Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-002-S2024/it114-chatroom-milestone-4-2024/grade/jeo29

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

Submissions:

Submission Selection

1 Submission [active] 4/27/2024 6:43:20 PM

Instructions

COLLAPSE

Implement the Milestone 4 features from the project's proposal document: https://docs.google.com/document/d/1ONmvEvel97GTFPGfVwwQC96xSsobbSbk56145Xi
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

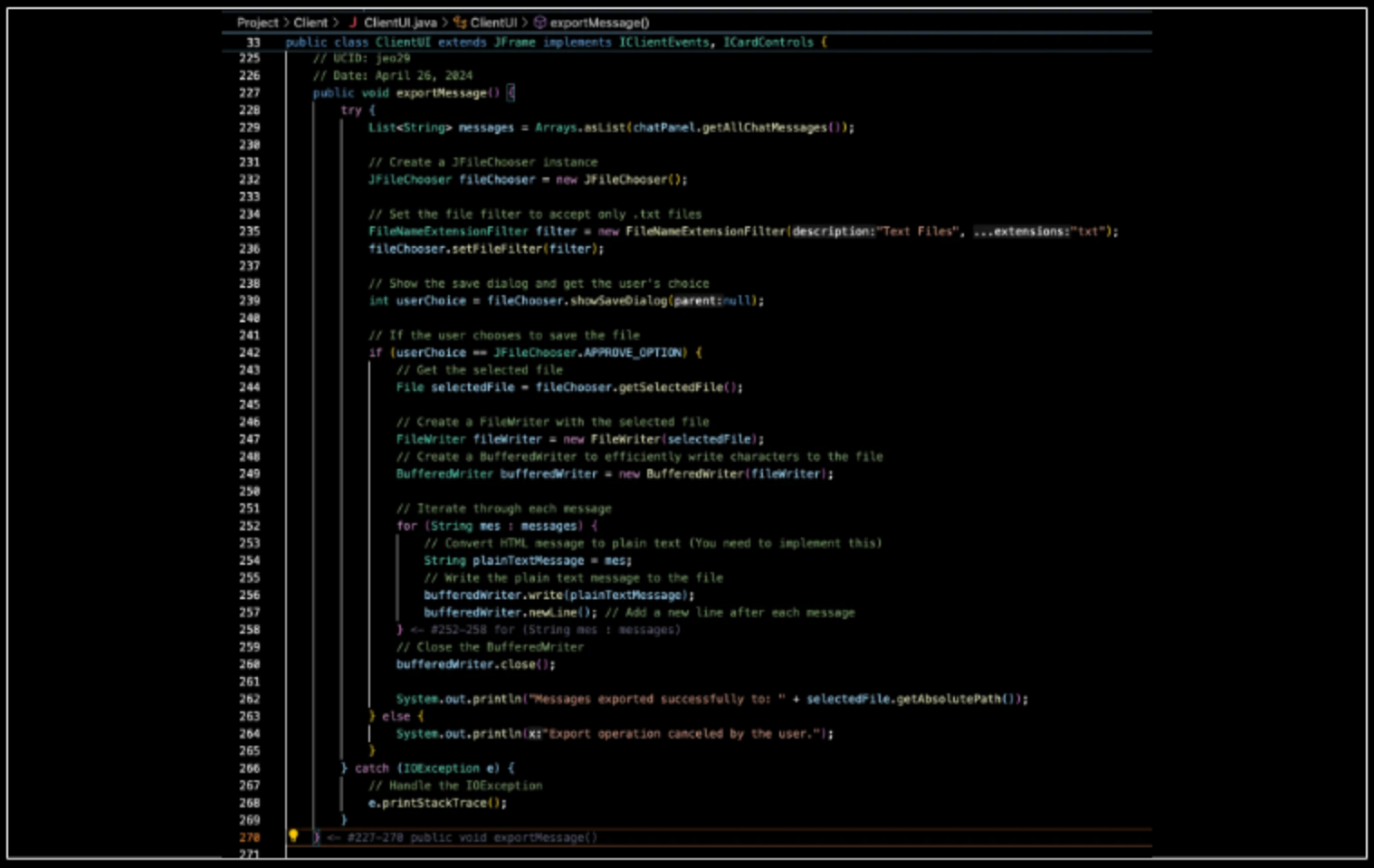
#	Points	Details

#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



```
Project > Client > ClientUI.java > ClientUI > exportMessage()
33  public class ClientUI extends JFrame implements IClientEvents, ICardControls {
225     // UCID: jeo29
226     // Date: April 26, 2024
227     public void exportMessage() {
228         try {
229             List<String> messages = Arrays.asList(chatPanel.getAllChatMessages());
230
231             // Create a JFileChooser instance
232             JFileChooser fileChooser = new JFileChooser();
233
234             // Set the file filter to accept only .txt files
235             FileNameExtensionFilter filter = new FileNameExtensionFilter(description:"Text Files", ...extensions:"txt");
236             fileChooser.setFileFilter(filter);
237
238             // Show the save dialog and get the user's choice
239             int userChoice = fileChooser.showSaveDialog(parent:null);
240
241             // If the user chooses to save the file
242             if (userChoice == JFileChooser.APPROVE_OPTION) {
243                 // Get the selected file
244                 File selectedFile = fileChooser.getSelectedFile();
245
246                 // Create a FileWriter with the selected file
247                 FileWriter fileWriter = new FileWriter(selectedFile);
248                 // Create a BufferedWriter to efficiently write characters to the file
249                 BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
250
251                 // Iterate through each message
252                 for (String mes : messages) {
253                     // Convert HTML message to plain text (You need to implement this)
254                     String plainTextMessage = mes;
255                     // Write the plain text message to the file
256                     bufferedWriter.write(plainTextMessage);
257                     bufferedWriter.newLine(); // Add a new line after each message
258                 } <- #252-258 for (String mes : messages)
259                 // Close the BufferedWriter
260                 bufferedWriter.close();
261
262                 System.out.println("Messages exported successfully to: " + selectedFile.getAbsolutePath());
263             } else {
264                 System.out.println(x:"Export operation canceled by the user.");
265             }
266         } catch (IOException e) {
267             // Handle the IOException
268             e.printStackTrace();
269         }
270     } <- #227-270 public void exportMessage()
271
```

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

Checklist Items (1)

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

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the file

Checklist *The checkboxes are for your own tracking

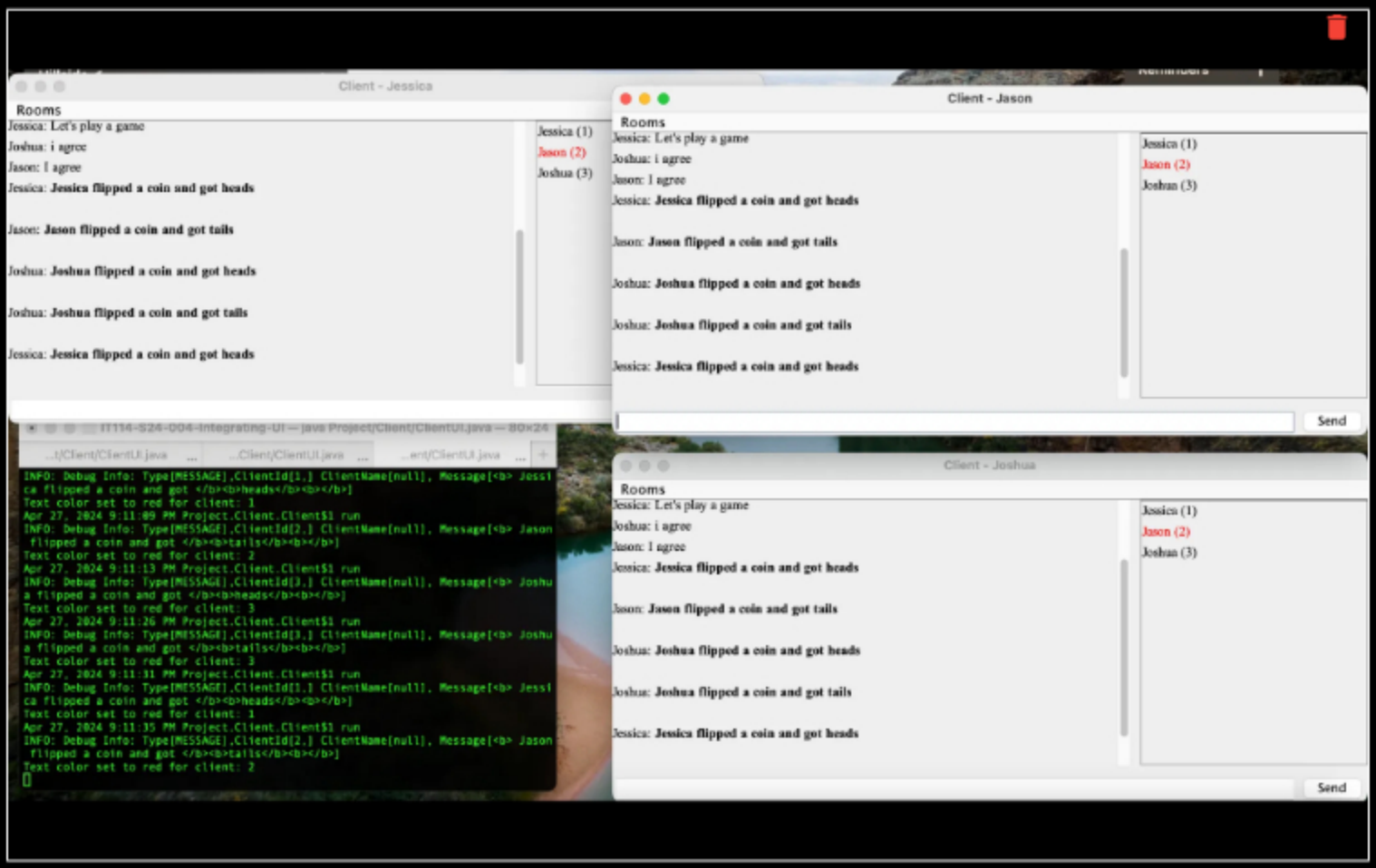
#	Points	Details

#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

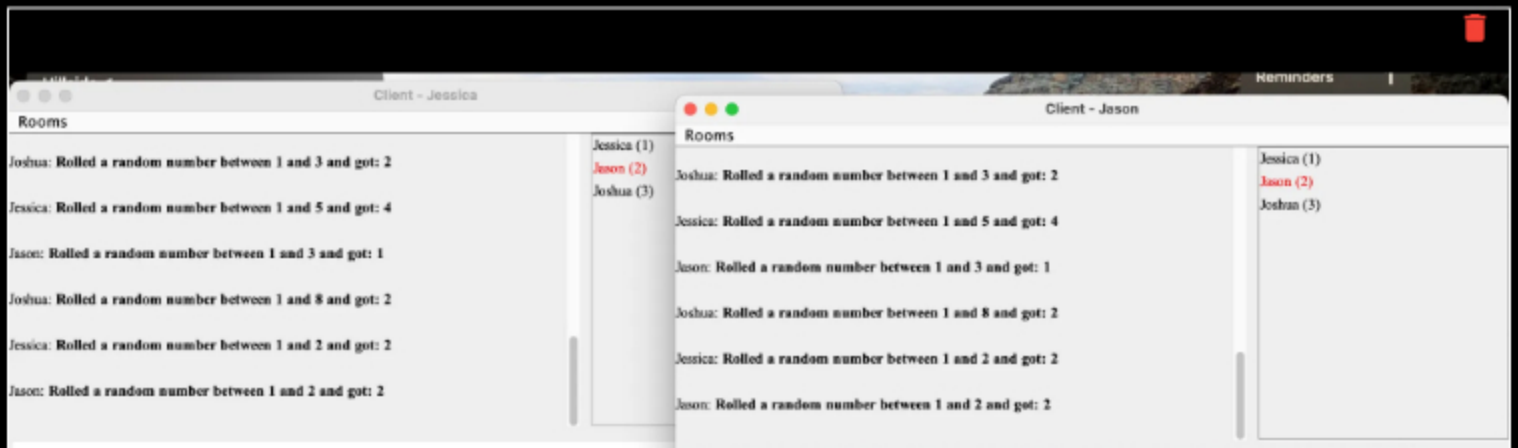
Small Medium Large

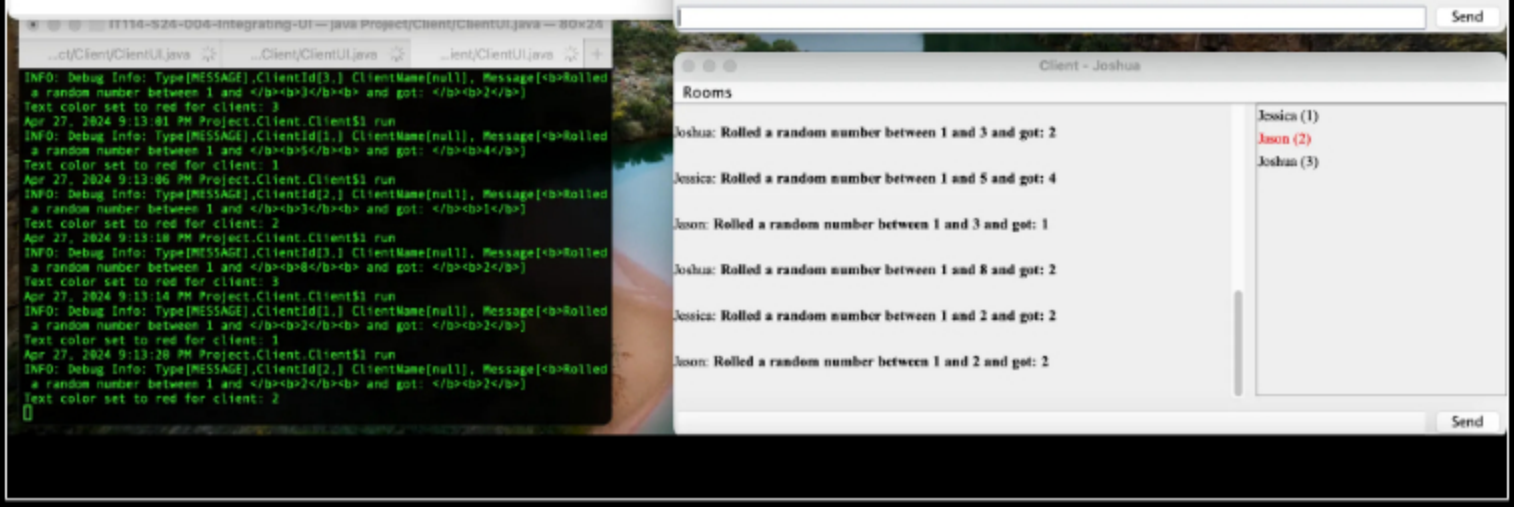


Flip

Checklist Items (1)

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

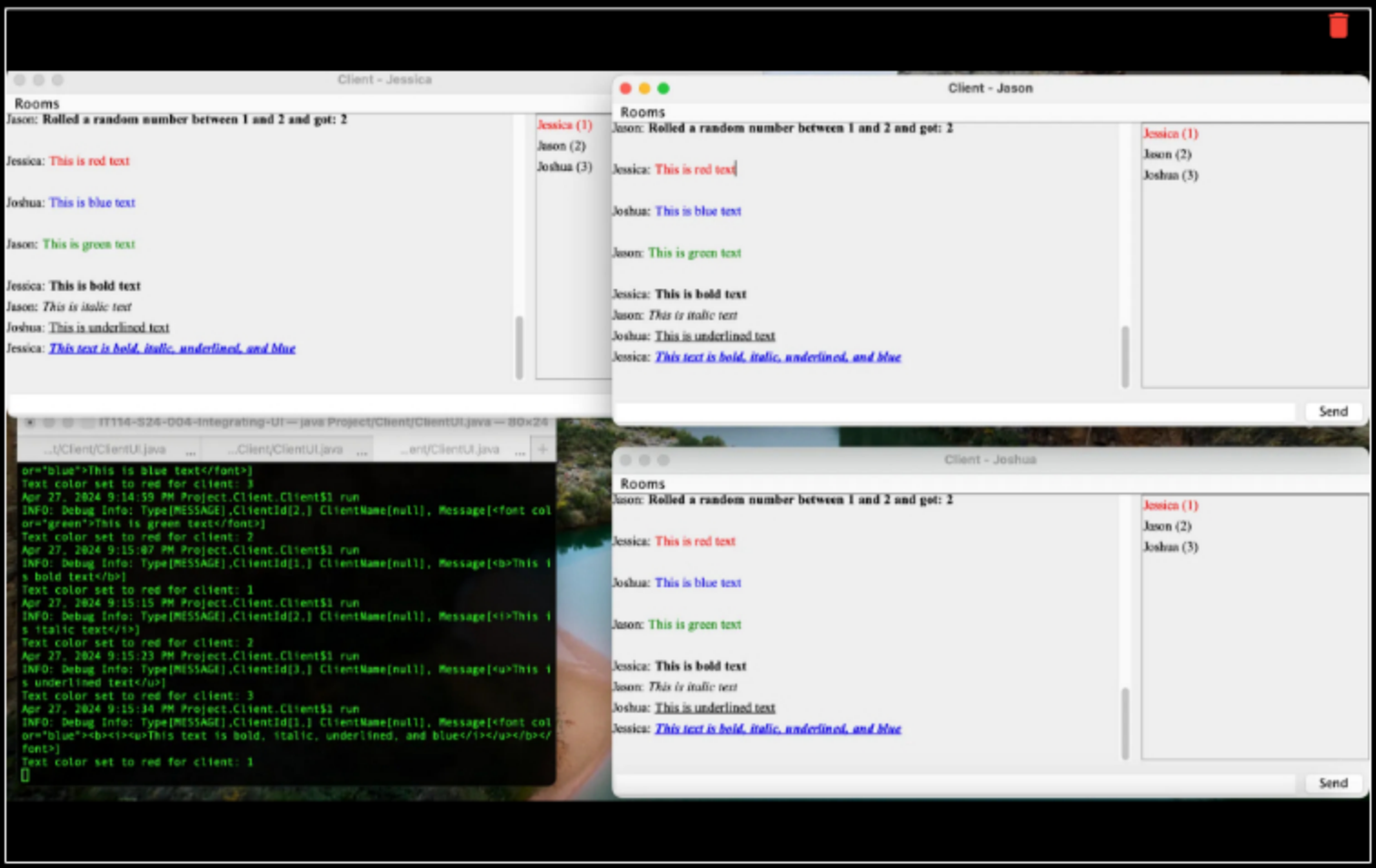




Roll

Checklist Items (1)

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



Formatting

Checklist Items (1)

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

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 messages are stored when a user is approved for download and is given a path. The file is generated and populated when we loop over all messages present in the chat panel and write them in.

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

Project > Server > Room.java > Room > muteUser(String, ServerThread)

```
11  public class Room implements AutoCloseable {
220     // UCID: jeo29
221     // Date: April 26, 2024
222     protected void muteUser(String usernameToMute, ServerThread mutingUser) {
223         if (findUserThreadInRoom(usernameToMute) != null) {
224             System.out.println("Muting user: " + usernameToMute);
225             mutedUsers.add(new MutedUser(usernameToMute, mutingUser.getClientName()));
```

```
226             // Notify the user that they have been muted by another user
227             ServerThread userThread = findUserThreadInRoom(usernameToMute);
228             if (userThread != null) {
229                 userThread.sendMessage(Constants.DEFAULT_CLIENT_ID, mutingUser.getClientName() + " muted you.");
230             }
231         } else {
232             System.out.println("User " + usernameToMute + " not found or not in this room.");
233         }
234     } <- #222-234 protected void muteUser(String usernameToMute, ServerThread m...
235
```

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

Checklist Items (1)

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

Project > Server > J ServerThread.java > ...

```
18  public class ServerThread extends Thread {
29
30  //    //mute/unmute feature
31  //    //UCID: jeo29
32  //    //DATE: April 13, 2024
33      public boolean sendMuteUser(ServerThread mutedUser, ServerThread mutingUser){
34          currentRoom.muteUser(mutedUser.getClientName(),mutingUser);
35          Payload p = new Payload();
36          p.setPayloadType(PayloadType.MUTE);
37          p.setClientName(mutedUser.getClientName());
38          p.setMutedClientID(mutedUser.getClientId());
39          return send(p);
40      } <- #33-40 public boolean sendMuteUser(ServerThread mutedUser, ServerThr...
41
42      // UCID: jeo29
43      // Date: April 26, 2024
44      public boolean sendUnmuteUser(ServerThread unMutedUser, ServerThread mutingUser){
45          currentRoom.unmuteUser(unMutedUser.getClientName());
46          Payload p = new Payload();
47          p.setPayloadType(PayloadType.UNMUTE);
48          p.setClientName(unMutedUser.getClientName());
49          p.setMutedClientID(unMutedUser.getClientId());
50
51          return send(p);
52      } <- #44-52 public boolean sendUnmuteUser(ServerThread unMutedUser, Serve...
53
```

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

Checklist Items (1)

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

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the demo

Checklist

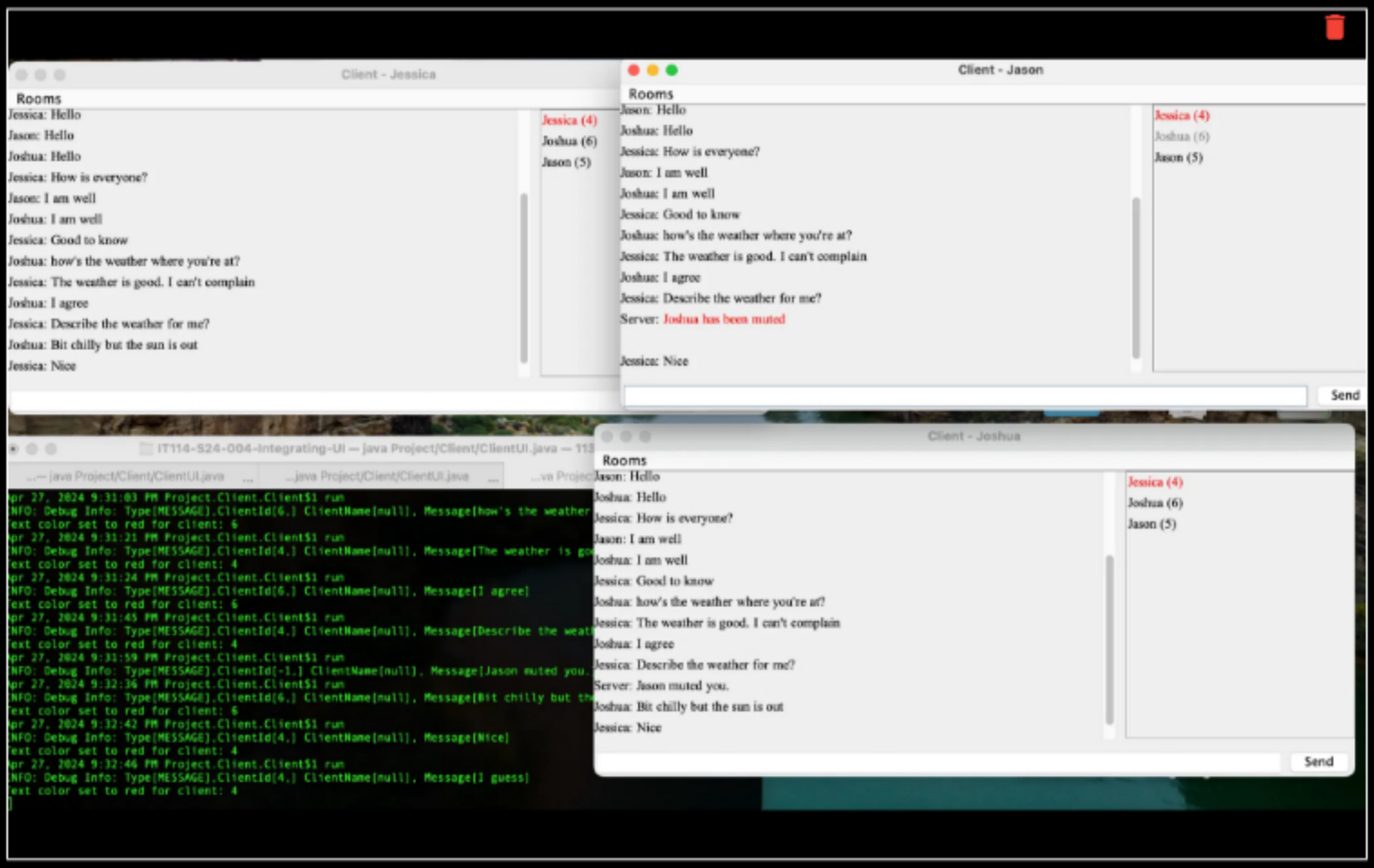
*The checkboxes are for your own tracking

#	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

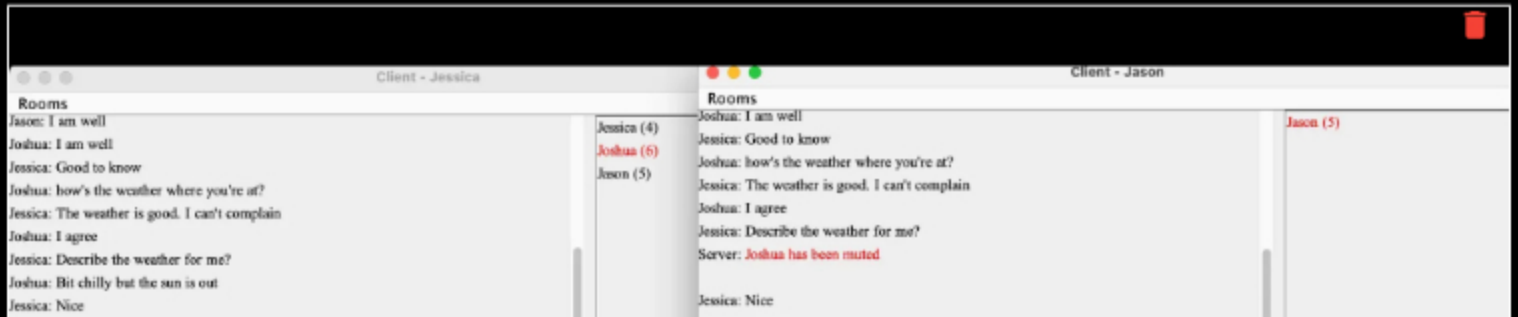
Small Medium Large

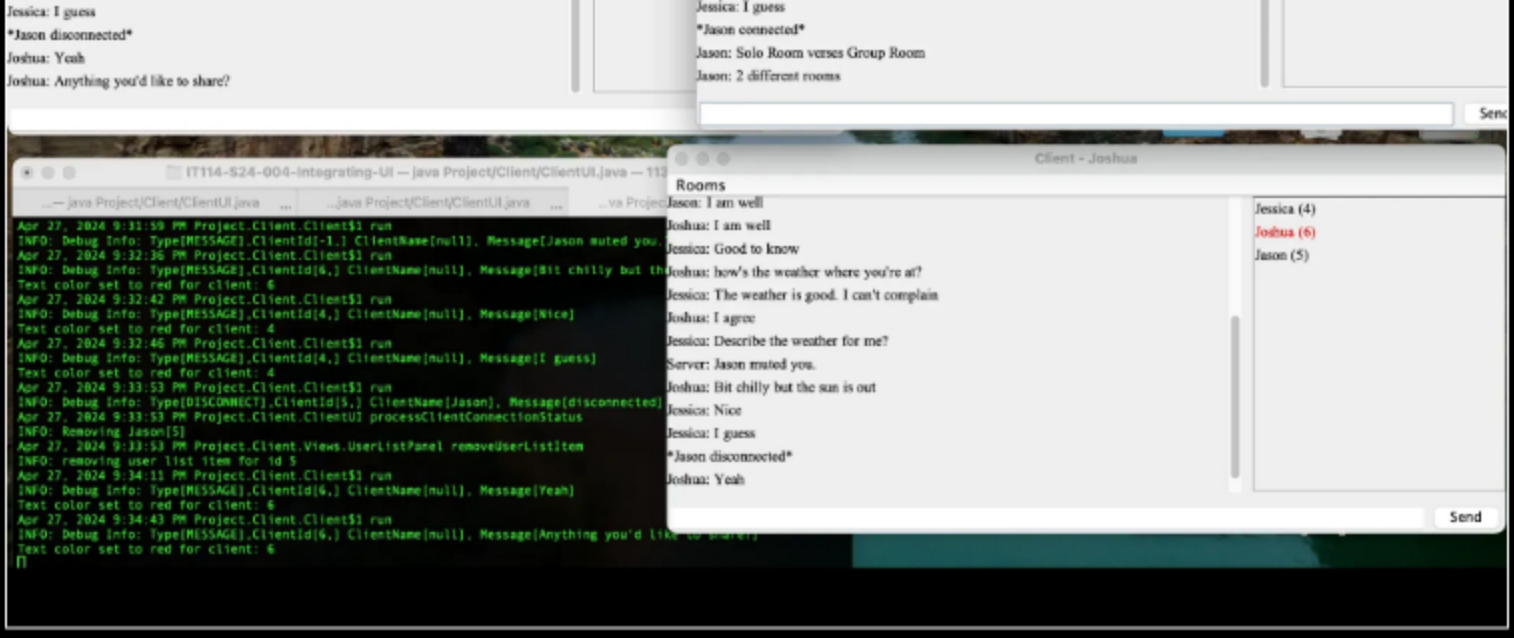


User muted

Checklist Items (1)

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

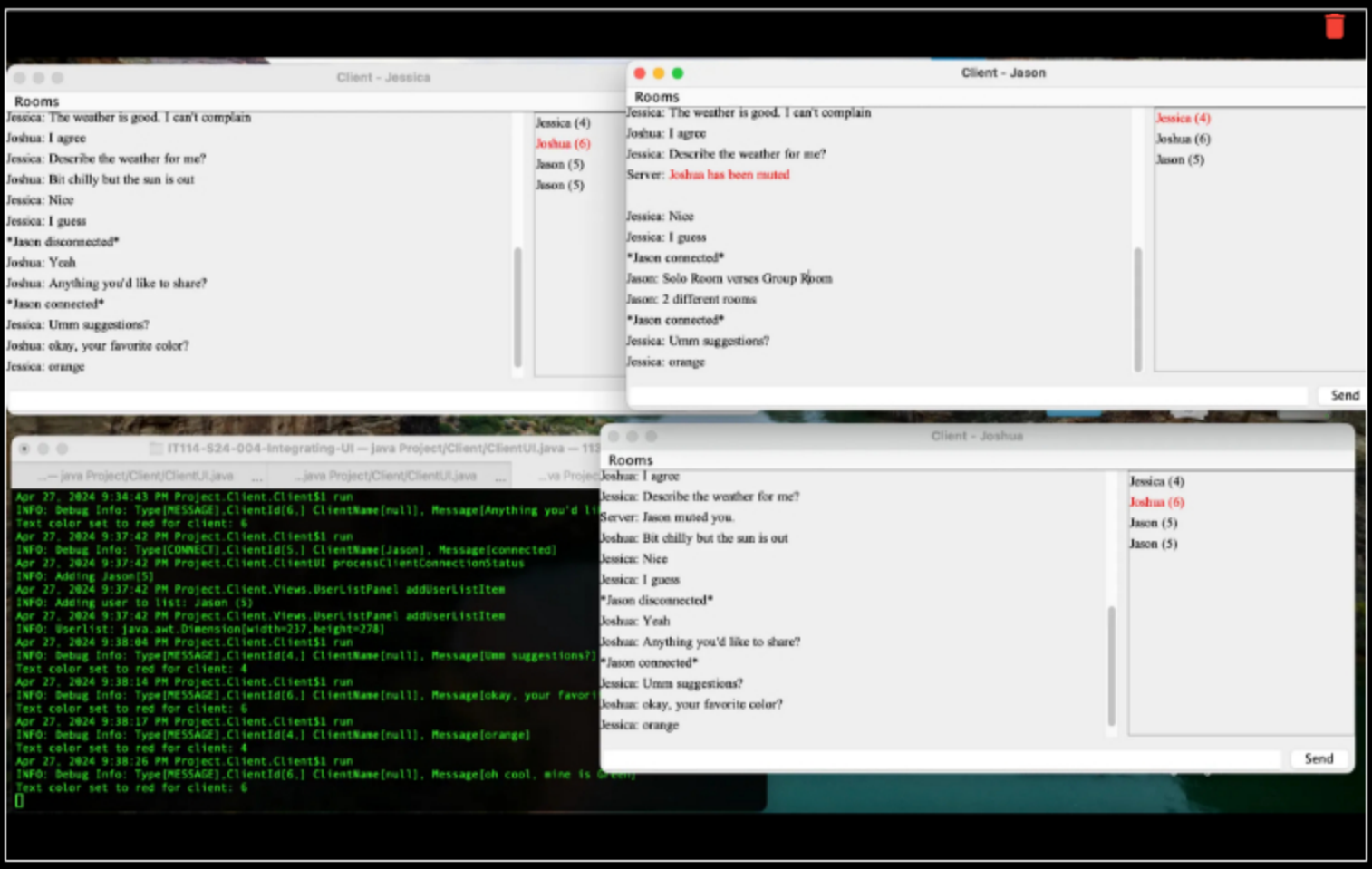




Jason disconnecting into his own room all while still having Joshua muted

Checklist Items (1)

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



Jason reconnecting and still has Joshua muted

Checklist Items (1)

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

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:

The mute list was saved and loaded by being stored in List and passed from Payload to Client UI, and from ClientUI users were grayed out from UI which were muted. Then back to being unmuted because they were no longer grayed out

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
#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

```
31  //    //UCID: jeo29
32  //    //DATE: April 13, 2024
33      public boolean sendMuteUser(ServerThread mutedUser, ServerThread mutingUser){
34          if (currentRoom.isUserMuted(mutedUser.getClientName(),mutingUser.getClientName())) {
35              return false; // Don't send mute notification if user is already muted
36          }
37          currentRoom.muteUser(mutedUser.getClientName(),mutingUser);
38          Payload p = new Payload();
39          p.setPayloadType(PayloadType.MUTE);
40          p.setClientName(mutedUser.getClientName());
41          p.setMutedClientID(mutedUser.getClientId());
42          return send(p);
```

```java
    } <- #33-43 public boolean sendMuteUser(ServerThread mutedUser, ServerThr...

    // UCID: jeo29
    // Date: April 27, 2024
    public boolean sendUnmuteUser(ServerThread unMutedUser, ServerThread mutingUser){
        if (!currentRoom.isUserMuted(unMutedUser.getClientName(),mutingUser.getClientName())) {
            return false; // Don't send mute notification if user is already muted
        }
        currentRoom.unmuteUser(unMutedUser.getClientName());
        Payload p = new Payload();
        p.setPayloadType(PayloadType.UNMUTE);
        p.setClientName(unMutedUser.getClientName());
        p.setMutedClientID(unMutedUser.getClientId());

        return send(p);
    } <- #47-58 public boolean sendUnmuteUser(ServerThread unMutedUser, Serve...
```

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)

Checklist Items (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)

^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshots of the demo

Checklist

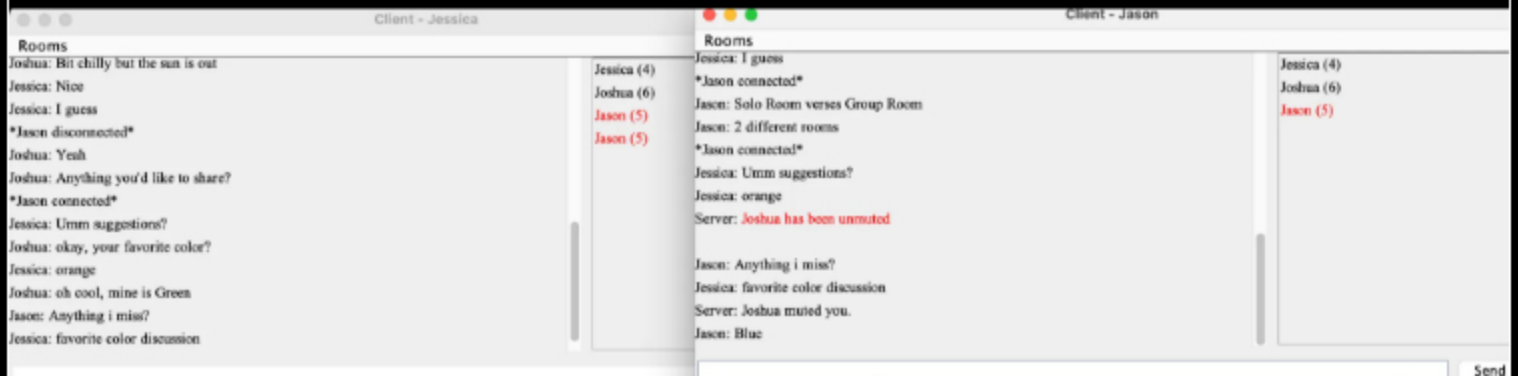
*The checkboxes are for your own tracking

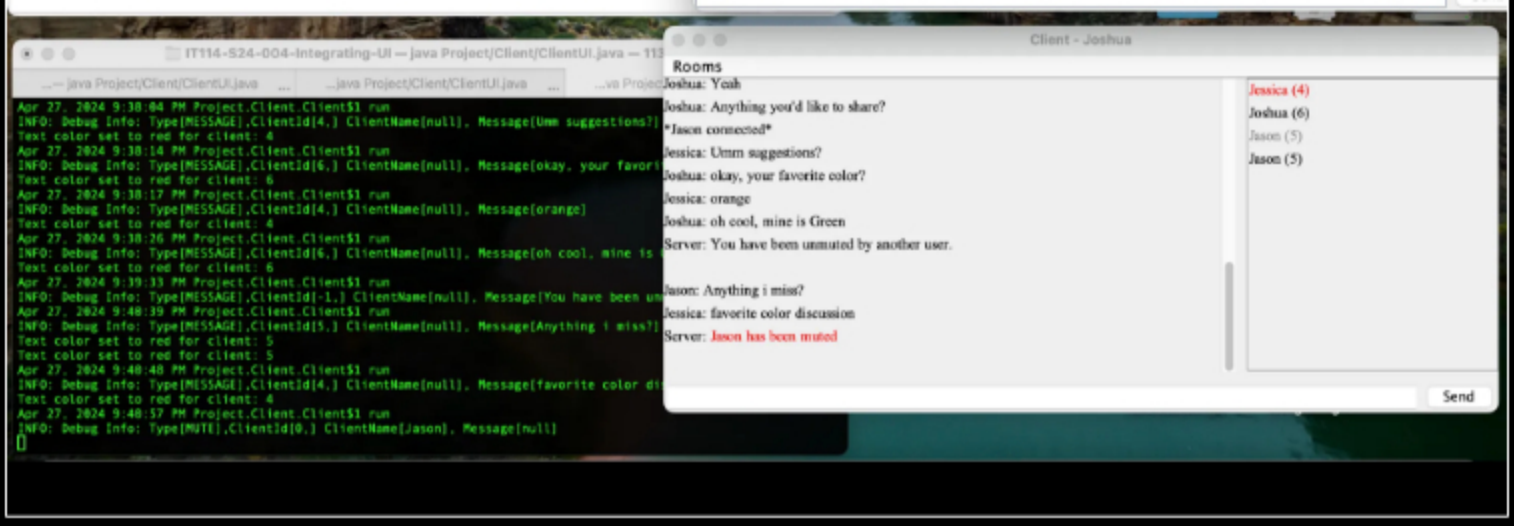
#	Points	Details
#1	1	Show examples of doing /mute twice in succession for the same user only yields one message
#2	1	Show examples of doing /unmute twice in succession for the same user only yields one message
#3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large

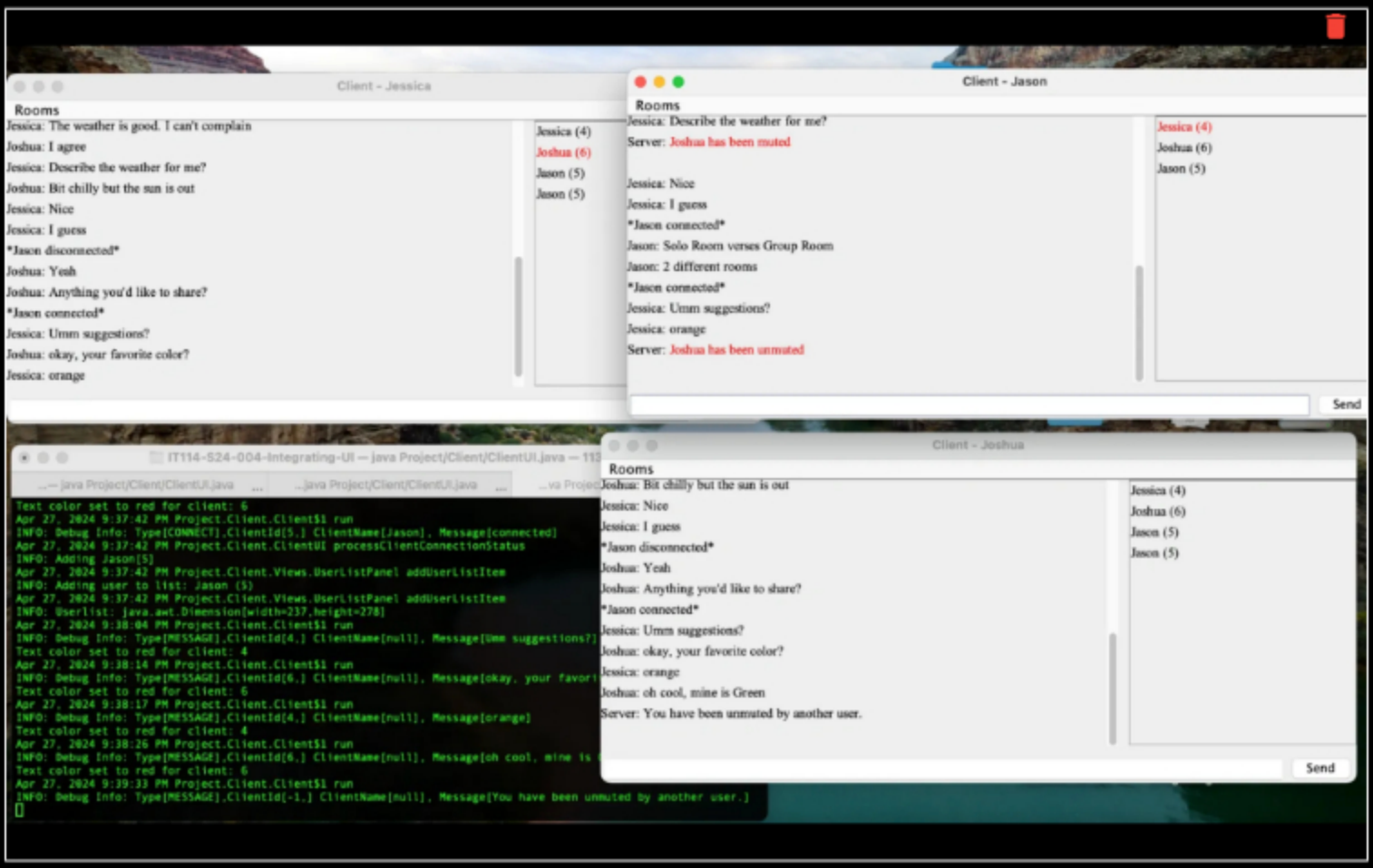




Mute Joshua

Checklist Items (1)

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



/unmute twice in succession for the same user only yields one message

Checklist Items (1)

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

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:

In each scenario, the messages are limited with the help of the mute/unmute feature. You can find the correct user to send a message to with the "@" feature to send a private message.

Demonstrate user list visual changes (2.25 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the code

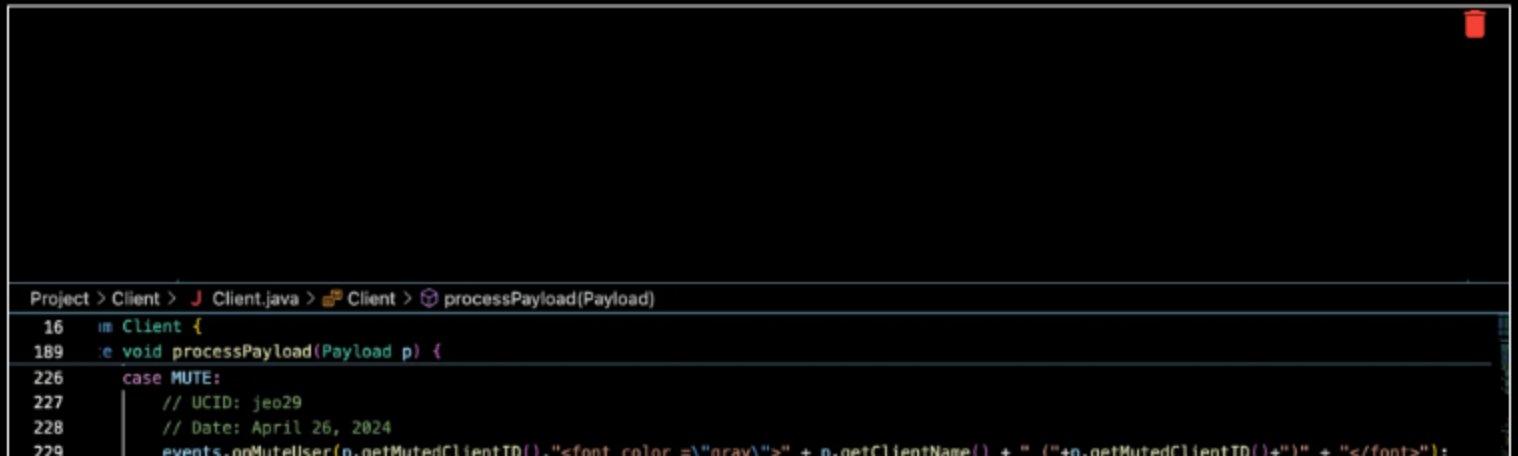
Checklist *The checkboxes are for your own tracking

#	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



```
230         events.onMessageReceive(p.getClientId(), "<font color =\"red\">" + p.getClientName() + " has been muted</font>");
231         break;
232     case UNMUTE:
233         // UCID: jeo29
234         // Date: April 26, 2024
235         events.onUnMuteUser(p.getMutedClientID(),"<font color =\"black\">" + p.getClientName() + " ("+p.getMutedClientID()+")" + "</font>");
236         events.onMessageReceive(p.getClientId(), "<font color =\"red\">" + p.getClientName() + " has been unmuted</font>");
237         break;
```

Show the code related to “graying out” muted users and returning them to normal when unmuted

Checklist Items (1)

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

```
62      // UCID: jeo29
63      // Date: April, 26, 2024
64      protected void recentUser(long clientId) {
65          // Iterate through the components in the userListArea
66          Component[] cs = userListArea.getComponents();
67          for (Component component : cs) {
68              // Check if the component is a JEditorPane and its name matches the clientId
69              if (component.getName().equals(clientId + " ")) {
70                  JEditorPane namePane = (JEditorPane) component;
71                  // Update the HTML content to change text color to red
72                  String htmlContent = namePane.getText().replace(target:"color=\"black\"", "color=\"" + "red" + "\"");
73                  namePane.setText(htmlContent);
74                  namePane.setName(clientId + " ");
75                  System.out.println("Text color set to red for client: " + clientId);
76              } <- #69-76 if (component.getName().equals(clientId + " "))
77              else {
78                  JEditorPane namePane = (JEditorPane) component;
79                  // Update the HTML content to change text color to black
80                  String htmlContent = namePane.getText().replace(target:"color=\"red\"", "color=\"" + "black" + "\"");
81                  namePane.setText(htmlContent);
82              } <- #77-82 else
83          } <- #67-83 for (Component component : cs)
84          userListArea.revalidate();
85          userListArea.repaint();
86      } <- #64-86 protected void recentUser(long clientId)
```

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

Checklist Items (1)

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

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the demo

Checklist

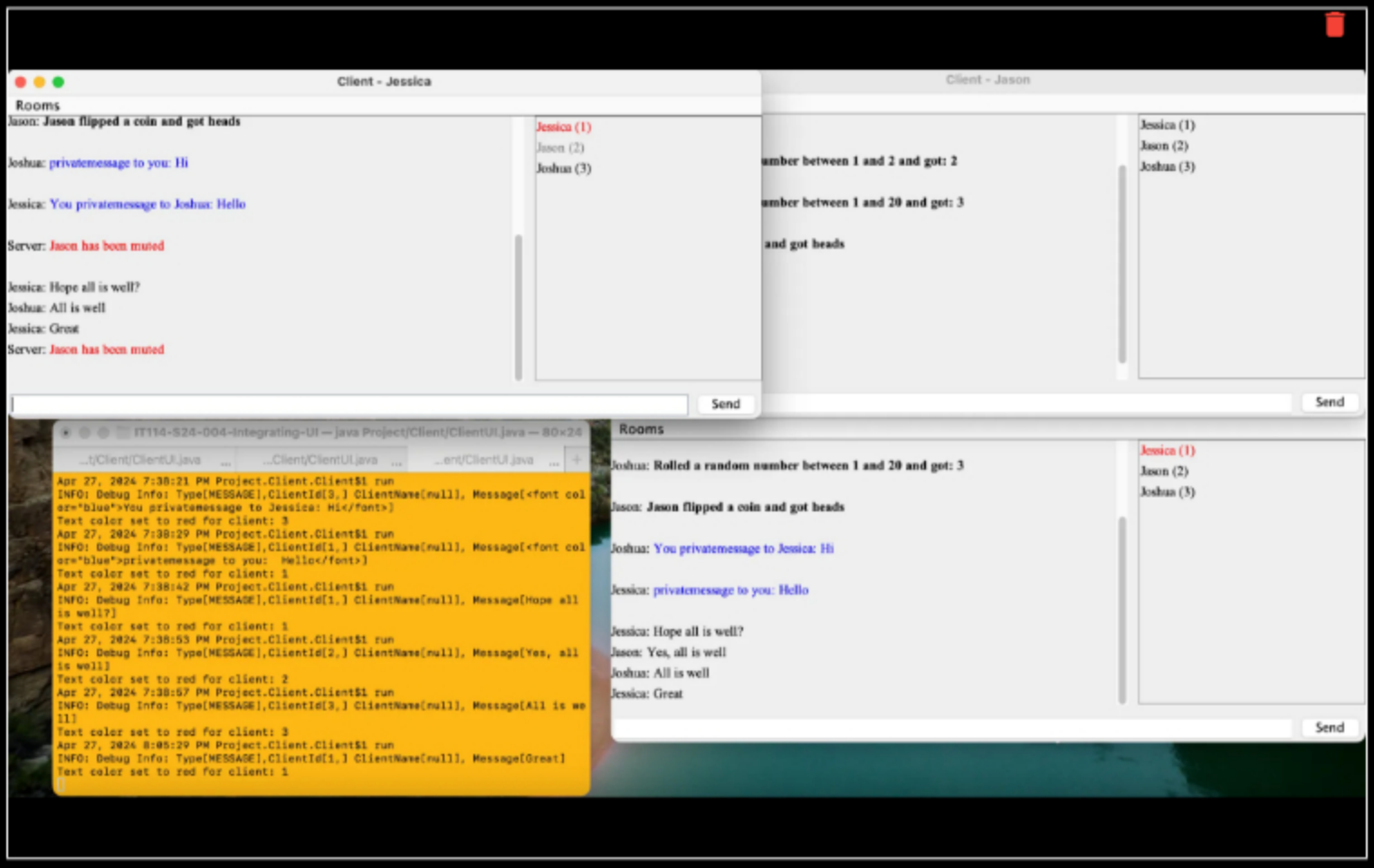
*The checkboxes are for your own tracking

#	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

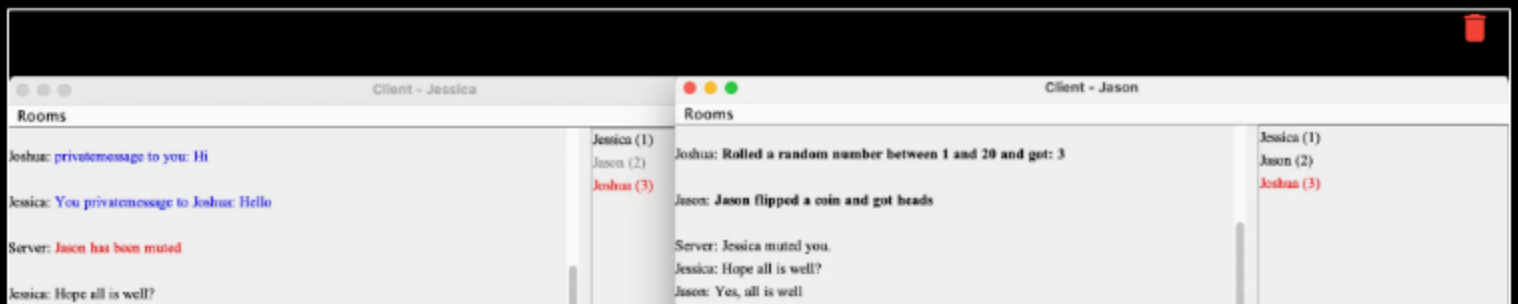
Small Medium Large

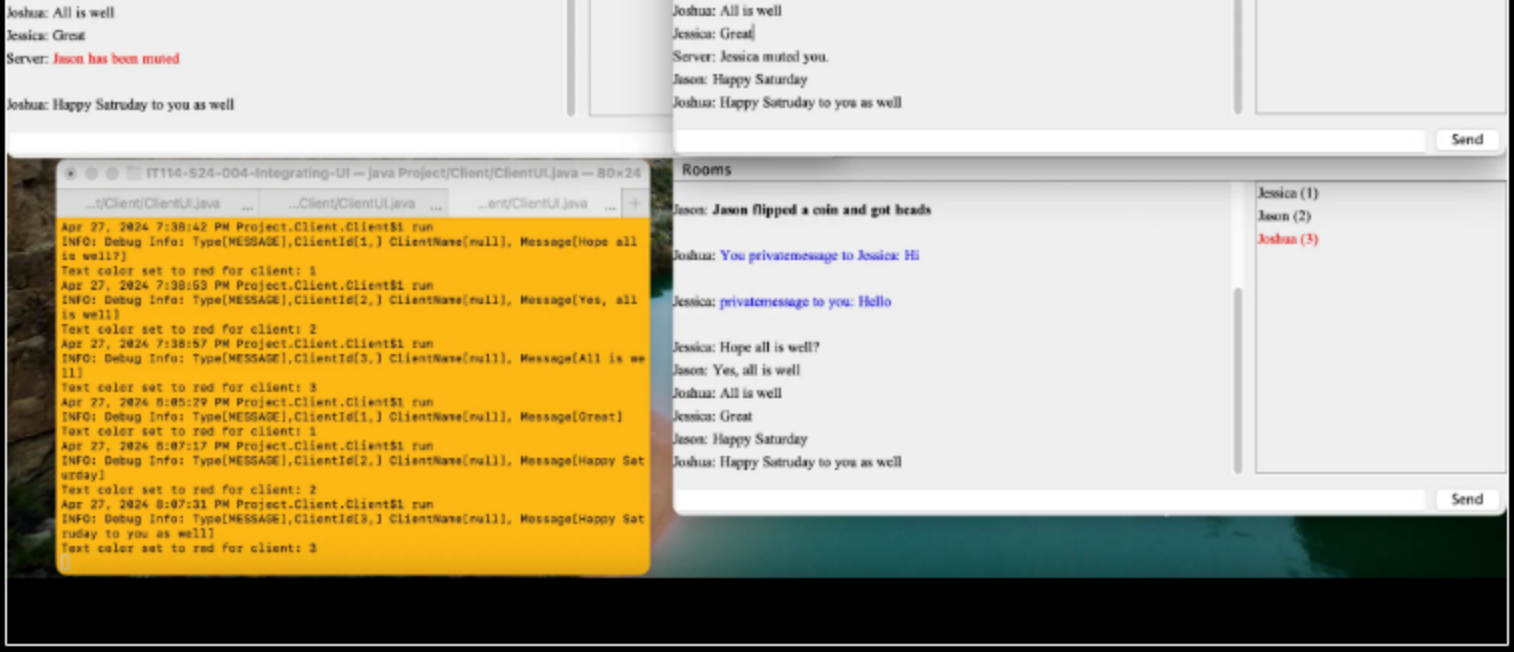


Jason muted

Checklist Items (1)

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

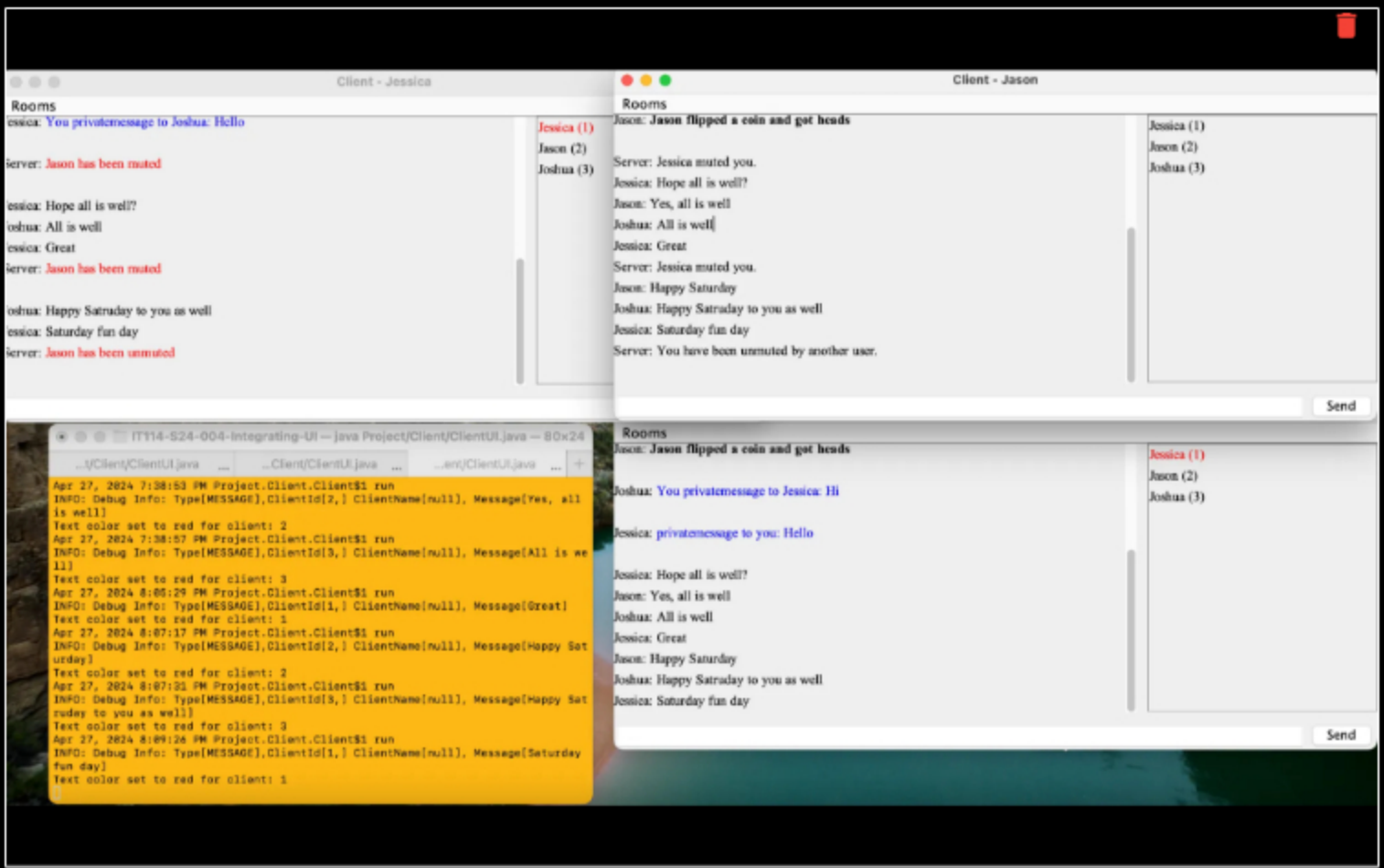




Jason muted Part 2

Checklist Items (1)

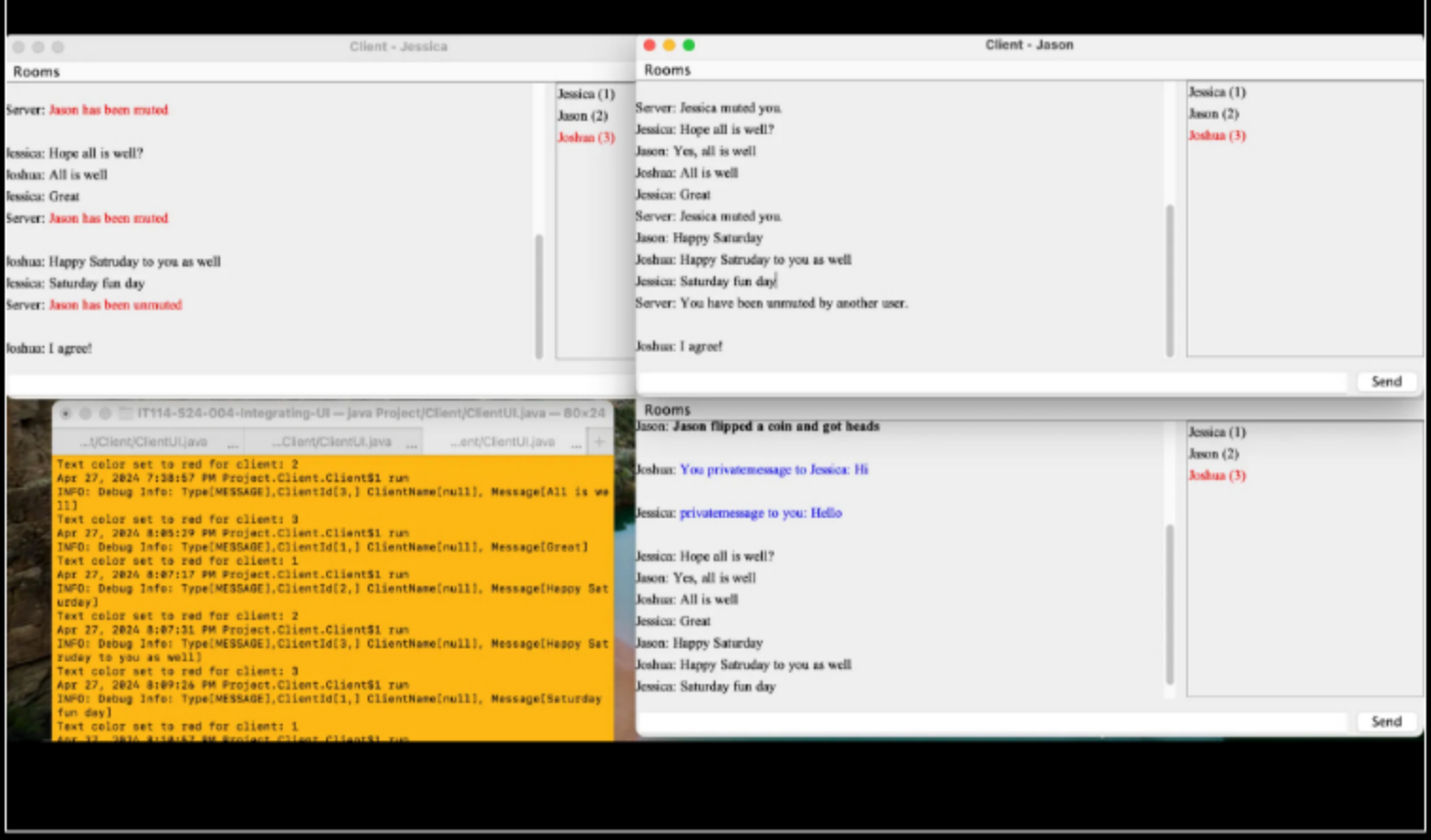
#1 Show before and after screenshots of the list updating upon mute and unmute



Jason unmute

Checklist Items (1)

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



"last person to send a message gets highlighted"

Checklist Items (1)

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

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:

When a mute/unmute payload is received and processed the client side is triggered. For the /mute action, the muted user's name is displayed in gray, and a red message indicates that the user has been muted. Similarly, for /unmute, the user's name is shown in black with the message showing the action that was completed.

The visual changes are based on exactly what is being done to the user.

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/jessicaodoom/jeo29-IT114-002/pull/19

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

The issues that I had during this assignment first stemmed from my laptop being overloaded when running the clients and then being able to export the chat. My device froze during the second time I tried to export a chat. I had to force quit. My next issue was trying to connect with the clients. The client would start, and allow me to enter a name, but once I selected connect, it wouldn't connect. For some odd reason, it wouldn't start up anymore. I reached out to the professor who told me to make sure nothing else was trying to connect to the port. That didn't help so I ended up uninstalling Java just starting the needed installations for this class from the beginning to see if that would help. I knew there was no issue with the code because I had not made any changes within my code when the client connect stopped working.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

Details:

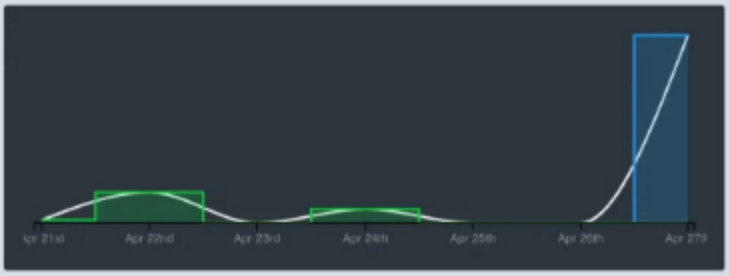
Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

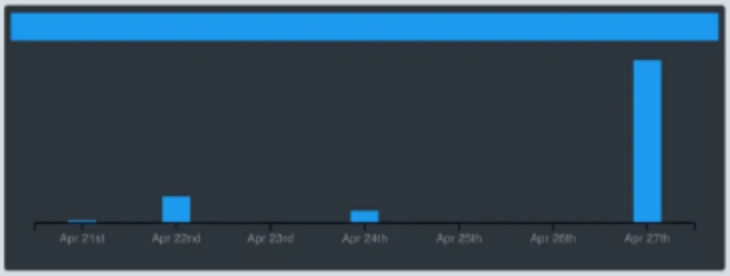
Task Screenshots:

Gallery Style: Large View

Small Medium Large

2 hrs 23 mins over the Last 7 Days.





1 hr 53 mins Today

IT114-S24-004-Integrating-UI 1:53

1 hr 53 mins Today

Java 1:53

Final time April 27, 2024

End of Assignment