

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-chatroom-milestone-4-2024/grade/bm47>

IT114-006-S2024 - [IT114] Chatroom Milestone 4 2024

Submissions:

Submission Selection

1 Submission [active] 4/29/2024 2:22:05 PM


Instructions

^ COLLAPSE ^

Implement the Milestone 4 features from the project's proposal document: <https://docs.google.com/document/d/1ONmvEvel97GTFPGfVwwQC96xSsobbSbk56145Xl/edit>
Make sure you add your ucid/date as code comments where code changes are done
All code changes should reach the Milestone4 branch
Create a pull request from Milestone4 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: Milestone4

Tasks: 15 Points: 10.00

 Demonstrate Chat History Export (2.25 pts.)

^ COLLAPSE ^



Task #1 - Points: 1

Text: Screenshots of code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
---	--------	---------

#1	1	Show the code that gets the messages and writes it to a file (recommended to use a StringBuilder)
#2	1	File name should be unique to avoid overwriting (i.e., incorporate timestamp)
#3	1	Screenshots should include ucid and date comment
#4	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

SmallMediumLarge

```
void exportChatHistory() {
    try {
        SimpleDateFormat dateFormat = new SimpleDateFormat("MM-dd-yyyy_HHms");
        String timestamp = dateFormat.format(new Date());

        File file = new File("chat_logs_" + timestamp + ".txt");
        FileWriter writer = new FileWriter(file);

        dateFormat.applyPattern("MM-dd-yyyy HH:mm:ss");

        for (Component component : chatArea.getComponents()) {
            if (component instanceof JEditorPane) {
                JEditorPane messagePane = (JEditorPane) component;
                String messageText = stripHtmlTags(messagePane.getText().trim());
                if (!messageText.isEmpty()) {
                    String finalMessage = "[" + dateFormat.format(new Date()) + " ] " + messageText; // Add the timestamp to the message

                    // Makes the chat logs easier to read by getting rid of unnecessary spaces and adding a new line after each message
                    //bm47 BRYAN MADEWELL it114 spring 2024
                    writer.write(finalMessage.trim());
                    writer.write(System.lineSeparator());
                }
            }
        }

        writer.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

This code gets the messages and writes it. I found this to be the simplest way to get all of the messages exported into a text file. It has a unique name every time, with the name being chat_logs_ and then the appropriate timestamp in MM-DD-YYYY format to avoid overwriting. My UCID is included. I also added some features to make the logs more legible, such as spaces where needed, and trimming the message. I also got carried away and just got rid of the HTML tags, because when I would export the file the HTML tags were very excessive.

Checklist Items (0)

```
212 // The logs were full of HTML tags and it was very hard to read, this code just removes any of the HTML tags
213 //bm47 BRYAN MADEWELL it114 spring 2024
```

```

214     private String stripHtmlTags(String html) {
215         Pattern pattern = Pattern.compile("<[^>]*>");
216         Matcher matcher = pattern.matcher(html);
217         return matcher.replaceAll("");
218     }
219 }

```

Very simple code that just gets rid of the HTML tags when I export the message.

Checklist Items (0)

ignore this please accidentally created an extra image

Checklist Items (0)



^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshot of the file

Checklist

*The checkboxes are for your own tracking

#	Points	Details
---	--------	---------

#1	1	Show content with variation of messages (i.e., flip, roll, formatting, etc)
#2	1	It should be clear who sent each message
#3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

SmallMediumLarge



This shows the content when using commands such as roll and flip, special formatting, along with regular messages.

Checklist Items (0)

Task #3 - Points: 1

Text: Explain solution

Checklist			*The checkboxes are for your own tracking
#	Points	Details	
#1	1	Mention where the messages are stored and how you fetched them	
#2	1	Mention how the file is generated and populated	

Response:

The messages are fetched by looping over the information in the chatArea. It will then be seen as a message, and is therefore written to the chat logs file using the filewriter. The message is trimmed so it is easier to read, and the timestamp is implemented using the SimpleDateFormat object in the MM/DD/YYYY format

timestamp is implemented using the SimpleDateFormat object in the MM-DD-YY HH:mm:ss format.

Demonstrate Mute List Persistence (2.25 pts.)

^COLLAPSE ^



^COLLAPSE ^

Task #1 - Points: 1

Text: Screenshots of the code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input checked="" type="checkbox"/> #1	1	Show the code that saves the mute list to a file with the name of the user it belongs to
<input type="checkbox"/> #2	1	Show the code that loads the mute list when a ServerThread is connected
<input checked="" type="checkbox"/> #3	1	Screenshots should include ucid and date comment
<input checked="" type="checkbox"/> #4	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```
153     });
154 }
155
156 List<String> mutedUsers = new ArrayList<>();
157
158 //bryan madewell bm47 IT114 SPRING 2024 Milestone 4
159
160 public void addUserListItem(long clientId, String clientName) {
161     userListPanel.addUserListItem(clientId, clientName);
162 }
163
164
165 public void removeUserListItem(long clientId) {
166     userListPanel.removeUserListItem(clientId);
167 }
168
169 public void clearUserList() {
170     userListPanel.clearUserList();
171 }
```

This code saved the mutedUsers to an arraylist.

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the demo

Checklist

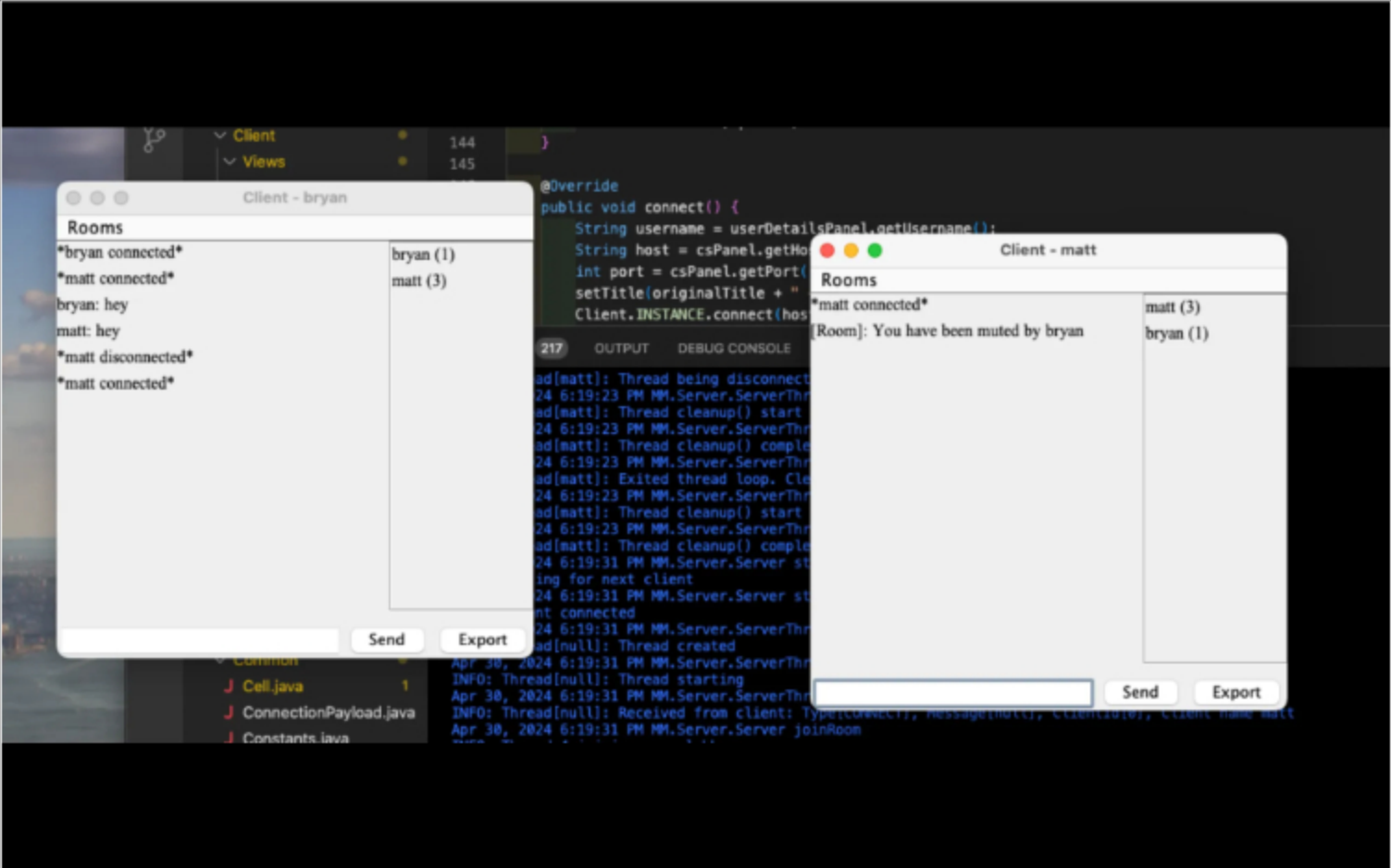
*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Show a user muting another user, disconnecting, reconnecting, and still having that user muted (same should be possible if the server restarts)
<input type="checkbox"/> #2	1	This should also be reflected in the UI per related feature in this milestone
<input checked="" type="checkbox"/> #3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

SmallMediumLarge



shows matt being muted by bryan

Checklist Items (0)

COLLAPSE

Task #3 - Points: 1

Text: Explain solution


Checklist

*The checkboxes are for your own tracking


#	Points	Details
<input type="checkbox"/> #1	1	Mention how you got the mute list to save and load
<input type="checkbox"/> #2	1	Discuss the steps to sync the data to the client/ui

Response:

The mute list contains names of muted users. Upon a user joining, the list is checked to see if there are any muted users, and if they are joining under a username that is muted or not.

 Demonstrate Mute/Unmute notification (2.25 pts.)

 COLLAPSE 

 Task #1 - Points: 1
Text: Screenshots of the code

Checklist			*The checkboxes are for your own tracking
#	Points	Details	
<input type="checkbox"/> #1	1	Show how the message is sent to the target user only if their mute/unmute state changes (i.e., doing mute twice for the same user shouldn't send two mute messages)	
<input type="checkbox"/> #2	1	Screenshots should include ucid and date comment	
<input type="checkbox"/> #3	1	Each screenshot should be clearly captioned	

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
//bm47 IT114 Spring 2024 milestone 4
private void muteUser(String userToMute, ServerThread muter) {
    for (ServerThread client : clients) {
        if (client.getClientName().equals(userToMute)) {
            if (!client.isMuted()) {
                client.setMuted(muted:true);
                client.sendMessage(Constants.DEFAULT_CLIENT_ID, "You have been muted by " + muter.getClientName());
                logger.info(userToMute + " has been muted by " + muter.getClientName());
            } else {
                logger.info(userToMute + " is already muted.");
            }
            return;
        }
    }
    logger.info("User " + userToMute + " not found in the room.");
}
```


This is the muteUser method. In the code, if a user is not muted, they will be muted after the command is sent and will receive a message. To prevent spamming, the code checks if a user is muted. If they are, the message that they are muted is not sent to prevent spamming.

Checklist Items (0)

●

^COLLAPSE ^

Task #2 - Points: 1

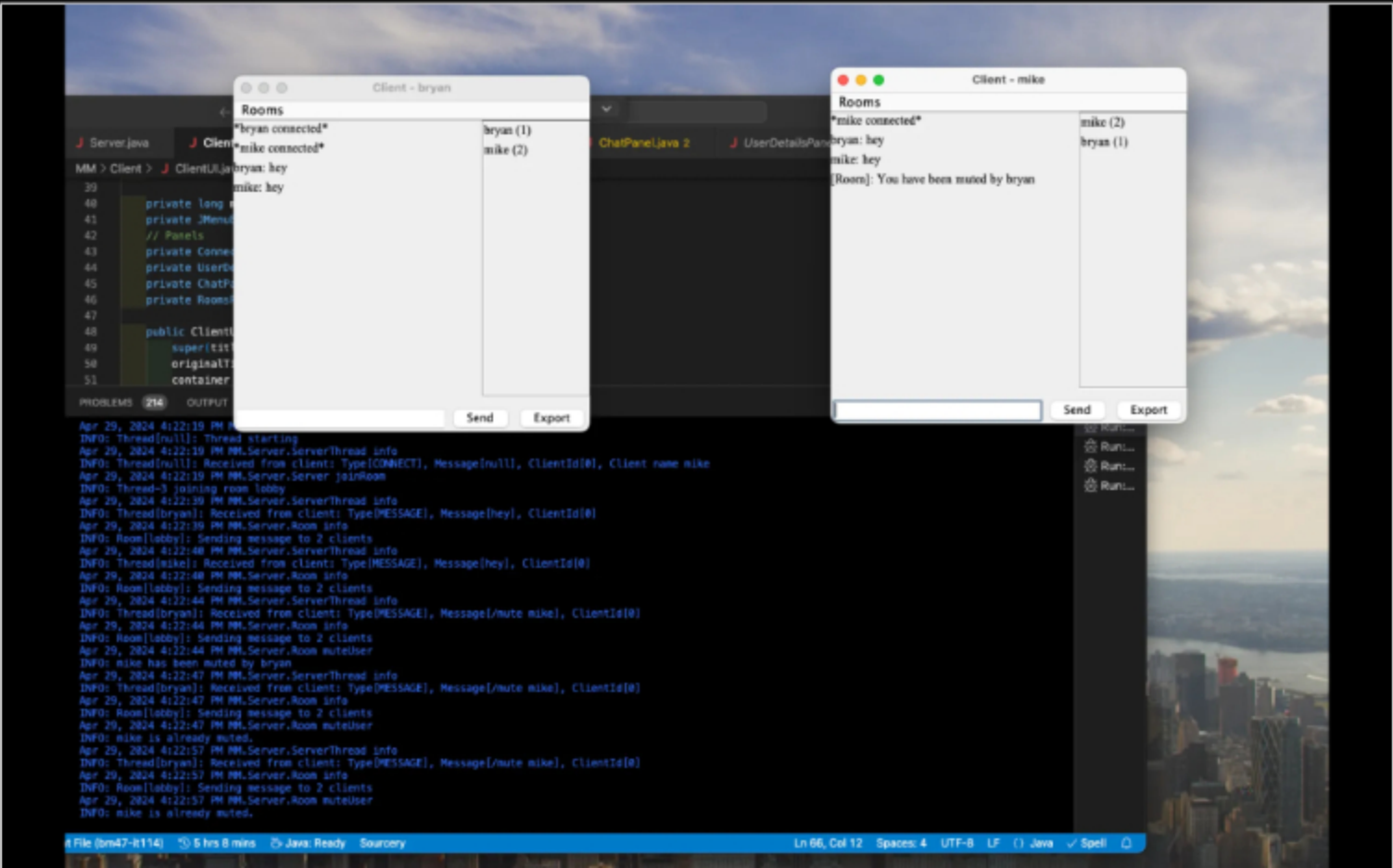
Text: Screenshots of the demo

Checklist		*The checkboxes are for your own tracking
#	Points	Details
<input type="checkbox"/> #1	1	Show examples of doing /mute twice in succession for the same user only yields one message
<input type="checkbox"/> #2	1	Show examples of doing /unmute twice in succession for the same user only yields one message
<input type="checkbox"/> #3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

SmallMediumLarge



This code shows bryan muting mike. After the first command is sent, mike is then muted. I sent /mute mike multiple times as bryan, and as you can see in the terminal the message that mike is already muted pops up, as well as the UI only letting mike know that he has been muted once.

Checklist Items (0)



^COLLAPSE ^

Task #3 - Points: 1

Text: Explain solution

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input checked="" type="checkbox"/> #1	1	Mention how you limit the messages in each scenario
<input checked="" type="checkbox"/> #2	1	Discuss how you find the correct user to send the message to

Response:

As previously explained, the mute method checks if a user is muted. If they are, they do not receive the message that they are muted more than once. If they are already muted and someone tries to mute them again, it is noted that they are already muted and no message is received. The correct user is found by using the `clientName`, which corresponds to the username they entered when logging into the chatroom.



Demonstrate user list visual changes (2.25 pts.)

^COLLAPSE ^



^COLLAPSE ^

Task #1 - Points: 1

Text: Screenshots of the code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Show the code related to "graying out" muted users and returning them to normal when unmuted
<input type="checkbox"/> #2	1	Show the code related to highlighting the user who last sent a message (and unhighlighting the remainder of the list)
<input checked="" type="checkbox"/> #3	1	Screenshots should include ucid and date comment
<input checked="" type="checkbox"/> #4	1	Each screenshot should be clearly captioned

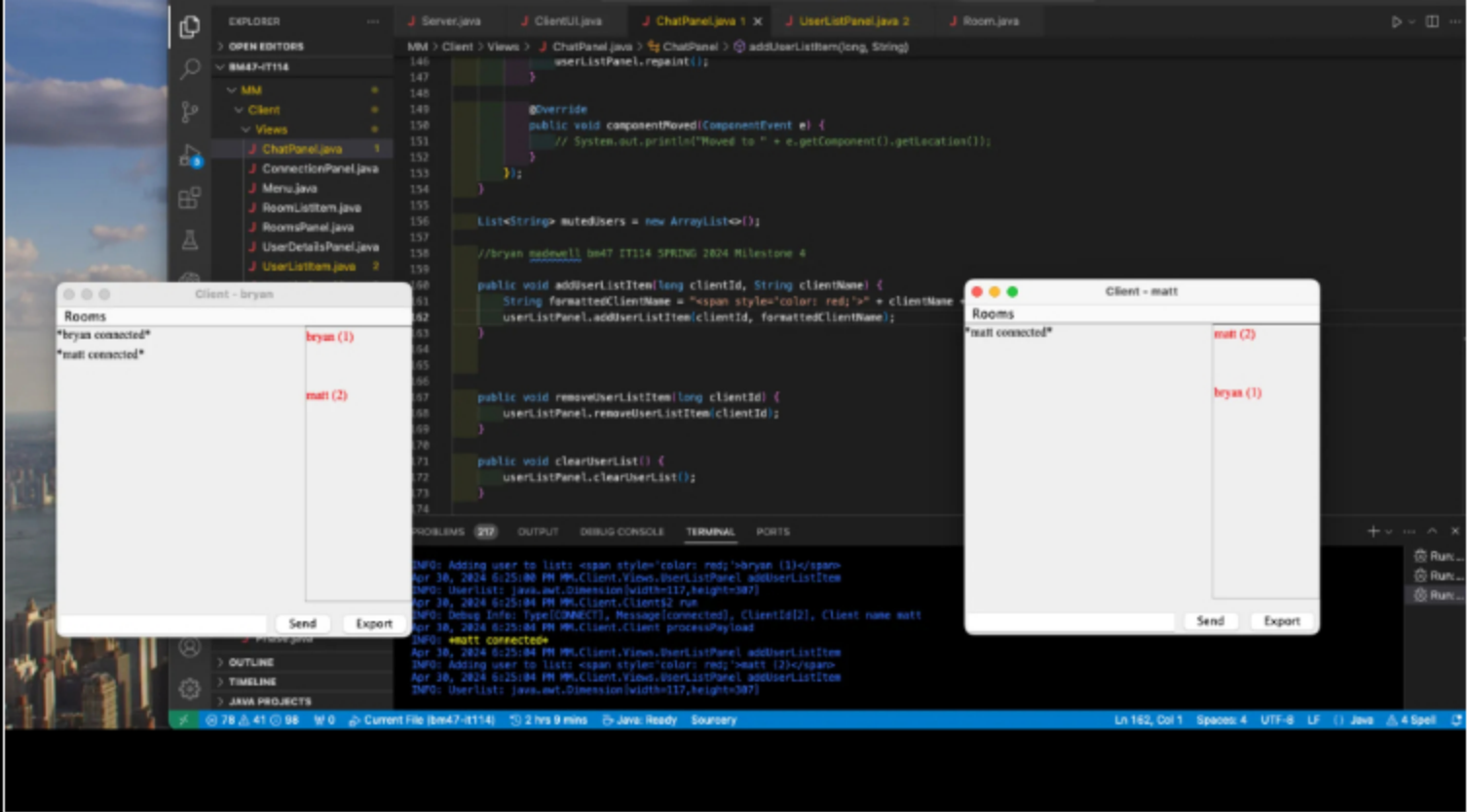
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



Unfortunately, i was not able to get this feature to work. I attempted changing the addUserListItem code, but it did not work because it needed to be updated whenever a user was muted in order to change the color. I was able to permanently change the color, but not change the color upon muting a user. In the code and UI, it shows that I was able to change the color of the usernames in the UI, but unfortunately I was not able to get it to update upon mute.

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the demo

Checklist			*The checkboxes are for your own tracking
#	Points	Details	
<input type="checkbox"/> #1	1	Show before and after screenshots of the list updating upon mute and unmute	
<input type="checkbox"/> #2	1	Capture variations of "last person to send a message gets highlighted"	
<input checked="" type="checkbox"/> #3	1	Each screenshot should be clearly captioned	

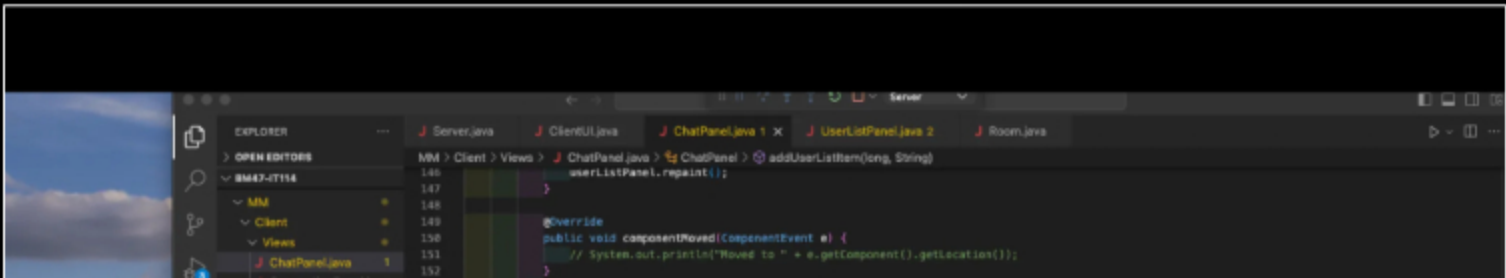
Task Screenshots:

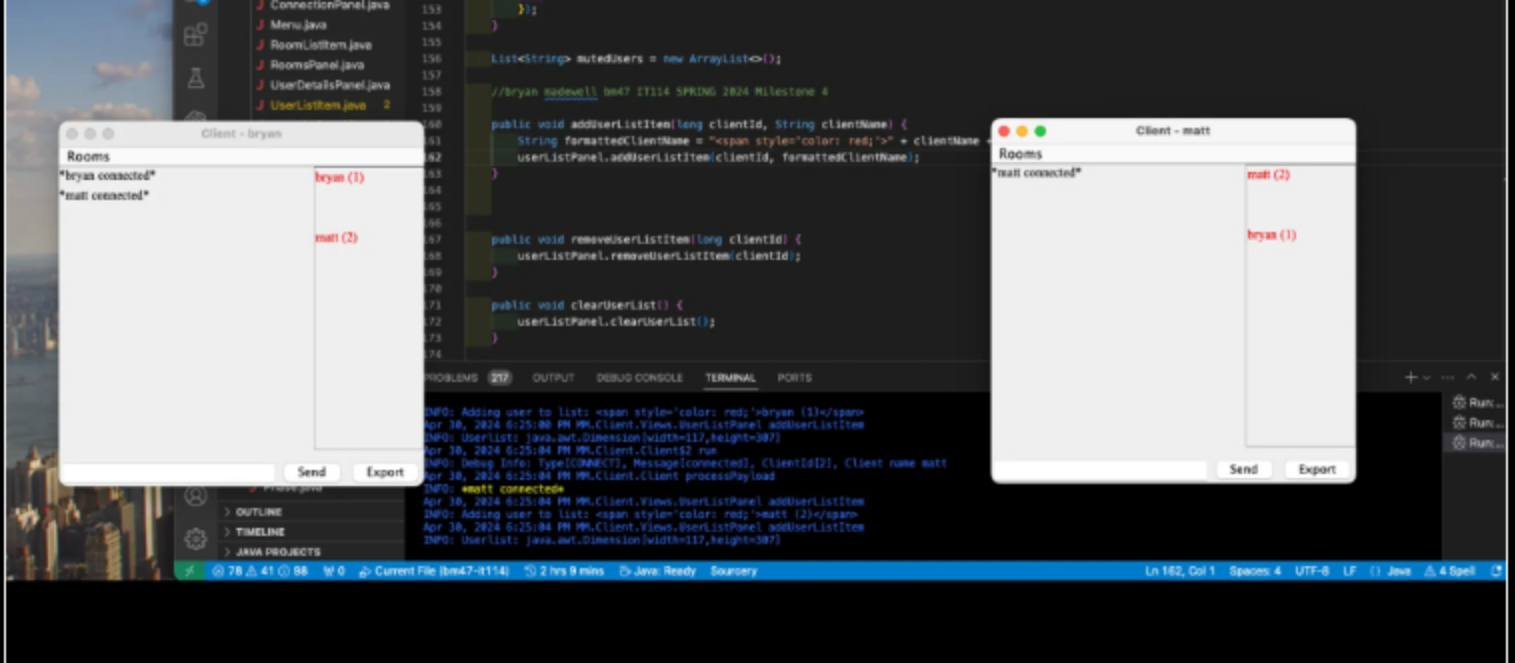
Gallery Style: Large View

Small

Medium

Large





As previously stated, I was not able to get these features to work. Hopefully I receive partial credit for being able to change the names in the user list panel, but not change them if they were the last person to send a message or if they were muted.

Checklist Items (0)

^COLLAPSE ^

Task #3 - Points: 1

Text: Explain solution

Checklist			*The checkboxes are for your own tracking
#	Points	Details	
<input checked="" type="checkbox"/> #1	1	Mention how you got the mute/unmute effect implemented	
<input checked="" type="checkbox"/> #2	1	Mentioned how you got the highlight effect implemented (including unhighlighting the other users)	

Response:

If I were to properly get it to work, it would likely involve having a method that checks whether or not a user is muted, and another method that checks who sent the most recent message. As far as the mute method goes, if the user is muted, it would change their username to gray, and upon unmute, their name would go back to default. For the most recent message sent, it would highlight the person that most recently sends the message, while simultaneously changing everyone elses name to default, if they are not muted.

^COLLAPSE ^

Misc (1 pt.)

^COLLAPSE ^

Task #1 - Points: 1

Text: Add the pull request link for the branch

Details:

Note: the link should end with /pull/#

URL #1

Missing URL



^COLLAPSE ^

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

I had quite a few issues during this assignment. Milestone 4 was probably the most challenging for me. I was able to include many of the features in Milestone 3, but adding the additional features in Milestone 4 was much more challenging to me. Some features I struggled with adding were any of the features that highlighted the username in the User List Panel. It was difficult to get these features to work properly, but I did learn a lot while attempting to do so. I enjoyed the chatroom project and found it to be both educational and challenging at the same time. I did become frustrated at times, but eventually figured out most of what I was supposed to.



^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

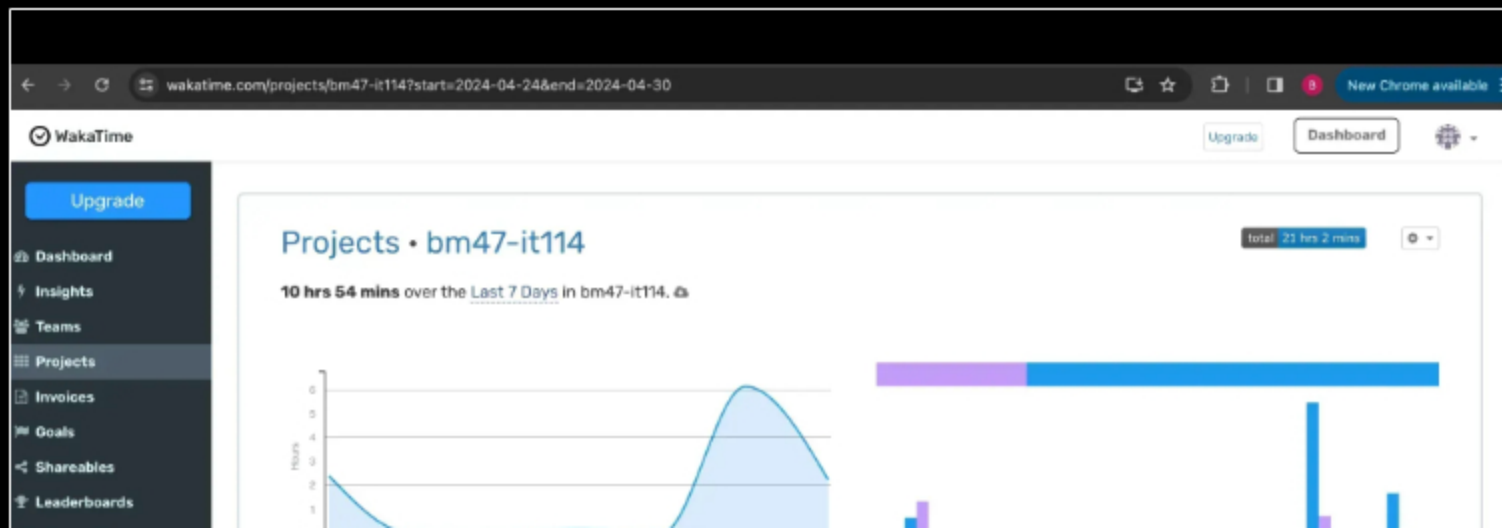
Task Screenshots:

Gallery Style: Large View

Small

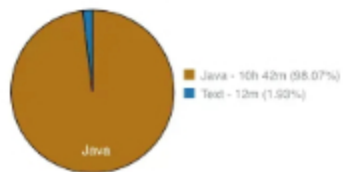
Medium

Large





Languages



Editors



Missing Caption

End of Assignment