

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-chatroom-milestone-3-2024/grade/fj28>

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

Submissions:

Submission Selection

1 Submission [active] 4/15/2024 2:45:49 PM

Instructions

^ COLLAPSE ^

Implement the Milestone 3 features from the project's proposal document: <https://docs.google.com/document/d/1ONmvEvel97GTFPGfVwwQC96xSsobbSbk56145Xl/edit>
Make sure you add your uid/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

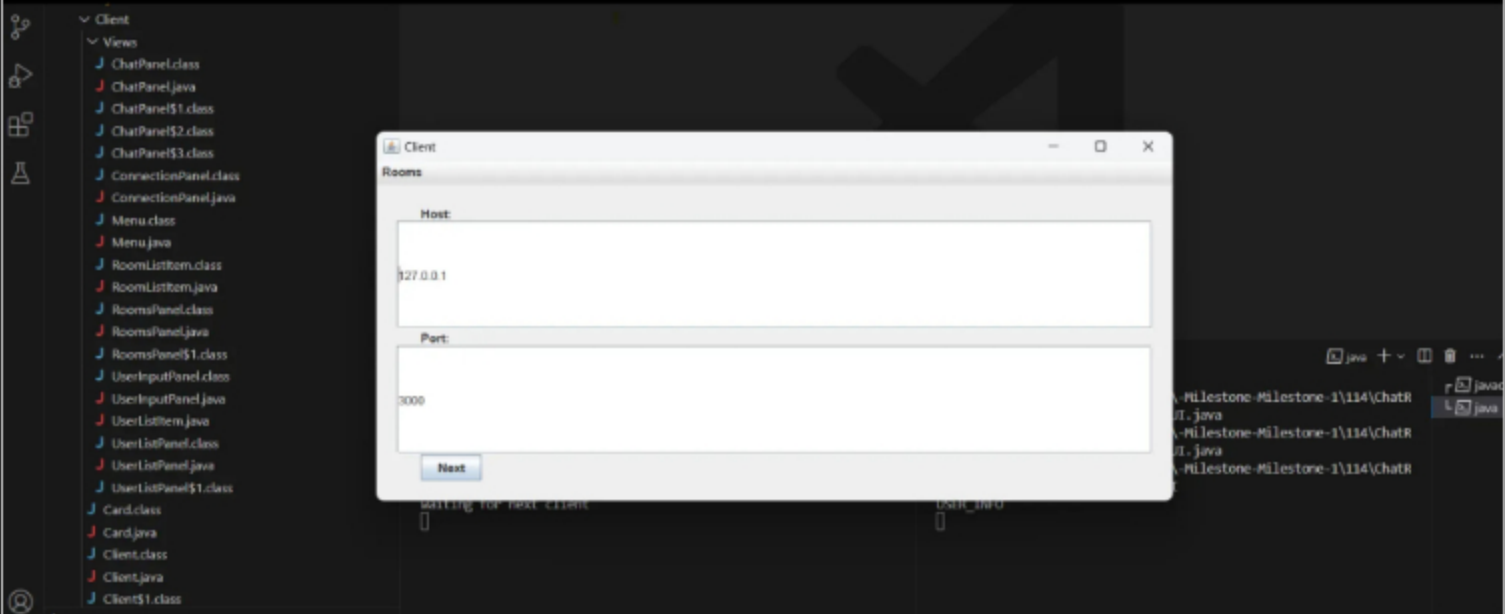
#	Points	Details

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

Task Screenshots:

Gallery Style: Large View

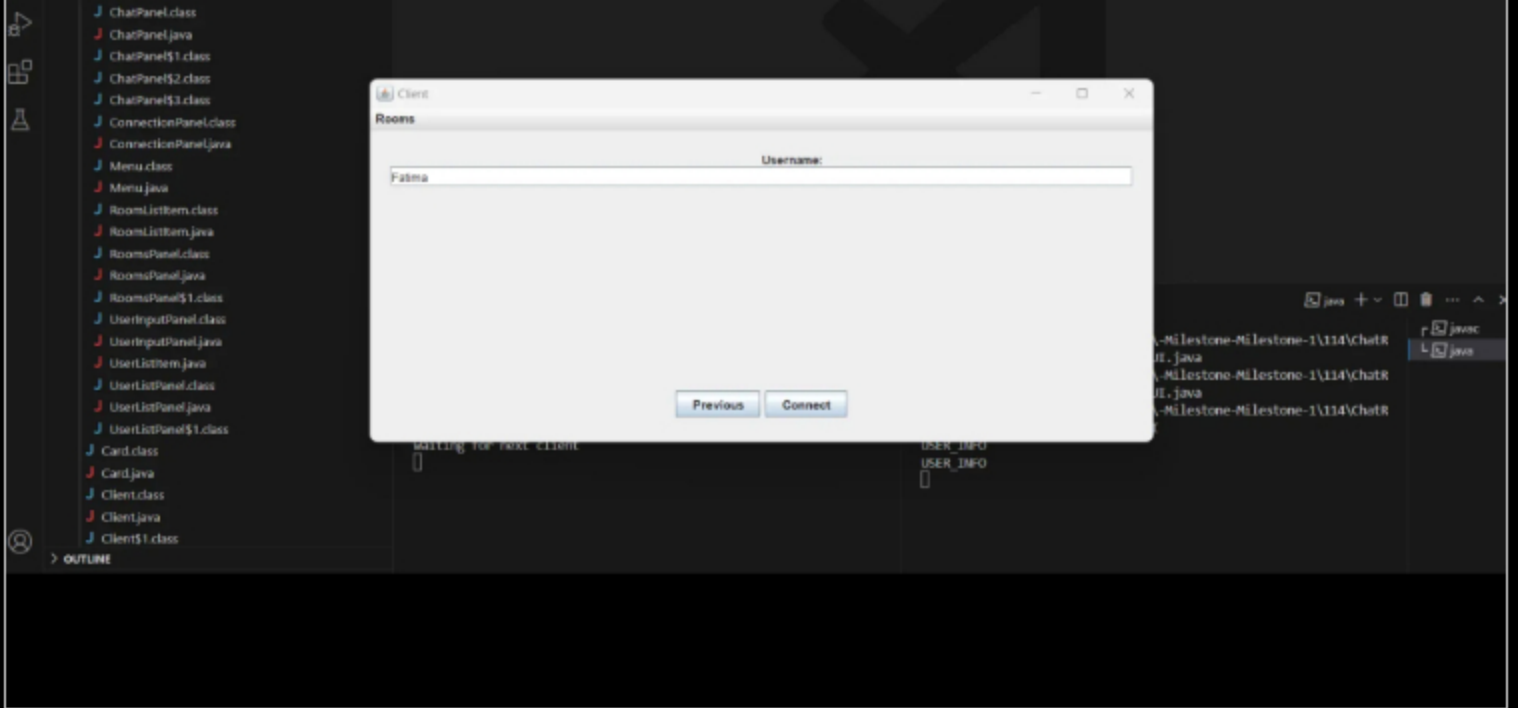
SmallMediumLarge



Connection Panel

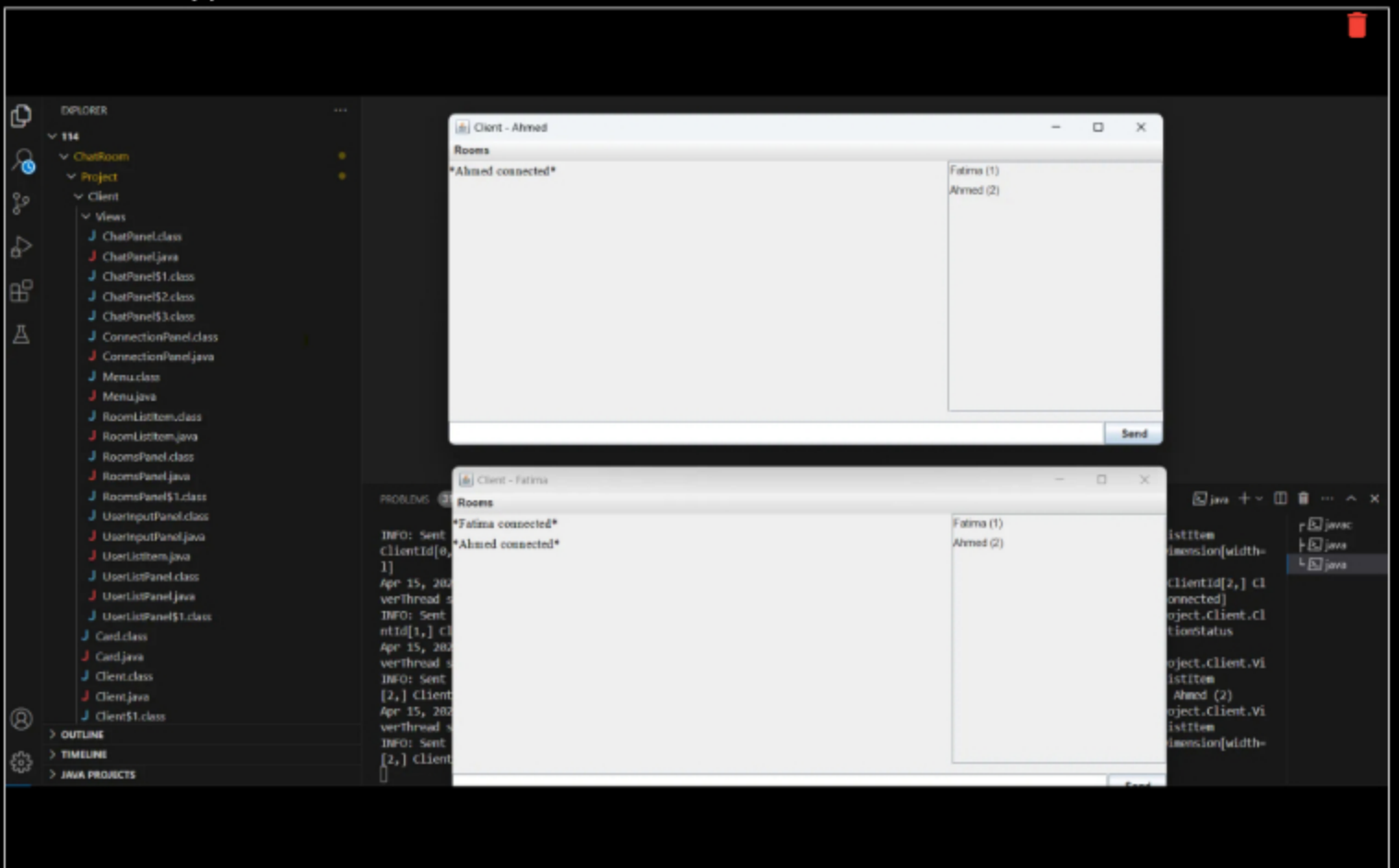
Checklist Items (4)

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



User Details Panel

Checklist Items (0)



Chat Panel

Checklist Items (0)

● Formatting (2 pts.)

⌵COLLAPSE⌶

^COLLAPSE ^

Task #1 - Points: 1

Text: Screenshots demoing flip and roll commands

Checklist

*The checkboxes are for your own tracking

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

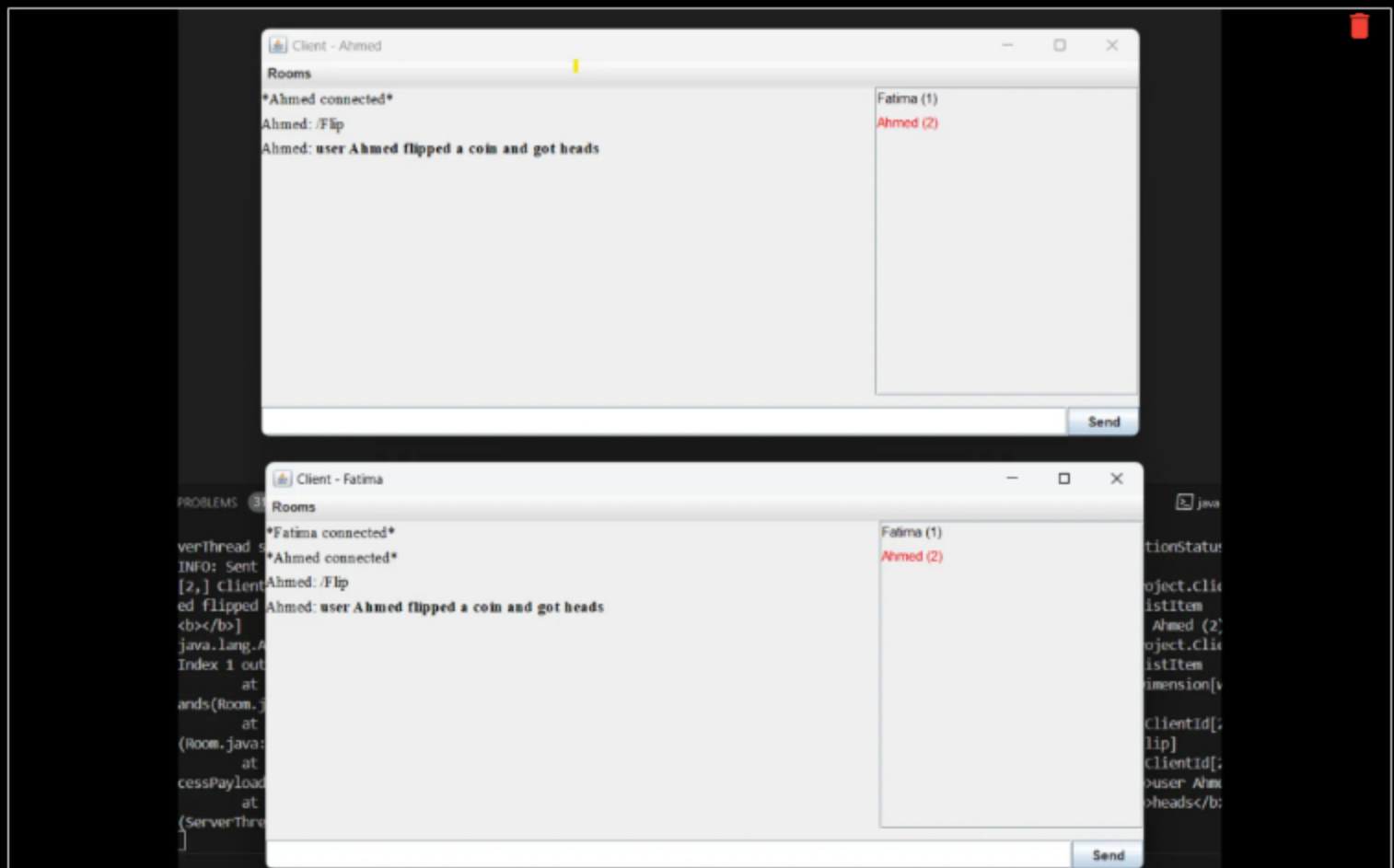
Task Screenshots:

Gallery Style: Large View

Small

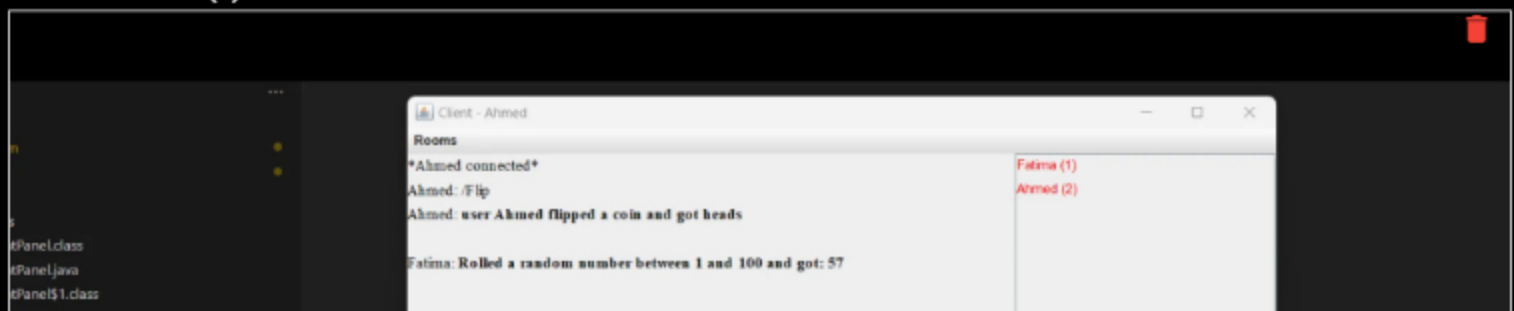
Medium

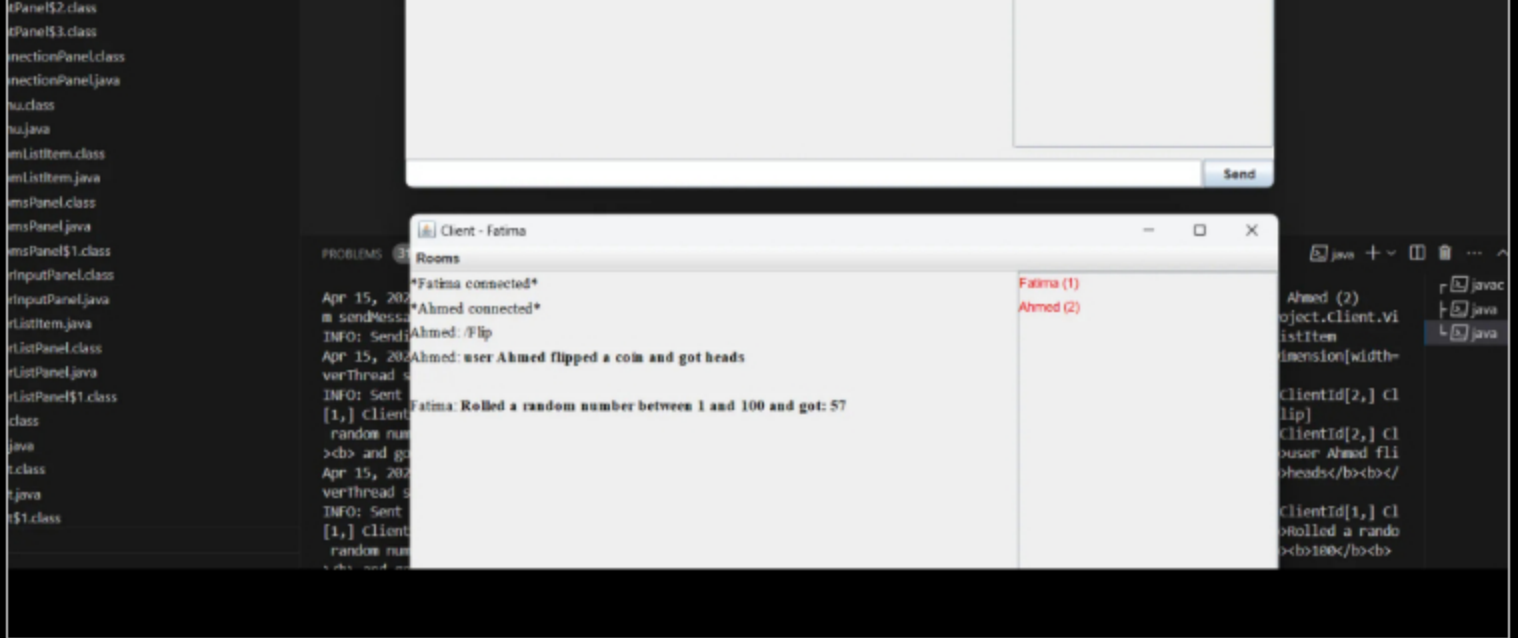
Large



Flip output in a different format than normal messages

Checklist Items (0)

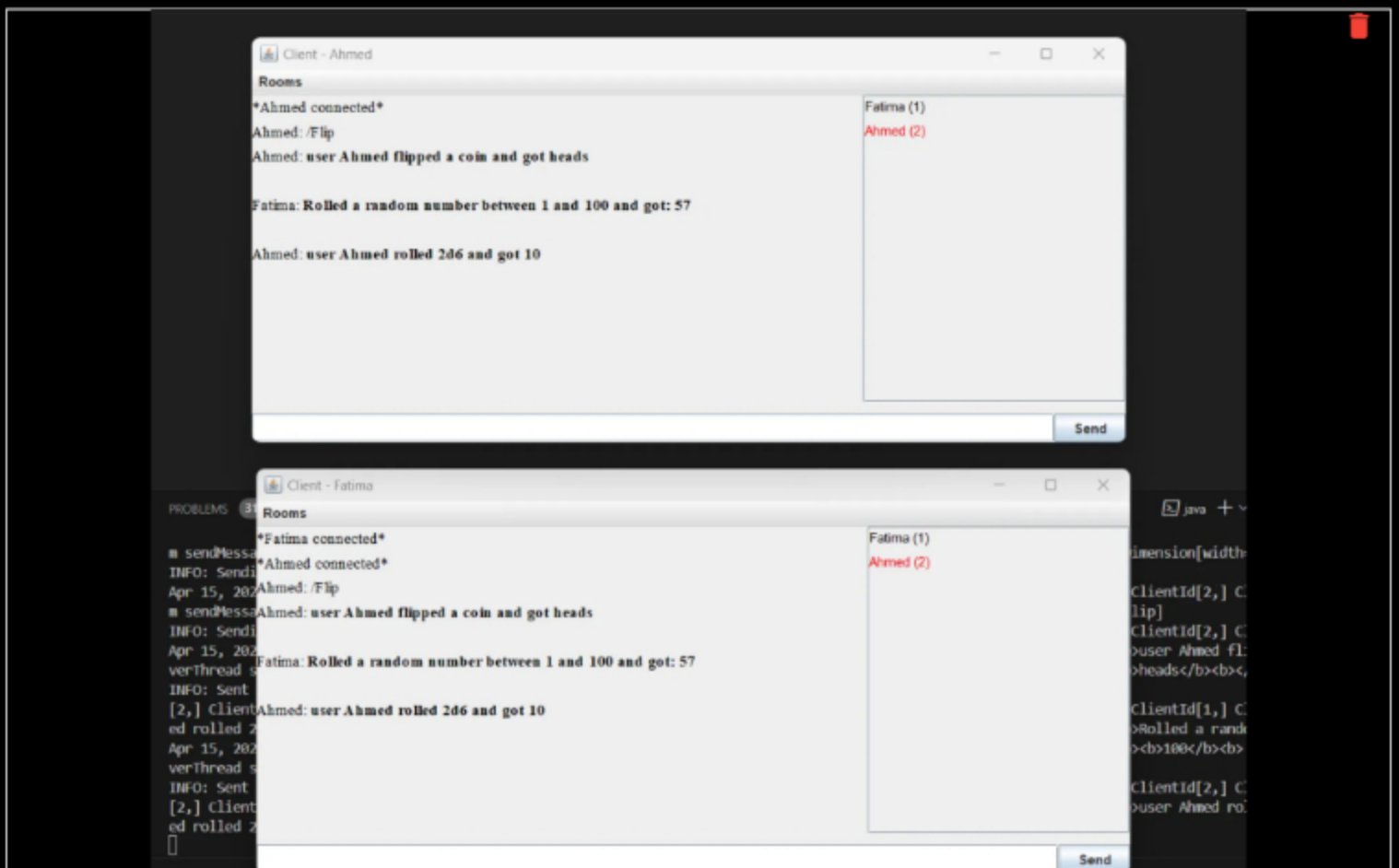




Roll # output in a different format than normal messages

Checklist Items (1)

#2 Roll # output in a different format than normal messages



Roll #d# output in a different format than normal messages

Checklist Items (0)

Task #2 - Points: 1

Text: Screenshots demoing custom text formatting

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input 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

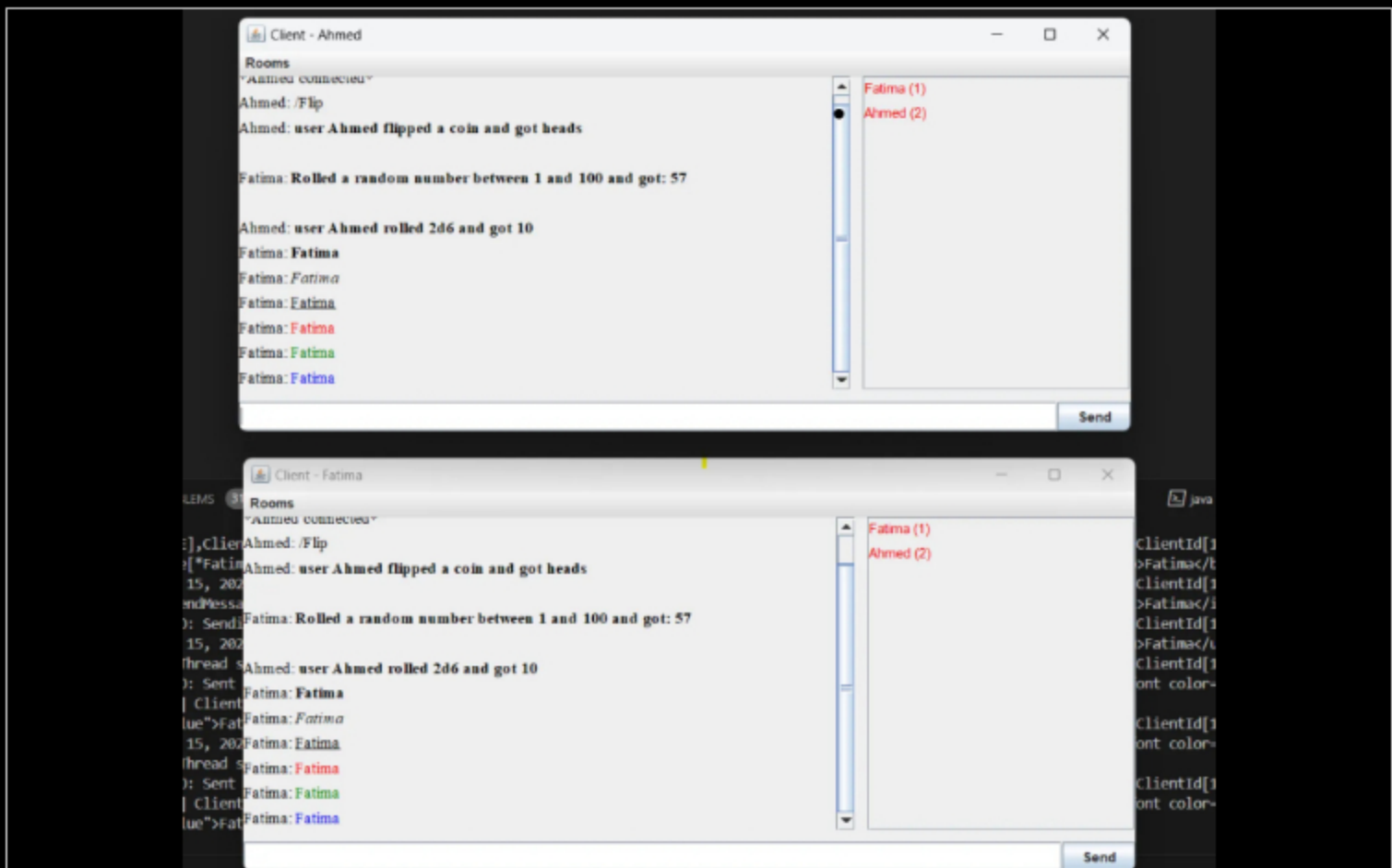
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



All items completed

Checklist Items (0)

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

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```
//FJ28
//Date: 4.1.204
if (message.contains(s:"~")){
    String[] tEffects = message.split(regex:"");
    message = "";
    int count = 0;
    int count2 = 0;
    int indexcount = 0;
    for (int i = 0; i < tEffects.length; i++){
        if (tEffects[i].equals(anObject:"~")){
            count++;
            count2++;
            if (count == 1){
                indexcount = i;
                tEffects[i] = "<b>";
            }
            if (count == 2){
                tEffects[i] = "</b>";
                count = 0;
            }
        }
    }
    if (count2%2 == 1){
        tEffects[indexcount] = "~";
    }
    for(String i: tEffects){
        message+= i;
    }
}
```

Screenshot of the code solving the formatting display

Checklist Items (0)

```
//FJ28
//Date: 4.1.204
if (message.contains(s:"!")){
    String[] tEffects = message.split(regex:"");
    message = "";
    int count = 0;
    int count2 = 0;
    int indexcount = 0;
    for (int i = 0; i < tEffects.length; i++){
        if (tEffects[i].equals(anObject:"!")){
            count++;
            count2++;
        }
    }
}
```



```

        if (count == 1){
            indexcount = i;
            tEffects[i] = "<i>";
        }
        if (count == 2){
            tEffects[i] = "</i>";
            count = 0;
        }
    }
}
if (count2%2 == 1){
    tEffects[indexcount] = "!";
}
for(String i: tEffects){
    message+= i;
}
}

```

Screenshot of the code solving the formatting display

Checklist Items (0)

```

111 //E208
112 //Date: 4.1.204
113
114 if (message.contains(c:"~")){
115     String[] tEffects = message.split(regex:"~");
116     message = "";
117     int count = 0;
118     int count2 = 0;
119     int indexcount = 0;
120     for (int i = 0; i < tEffects.length; i++){
121         if (tEffects[i].equals(mObject("~"))){
122             count++;
123             count2++;
124             if (count == 1){
125                 indexcount = i;
126                 tEffects[i] = "<u>";
127             }
128             if (count == 2){
129                 tEffects[i] = "</u>";
130                 count = 0;
131             }
132         }
133     }
134     if (count2%2 == 1){
135         tEffects[indexcount] = "~";
136     }
137     for(String i: tEffects){
138         message+= i;
139     }
140 }
141 //E209
142 //Date: 4.1.204
143
144 if (message.contains(c:"^")){
145     String[] tEffects = message.split(regex:"^");
146     message = "";
147     int count = 0;
148     int count2 = 0;
149     int indexcount = 0;
150     for (int i = 0; i < tEffects.length; i++){
151         if (tEffects[i].equals(mObject("^"))){
152             count++;
153             count2++;
154             if (count == 1){
155                 indexcount = i;
156                 tEffects[i] = "<font color='red'>";
157             }
158             if (count == 2){
159                 tEffects[i] = "</font>";
160                 count = 0;
161             }
162         }
163     }
164     if (count2%2 == 1){
165         tEffects[indexcount] = "^";
166     }
167     for(String i: tEffects){
168         message+= i;
169     }
170 }
171 }
172

```

Screenshot of the code solving the formatting display

Checklist Items (0)

```

71 //E208
72 //Date: 4.1.204
73
74 if (message.contains(c:"X")){
75     String[] tEffects = message.split(regex:"X");
76     message = "";
77     int count = 0;
78     int count2 = 0;
79     int indexcount = 0;
80     for (int i = 0; i < tEffects.length; i++){
81         if (tEffects[i].equals(mObject("X"))){
82             count++;
83             count2++;
84             if (count == 1){
85                 indexcount = i;
86                 tEffects[i] = "<font color='green'>";
87             }
88             if (count == 2){
89                 tEffects[i] = "</font>";
90                 count = 0;
91             }
92         }
93     }
94     if (count2%2 == 1){
95         tEffects[indexcount] = "X";
96     }
97     for(String i: tEffects){
98         message+= i;
99     }
100 }

```



```

36     if (count2%2 == 1){
37         tEffects[indexcount] = "B";
38     }
39     for(String i: tEffects){
40         message+= i;
41     }
42 }
43
44 //E338
45 //Date: 4.1.206
46
47 if (message.contains(":")){
48     String[] tEffects = message.split(":");
49     message = "";
50     int count = 0;
51     int count2 = 0;
52     int indexcount = 0;
53     for (int i = 0; i < tEffects.length; i++){
54         if (tEffects[i].equals("object:")){
55             count++;
56             count2++;
57             if (count == 1){
58                 indexcount = i;
59                 tEffects[i] = "<font color='blue'>";
60             }
61             if (count == 2){
62                 tEffects[i] = "</font>";
63                 count = 0;
64             }
65         }
66     }
67     if (count2%2 == 1){
68         tEffects[indexcount] = "B";
69     }
70     for(String i: tEffects){
71         message+= i;
72     }
73 }

```

Screenshot of the code solving the formatting display

Checklist Items (0)



^COLLAPSE ^

Task #4 - Points: 1

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

Details:

Note each scenario

Response:

In ChatPanel.java, I modified JEditorPane. It was set to "text/plain" to "text/html." This change enabled the use of HTML tags for formatting sentences.



Private Message with @ (2 pts.)

^COLLAPSE ^



^COLLAPSE ^

Task #1 - Points: 1

Text: Screenshots demoing private message

Checklist

*The checkboxes are for your own tracking

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

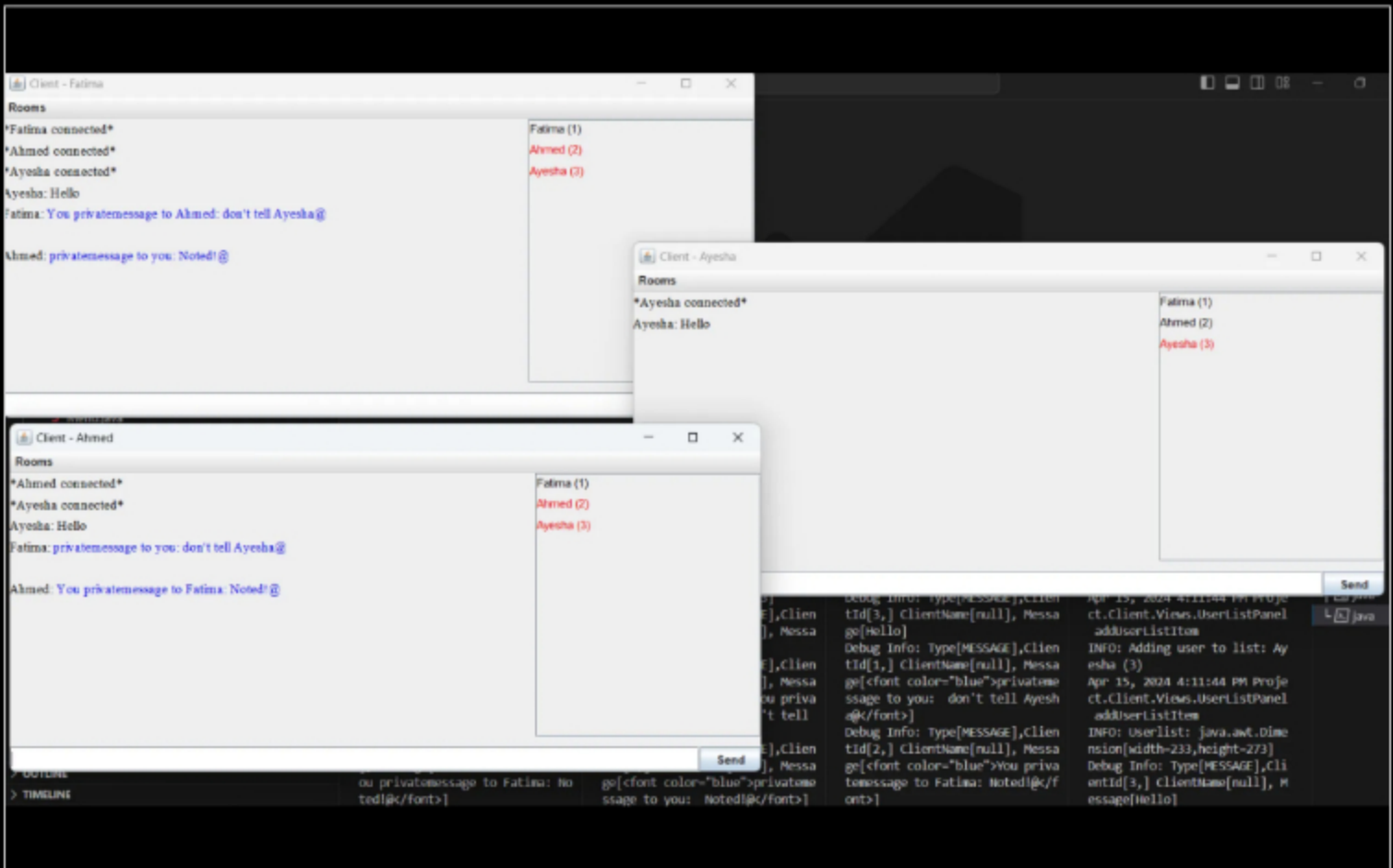
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



Demo

Checklist Items (0)






^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshots of the related code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
 #1	1	Show what code processes and handles the private message
 #2	1	The message should only be sent to the receiver and the target
 #3	1	The client should be targeting the username and the server side should be fetching the correct recipient
 #4	1	Include ucid and date comment
 #5	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```
114 ChatRoom > Project > Server > Room.java > Room > sendMessage(ServerThread sender, String message)
115 class Room implements AutoCloseable {
116     synchronized void sendMessage(ServerThread sender, String message) {
117         //F328
118         //Date: 4.15.2024
119         if (message.contains("@")){
120             String[] words = message.split(regex: "\\s+");
121             for(String word : words){
122                 if (word.startsWith(prefix:"@")){
123                     String privatmessageName = word.substring(beginIndex:1);
124                     ServerThread targetUser = findUser(privatmessageName);
125                     if (targetUser != null){
126                         String senderMessage = "<font color='blue'>You privatmessage to " + targetUser.getClientName() + ":" + message.substring(privatmessageName.length()+1) + "</font>";
127                         String receiverMessage = "<font color='blue'>privatmessage to you: " + message.substring(privatmessageName.length()+1) + "</font>";
128                         targetUser.sendMessage(sender.getClientId(), receiverMessage);
129                         sender.sendMessage(sender.getClientId(), senderMessage);
130                         return;
131                     }
132                 }
133             }
134         }
135     }
136 }
137 }
138 }
139 }
140 }
```

Private message code

Checklist Items (5)

- #1 Show what code processes and handles the private message
- #2 The message should only be sent to the receiver and the target
- #3 The client should be targeting the username and the server side should be fetching the correct recipient
- #4 Include ucid and date comment
- #5 Clearly caption screenshots

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

Response:

Finding the Target User: If a word starts with "@", it is assumed to be a username. The username is extracted by removing the "@" symbol. This username is then used to find the corresponding ServerThread object representing the

removing the @ symbol. This username is then used to find the corresponding ServerThread object representing the target user by calling the findUser method.

Sending the Private Message:

Sender's Perspective:

The sender constructs a message (senderMessage) formatted with a blue color, indicating it's a private message. This message includes the sender's message content and the target user's name (targetUser.getClientName()). The sender sends this message to themselves using sender.sendMessage(sender.getClientId(), senderMessage).

Receiver's Perspective:

The receiver constructs a message (receiverMessage) formatted similarly in blue, indicating it's a private message. This message includes only the message content. The receiver sends this message back to the sender using targetUser.sendMessage(sender.getClientId(), receiverMessage).

Handling Username to ID Mapping:

The code assumes that the findUser method takes a username as an argument and returns the corresponding ServerThread object representing that user. The targetUser object retrieved is used to send the message to the correct recipient.

Mute/Unmute Users (3 pts.)

^COLLAPSE ^

Task #1 - Points: 1

^COLLAPSE ^

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 unmutee

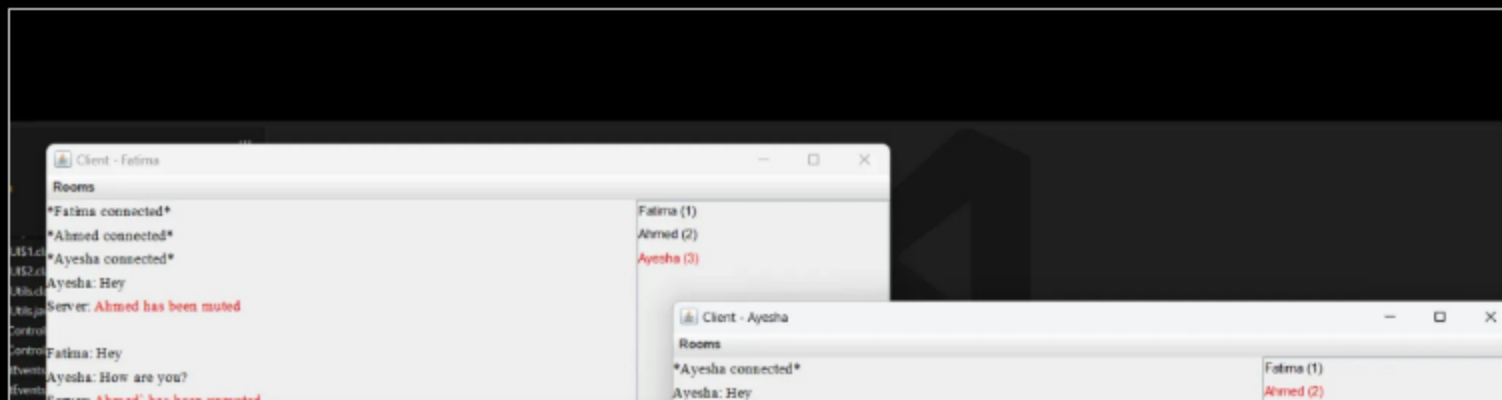
Task Screenshots:

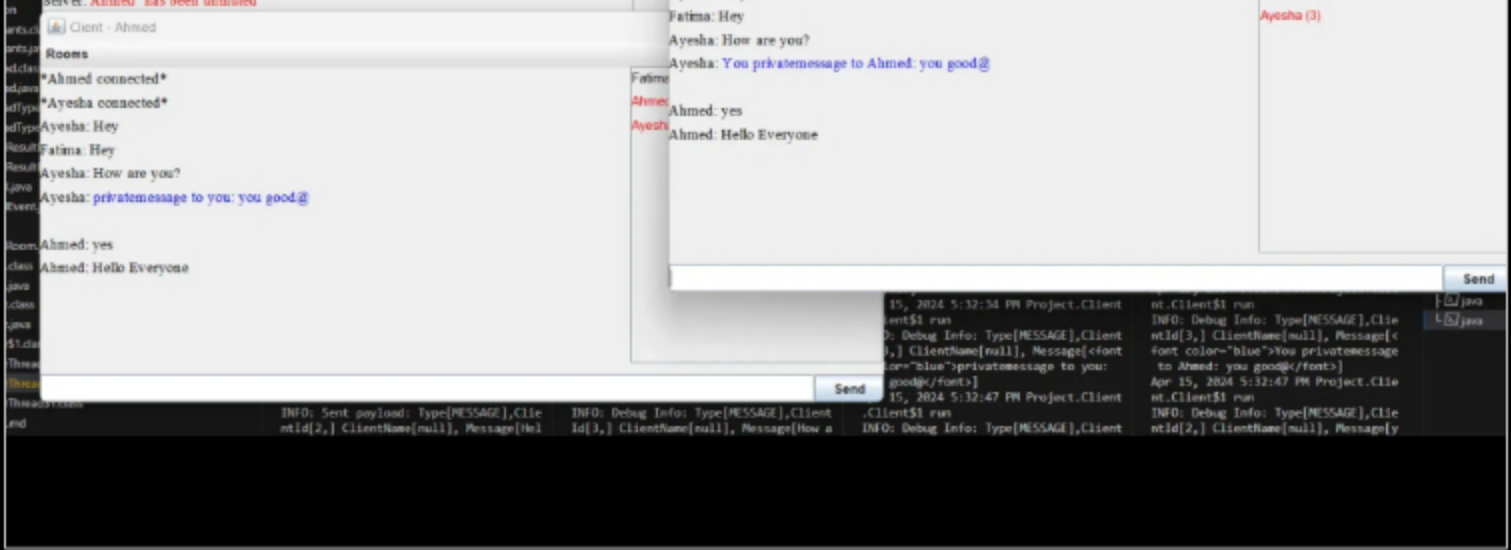
Gallery Style: Large View

Small

Medium

Large





Screenshots demoing feature working

Checklist Items (4)

- #1 Should have 3 clients in the same room
- #2 Demo mute preventing messages between the muter and the target
- #3 Demo mute also being accounted for with private messages
- #4 Demo unmute allowing the messages again from the target to the unmutter



^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

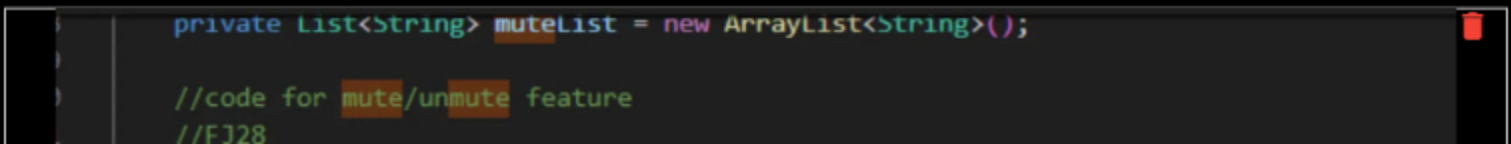
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



```

// F28
// 04/15/24
public boolean sendMuteUser(String name){
    Payload p = new Payload();
    p.setPayloadType(PayloadType.MUTE);
    p.setClientName(name);
    return send(p);
}

public boolean sendUnmuteUser(String name){
    Payload p = new Payload();
    p.setPayloadType(PayloadType.UNMUTE);
    p.setClientName(name);
    return send(p);
}

public boolean isMuted(String name){
    for(String i: mutelist){
        if(i.equals(name)){
            return true;
        }
    }
    return false;
}

```

ServerThread

Checklist Items (2)

#1 ServerThread should have a list of who they muted

#2 ServerThread should expose and add, remove, and is muted check to room

```

187     }
188
189     // F328
190     //DATE: 04/15/24
191
192     public ServerThread findMute(String username) {
193         for(ServerThread user : clients) {
194             if(user.getClientName().equals(username)) {
195                 return user;
196             }
197         }
198         return null;
199     }
200

```

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:

In server thread, three methods are created: `sendMuteUser`, `sendUnmuteUser`, and `isMuted`. The `isMuted` method communicates with the room and involves Payload types: `COMMAND`, `MUTE`, and `UNMUTE`. These payloads are then processed through `muteUser`, `unmuteUser`, and `processMute` methods. A switch case is used: if the action is `MUTE`, it goes to `muteUser`; if it's `UNMUTE`, it goes to `unmuteUser`. Each method generates a new payload object named "p".



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/fj29/-Milestone-Milestone-1/pull/3>

^COLLAPSE ^

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

The assignment went quite well overall, and I didn't encounter any major issues. I learned about the `ChatPanels` and more about the other java files. The `ClientUI` code was detailed and insightful



^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

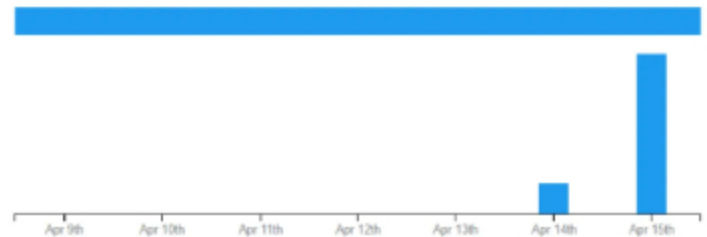
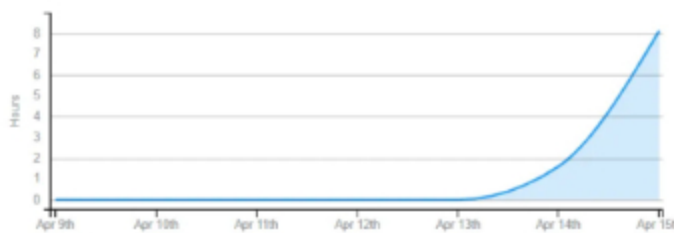
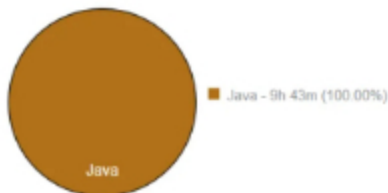
Large

Projects • -Milestone-Milestone-1

total 9 hrs 43 mins



9 hrs 43 mins over the Last 7 Days in -Milestone-Milestone-1 under all branches. 📄

**Languages****Editors**

WakaTime

End of Assignment