Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-002-S2024/it114-chatroom-milestone-4-2024/grade/ns87

IT114-002-S2024 - [IT114] Chatroom Milestone 4 2024

Submissions:

Submission Selection

1 Submission [active] 4/26/2024 1:12:36 PM

Instructions

COLLAPSE

Implement the Milestone 4 features from the project's proposal document: https://docs.google.com/document/d/1QNmvEvel97GTFPGfVwwQC96xSsobbSbk56145X
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

```
        //NS87 - Date 4-30-2024

public void chatExport(){
        Component[] chathis = chatArea.getComponents();
        SimpleDateFormat sdf = new SimpleDateFormat(pattern:"yyyyMMddHHmmss");
        String timestamp = sdf.format(new Date());
        try (FileWriter chatfile = new FileWriter("chathistory_" + timestamp + ".html")) {
            for (Component i : chathis) {
                String message = ((JEditorPane) i).getText();
                chatfile.write("<br>" + message + "</br>");
            }
            Client.INSTANCE.sendMessage(message:"Export successful");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

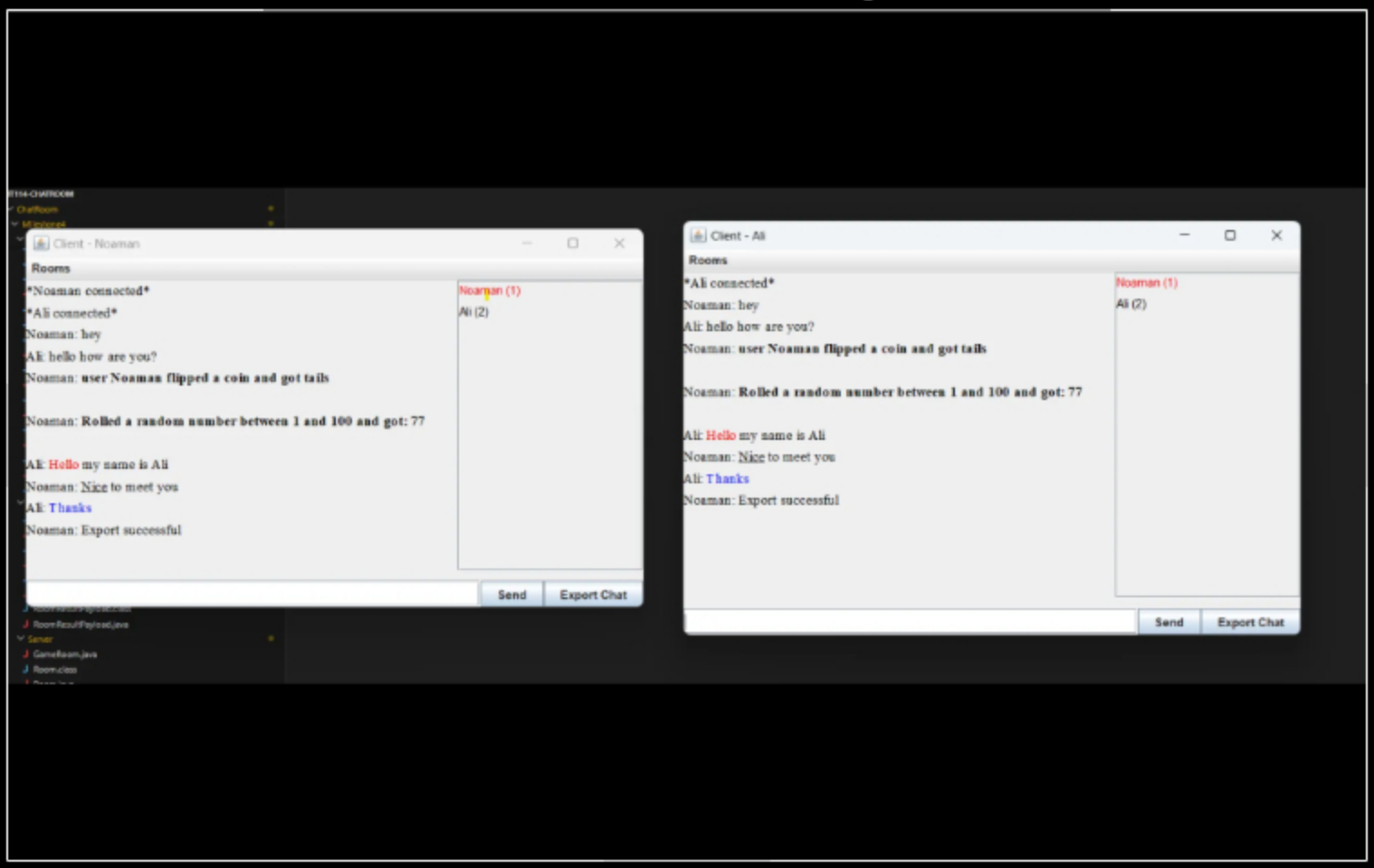
*The checkboxes are for your own tracking

#	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



Formatting/Roll/Flip

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:

The Java program export chat messages from a chat window to an HTML file. When user clicks the "Export Chat" button, the program saves all messages displayed in the chat window to an HTML file. Each export gets a unique timestamp in its filename to avoid overwriting previous exports. The exported HTML files are stored in the same folder as the program.

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

```
    //NS87-4-25-2024
void processPayload(Payload p) {
    switch (p.getPayloadType()) {
        case CONNECT:
            setClientName(p.getClientName());
            mutePersistList = "C://Users//Public//IT114-ChatRoom//ChatRoom/" + p.getClientName() + ".txt";
            loadMuteList();
            break;
        case DISCONNECT:
            Room.disconnectClient(this, getCurrentRoom());
            break;
        case MESSAGE:
```

```java
            if (currentRoom != null) {
                currentRoom.sendMessage(this, p.getMessage());
            } else {
                // TODO migrate to lobby
```

code that loads the mute list when a ServerThread is connected

Checklist Items (3)

#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

```java
    }
    //Methods that create/update a text file containing the muted usernames and uses the same file upon server start-up respectively
    //NS87-4-30-2024
    public void updateMuteList() {
        try(PrintWriter writer = new PrintWriter(new FileWriter(mutePersistList))) {
            for(String mutedUser : muteList) {
                writer.println(mutedUser);
            }
            writer.flush();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void loadMuteList() {
        try(BufferedReader reader = new BufferedReader(new FileReader(mutePersistList))) {
            String line;
            while ((line = reader.readLine()) != null) {
                muteList.add(line);
            }
```

Code that saves the mute list to a file with the name of the user it belongs to

Checklist Items (0)

^COLLAPSE^

Task #2 - Points: 1

Text: Screenshots of the demo

Checklist

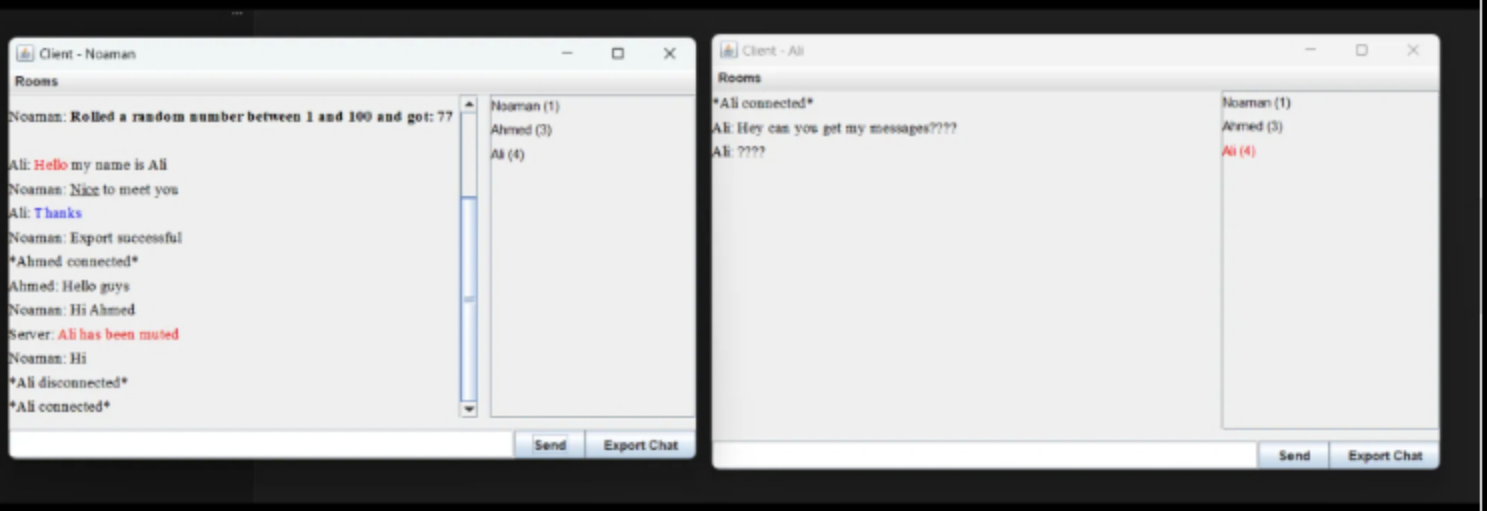
*The checkboxes are for your own tracking

#	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



Noaman mutes Ali. Ali is not able to message Noaman even after connecting again.

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:

To save the mute list, we use a method called updateMuteList(). It writes each muted user's name to a file using a PrintWriter.
When we want to load the mute list, we use loadMuteList().
This method reads each line from the file using a BufferedReader and adds the muted users to our muteList.
When a client connects to the server (CONNECT case), the server sets the client's name and then creates a file path for that client's mute list based on their name. After that, it loads the mute list for that specific client using loadMuteList().

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

```
////N587-4-30-2024
case MUTE:
    if (!isRedundantMute(p.getClientName())) {
        // Process the mute action
        muteList.add(p.getClientName());
        updateMuteList();
        sendMuteUser(p.getClientName());
        Room mroom = getCurrentRoom();
        if (mroom != null) {
            ServerThread mutedUser = mroom.findMute(p.getClientName());
            mutedUser.sendMessage(p.getClientId(), "<font color=\"red\">You have been muted by " + getClientName() + "</font>");
        }
        // Add current timestamp to lastMuteTimestamps map
        lastMuteTimestamps.put(p.getClientName(), System.currentTimeMillis());
    }
    break;
case UNMUTE:
    if (!isRedundantUnmute(p.getClientName())) {
        // Process the unmute action
        muteList.remove(p.getClientName());
        updateMuteList();
```

```java
        updateMuteList();
        sendUnmuteUser(p.getClientName());
        Room mroom = getCurrentRoom();
        if (mroom != null) {
            ServerThread mutedUser = mroom.findMute(p.getClientName());
            mutedUser.sendMessage(p.getClientId(), "<font color=\"red\" >You have been unmuted by " + getClientName() + "</font>");
            // Inform the client who muted that the user has been unmuted
             ServerThread mutingClient = mroom.findClient(getClientName());
             lastUnmuteTimestamps.put(p.getClientName(), System.currentTimeMillis());
            // Check if the client who muted is not the same as the unmuted client
             if (!p.getClientName().equals(getClientName())) {
    }
```

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)

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

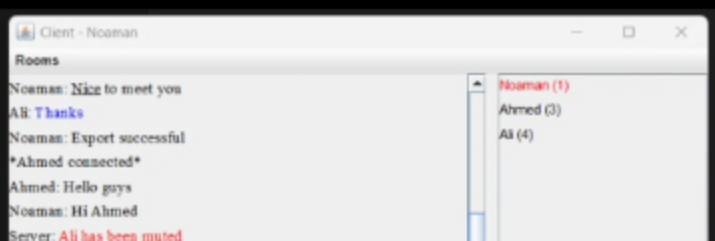
*The checkboxes are for your own tracking

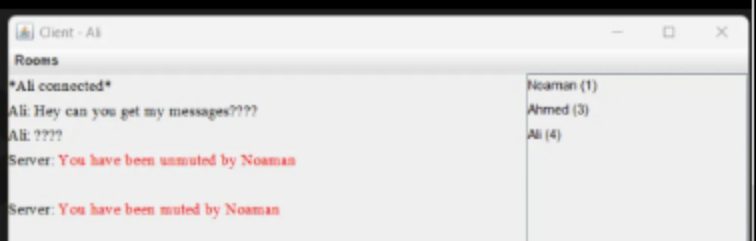
#	Points	Details
#1	1	Show examples of doing /mute twice in succession for the same user only yields one message
#2	1	Show examples of doing /unmute twice in succession for the same user only yields one message
#3	1	Each screenshot should be clearly captioned

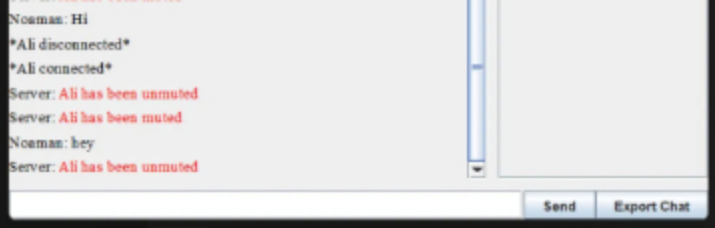
Task Screenshots:

Gallery Style: Large View

Small Medium Large







Both mute and unmute were performed twice. Only one message is displayed

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:

Limiting Messages:
We use timestamps to check when a mute or unmute action was last performed for a user.
If less than 5 seconds have passed since the last action, we don't send another message to avoid duplicates.
Finding the Correct User:
When muting or unmuting, we know the username of the target user (p.getClientName()).
We use this username to find their connection thread (ServerThread) in the room.
This helps us send the mute/unmute message only to the right user.

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
☐ #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
//NS87 - 4-30-2024
// Method to update the style of muted users
protected void updateUserListStyle() {
    Component[] cs = userListArea.getComponents();
    for (Component c : cs) {
        if (c instanceof JEditorPane) {
            JEditorPane textContainer = (JEditorPane) c;
            if (isMuted(Long.parseLong(textContainer.getName()))) {
                textContainer.setForeground(Color.GRAY); // Apply grayed-out style to muted users
            } else {
                textContainer.setForeground(Color.BLACK); // Reset color for non-muted users
            }
        }
    }
}
```

Code related to "graying out" muted users and returning them to normal when unmuted

Checklist Items (3)

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

#3 Screenshots should include ucid and date comment

#4 Each screenshot should be clearly captioned

```java
//NS87 - 4-30-2024
// Method for highlighting the user who last sent a message
public void recentUser(long clientId) {
    updateUserListStyle();
    Component[] cs = userListArea.getComponents();
    for (Component c : cs) {
        if (c.getName().equals(clientId + "")) {
            c.setForeground(Color.RED);
            break;
        } else {
            c.setForeground(Color.BLACK);
        }
    }
}
```

Code related to highlighting the user who last sent a message (and unhighlighting the remainder of the list)

Checklist Items (3)

#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

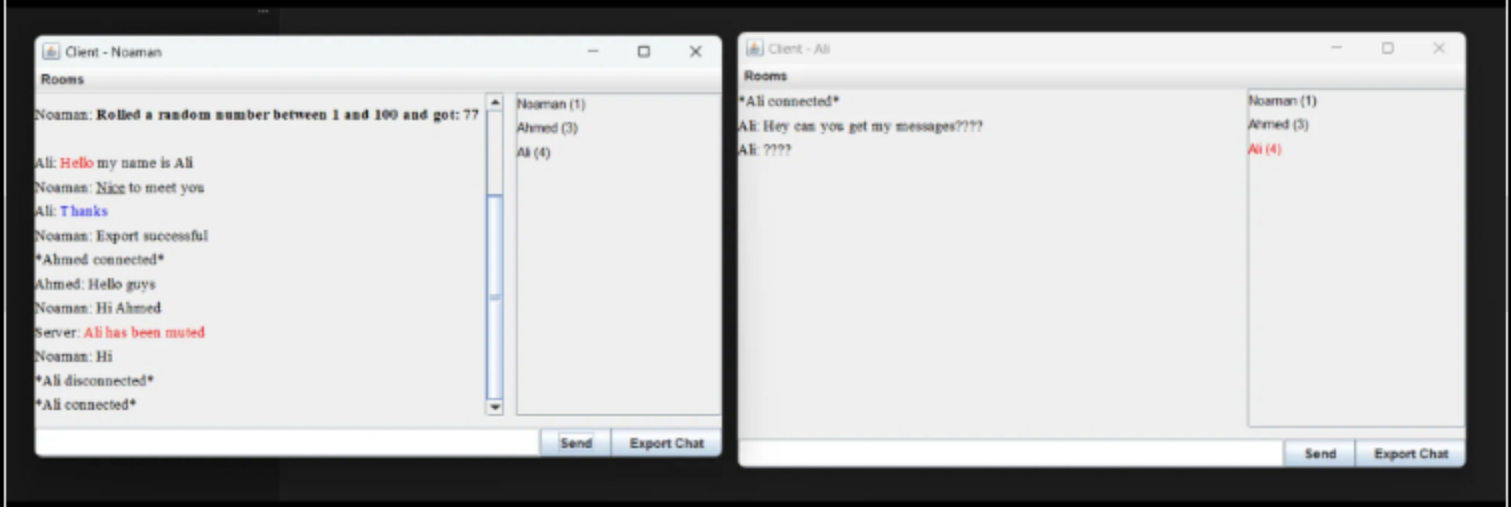
*The checkboxes are for your own tracking

#	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



Noaman has muted Ali and the user Ali has been greyed out from Noaman panel. Even Ali sent the last message

Checklist Items (0)

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:

Mute/Unmute Effect:

First, we created a method called updateUserListStyle() that checks each user in the list.
If isMuted() says a user is muted, we made their text color gray (Color.GRAY); otherwise, their text stays black (Color.BLACK).

Highlight Effect (Including Unhighlighting):

Next, a method called recentUser(long clientId) to highlight the user who last sent a message.
In this method, the style was updated using updateUserListStyle() first to make sure everyone's colors are correct.
Then, we checked each user in the list to see if their ID matches the ID of the user who sent the last message (clientId).
If a user's ID matches, we made their text color red (Color.RED) to highlight them; otherwise, their text color stayed

black to unhighlight them.

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/Noaman4/IT114/pull/4

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

Some issues I faced was understanding how to correctly implement the mute/unmute and greying out effect. Also ensuring that the logic for checking muted users and highlighting the last message sender is accurate and efficient. But overall the project helped me learn in detail about the function especially when applying mute/unmute function.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

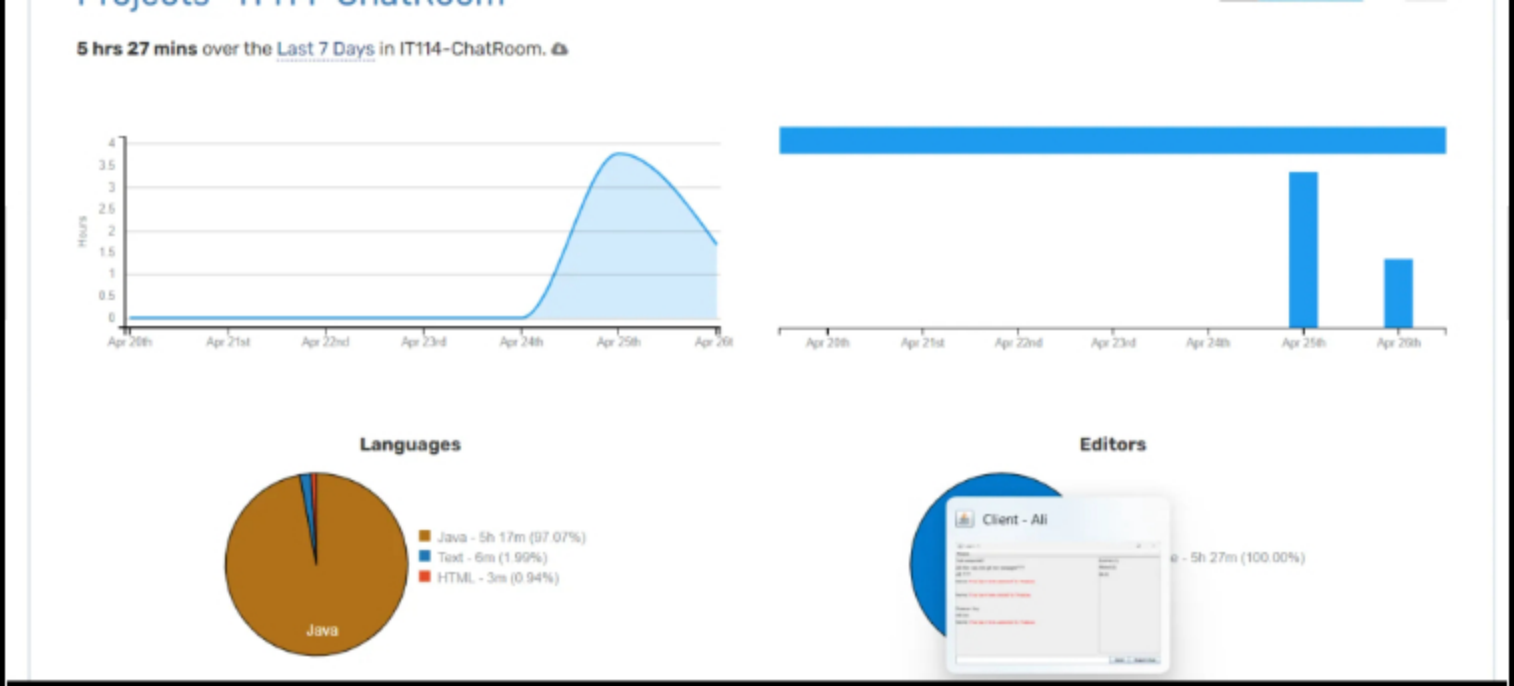
Details:

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



Wakatime

End of Assignment