Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-004-S2024/it114-chatroom-milestone-4-2024/grade/mbh3

IT114-004-S2024 - [IT114] Chatroom Milestone 4 2024

Submissions:

Submission Selection

1 Submission [active] 4/30/2024 5:55:09 PM

Instructions

COLLAPSE

Implement the Milestone 4 features from the project's proposal document: https://docs.google.com/document/d/1ONmvEvel97GTFPGfVwwQC96xSsobbSbk56145Xf
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

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of code

Checklist

*The checkboxes are for your own tracking

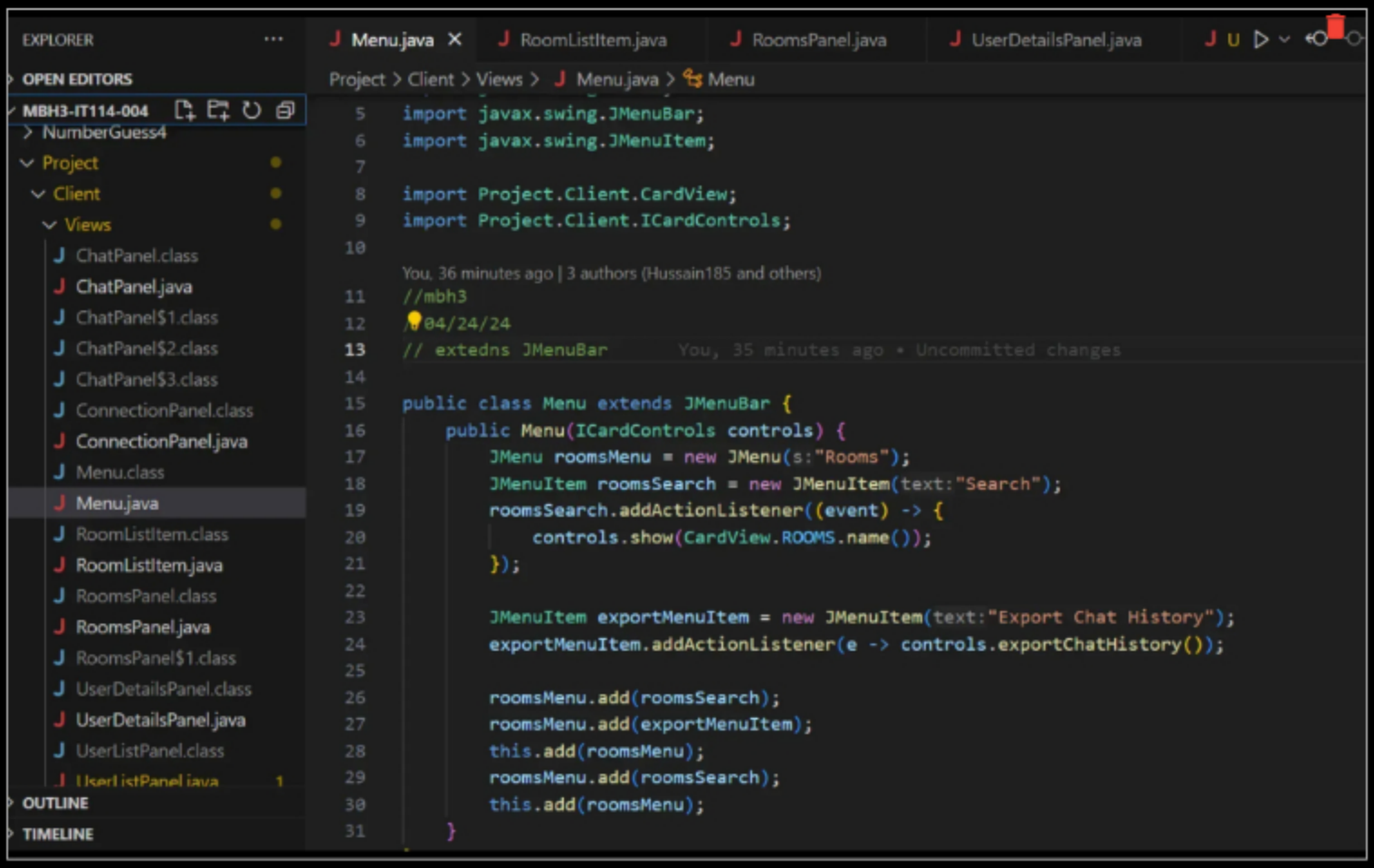
#	Points	Details

#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

Small Medium Large



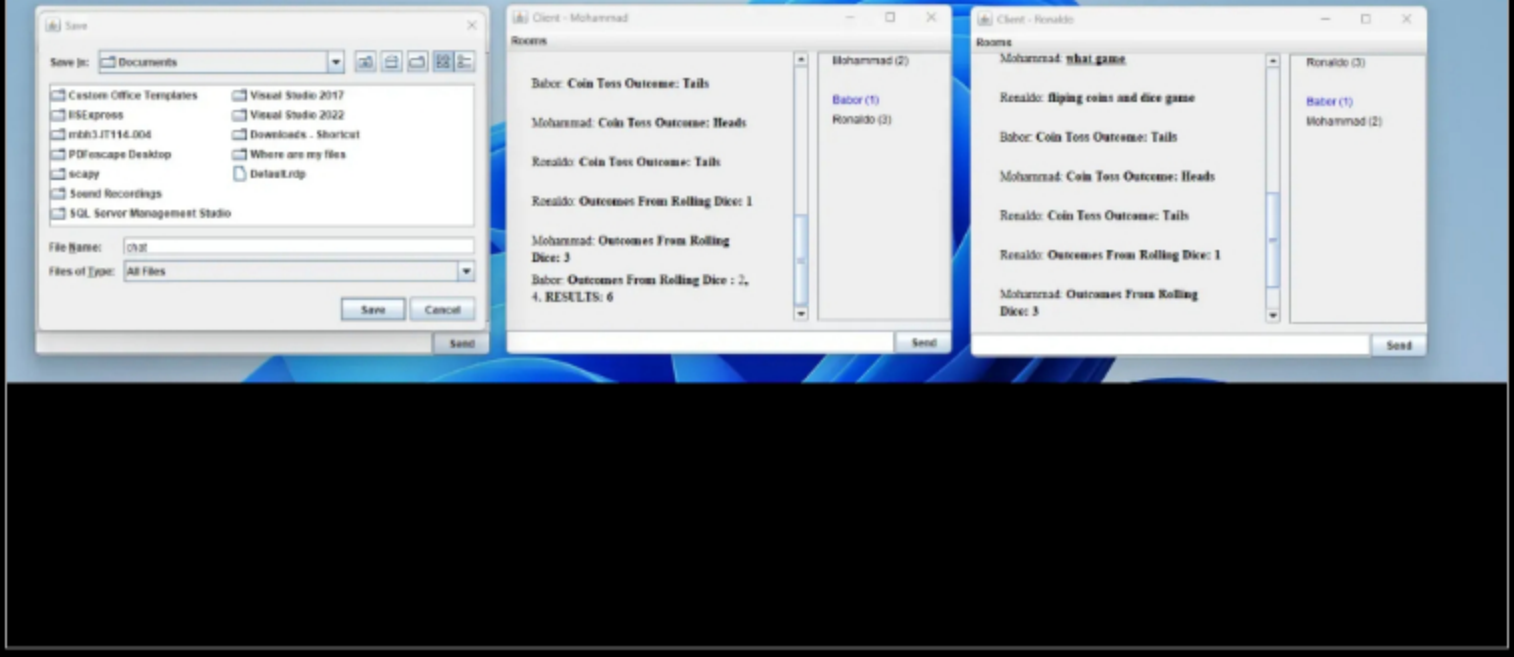
related code

Checklist Items (3)

#1 Show the code that gets the messages and writes it to a file (recommended to use a StringBuilder)

#3 Screenshots should include ucid and date comment

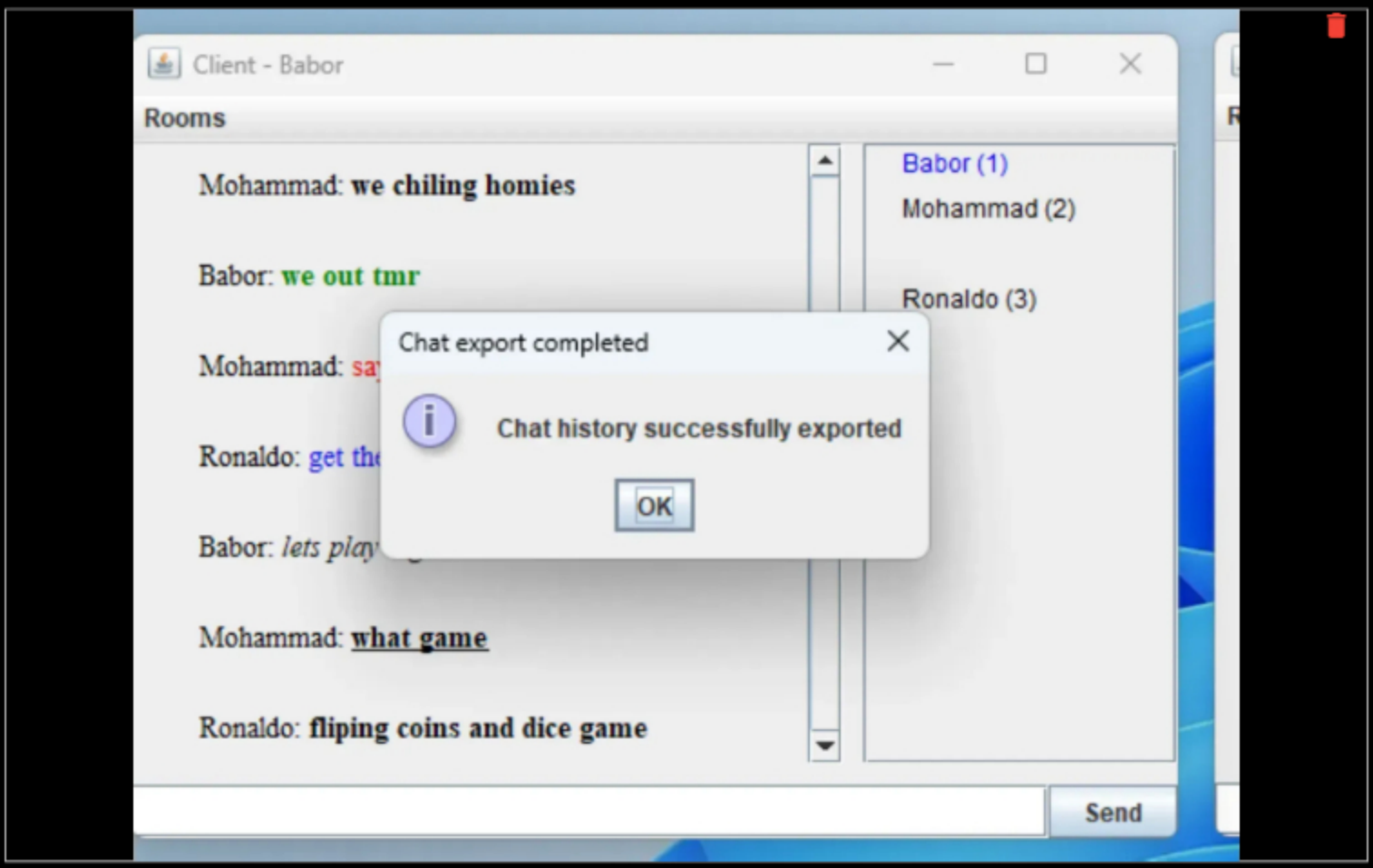
#4 Each screenshot should be clearly captioned



file name of export chat and where I saved it.

Checklist Items (1)

#2 File name should be unique to avoid overwriting (i.e., incorporate timestamp)

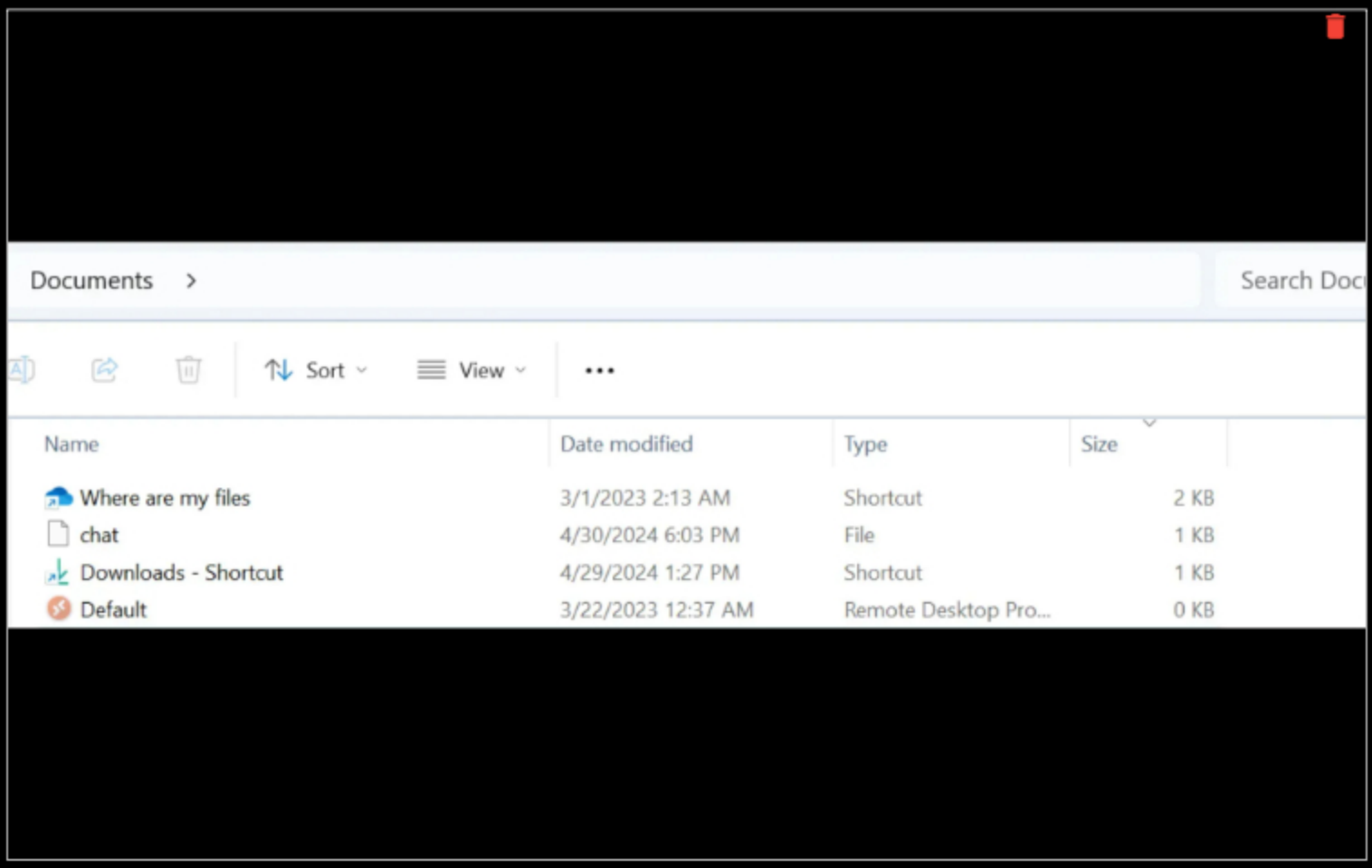


successfully imported the chat

Checklist Items (2)

#2 File name should be unique to avoid overwriting (i.e., incorporate timestamp)

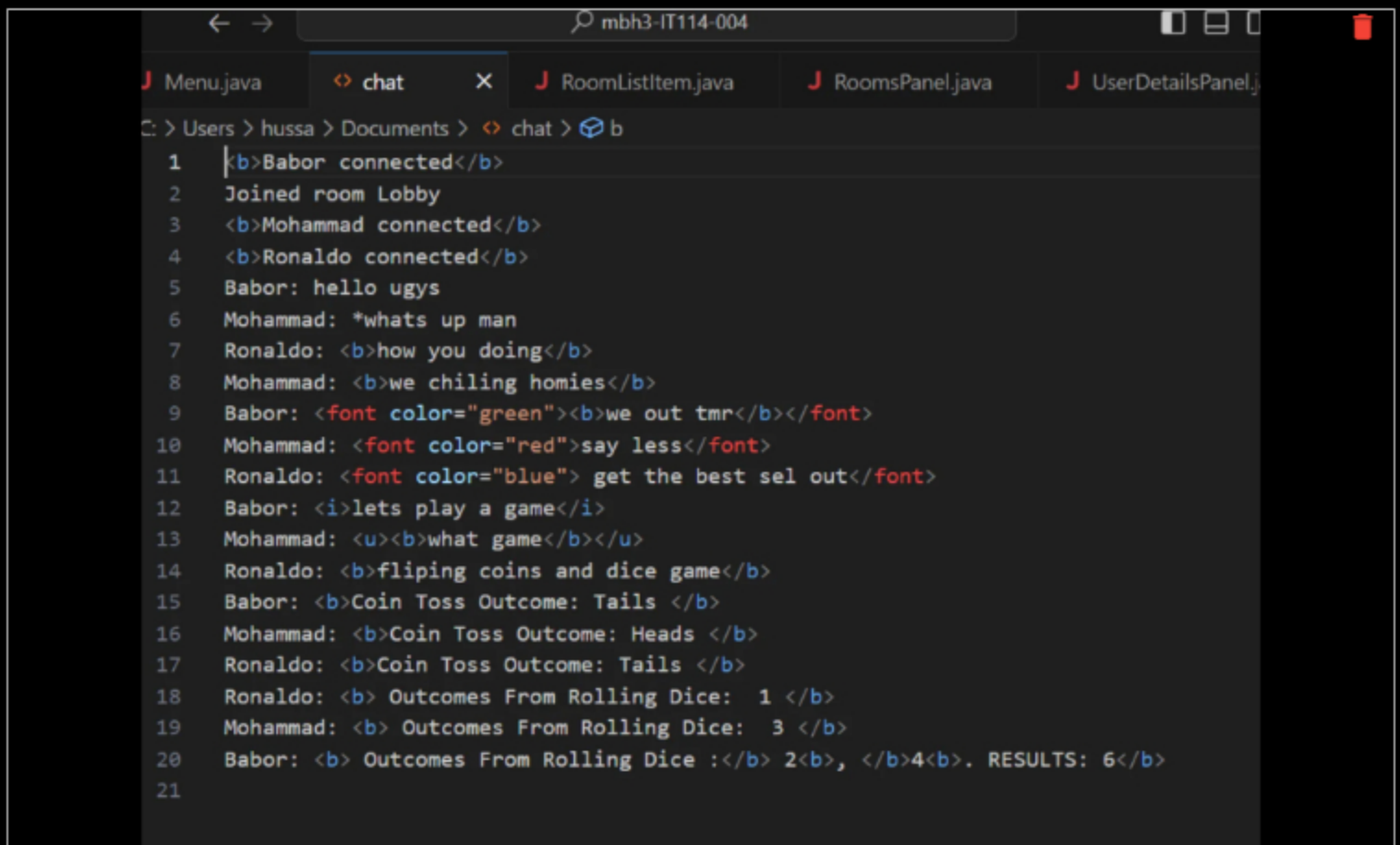
#4 Each screenshot should be clearly captioned



timestamps of the chat file

Checklist Items (1)

#2 File name should be unique to avoid overwriting (i.e., incorporate timestamp)

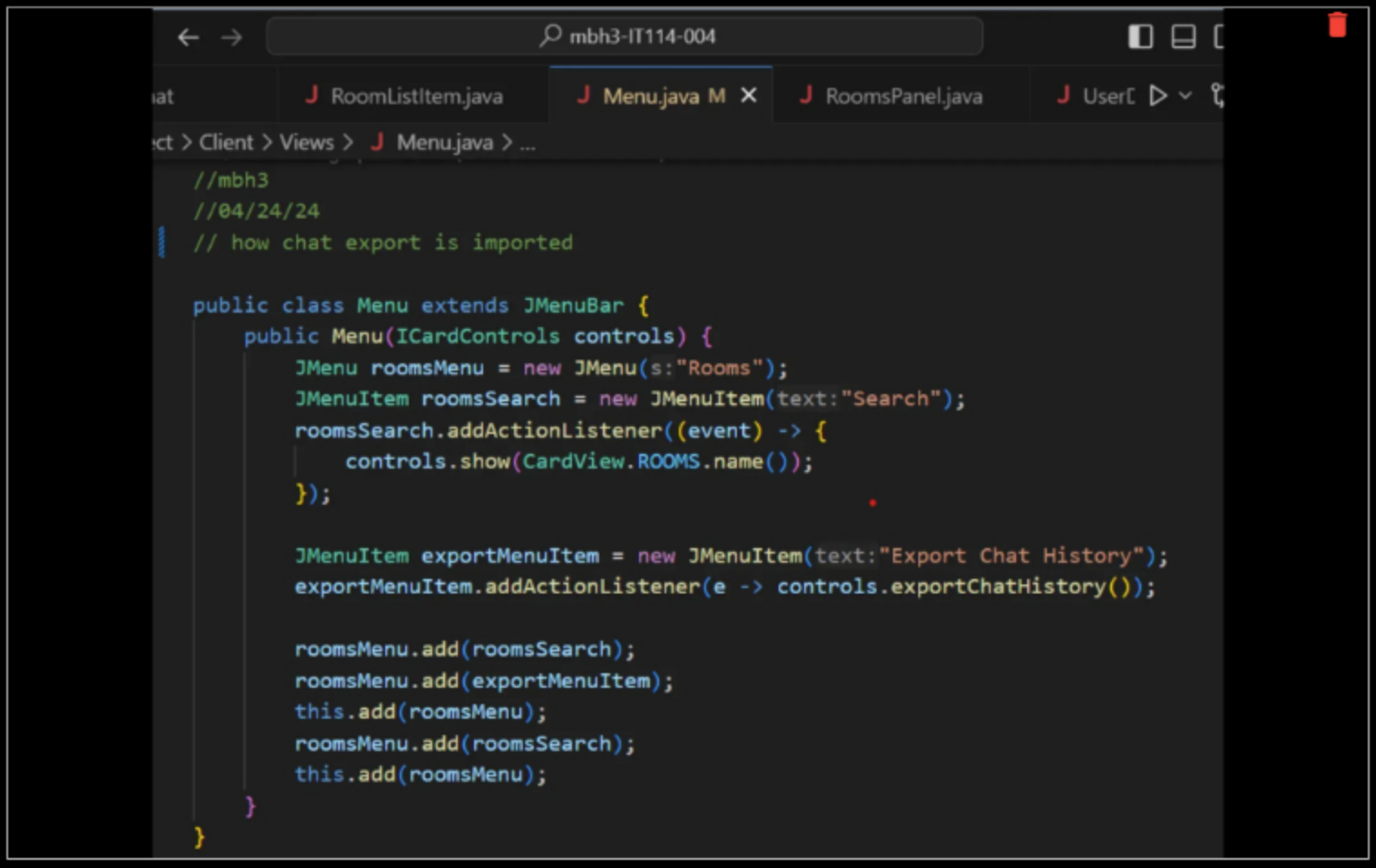


chat history

Checklist Items (2)

#2 File name should be unique to avoid overwriting (i.e., incorporate timestamp)

#4 Each screenshot should be clearly captioned



code

Checklist Items (4)

#1 Show the code that gets the messages and writes it to a file (recommended to use a StringBuilder)

#2 File name should be unique to avoid overwriting (i.e., incorporate timestamp)

#3 Screenshots should include ucid and date comment

#4 Each screenshot should be clearly captioned

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the file

Checklist

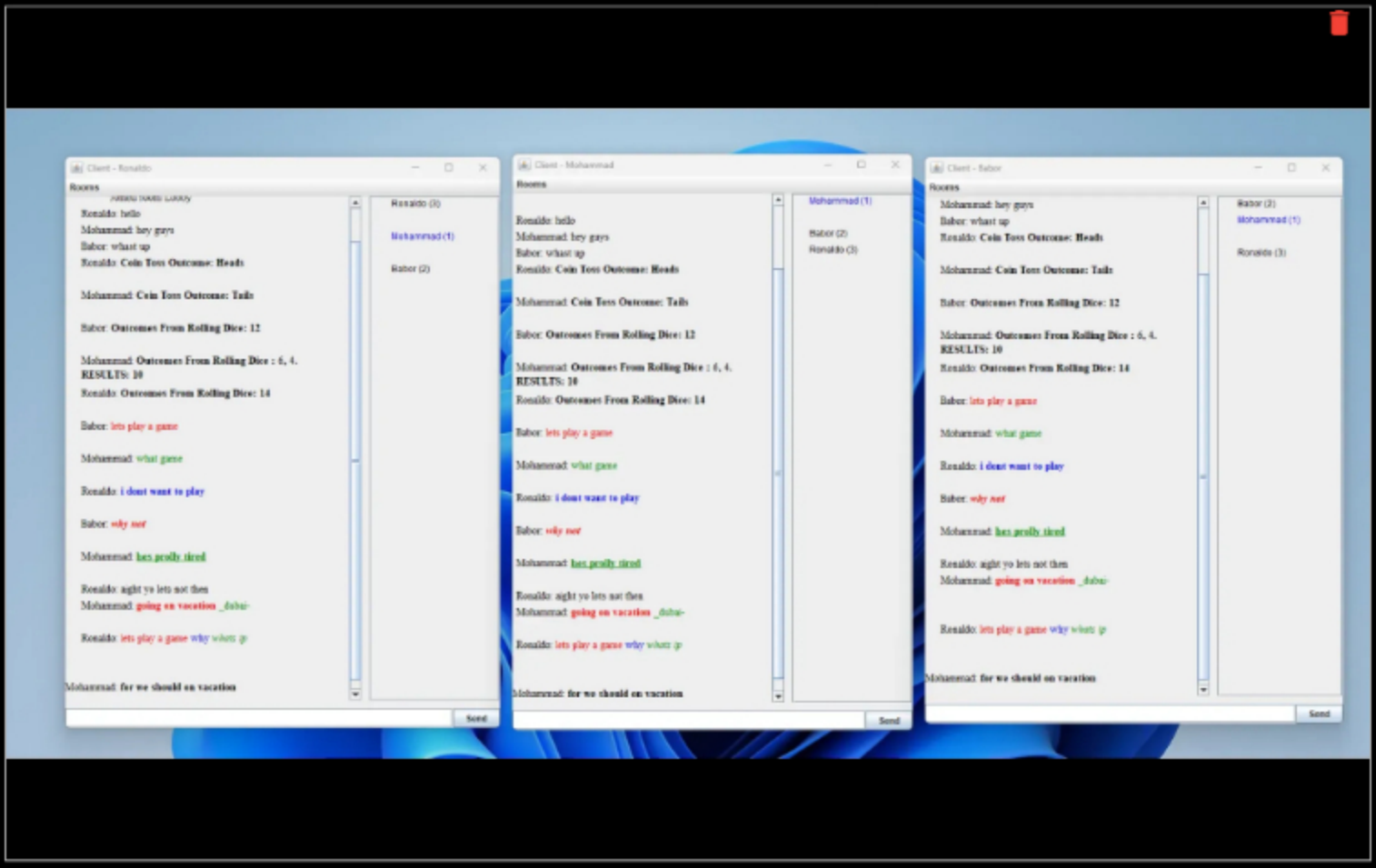
*The checkboxes are for your own tracking

#	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

Small Medium Large



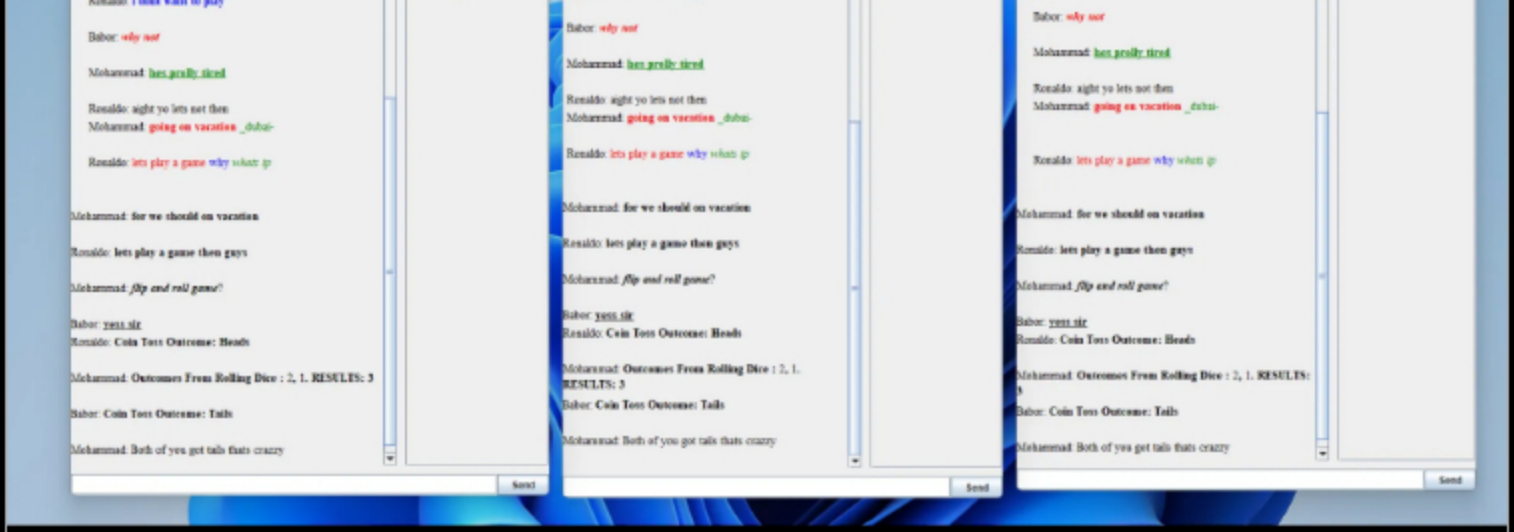
different types of message and formating and who sent the text

Checklist Items (3)

#1 Show content with variation of messages (i.e., flip, roll, formatting, etc)

#2 It should be clear who sent each message

#3 Each screenshot should be clearly captioned



chat mesaages

Checklist Items (3)

#1 Show content with variation of messages (i.e., flip, roll, formatting, etc)

#2 It should be clear who sent each message

#3 Each screenshot should be clearly captioned

COLLAPSE

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:

I store chat messages in a list called chatHistory within the ChatPanel class. As users interact in the chat, their messages are collected in this list and formatted for display using HTML to apply styles such as bold, italic, and colored text. I display these messages in a JEditorPane located in the chatArea.

I've implemented a feature to export the chat history, which users can access through a menu item labeled "Export Chat History." It collets messages from chatHistory to a file which allows users to save the chat logs for later review or archival purposes, making it easy to retrieve past conversations whenever needed.

Demonstrate Mute List Persistence (2.25 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the code

Checklist

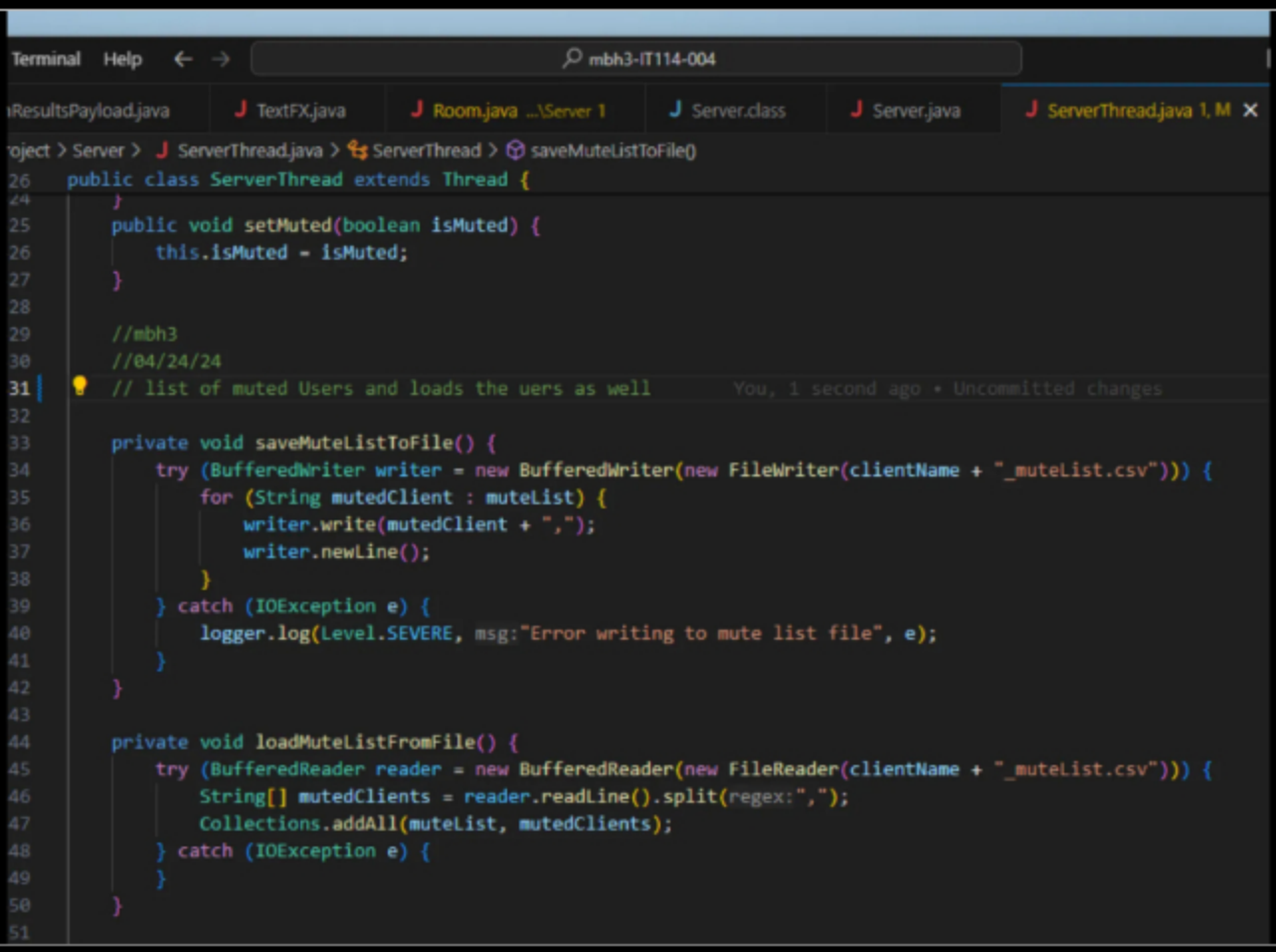
*The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



save mute and load mute files

Checklist Items (4)

#1 Show the code that saves the mute list to a file with the name of the user it belongs to

#2 Show the code that loads the mute list when a ServerThread is connected

#3 Screenshots should include ucid and date comment

#4 Each screenshot should be clearly captioned

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the demo

Checklist

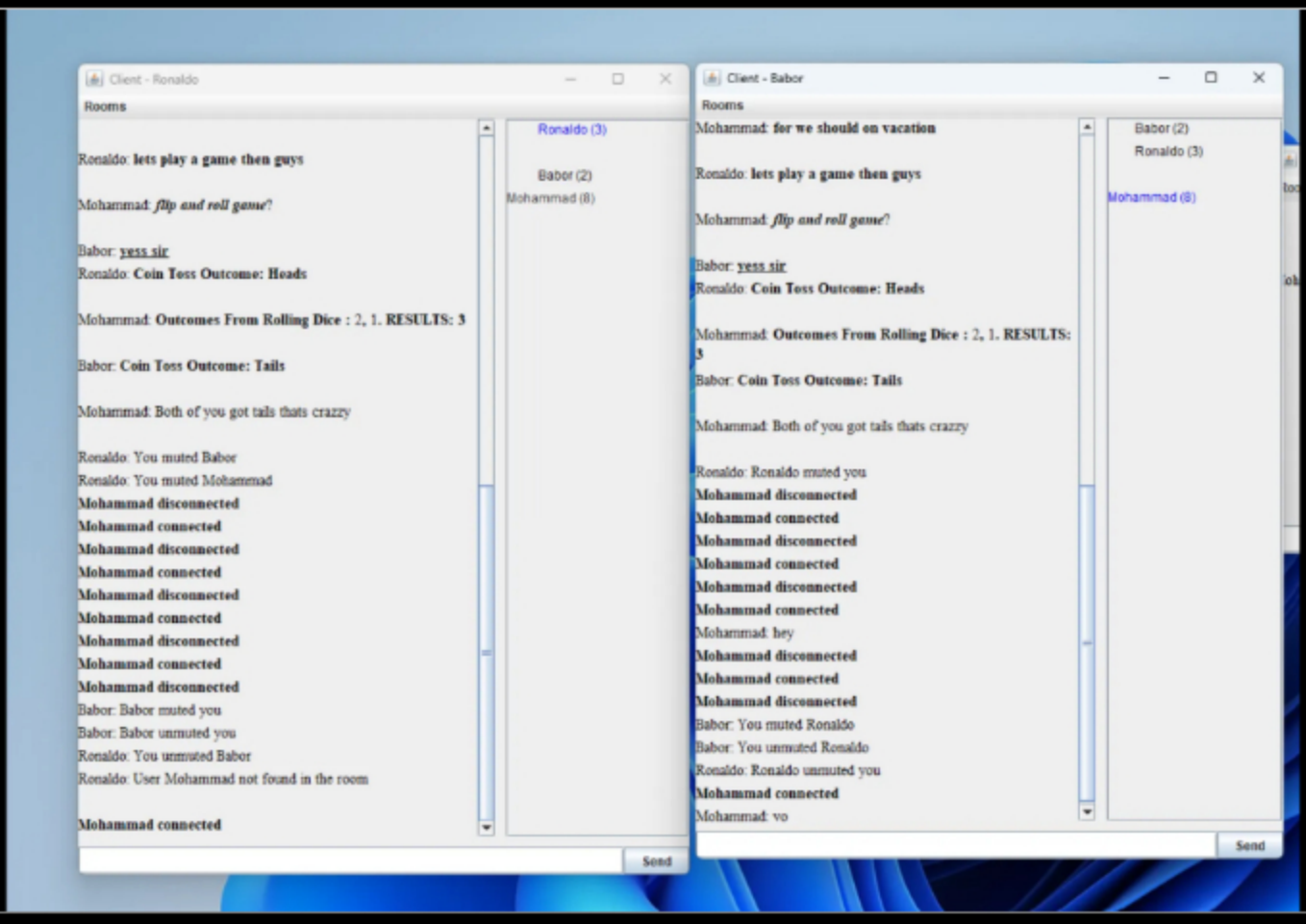
*The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



muting unmuting users and connecting and disconneting

Checklist Items (3)

#1 Show a user muting another user, disconnecting, reconnecting, and still having that user muted (same should be possible if the server restarts)

#2 This should also be reflected in the UI per related feature in this milestone

#3 Each screenshot should be clearly captioned

COLLAPSE

Task #3 - Points: 1

Text: Explain solution

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Mention how you got the mute list to save and load
☐ #2	1	Discuss the steps to sync the data to the client/ui

Response:

I have implemented functionality to manage a list of muted users, allowing me to save and load this list to and from a file. I use a method called saveMuteListToFile() to write the names of muted users to a CSV file. When saving, I iterate over the list of muted clients and write each one to the file along with a comma separator, ensuring each entry is on a new line. If there's an error during the file writing process, I log this with a severity level of SEVERE. To load the mute list back into the application, I use loadMuteListFromFile(), which reads the CSV file. Each line of the file, representing a muted user, is split by commas to retrieve individual usernames, which are then added back into the application's mute list. If there's an issue during the reading process, I also log this error.

When the application runs, it immediately load the mute list from the file using loadMuteListFromFile(). This ensures that any previously muted users are still recognized by the system. And whenever the mute list changes (either by adding or removing a user), it update the relevant UI components. In addiiton, any action taken on the UI that alters the mute list (like muting or unmuting a user) triggers an immediate save to the file by calling saveMuteListToFile(), ensuring that the file always reflects the current state of muted users.

Demonstrate Mute/Unmute notification (2.25 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the code

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #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)
☐ #2	1	Screenshots should include ucid and date comment

#3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
371
372     //mbh3
373     //04/24/24
374     //mutes target users command
375
376     private void processMuteCommand(String message) {
377         String targetUsername = message.substring(beginIndex:5).trim();
378
379         if (currentRoom != null) {
380             ServerThread targetClient = currentRoom.findClientByName(targetUsername);
381
382             if (targetClient != null) {
383                 if (muteList.contains(targetUsername)) {
384                     sendMessage(getClientId(), targetUsername + " is already muted");
385                 } else {
386                     muteList.add(targetUsername);
387                     saveMuteListToFile();
388                     targetClient.sendMessage(getClientId(), getClientName() + " muted you");
389                     sendMessage(getClientId(), "You muted " + targetUsername);
390                 }
391             } else {
392                 sendMessage(getClientId(), "User " + targetUsername + " not found in the room");
393             }
394         } else {
395             sendMessage(getClientId(), message:"Currently not in a room");
396         }
397     }
398
399
400
```

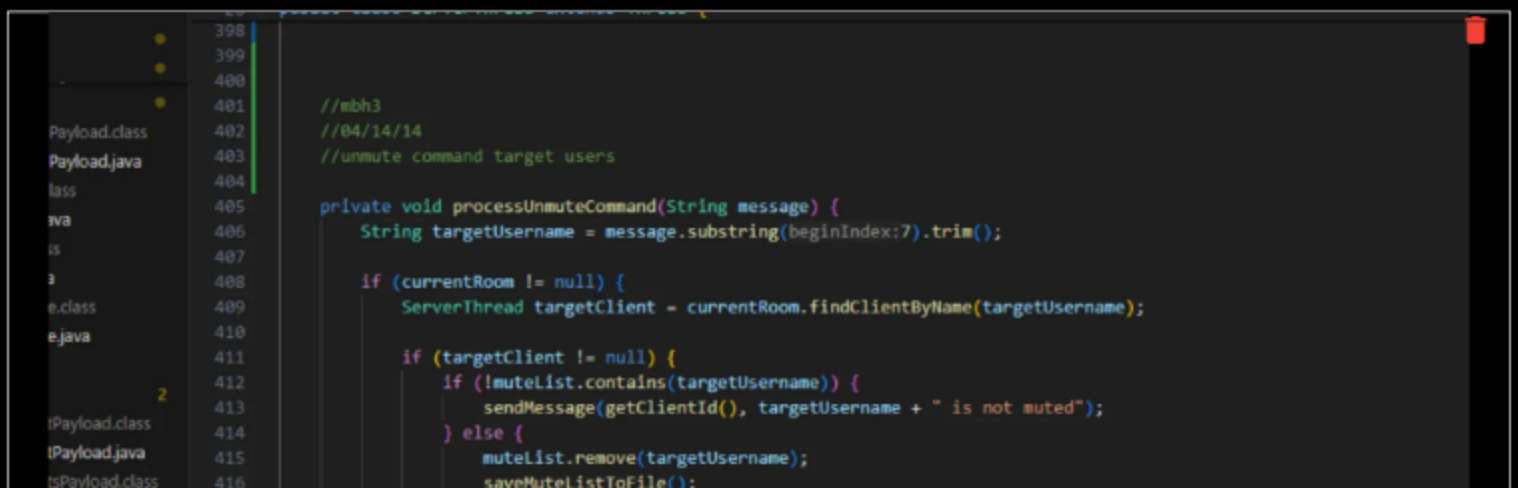
Mute command code

Checklist Items (3)

#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)

#2 Screenshots should include ucid and date comment

#3 Each screenshot should be clearly captioned



```
398
399
400
401     //mbh3
402     //04/14/14
403     //unmute command target users
404
405     private void processUnmuteCommand(String message) {
406         String targetUsername = message.substring(beginIndex:7).trim();
407
408         if (currentRoom != null) {
409             ServerThread targetClient = currentRoom.findClientByName(targetUsername);
410
411             if (targetClient != null) {
412                 if (!muteList.contains(targetUsername)) {
413                     sendMessage(getClientId(), targetUsername + " is not muted");
414                 } else {
415                     muteList.remove(targetUsername);
416                     saveMuteListToFile();
```

```
417                     targetClient.sendMessage(getClientId(), getClientName() + " unmuted you");
418                     sendMessage(getClientId(), "You unmuted " + targetUsername);
419                 }
420             } else {
421                 sendMessage(getClientId(), "User " + targetUsername + " not found in the room");
422             }
423         } else {
424             sendMessage(getClientId(), message:"Currently not in a room");
425         }
426     }
427
428     public boolean isMuted() {
429         return isMuted;
430     }
431     public void setMuted(boolean isMuted) {
432         this.isMuted = isMuted;
433     }
434
```

unmute command code

Checklist Items (3)

#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)

#2 Screenshots should include ucid and date comment

#3 Each screenshot should be clearly captioned

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the demo

Checklist

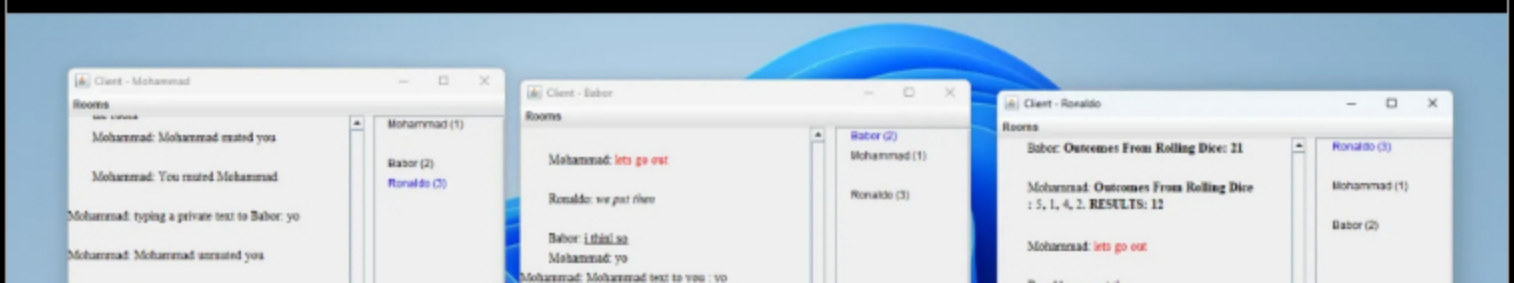
*The checkboxes are for your own tracking

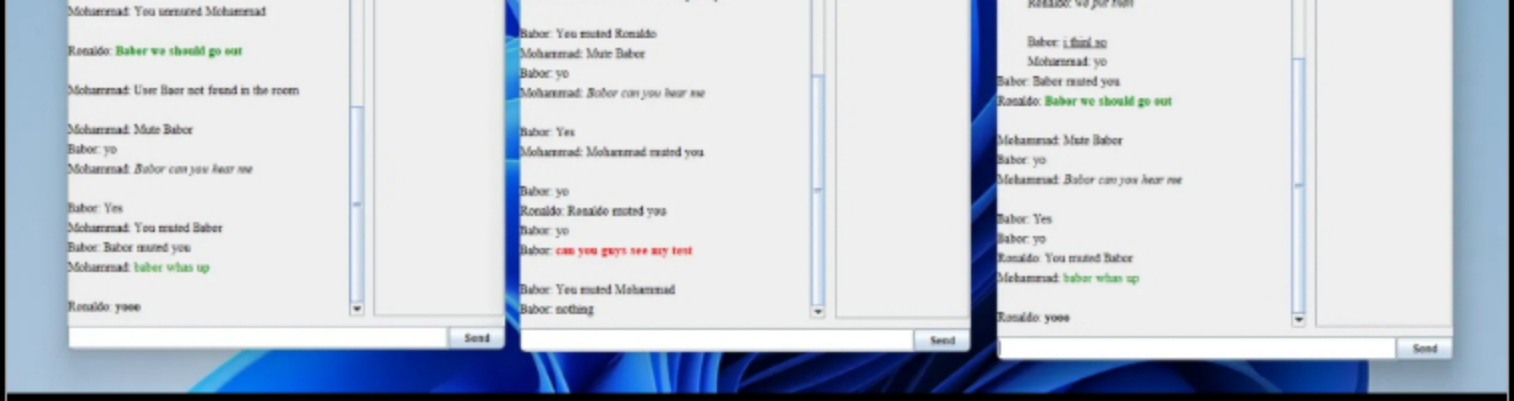
#	Points	Details
☐ #1	1	Show examples of doing /mute twice in succession for the same user only yields one message
☐ #2	1	Show examples of doing /unmute twice in succession for the same user only yields one message
☐ #3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large





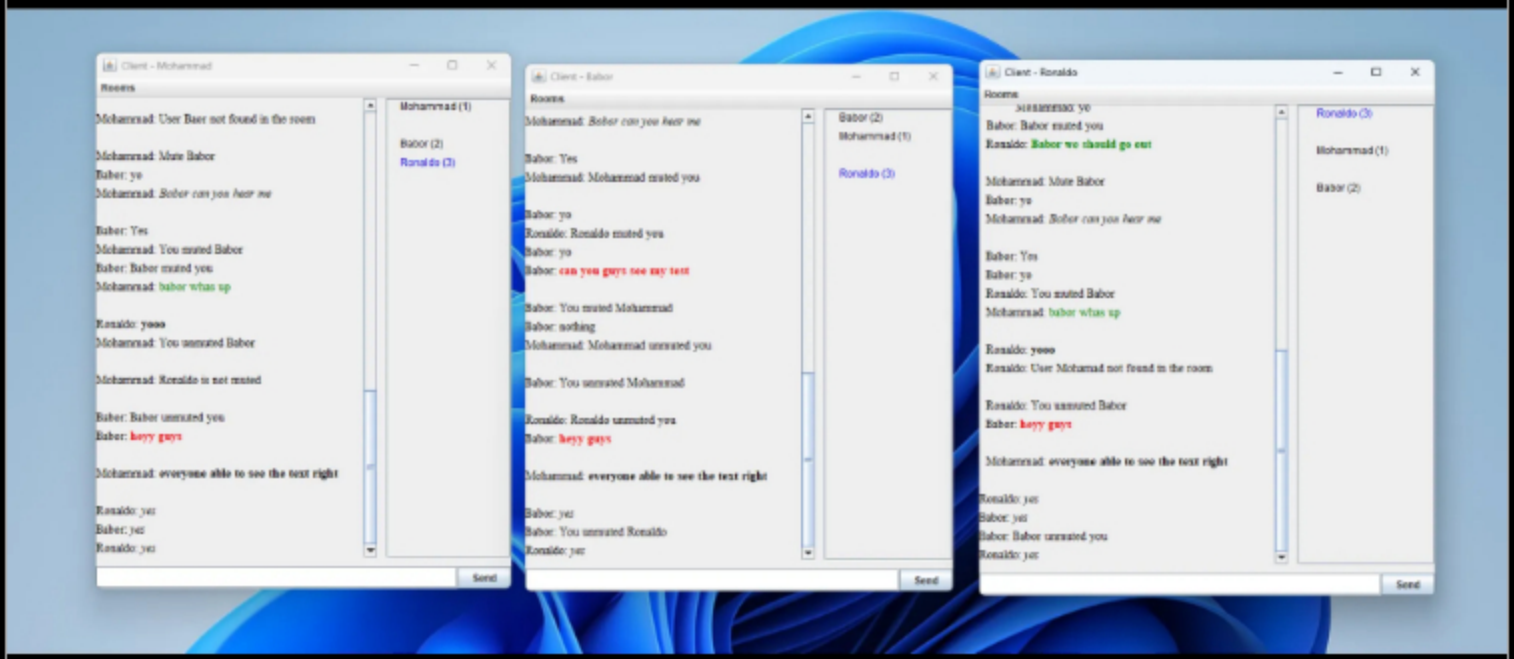
As you can see how when they are muted they cannot see each other text or anything like that.

Checklist Items (3)

#1 Show examples of doing /mute twice in succession for the same user only yields one message

#2 Show examples of doing /unmute twice in succession for the same user only yields one message

#3 Each screenshot should be clearly captioned



unmute now and everyone can send messages

Checklist Items (3)

#1 Show examples of doing /mute twice in succession for the same user only yields one message

#2 Show examples of doing /unmute twice in succession for the same user only yields one message

#3 Each screenshot should be clearly captioned

COLLAPSE

Task #3 - Points: 1

Text: Explain solution

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Mention how you limit the messages in each scenario
#2	1	Discuss how you find the correct user to send the message to

Response:

For ensuring the message reaches the correct user, I use the target user's identifier obtained during the room search to send feedback directly to both the command issuer and the target user. This feedback includes notifying the target user that they have been muted or unmuted, and confirming the action to the user who issued the command. This system ensures that all parties are immediately aware of the changes in their interaction capabilities within the chat room, maintaining clarity and order in user communications.

Demonstrate user list visual changes (2.25 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the code

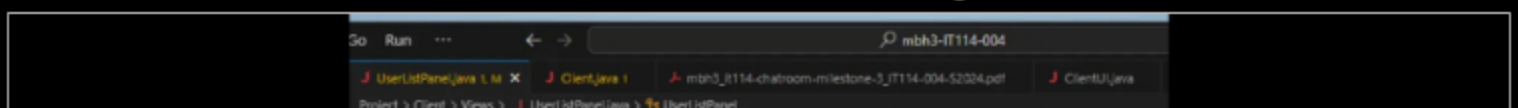
Checklist *The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```java
public class UserListPanel extends JPanel {
    protected void addUserListItem(long clientId, String clientName) {
    }

    //msh3
    //04/24/24
    //background color, graying out user,

    protected void updateUserListItem(long clientId) {
        logger.log(Level.INFO, "Updating display color for user with ID: " + clientId);

        Component[] cs = userListArea.getComponents();

        for (Component c : cs) {
            boolean isUser = c.getName().equals(clientId + "");

            ((JEditorPane) c).setForeground((isUser ? Color.BLUE : Color.black));
        }
    }

    public void muteUser(long userId) {
        logger.log(Level.INFO, "Muting user list item for id " + userId);
        Component[] components = userListArea.getComponents();
        for (Component component : components) {
            if (component.getName().equals(Long.toString(userId))) {
                component.setForeground(Color.GRAY);
                break;
            }
        }
        userListArea.revalidate();
        userListArea.repaint();
    }

    public void unmuteUser(long userId) {
        logger.log(Level.INFO, "Unmuting user list item for id " + userId);
        Component[] components = userListArea.getComponents();
        for (Component component : components) {
            if (component.getName().equals(Long.toString(userId))) {
                component.setForeground(Color.BLACK);
                break;
            }
        }
        userListArea.revalidate();
        userListArea.repaint();
    }
```

code snipped for background color and but graying out mute user didn't work

Checklist Items (4)

#1 Show the code related to "graying out" muted users and returning them to normal when unmuted

#2 Show the code related to highlighting the user who last sent a message (and unhighlighting the remainder of the list)

#3 Screenshots should include ucid and date comment

#4 Each screenshot should be clearly captioned

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the demo

Checklist

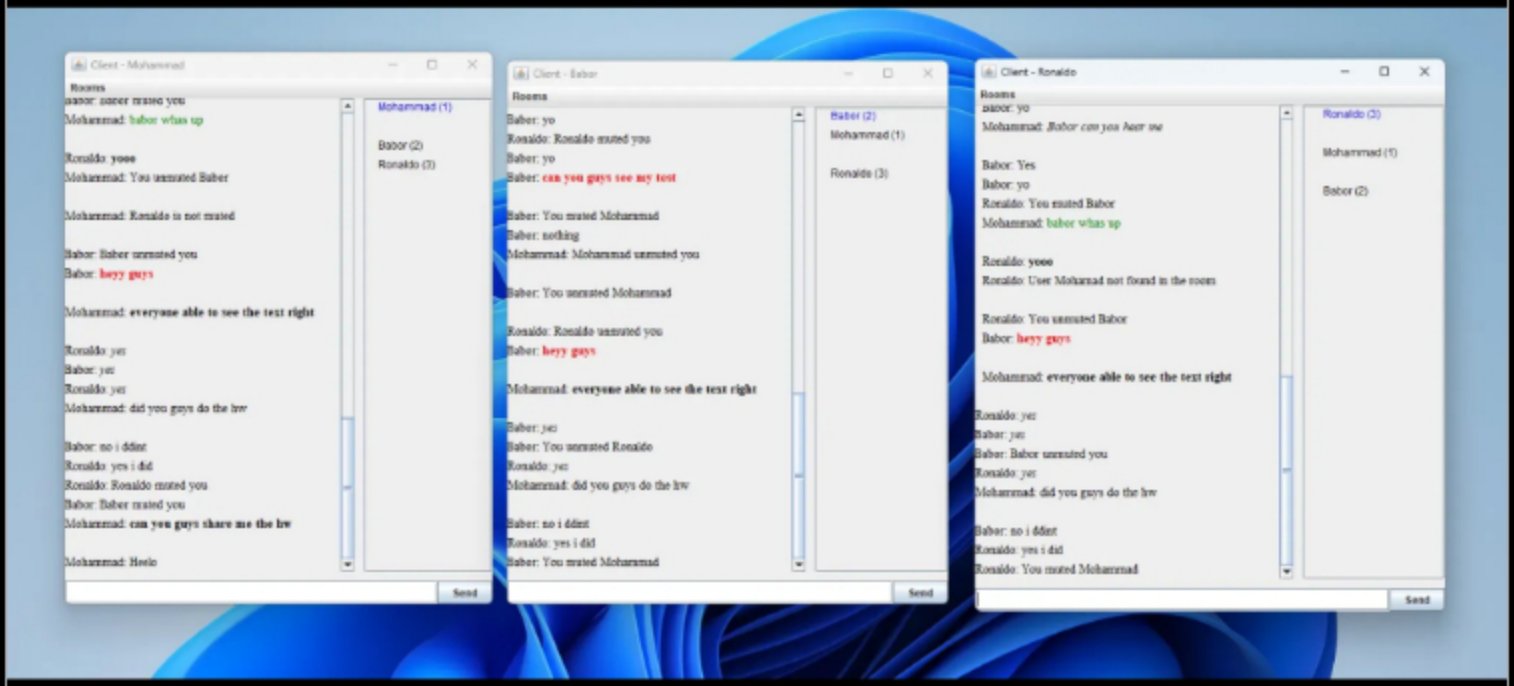
*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Show before and after screenshots of the list updating upon mute and unmute
☐ #2	1	Capture variations of "last person to send a message gets highlighted"
☐ #3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large



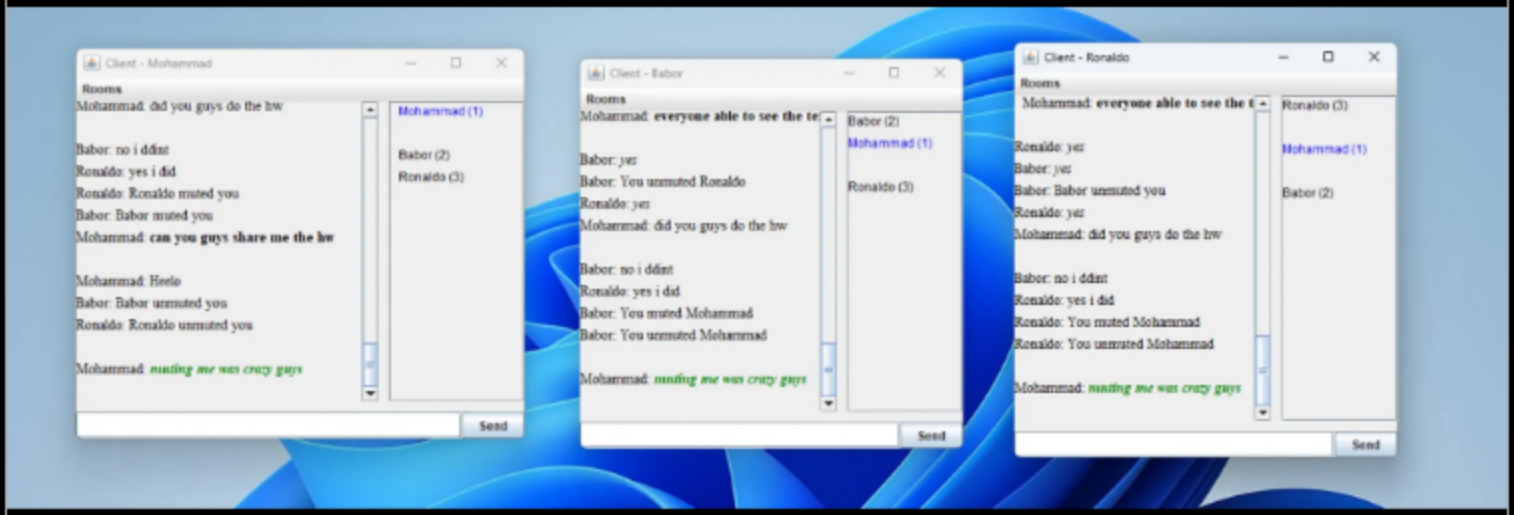
screenshot

Checklist Items (3)

#1 Show before and after screenshots of the list updating upon mute and unmute

#2 Capture variations of “last person to send a message gets highlighted”

#3 Each screenshot should be clearly captioned



as you can see last person who sent the message was Mohammad where he's highlighted in Blue

Checklist Items (3)

#1 Show before and after screenshots of the list updating upon mute and unmute

#2 Capture variations of "last person to send a message gets highlighted"

#3 Each screenshot should be clearly captioned

COLLAPSE

Task #3 - Points: 1

Text: Explain solution

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Mention how you got the mute/unmute effect implemented
☐ #2	1	Mentioned how you got the highlight effect implemented (including unhighlighting the other users)

Response:

For highlighting a specific user, I use the updateUserListItem method, where the user's name is highlighted in blue to indicate they are the current speaker or the focus of attention. When updating, I check each user ID in the list, if it matches the one I'm updating, I set their color to blue. If not, I ensure their color is set to black to unhighlight them. This keeps only the relevant user highlighted at any time, making it easy to visually track who is currently active or being discussed in the chat.

For muting and unmuting color, when a user is muted, I use the muteUser method to change the text color of their name to gray in the user list, indicating they are muted. This change is achieved by iterating through the components in the user list area, checking for the matching user ID, and setting the foreground color of the respective JEditorPane to gray. Similarly, when unmuting, I revert the color back to black using the unmuteUser method, following the same process but reversing the color change.

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/Mohammadh222/mbh3-IT114-004/compare/Milestone4?expand=1

COLLAPSE

Task #2 - Points: 1
Text: Talk about any issues or learnings during this assignment

Response:

I faced lost of issues, particularly with the muting and unmuting functionality. Initially, I struggled to correctly apply the mute and unmute effects. The main issue was ensuring that the GUI accurately reflected the mute status of each user. I had problems with the code where the background color of the user list items wasn't updating consistently. Sometimes, when I tried to mute a user, their name wouldn't turn gray as expected, and didn't really work. Navigating the structure of the code was difficult at times. had to spend a lots of amount of time understanding the flow of data and the interaction between components to ensure that actions like muting and unmuting were reflected correctly across the user interface. Through this, I learned a lot about the importance of maintaining clear and modular code to simplify troubleshooting and updates in a multi-component environment. It was heck of work and very difficult.

COLLAPSE

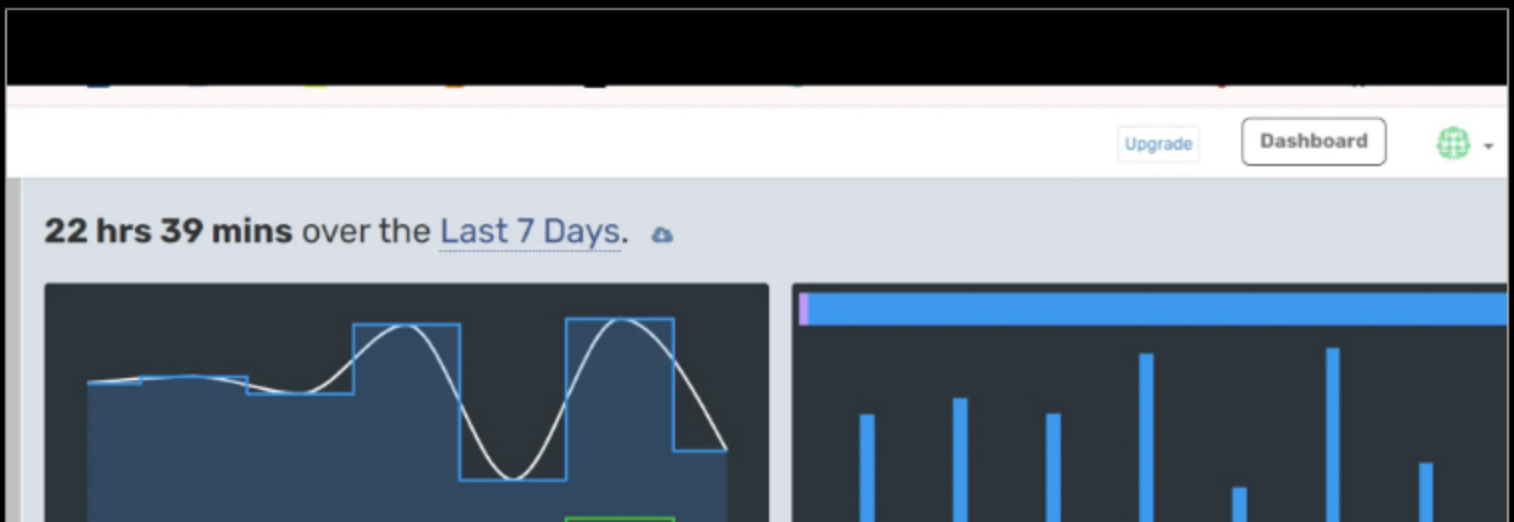
Task #3 - Points: 1
Text: WakaTime Screenshot

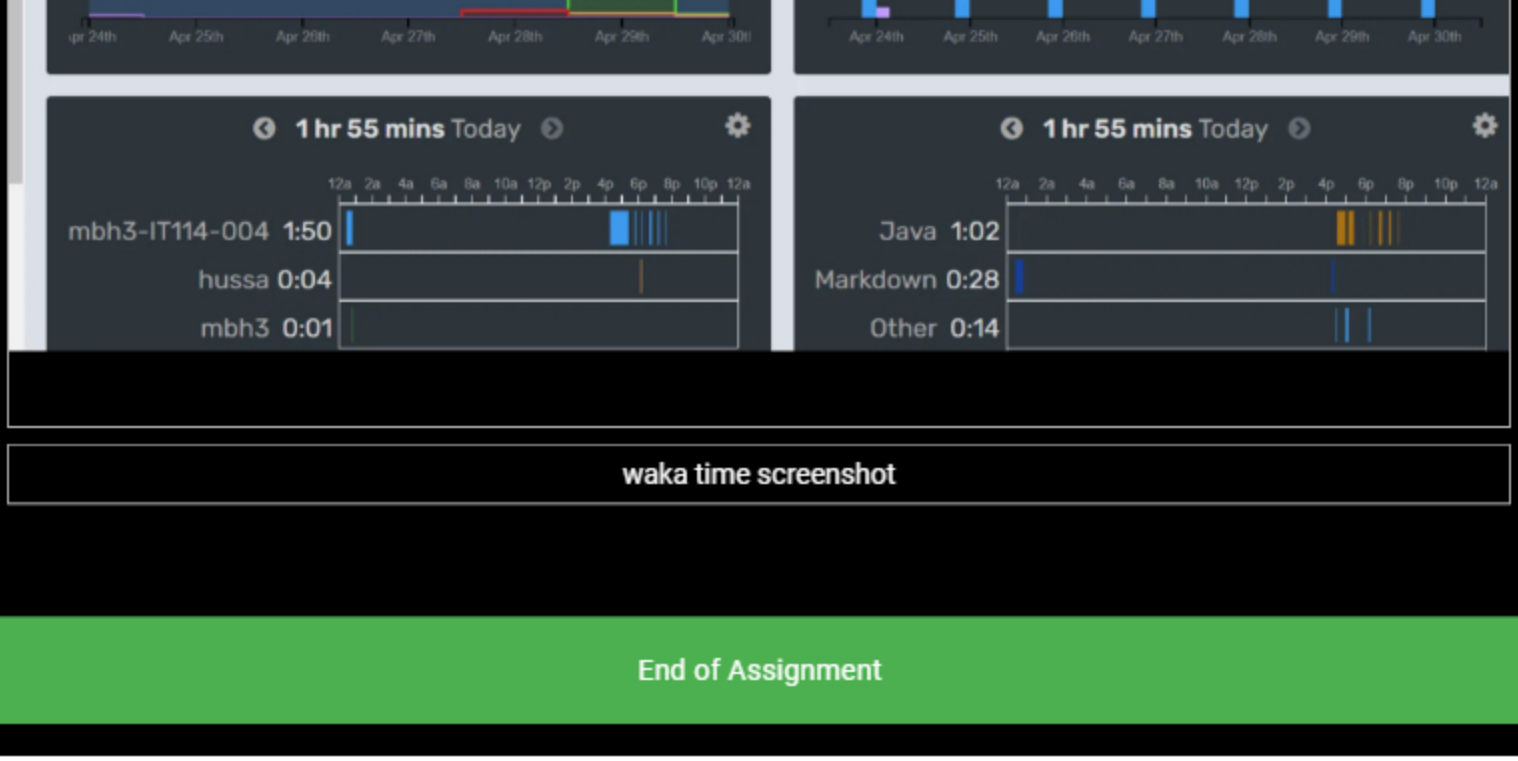
Details:
Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

Task Screenshots:

Gallery Style: Large View

Small Medium Large





waka time screenshot

End of Assignment