

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-chatroom-milestone-3-2024/grade/oha2>

IT114-006-S2024 - [IT114] Chatroom Milestone 3 2024

Submissions:

Submission Selection

1 Submission [active] 4/30/2024 11:11:56 PM


Instructions

^ COLLAPSE ^

Implement the Milestone 3 features from the project's proposal document: <https://docs.google.com/document/d/1ONmvEvel97GTFPGfVwwQC96xSsobbSbk56145X/>
Make sure you add your ucid/date as code comments where code changes are done
All code changes should reach the Milestone3 branch
Create a pull request from Milestone3 to main and keep it open until you get the output PDF from this assignment.
Gather the evidence of feature completion based on the below tasks.
Once finished, get the output PDF and copy/move it to your repository folder on your local machine.
Run the necessary git add, commit, and push steps to move it to GitHub
Complete the pull request that was opened earlier
Upload the same output PDF to Canvas

Branch name: Milestone3

Tasks: 14 Points: 10.00

 Basic UI (2 pts.)

^ COLLAPSE ^

 Task #1 - Points: 1

Text: Screenshots of the following

Checklist			*The checkboxes are for your own tracking
#	Points	Details	

#1	1	Connection Panel
#2	1	User Details Panel
#3	1	Chat Panel
#4	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

Client - bub

Rooms

bub connected

zap connected

[Room]: You got heads!

[Room]: You got heads!

[Room]: You got tails!

[Room]: Rolled a 5-sided die, result: 4

bub (2)

d (1)

zap (3)

Send

chat panel

Checklist Items (0)

Client

Rooms

Host:

127.0.0.1

Port:

3000

Next

connection panel

Checklist Items (0)

Client

Rooms

Username:

Previous Connect

username panel

Checklist Items (0)



Formatting (2 pts.)

^COLLAPSE ^



Task #1 - Points: 1

Text: Screenshots demoing flip and roll commands

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Flip output in a different format than normal messages

<input checked="" type="checkbox"/> #2	1	Roll # output in a different format than normal messages
<input checked="" type="checkbox"/> #3	1	Roll #d# output in a different format than normal messages
<input checked="" type="checkbox"/> #4	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

SmallMediumLarge



The screenshot shows a chat application window titled "Client - bub". It has a "Rooms" list on the right side containing "bub (2)", "d (1)", and "zap (3)". The main chat area displays the following messages: "*bub connected*", "*zap connected*", "[Room]: You got heads!" (in blue), "[Room]: You got heads!" (in blue), "[Room]: You got tails!" (in orange), and "[Room]: Rolled a 5-sided die, result: 4" (in blue). At the bottom, there is a text input field and a "Send" button.

flip and roll commands with different formatting

Checklist Items (0)

☒

^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshots demoing custom text formatting

Checklist			*The checkboxes are for your own tracking
#	Points	Details	
<input checked="" type="checkbox"/> #1	1	Custom text formatting for bold working (Part of the message should appear bold)	
<input type="checkbox"/> #2	1	Custom text formatting for italic working (Part of the message should appear italic)	
<input type="checkbox"/> #3	1	Custom text formatting for underline working (Part of the message should appear underline)	

<input type="checkbox"/> #4	1	Custom text formatting for red working (Part of the message should appear red)
<input type="checkbox"/> #5	1	Custom text formatting for blue working (Part of the message should appear blue)
<input type="checkbox"/> #6	1	Custom text formatting for green working (Part of the message should appear green)
<input type="checkbox"/> #7	1	Custom text formatting for combined bold, italic, underline, and a color working (Part of the message should have all 4 formats applied at once)
<input type="checkbox"/> #8	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

SmallMediumLarge

Client - b

Rooms

b connected

c connected

a: *muted c*

c: why :(

b: **bold**

b: underline

b: *italics*

b: red

b: #all the things#R and nothing is the same text

b: oop messed up

b: all the things and nothing is the same text

b: green blue

b (2)

a (1)

c (3)

Send

just all of the formatting in 1 screenshot

Checklist Items (0)

Task #3 - Points: 1

Text: Screenshot of the code solving the formatting display

Checklist			*The checkboxes are for your own tracking
#	Points	Details	
<input type="checkbox"/> #1	1	Show each relevant file this was done in (may be one or more)	

<input type="checkbox"/> #2	1	Include uid and date comment
<input type="checkbox"/> #3	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
//oha2 4/19
public void addText(String text) {
    JPanel content = chatArea;
    // add message
    JEditorPane textContainer = new JEditorPane("text/html", text);


    // sizes the panel to attempt to take up the width of the container
    // and expand in height based on word wrapping
    textContainer.setLayout(null);
    textContainer.setPreferredSize(
        new Dimension(content.getWidth(), ClientUtils.calcHeightForText(this, text, content.getWidth()))
    );
    textContainer.setMaximumSize(textContainer.getPreferredSize());
    textContainer.setEditable(false);
    ClientUtils.clearBackground(textContainer);
    // add to container and tell the layout to revalidate
    content.add(textContainer);
    // scroll down on new message
    JScrollBar vertical = ((JScrollPane) chatArea.getParent().getParent()).getVerticalScrollBar();
    vertical.setValue(vertical.getMaximum());
}
```

code for formatting being in html instead of plain text

Checklist Items (0)

Task #4 - Points: 1

Text: Explain how the formatting was made to be visible/rendered in the UI

 Details:

Note each scenario

Response:

The JPanel is showing everything formatted like and html page using html formatting do using html tags you can format things like colors and font and such

Task #1 - Points: 1

Text: Screenshots demoing private message

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Should have 3 clients in the same room
<input type="checkbox"/> #2	1	Demo a private message where only the sender and target see the message
<input type="checkbox"/> #3	1	Clearly caption screenshots

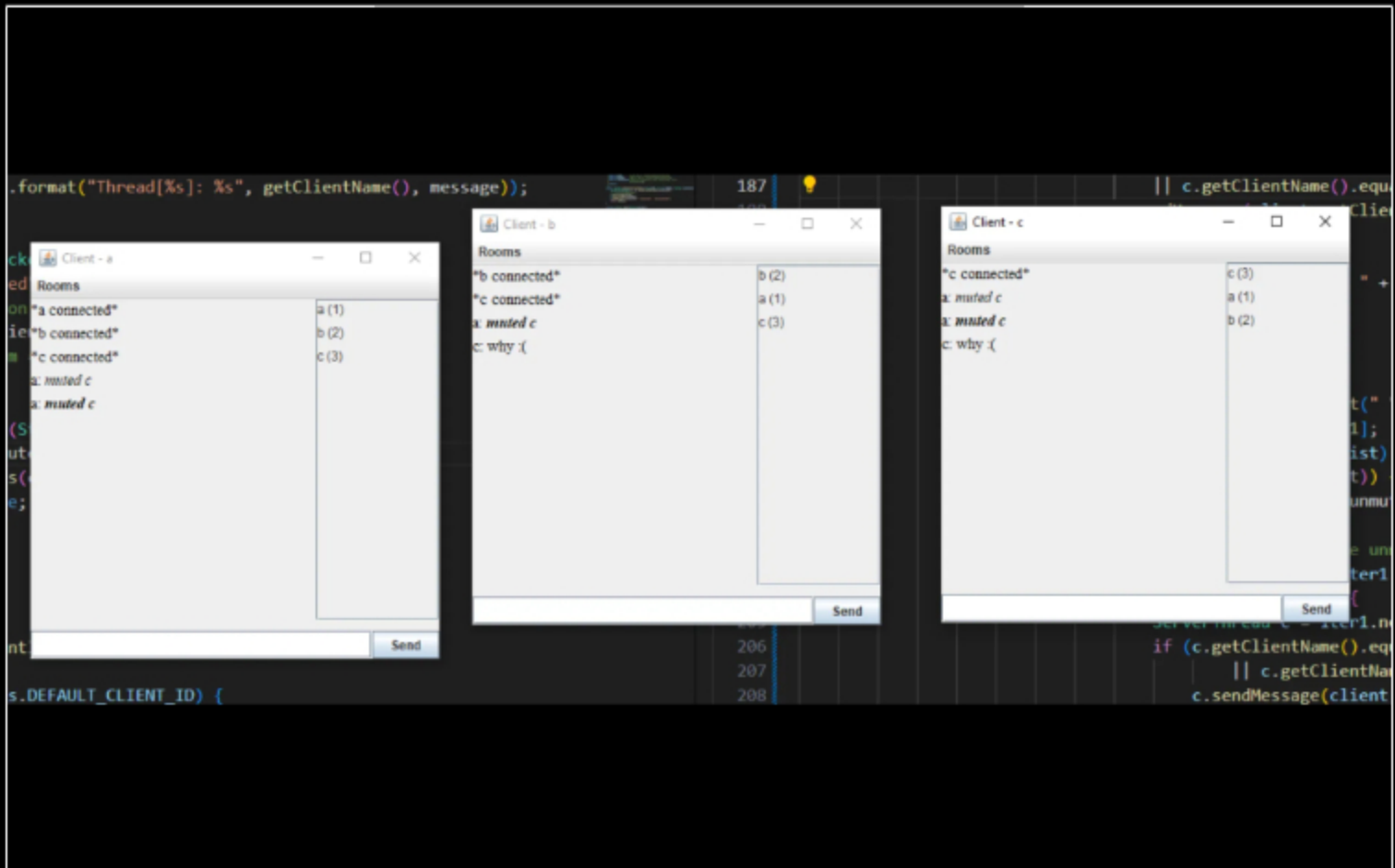
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



a muted c and can not see c's messages

Checklist Items (0)

Task #2 - Points: 1

Text: Screenshots of the related code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Show what code processes and handles the private message
<input type="checkbox"/> #2	1	The message should only be sent to the receiver and the target
<input type="checkbox"/> #3	1	The client should be targeting the username and the server side should be fetching the correct recipient
<input type="checkbox"/> #4	1	Include uid and date comment
<input type="checkbox"/> #5	1	Clearly caption screenshots

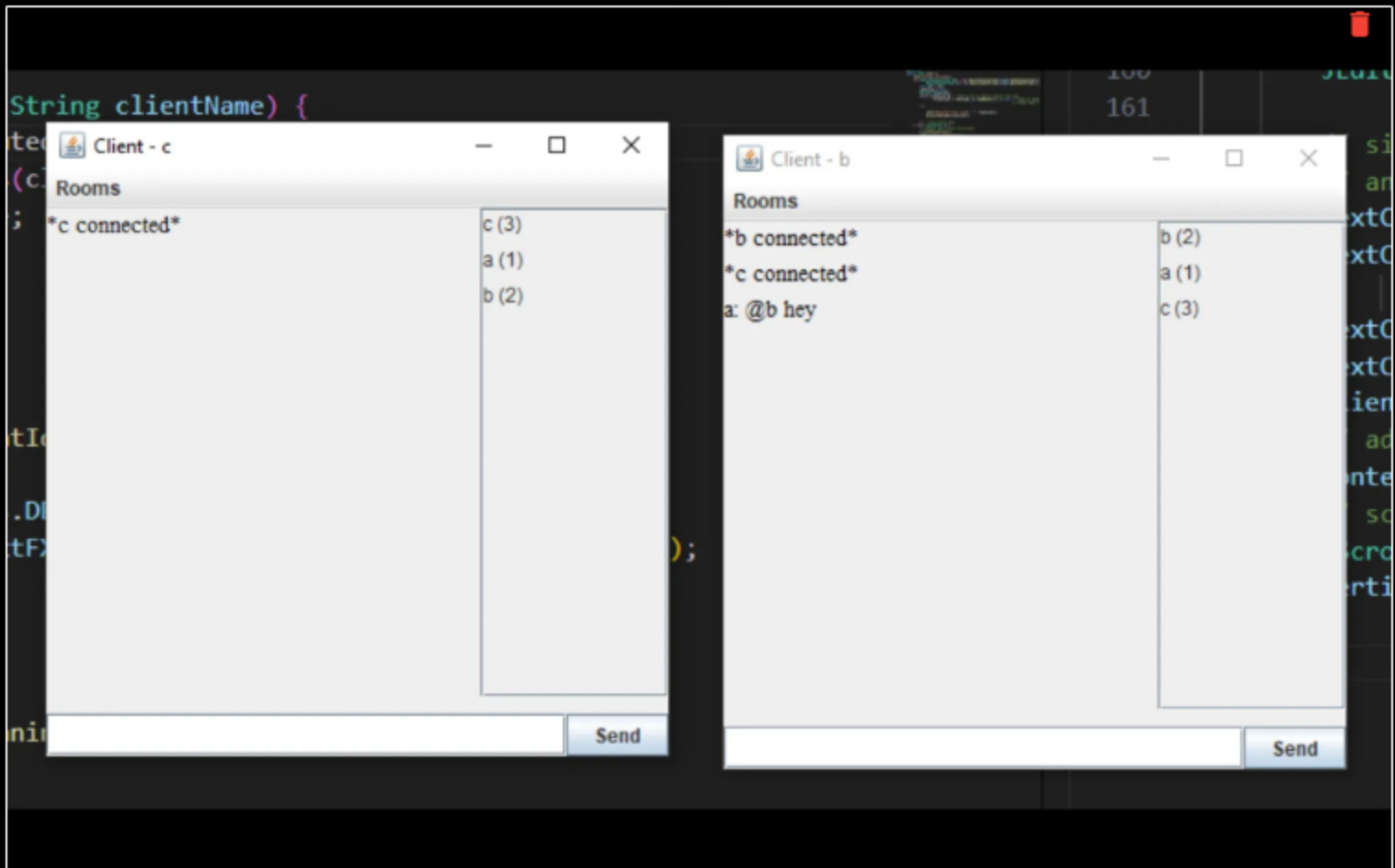
Task Screenshots:

Gallery Style: Large View

Small

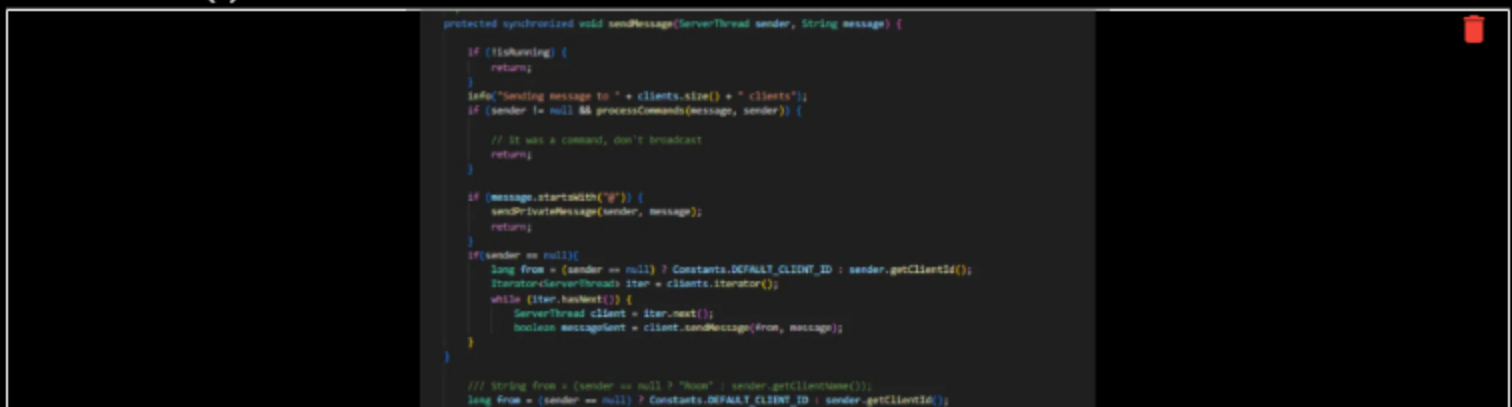
Medium

Large



only b getting the dm from and not c

Checklist Items (0)




```

        Iterator<ServerThread> iter = clients.iterator();
        while (iter.hasNext()) {
            ServerThread client = iter.next();
            if (!client.isMuted(sender.getClientName())) {
                boolean messageSent = client.sendMessage(sender.getClientId(), message);
                if (messageSent) {
                    handledDisconnect(iter, client);
                }
            }
        }
    }

    //P202 4/25
    protected synchronized void sendPrivateMessage(ServerThread sender, String message) {
        if (!isRunning) {
            return;
        }
        info("Sending message private");

        long from = (sender == null) ? Constants.DEFAULT_CLIENT_ID : sender.getClientId();
        Iterator<ServerThread> iter = clients.iterator();
        String recipient = null;
        String[] ws = message.split(" ");
        for (String w : ws) {
            if (w.startsWith("@")) {
                recipient = w.substring(1);

                while (iter.hasNext()) {
                    ServerThread c = iter.next();
                    if (c.getClientName().equals(recipient)) {
                        c.sendMessage(from, message);
                    }
                }
            }
        }
    }
}

```

private message code

Checklist Items (0)



^COLLAPSE ^

Task #3 - Points: 1

Text: Explain how private message works related to the code above

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Include how the sender and receiver are handled
<input type="checkbox"/> #2	1	Include how the username is used to get the proper id

Response:

if a message starts with an @ then sendMessage passes it to sendPrivateMessage. The message is turned into an array of strings and it gets the word after the @ and gets the client id of the user who has a clientname through the serverthread functions that matches it and only it them while still knowing who the original sender was and listing them as the sender.



Mute/Unmute Users (3 pts.)

^COLLAPSE ^



^COLLAPSE ^

Task #1 - Points: 1

Text: Screenshots demoing feature working

Checklist

*The checkboxes are for your own tracking

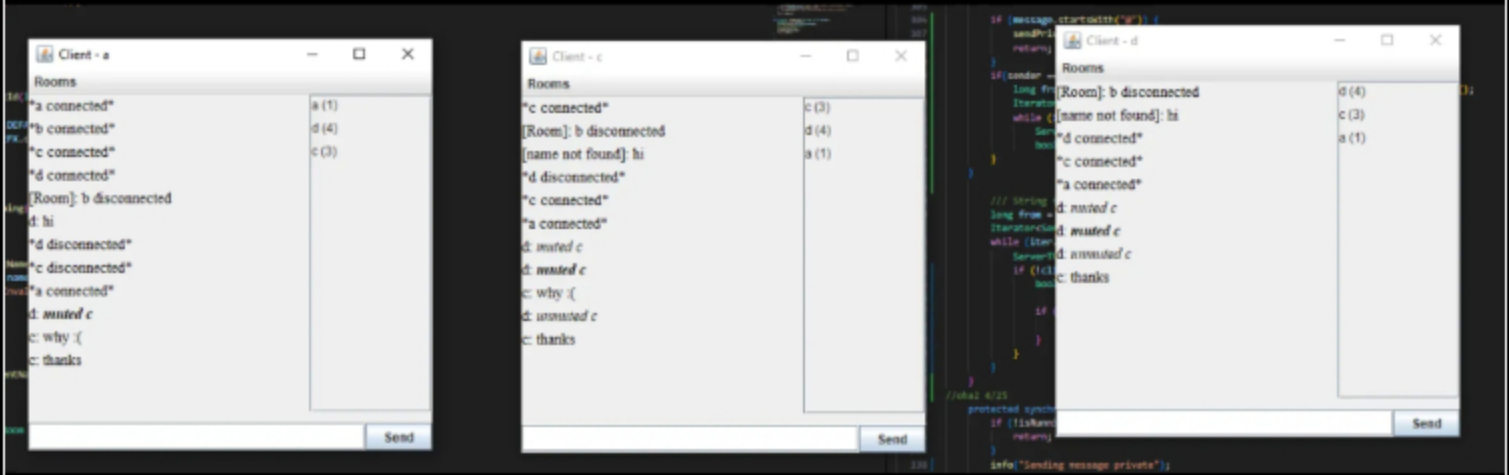
#	Points	Details
<input type="checkbox"/> #1	1	Should have 3 clients in the same room
<input type="checkbox"/> #2	1	Demo mute preventing messages between the muter and the target
<input type="checkbox"/> #3	1	Demo mute also being accounted for with private messages

<input type="checkbox"/> #4	1	Demo unmute allowing the messages again from the target to the unmutter
-----------------------------	---	---

Task Screenshots:

Gallery Style: Large View

SmallMediumLarge



c getting muted and unmuted by d

Checklist Items (0)

●

^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshots of the related code

Checklist			*The checkboxes are for your own tracking
#	Points	Details	
<input type="checkbox"/> #1	1	ServerThread should have a list of who they muted	
<input type="checkbox"/> #2	1	ServerThread should expose and add, remove, and is muted check to room	
<input type="checkbox"/> #3	1	Room should handle the mute list when receiving the appropriate payloads	
<input type="checkbox"/> #4	1	Room should check the mute list during send message and private messages	
<input type="checkbox"/> #5	1	Include ucid and date comment	
<input type="checkbox"/> #6	1	Clearly caption screenshots	

Task Screenshots:

Small Medium Large

```
Project > Project > Server > J ServerThread.java > ServerThread > isMuted(String)
23 public class ServerThread extends Thread {
24     private boolean isRunning = false;
27     private long clientId = Constants.DEFAULT_CLIENT_ID;
28     private ObjectOutputStream out; // exposed here for send()
29     // private Server server; // ref to our server so we can call methods on it
30     // more easily
31     private Room currentRoom;
32     private Logger logger = Logger.getLogger(ServerThread.class.getName());
33     public List<String> mutedList = new ArrayList<String>();
34
35     private void info(String message) {
36         logger.info(String.format("Thread[%s]: %s", getClientName(), message));
37     }
38
39     public ServerThread(Socket myClient /* , Room room */) {
40         info("Thread created");
41         // get communication channels to single client
42         this.client = myClient;
43         // this.currentRoom = room;
44     }
45
46     // oha2 4/25
47     public boolean isMuted(String clientName) {
48         for (String name : mutedList) {
49             if (name.equals(clientName)) {
50                 return true;
51             }
52         }
53         return false;
54     }
55 }
```

muted list in server thread

Checklist Items (0)

```
// oha2 4/25
case "mute":
    String[] splitMsg = message.split(" ");
    String mutedClient = splitMsg[1];
    client.mutedList.add(mutedClient);

    // sends a message to the muted user and the client that muted them
    Iterator<ServerThread> iter = clients.iterator();
    while (iter.hasNext()) {
        ServerThread c = iter.next();
        if (c.getClientName().equals(mutedClient)
            || c.getClientName().equals(client.getClientName())) {
            c.sendMessage(client.getClientId(), " <i>muted " + mutedClient + "</i>");
        }
    }
    sendMessage(client, " <i><b>muted " + mutedClient + "</b></i>");

    break;
case "unmute":
    String[] splitArr = message.split(" ");
    String unmutedClient = splitArr[1];
    for (String name : client.mutedList) {
        if (name.equals(unmutedClient)) {
            client.mutedList.remove(unmutedClient);
        }
    }

    // sends a message to the unmuted user and the client that unmuted them
    Iterator<ServerThread> iter1 = clients.iterator();
    while (iter1.hasNext()) {
        ServerThread c = iter1.next();
        if (c.getClientName().equals(unmutedClient)
            || c.getClientName().equals(client.getClientName())) {
            c.sendMessage(client.getClientId(), " <i>unmuted " + unmutedClient + "</i>");
        }
    }
    // sendMessage(client, " <i>unmuted " + unmutedClient + "</i>");

    break;
}
```

room adding and removing from the muted list

Checklist Items (0)

```
protected synchronized void sendMessage(ServerThread sender, String message) {  
  
    if (!isRunning) {  
        return;  
    }  
    info("Sending message to " + clients.size() + " clients");  
    if (sender != null && processCommands(message, sender)) {  
  
        // It was a command, don't broadcast  
        return;  
    }  
  
    if (message.startsWith("@")) {  
        sendPrivateMessage(sender, message);  
        return;  
    }  
    if (sender == null) {  
        long from = (sender == null) ? Constants.DEFAULT_CLIENT_ID : sender.getClientId();  
        Iterator<ServerThread> iter = clients.iterator();  
        while (iter.hasNext()) {  
            ServerThread client = iter.next();  
            boolean messageSent = client.sendMessage(from, message);  
        }  
    }  
  
    /// String from = (sender == null) ? "Room" : sender.getClientName();  
    long from = (sender == null) ? Constants.DEFAULT_CLIENT_ID : sender.getClientId();  
    Iterator<ServerThread> iter = clients.iterator();  
    while (iter.hasNext()) {  
        ServerThread client = iter.next();  
        if (!client.isMuted(sender.getClientName())) {  
            boolean messageSent = client.sendMessage(sender.getClientId(), message);  
  
            if (!messageSent) {  
                handleDisconnect(iter, client);  
            }  
        }  
    }  
}  
// oha2 4/25
```

sendmessage accounting for if someone is muted

Checklist Items (0)

```
// oha2 4/25  
protected synchronized void sendPrivateMessage(ServerThread sender, String message) {  
    if (!isRunning) {  
        return;  
    }  
    info("Sending message private");  
  
    long from = (sender == null) ? Constants.DEFAULT_CLIENT_ID : sender.getClientId();  
    Iterator<ServerThread> iter = clients.iterator();  
    String recipient = null;  
    String[] ws = message.split(" ");  
    if (client.isMutedsender.getClientName()){  
        return;  
    }  
    for (String w : ws) {  
        if (w.startsWith("@")) {  
            recipient = w.substring(1);  
  
            while (iter.hasNext()) {  
                ServerThread c = iter.next();  
                if (c.getClientName().equals(recipient)) {  
                    c.sendMessage(from, message);  
                }  
            }  
        }  
    }  
    // sender.sendMessage(sender, message);  
}
```

```
    // String from = (sender == null ? "Room" : sender.getClientName());  
}
```

send private message just returning and doing nothing if muted

Checklist Items (0)



^COLLAPSE ^

Task #3 - Points: 1

Text: Explain how the mute and unmute logic works in relation to the code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Explain how your mute list is handled
<input type="checkbox"/> #2	1	Explain how it's handled/processed in send message and private message

Response:

The muted list is saved in each person's server thread. When a message is sent the room checks if the person who sent it is not on your muted list and sends only if they are.

if it's a pm then it checks if they are on the muted list and then just does nothing.



Misc (1 pt.)

^COLLAPSE ^



^COLLAPSE ^

Task #1 - Points: 1

Text: Add the pull request link for the branch

Details:

Note: the link should end with /pull/#

URL #1

<https://github.com/Jomarrrrr/Oha2-it114-006/pull/9>



^COLLAPSE ^

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

I got really sick midway through the semester there are also just a lot of little things that don't work. Have not had time to ask for help because I'm swamped by everything.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

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:

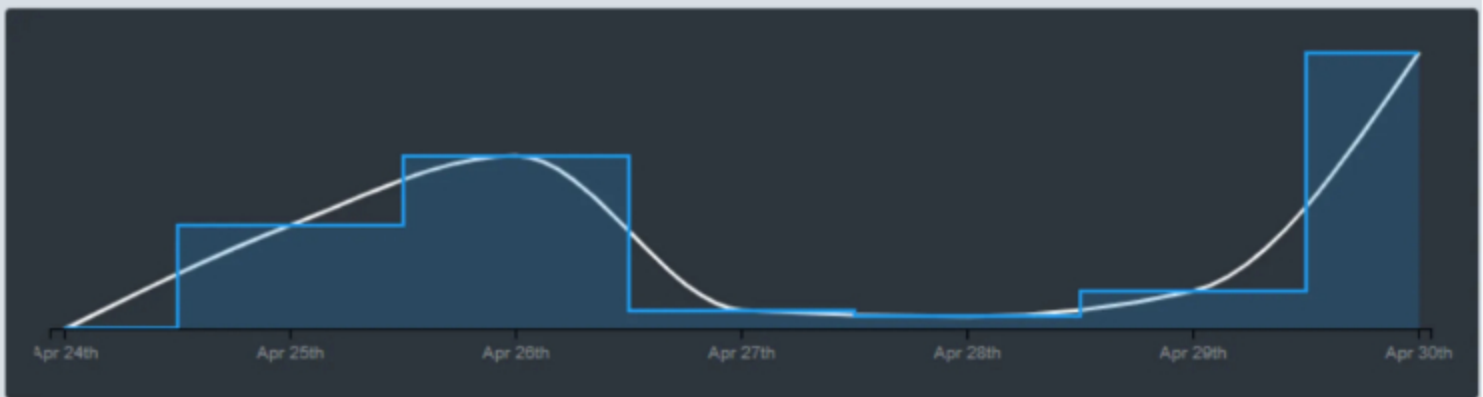
Gallery Style: Large View

Small

Medium

Large

24 hrs 1 min over the Last 7 Days. 



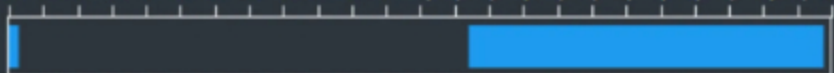
10 hrs 38 mins Today



Oha2-it114-006

10:38

12a 1a 2a 3a 4a 5a 6a 7a 8a 9a 10a 11a 12p 1p 2p 3p 4p 5p 6p 7p 8p 9p 10p 11p 12a



wakatime

End of Assignment