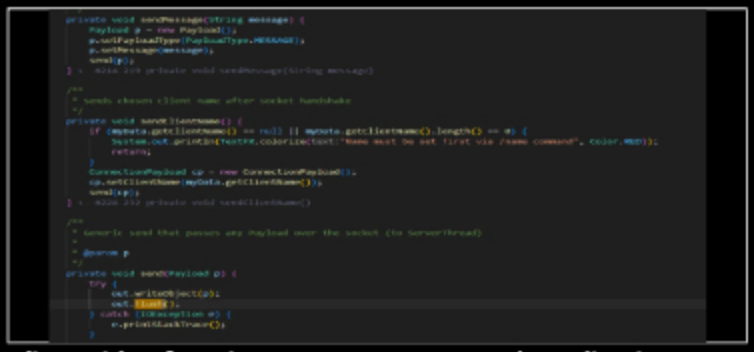
100%

Sub Task #2: Show the code related to the Client-side of getting a user message and sending it over the socket

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Client Side of getting a user message and sending it over the socket

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

The send message method uses a scanner that reads user input and then asks the user to enter their message. The input will create a new payload object that will set the message and then it will get sent over the socket with "out.writeObject(payload);" and "out.flush()".

Sub-Task

100%

Group: Communication

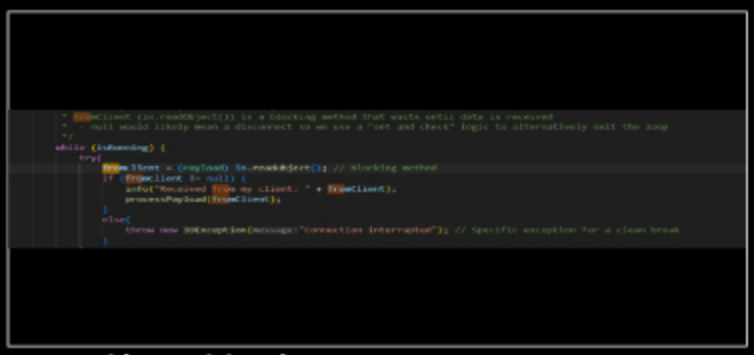
Task #1: Communication

Sub Task #3: Show the code related to the Server-side receiving the message and relaying it to each connected Client

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Server side receiving the message

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

This, fromClient = (Payload) in.readObject(); is what reads the message received and once it's received it goes to the payload and then that payload is passed to the processpayload method. This method will be responsible for processing the message and the message is sent in the current room.

Sub-Task

100%

Group: Communication
Task #1: Communication
Sub Task #4: Show the code related to the Client receiving messages from the Server-side and presenting them

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Client receiving messages from the server-side and presenting them

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

This Payload fromServer = (Payload) in.readObject();, is what waits for a message from the server. if that message is received it calls on processPayload to handle the message. This method processes the message and then sends the message to the other clients.

End of Task 1

Task

100%

Group: Communication
Task #2: Rooms
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:

Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 1

Sub-Task

100%

Group: Communication

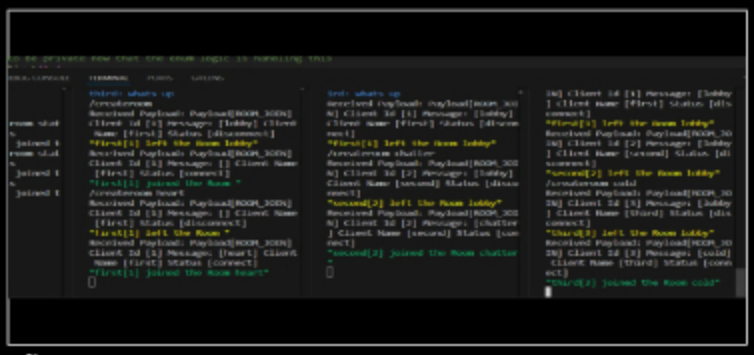
Task #2: Rooms

Sub Task #1: Show Clients can Create Rooms

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Client can create rooms

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Communication

Task #2: Rooms

Sub Task #2: Show Clients can Join Rooms (leave/join messages should be visible)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Clients can join rooms

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Communication
Task #2: Rooms
Sub Task #3: Show the Client code related to the create/join room commands

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Client code related to create/join

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

The processClientCommand method checks the command input and if the command is "createroom" it calls the SendCreateRoom method which creates a new payload object, which is sent to the server. This also applies to the "joinroom" command but with the send join room method instead.

Sub-Task

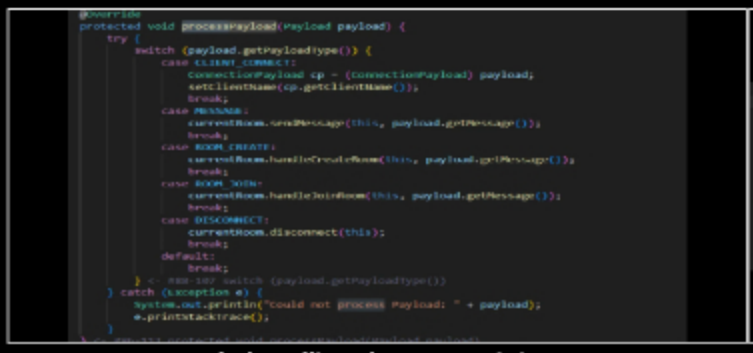
100%

Group: Communication
Task #2: Rooms
Sub Task #4: Show the ServerThread/Room code handling the create/join process

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Server/Room code handling the create/join process

Server/Room code handling the create/join process

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

Whenever a client wants to join or create a room a payload is sent to the server. In the Server code the processPayload method is what handles the payloads where handleCreateRoom and handleJoinRoom are called, allowing the user to create or join a room. In room, the handleCreateRoom verifies that the room can be made and if it can it adds the sender to the new room, the same applies to handleJoinRoom in Room.

Sub-Task

100%

Group: Communication

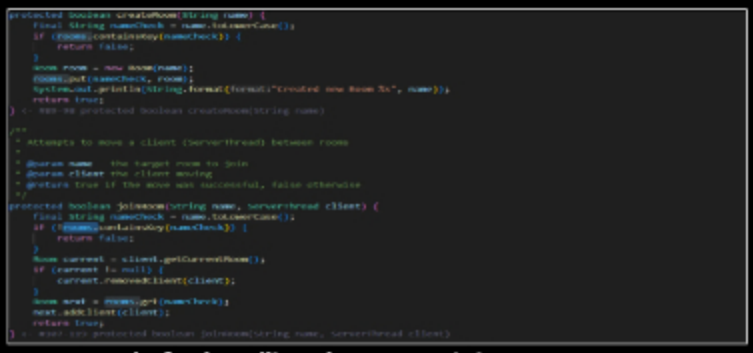
Task #2: Rooms

Sub Task #5: Show the Server code for handling the create/join process

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Server code for handling the create/join process

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

When the user creates a room, the room is created and if the room does not already exist, there should be no issues in the creation of the room. In order to join a room, the method rooms.get(namecheck) looks for an existing room and if the room does not exist, it will return false. If the room exists it will show that the client has joined the room and its true.

Sub-Task

100%

Group: Communication

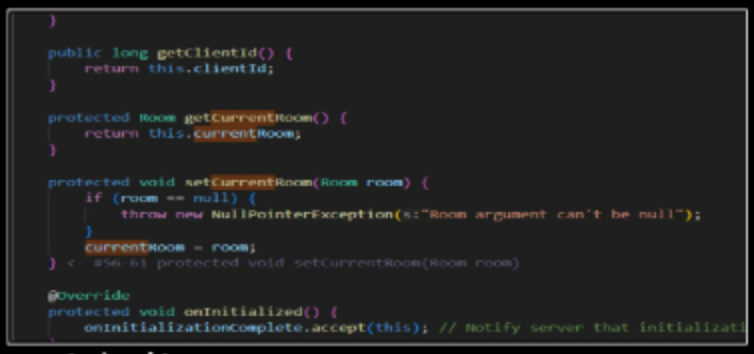
Task #2: Rooms

Sub Task #6: Show that Client messages are constrained to the Room (clients in different Rooms can't talk to each other)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



constrained to room

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain why/how it works this way

Response:

In the ServerThread class a client is tied to a specific room because of currentRoom. The messages in the payload are sent to the current room and to the current users, and during that time only.

End of Task 2

End of Group: Communication
Task Status: 2/2

Group

100%

Group: Disconnecting/Termination
Tasks: 1
Points: 3

COLLAPSE

Task

100%

Group: Disconnecting/Termination
Task #1: Disconnecting
Weight: ~100%
Points: ~3.00

COLLAPSE

Details:

Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 1

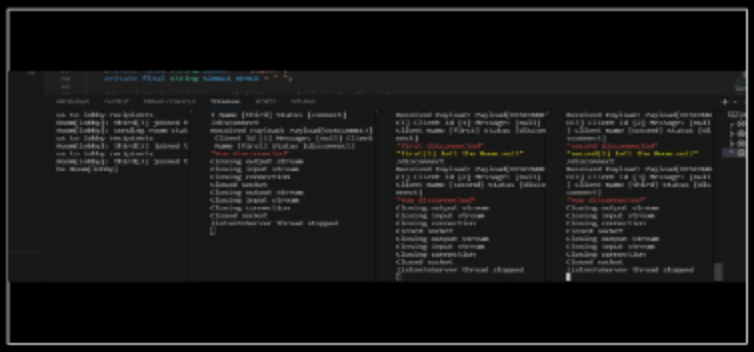
Sub-Task 100%

Group: Disconnecting/Termination
Task #1: Disconnecting
Sub Task #1: Show Clients gracefully disconnecting (should not crash Server or other Clients)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Clients gracefully disconnecting

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

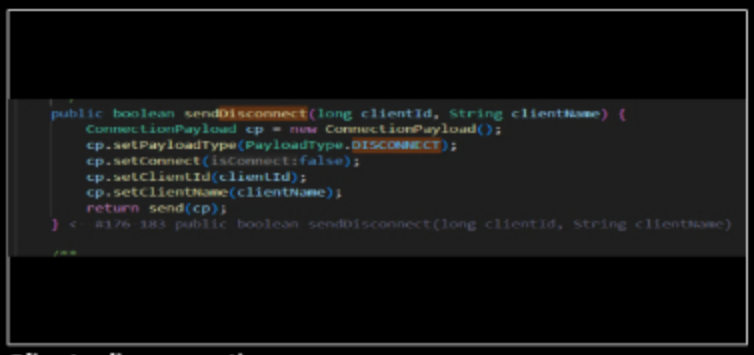
Sub-Task 100%

Group: Disconnecting/Termination
Task #1: Disconnecting
Sub Task #2: Show the code related to Clients disconnecting

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Clients disconnecting

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

The disconnect method in the server thread checks if the room is running , then the client. disconnect method is called to handle the disconnect. Once the client is removed from the room a message is printed stating that the

client has been disconnected. This message gets sent to all remaining clients in the room.

Sub-Task

100%

Group: Disconnecting/Termination
Task #1: Disconnecting
Sub Task #3: Show the Server terminating (Clients should be disconnected but still running)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Server Terminating

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

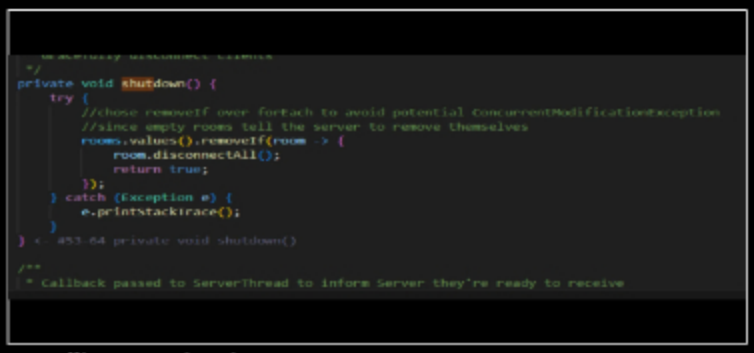
100%

Group: Disconnecting/Termination
Task #1: Disconnecting
Sub Task #4: Show the Server code related to handling termination

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
 */
private void shutdown() {
    try {
        //chose removeIf over forEach to avoid potential ConcurrentModificationException
        //since empty rooms tell the server to remove themselves
        rooms.values().removeIf(room -> {
            room.disconnectAll();
            return true;
        });
    } catch (Exception e) {
        e.printStackTrace();
    }
} <  453:64  private void shutdown()

/**
 * Callback passed to ServerThread to inform Server they're ready to receive
```

Handling termination

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

In order to terminate the server the shutdown method is invoked. This method goes through all the rooms and for

In order to terminate the server the shutdown method is invoked. This method goes through all the rooms and for every room, the room. disconnected method ensuring that all clients in the room have been disconnected. If all clients have been disconnected, then the the removeIf() becomes true and then the room gets taken out from the server.

End of Task 1

End of Group: Disconnecting/Termination
Task Status: 1/1

Group

Group: Misc
Tasks: 3
Points: 1

COLLAPSE

Task

Group: Misc
Task #1: Add the pull request link for this branch
Weight: ~33%
Points: ~0.33

COLLAPSE

Task URLs

URL #1
Missing URL

URL

End of Task 1

Task

Group: Misc
Task #2: Talk about any issues or learnings during this assignment
Weight: ~33%
Points: ~0.33

COLLAPSE

Details:

Few related sentences about the Project/sockets topics

Task Response Prompt

Response:

I didn't have any issues in particular while completing this assignment

End of Task 2

Task

100%

Group: Misc
Task #3: WakaTime Screenshot
Weight: ~33%
Points: ~0.33

COLLAPSE

Details:
Grab a snippet showing the approximate time involved that clearly shows your repository.

The duration isn't considered for grading, but there should be some time involved.

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Wakatime screenshot

End of Task 3

End of Group: Misc
Task Status: 2/3

End of Assignment