Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-002-S2024/it114-chatroom-milestone-4-2024/grade/ob75

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

Submissions:

Submission Selection

1 Submission [active] 4/26/2024 9:32:18 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 > Views > ChatPanel.java > ChatPanel > ChatPanel(ICardControls) > new ComponentAdapter() {...} > componentResized(ComponentEvent)

```
34  public class ChatPanel extends JPanel {
211
212     // UCID - ob75 - April 18, 2024
213     public void extractText() {
214         StringBuilder exportText = new StringBuilder();
215         Component[] cs = chatArea.getComponents();
216         for (Component textContain : cs) {
217             if (textContain instanceof JEditorPane) {
218                 JEditorPane jText = (JEditorPane) textContain;
219                 String text = jText.getText();
220                 exportText.append(text);
221             }
222
223         }try {
224             Date Date = new Date() ;
225             SimpleDateFormat dateFormat = new SimpleDateFormat(pattern:"yyyy-MM-dd HH-mm-ss") ;
226             FileWriter exportedWriter = new FileWriter(dateFormat.format(Date) + ".txt");
227             exportedWriter.write(exportText.toString());
228             exportedWriter.close();
229
230           } catch (IOException e) {
231             System.out.println(x:"An error occurred.");
232             e.printStackTrace();
233           }
234     }
235 }
236
```

Starts under UCID on line 213 - 234 for extracting text to a file. All code is in ChatPanel class.

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

```
Project > Client > Views > J ChatPanel.java > ChatPanel > ChatPanel(ICardControls) > new ComponentAdapter() {...} > componentResized(ComponentEvent)
34  public class ChatPanel extends JPanel {
39      public ChatPanel(ICardControls controls) {
62          //UCID - ob75 - April 18, 2024
63          JButton exportButton = new JButton(text:"Export");
```

Created button for the ChatPanel named export. Under UCID on line 63 is relevant code.

Checklist Items (2)

#3 Screenshots should include ucid and date comment

#4 Each screenshot should be clearly captioned

```
Project > Client > Views > J ChatPanel.java > ChatPanel > ChatPanel(ICardControls) > new ComponentAdapter() {...} > componentResized(ComponentEvent)
34  public class ChatPanel extends JPanel {
39      public ChatPanel(ICardControls controls) {
105         //UCID - ob75 - April 18, 2024
106         exportButton.addActionListener((event) -> {
107             try {
108                 extractText();
109
110
111             } catch (NullPointerException e) {
112                 e.printStackTrace();
113             }
114         });
115         chatArea = content;
116         input.add(button);
117
118         //UCID - ob75 - April 18, 2024
119         input.add(exportButton);
120
```

Implementing button to do the method of extracting text and exporting to file once pressed. Under UCID comment, line

106 - 114 and also under UCID comment on bottom screen line 119 is also relevant code.

Checklist Items (2)

#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

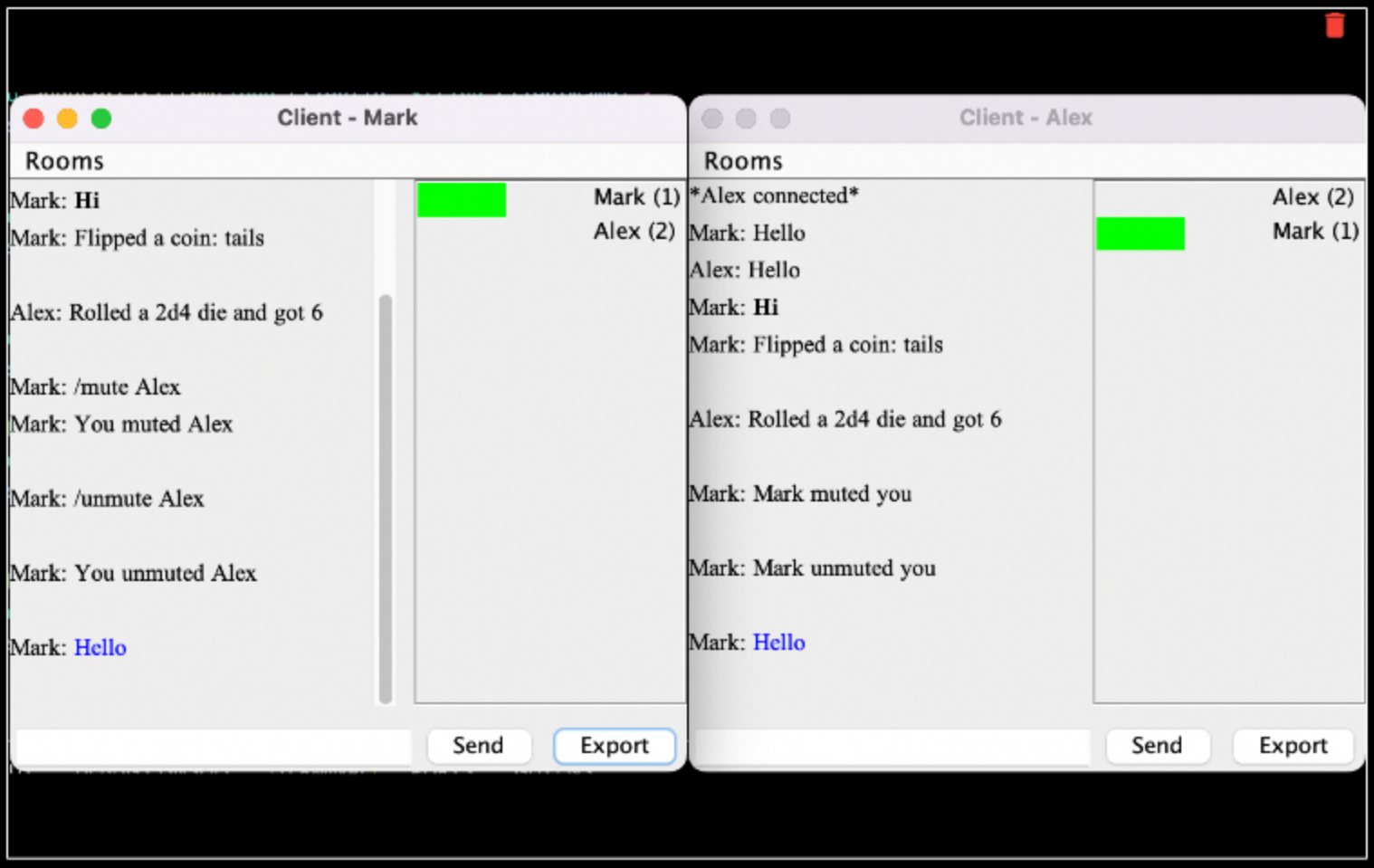
*The checkboxes are for your own tracking

#	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



Shows client ui of messages before export and showing the export button.

Checklist Items (1)

#3 Each screenshot should be clearly captioned

Exported .txt file with message being displayed and showing from who sent the message. (top half of .txt file)

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

```
  </head>
  <body>
    Mark: You muted Alex
  </body>
</html>
<html>
  <head>

  </head>
  <body>
    Mark: /unmute Alex
  </body>
</html>
<html>
  <head>

  </head>
  <body>
    Mark: You unmuted Alex
  </body>
</html>
```

```
</html>
<html>
  <head>

  </head>
  <body>
    Mark: <font color="blue">Hello</font>
  </body>
</html>
```

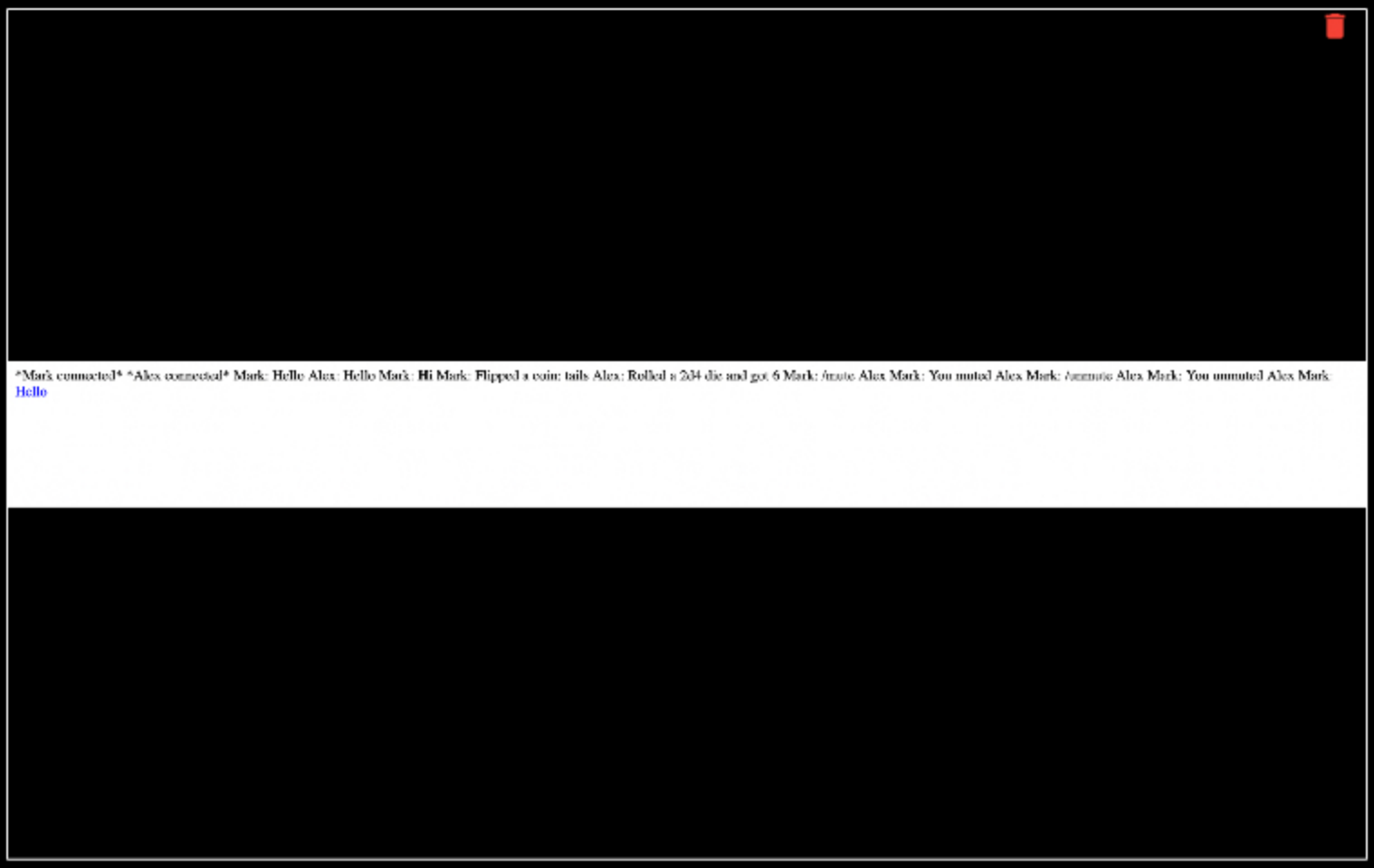
Exported .txt file containing message and from who sent it. (bottom half of .txt file).

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



Exported file converted into html to clearly show bold and color formatted messages.

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:

For implementing the export feature into Milestone 4, a new method was created in the ChatPanel class where the method is name extractText. The extractText method is meant to extract the text from the messages and out them into a file and write it out to view in .txt format but once changed. to html, it becomes much easier to see chat commands and text. Firstly in the method, a StringBuilder was initialized named exportText to be used later after grabbing the text from the messages. Then a component object named cs was made to be set to the chatArea's components. A for loop was then made for checking through the components with textContain being a new component made in the for loop argument and checking through cs which is the content's children components since content was set equal to chatArea. Then an if statement is made to check when textContain reaches the JEditorPane class which is where textContainer belongs to and textContainer holds the messages stored so its crucial to check for this field. The JEditorPane textContain field is then set to a new JEditorPane variable named jText and then to extract that text into a string, a new String variable was named text and gets the text from the JEditorPane.This text is then appended to the StringBuilder made previously named exportText. Now that the text has been extracted, it is now then to write it out to a file but first, there was Date named Date and SimpleDateFormat named dateFormat that was imported into the class and made in the try block so that every time the file is exported, there can be a unique name using a timestamp for the .txt file. After that, finally a file writer was imported and created named exportedWriter and converts the StringBuilder to String and writes it out to the file and closes it once done. The file is then created with the timestamp as the name and in a .txt file 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
#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

```
29   public class ServerThread extends Thread {
174
175      // UCID - ob75 - April 29, 2024
176      public void saveClientMuteFile() {
177          try {
178              FileWriter clientMuteFile = new FileWriter(getClientName() + ".txt");
179              clientMuteFile.write(String.join(delimiter:",", isMutedClients));
180              clientMuteFile.close();
181              System.out.println("Sucessfully wrote to file " + clientMuteFile);
182          } catch (IOException e) {
183              e.printStackTrace();
184          }
185
186      }
187
188      // UCID - ob75 - April 29, 2024
189      public void loadClientMuteFile() throws IOException {
190          File file = new File(getClientName() + ".txt");
191          if (client.isConnected()) {
192              if (file.exists()) {
193                  try (Scanner reader = new Scanner(file)) {
194                      reader.hasNextLine();
195                          String users = reader.nextLine();
196                              String[] mutedUsers = users.split(regex:",");
197                              for(int i = 0; i < mutedUsers.length; i++){
198                                  addMute(mutedUsers[i]);
199                                  sendMuteClient(currentRoom.getClientIdFromName(mutedUsers[i]));
200                                  System.out.println("adding muted user to list" + mutedUsers[i]);
201                              }
202                  } catch (FileNotFoundException e) {
203                      e.printStackTrace();
204                  } catch (Exception e2) {
205                      e2.printStackTrace();
206                  }
207              }
208          }
209      }
210
```

Whole screenshot shows relevant code for save method and a load method. UCID comment and under is lines 176 - 209 for relevant code.

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

Project > Server > ServerThread.java > ServerThread > loadClientMuteFile()

```
29   public class ServerThread extends Thread {
145
146      // UCID - ob75 - April 13, 2024
147      public void addMute(String clientMuteName) {
148
149          if (!isMutedClients.contains(clientMuteName)) {
150              isMutedClients.add(clientMuteName);
151          }
152          // UCID - ob75 - April 29, 2024
153          saveClientMuteFile();
154      }
```

```
154     }
155
156     // UCID - ob75 - April 13, 2024
157     public void removeMute(String clientUnmuteName) {
158         if (isMutedClients.contains(clientUnmuteName)) {
159             isMutedClients.remove(clientUnmuteName);
160         }
161         // UCID - ob75 - April 29, 2024
162         saveClientMuteFile();
163     }
```

Included save method inside add method and remove method to track state. Under UCID comment, line 153 and line 162 in screenshot is new code.

Checklist Items (3)

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

#3 Screenshots should include ucid and date comment

#4 Each screenshot should be clearly captioned

Project > Server > ServerThread.java > ServerThread > loadClientMuteFile()

```
29  public class ServerThread extends Thread {
66
67      protected void setClientName(String name) {
68          if (name == null || name.isBlank()) {
69              logger.severe(msg:"Invalid client name being set");
70              return;
71          }
72          clientName = name;
73          //UCID - ob75 - April 29, 2024
74          try {
75              loadClientMuteFile();
76          } catch (IOException e) {
77              e.printStackTrace();
78          }
79      }
```

Load method called inside setClientName method for loading mute list to client when joining. Under UCID comment lines 74 - 78 is new code.

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

```
12  public class Room implements AutoCloseable {
42      protected synchronized void addClient(ServerThread client) {
43          if (!isRunning) {
44              return;
45          }
46          client.setCurrentRoom(this);
47          client.sendJoinRoom(getName());// clear first
48          if (clients.indexOf(client) > -1) {
49              info(message:"Attempting to add a client that already exists");
50          } else {
51              clients.add(client);
52              // connect status second
53              sendConnectionStatus(client, isConnected:true);
54              syncClientList(client);
55
56              // UCID - ob75 - April 29, 2024
57              Iterator<ServerThread> iter = clients.iterator();
58              while (iter.hasNext()) {
59                  ServerThread user = iter.next();
60                  System.out.println(user.getClientName() + " and " + client.getClientName());
61                  if (user.checkMutedList(client.getClientName())) {
62                      try {
63                          user.sendMuteClient(client.getClientId());
64                      } catch (IOException e) {
65                          e.printStackTrace();
66                      }
67                  }
68              }      You, 7 minutes ago • Uncommitted changes
69              try {
70                  client.loadClientMuteFile();
71              } catch (IOException e) {
72
73                  e.printStackTrace();
74              }
75          }
76      }
```

Added Load method onto addClient method in Room side and iteration through clients in room to send payload for UI effect. Under UCID comment from lines 57 - 74.

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

```
12  public class Room implements AutoCloseable {
77
78      // UCID - ob75 - April 29, 2024
79      protected long getClientIdFromName(String clientName) {
80          Iterator<ServerThread> iter = clients.iterator();
81          while (iter.hasNext()) {
82              ServerThread client = iter.next();
83              if (client.getClientName().equals(clientName)) {
84                  return client.getClientId();
```

```
84                 return client.getClientId();
85             }
86
87         }
88         return Constants.DEFAULT_CLIENT_ID;
89
90     }
91
```

New method in Room side called getClientIdFromName to get the the id of the client from the name and be called in load method. Under UCID comment lines 79 - 90.

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

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the demo

Checklist

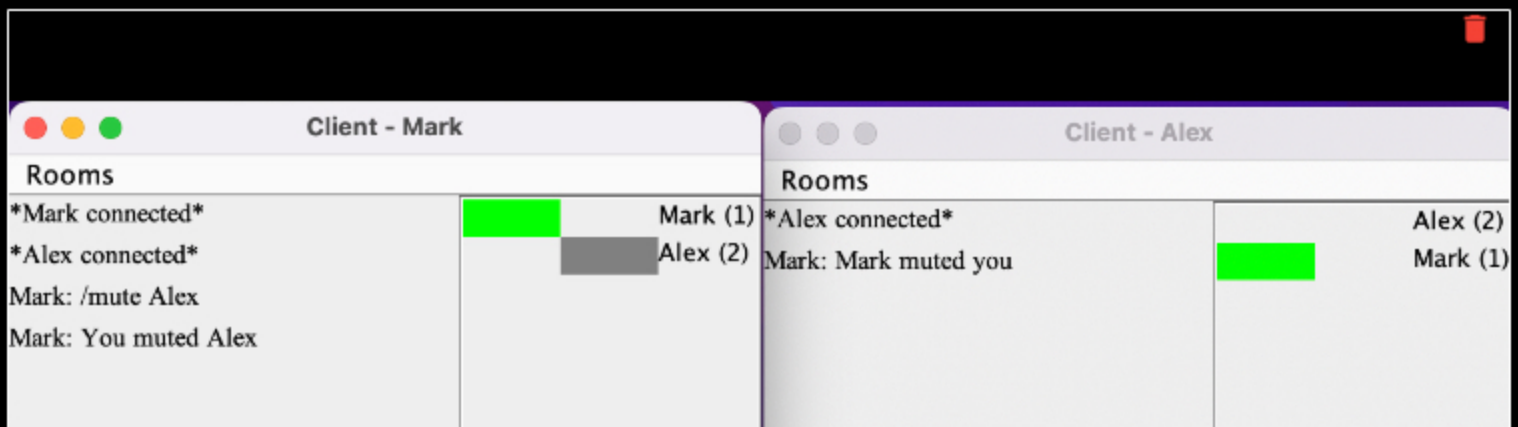
*The checkboxes are for your own tracking

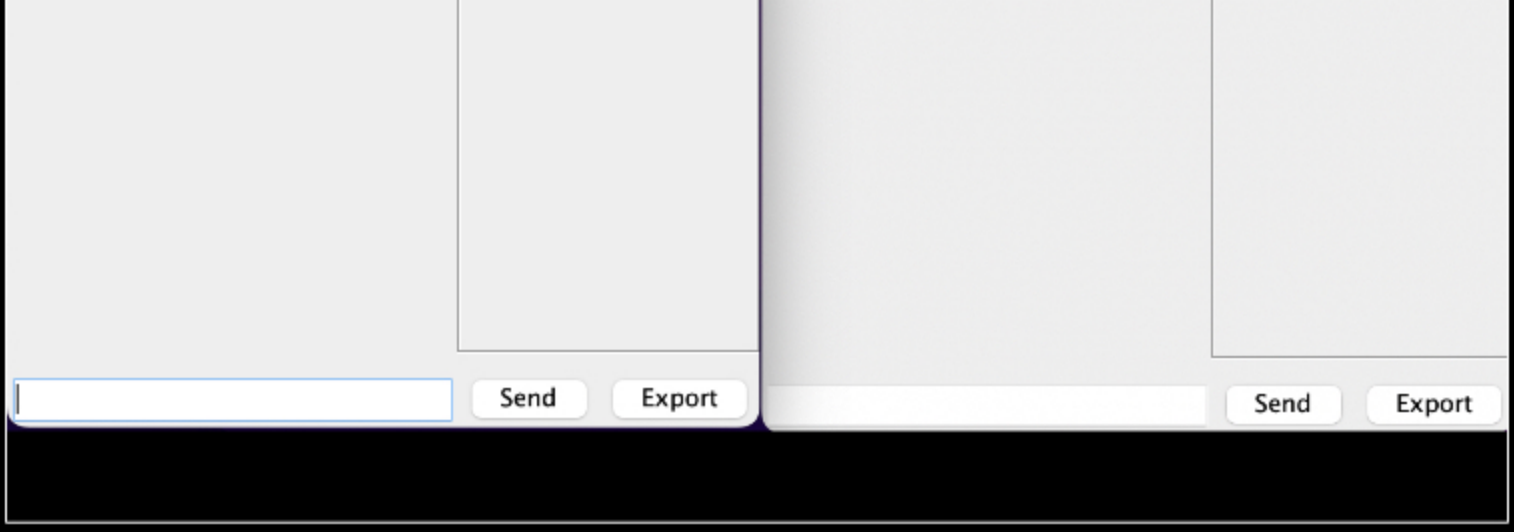
#	Points	Details
#1	1	Show a user muting another user, disconnecting, reconnecting, and still having that user muted (same should be possible if the server restarts)
#2	1	This should also be reflected in the UI per related feature in this milestone
#3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large



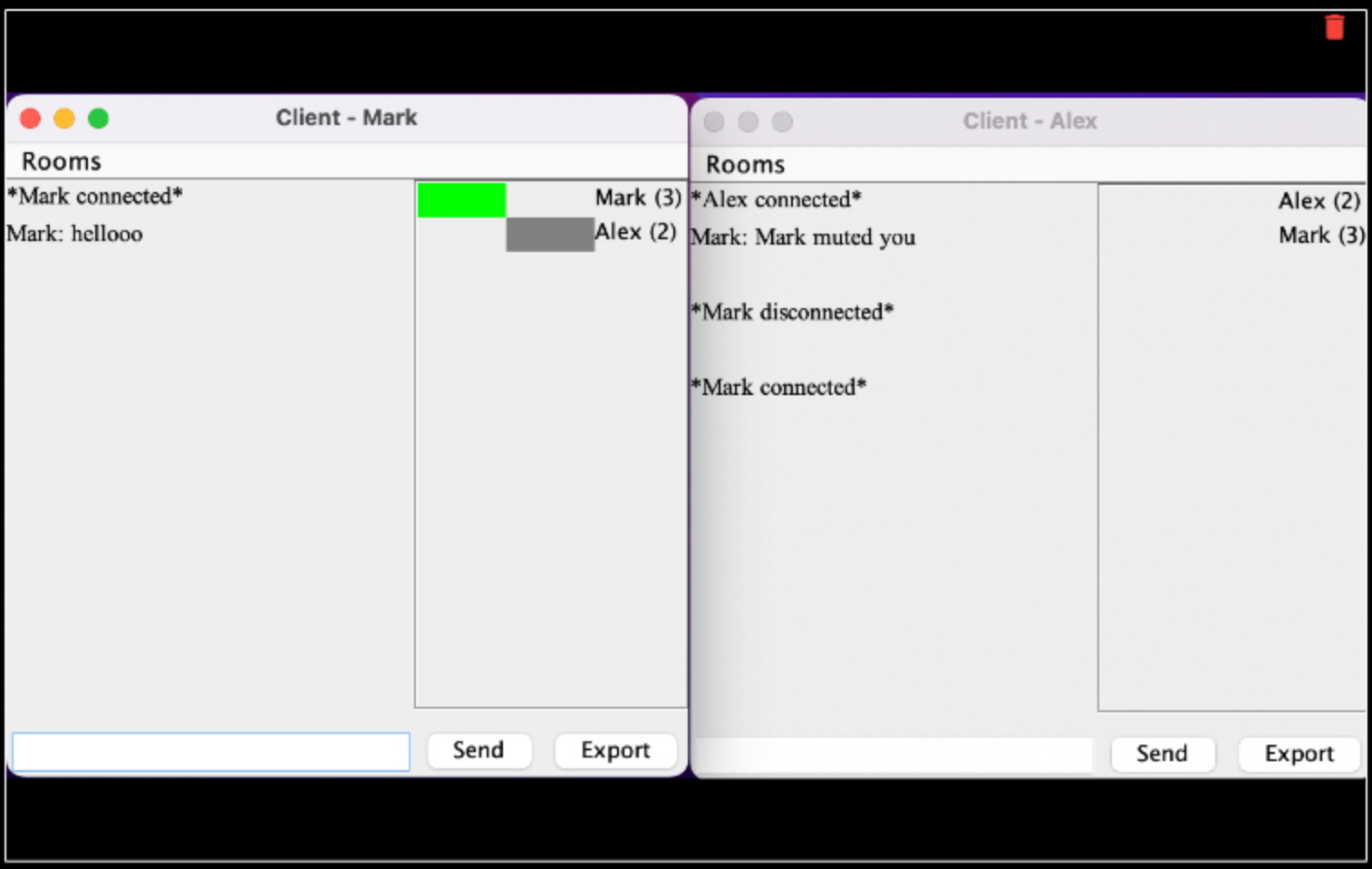


Demo of Client Mark muting client Alex before disconnecting.(Left window muting right window).

Checklist Items (2)

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

#3 Each screenshot should be clearly captioned



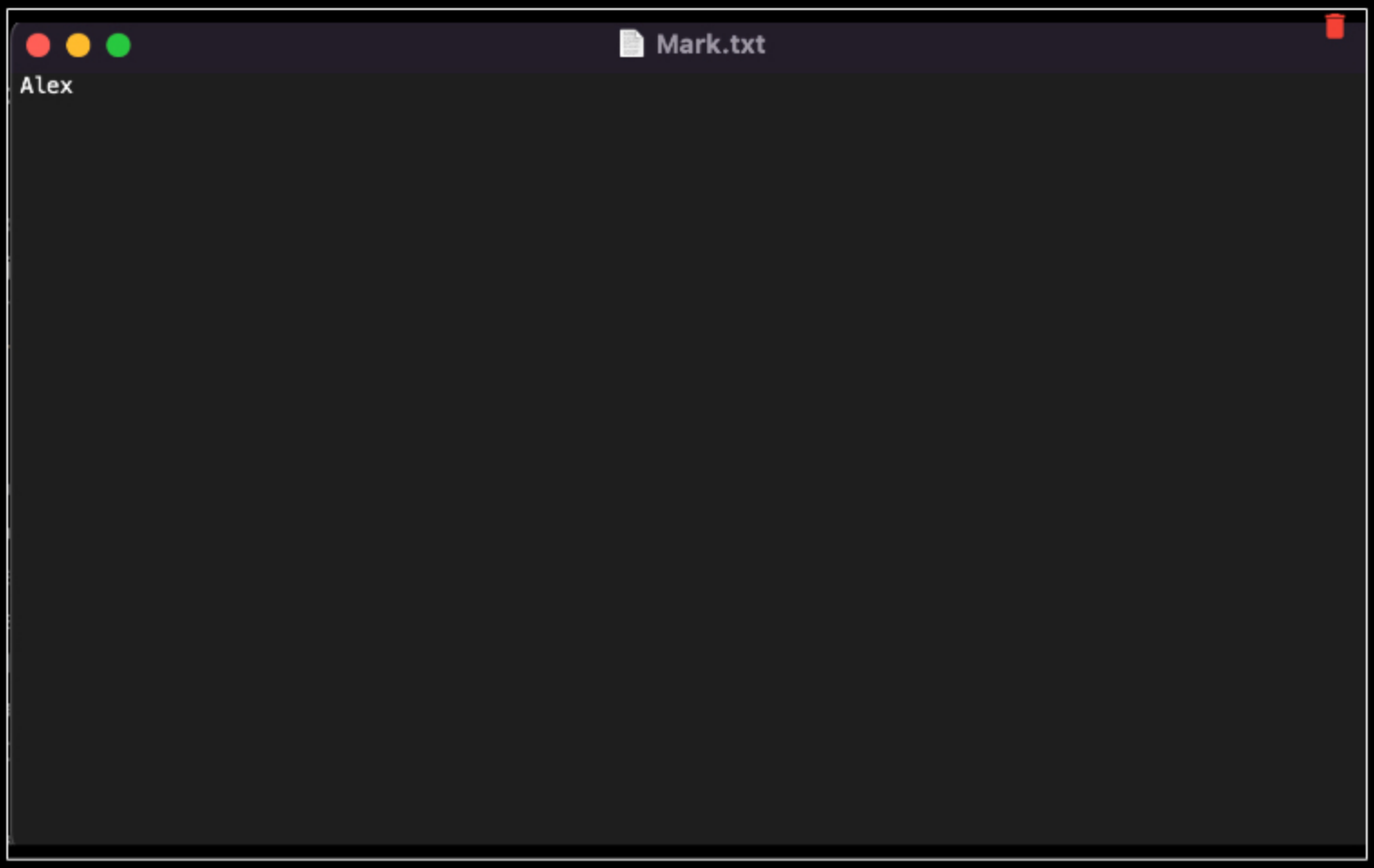
Demo of Client Mark reconnecting after disconnecting from room and still having Alex muted and still showing UI effect being the gray box next to client Alex.

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



The .txt file that is saved from muting the client Alex and the muted client's name appearing in the file and having the client Mark's name as the name of the file.

Checklist Items (2)

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

#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:

First thing that was done was to create 2 methods on the ServerThread side, one named saveClientMuteFile which would create a new .txt file with file writer that would contain the users name as the name of the file and would hold in

this file all the muted users in the list. This file would be called at the end in both the addMute and removeMute to track the state changes and would change the file according to when the muted list changes. The second method is loadClientMuteFile in which would get the file name first of the user who's logging back in the server and then would check if that file exist. If it does then it would read from the file using scanner and read the line of muted clients list that the file contains. It then splits the users and adds the users back into the muted clients list that was already made on the server thread side and sending a payload out so that ClientUI can still make the gray box for the muted client. A method was also made on the Room side named getClientIdFromName that grabs the Id of the client from the name by iterating through the list of clients in the room and then matching the client and returning the ID of the client and is called in the load method after adding the mute list so that when sending out the payload, it goes to the correct client. The loadClientMuteFile method is then called inside the setClientName so that once the client joins back and sets their name, it would load in the list of muted clients if the had one previously. Lastly in addClient on the Room side, an iterator was placed after the client is coming into the room and iterates to check the muted list and compare it to the muted list in the server thread so that the clients muted can be grayed out. The method of loadClientMuteFile was then called again here to load the file of the client that has muted clients in their list and so that it can persist on incoming clients.

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

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

```
13  public class Room implements AutoCloseable {
102     // UCID - ob75 - April 13, 2024
103     public void setMute(ServerThread client, String clientMuteName) {
104         client.sendMessage(client.getClientId(), String.format(format:"/mute %s", clientMuteName));
105         Iterator<ServerThread> muteIter = clients.iterator();
106         while (muteIter.hasNext()) {
107             ServerThread mClients = muteIter.next();
108             if (mClients.getClientName().equalsIgnoreCase(clientMuteName)) {
109                 if (client.checkMutedList(mClients.getClientName()) == false) {
110                     // UCID - ob75 - April 20, 2024
111                     client.sendMessage(client.getClientId(), String.format(format:" You muted %s", clientMuteName));
112                     mClients.sendMessage(client.getClientId(), String.format(format:" %s muted you", client.getClientName()));
113                     client.addMute(clientMuteName);
114                     // UCID - ob75 - April 24, 2024
115                     try {
```

```
116                 client.sendMuteClient(mClients.getClientId());
117             } catch (IOException e) {
118                 e.printStackTrace();
119             }
120
121         }
122
123     }
124
125 }
126 }
```

Sending mute message notification to both client and target client, relevant code is under UCID from line 111-112.

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

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

```
13  public class Room implements AutoCloseable {
127
128     // UCID - ob75 - April 13, 2024
129     public void setUnmute(ServerThread client, String clientUnmuteName) {
130         client.sendMessage(client.getClientId(), String.format(format:"/unmute %s", clientUnmuteName));
131         Iterator<ServerThread> muteIter = clients.iterator();
132         while (muteIter.hasNext()) {
133             ServerThread umClients = muteIter.next();
134             if (umClients.getClientName().equalsIgnoreCase(clientUnmuteName)) {
135                 if (client.checkMutedList(umClients.getClientName()) == true) {
136                     // UCID - ob75 - April 20, 2024      You, 1 second ago • Uncommitted changes
137                     client.sendMessage(client.getClientId(), String.format(format:" You unmuted %s", clientUnmuteName));
138                     umClients.sendMessage(client.getClientId(), String.format(format:" %s unmuted you", client.getClientName()));
139                     client.removeMute(clientUnmuteName);
140                     // UCID - ob75 - April 24, 2024
141                     try {
142                         client.sendUnmuteClient(umClients.getClientId());
143                     } catch (IOException e) {
144                         e.printStackTrace();
145                     }
146                 }
147
148             }
149
150         }
151
152     }
```

Sending unmute message notification to both client and target client, relevant code is under UCID comment from line 137 -138.

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

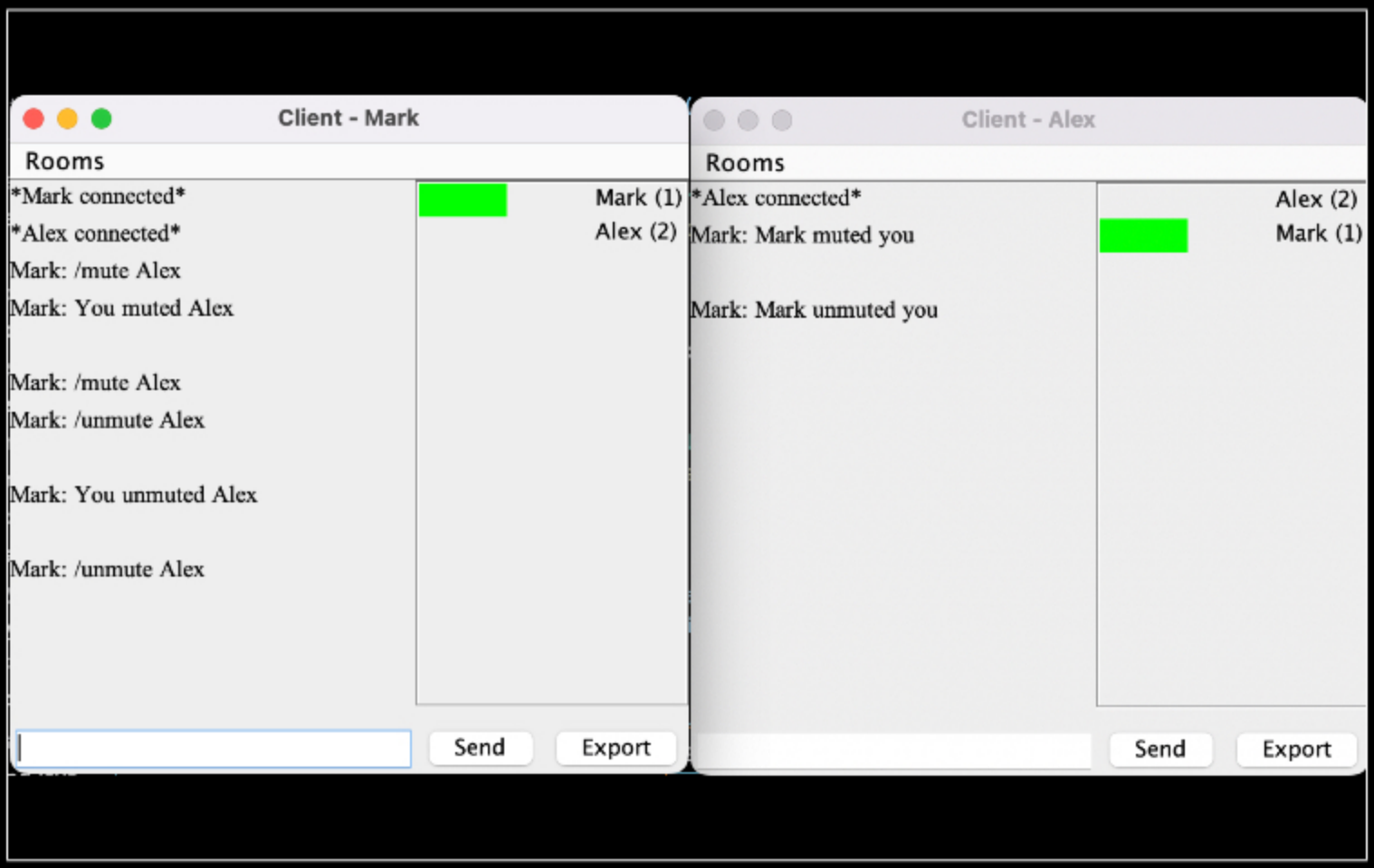
*The checkboxes are for your own tracking

#	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



Shows both mute and unmute notifications being sent to both clients once even though the command is placed twice in succession. /unmute is just the command for screenshot purposes to show command is being placed so it does not count as notification of mute or unmute as shown by left client window receiving only one notification for mute and one for unmute.

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:

In the setMute and setUnmute methods in the Room side, a message is first to the client by doing client.sendMessage inside of the for loop of checking if the client is or is not in the muted list and then grabbing the clients ID and then using a String.format to create a custom message and including who they muted for example in this code it is "you muted" or "you unmuted" and then the muted client name or unmuted client name. This is also done for the target client right after the client.sendMessage for them to receive a message by getting the server thread matched user target name in the room which in the setMute method was mClient.sendMessage and in setUnmute was umClient.sendMessage. Both mute and unmute notifications to user and target go through the same process with the difference being that the message saying unmute instead of mute. Now the way the message is sent once only is through after the message are sent, the addMute is done to add the target to the client's mute list and if they are in that mute list, then the for loop will not occur and won't send the message again if the command /mute is done again since the condition is no longer false and they are in the muted list now and is instead true as long as they are not removed from the list. This works as well for setUnmute with removeMute being after the sendMessages for the clients so that if the user is in the list or is true, then client will receive a message saying they successfully unmuted a target client and the target client will receive the message they are unmuted from the client but will not receive the message again if the command /unmute is done again on the same client if they stay unmuted since the condition is now false that they are not in the list and the condition requires to be true for it to be sent.

Note: "/mute clientName" or "/unmute clientName will repeat to just show the command and that the command is being repeated but no actual message of being muted or unmuted is repeated in succession if that is being done after the first time.

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

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

```
13   public class Room implements AutoCloseable {
102      // UCID - ob75 - April 13, 2024
103      public void setMute(ServerThread client, String clientMuteName) {
104          client.sendMessage(client.getClientId(), String.format(format:"/mute %s", clientMuteName));
105          Iterator<ServerThread> muteIter = clients.iterator();
106          while (muteIter.hasNext()) {
107              ServerThread mClients = muteIter.next();
108              if (mClients.getClientName().equalsIgnoreCase(clientMuteName)) {
109                  if (client.checkMutedList(mClients.getClientName()) == false) {
110                      // UCID - ob75 - April 20, 2024
111                      client.sendMessage(client.getClientId(), String.format(format:" You muted %s", clientMuteName));
112                      mClients.sendMessage(client.getClientId(), String.format(format:" %s muted you", client.getClientName()));
113                      client.addMute(clientMuteName);
114                      // UCID - ob75 - April 24, 2024
115                      try {
116                          client.sendMuteClient(mClients.getClientId());
117                      } catch (IOException e) {
118                          e.printStackTrace();
119                      }
120
121                  }
122
123              }
124
125          }
126      }
```

Sending payload of client with target ID for mute, relevant code is under UCID from line 115-119 is new code for feature. In Room class.

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

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

```
13   public class Room implements AutoCloseable {
127
128      // UCID - ob75 - April 13, 2024
129      public void setUnmute(ServerThread client, String clientUnmuteName) {
130          client.sendMessage(client.getClientId(), String.format(format:"/unmute %s", clientUnmuteName));
131          Iterator<ServerThread> muteIter = clients.iterator();
132          while (muteIter.hasNext()) {
133              ServerThread umClients = muteIter.next();
134              if (umClients.getClientName().equalsIgnoreCase(clientUnmuteName)) {
135                  if (client.checkMutedList(umClients.getClientName()) == true) {
136                      // UCID - ob75 - April 20, 2024      You, 1 second ago • Uncommitted changes
137                      client.sendMessage(client.getClientId(), String.format(format:" You unmuted %s", clientUnmuteName));
138                      umClients.sendMessage(client.getClientId(), String.format(format:" %s unmuted you", client.getClientName()));
139                      client.removeMute(clientUnmuteName);
140                      // UCID - ob75 - April 24, 2024
141                      try {
142                          client.sendUnmuteClient(umClients.getClientId());
143                      } catch (IOException e) {
144                          e.printStackTrace();
145                      }
146                  }
147
148              }
149
150          }
151
152      }
```

Sending payload of client with target ID for unmute, relevant code is under UCID from line 141-145 is new code for feature. In Room class.

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

Project > Server > ServerThread.java > ServerThread > sendUnmuteClient(long)

```
25   public class ServerThread extends Thread {
163
164      //UCID - ob75 - April 24, 2024
165      public void sendMuteClient(long clientId) throws IOException {
166          Payload mu = new Payload();
167          mu.setClientId(clientId);
168          mu.setPayloadType(PayloadType.MUTE);
169          out.writeObject(mu);
170      }
171
172      //UCID - ob75 - April 24, 2024      Omar Barrera, 3 days ago • Working on Milestone4 MuteColor
173      public void sendUnmuteClient(long clientId) throws IOException {
174          Payload um = new Payload();
175          um.setClientId(clientId);
176          um.setPayloadType(PayloadType.UNMUTE);
177          out.writeObject(um);
178      }
179
```

Both sendMute and sendUnmute methods in screenshot in server thread side. Under UCID comment lines 165-178 is relevant code for graying out muted users.

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

Project > Client > IClientEvents.java > IClientEvents

```
5   public interface IClientEvents {
66
67      //UCID - ob75 - April 24, 2024
68      void onReceiveMute(long clientId);
69          Omar Barrera, 3 days ago • Working on Milestone4 MuteColor
70      //UCID - ob75 - April 24, 2024
71      void onReceiveUnmute(long clientId);
72  }
```

IClientEvents methods for onReceiveMute and onReceiveUnmute methods, under UCID comment line 68 and line 71 is relevant code.

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

Project > Client > Client.java > Client > processPayload(Payload)

```
26   public enum Client {
507      private void processPayload(Payload p) {
578          // UCID - ob75 - April 24, 2024
579          case MUTE:
580          System.out.println("This is the client ID " + p.getClientId());
581              try {
582                  events.onReceiveMute(p.getClientId());
583
584              } catch (Exception e) {
585                  e.printStackTrace();
586              }
587
588              break;
589
590          // UCID - ob75 - April 24, 2024
591          case UNMUTE:
592          System.out.println("This is the Unmuted client ID " + p.getClientId());
593              try {
594              events.onReceiveUnmute(p.getClientId());
595
596              } catch (Exception e) {
597              e.printStackTrace();
598              }
599              break;
600                  Omar Barrera, 3 days ago • Working on Milestone4 MuteColor
601          default:
602              break;
```

Client side receiving payloads for case MUTE and UNMUTE and calling Client events. Under UCID comment from line 579-599 is relevant code.

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

Project > Client > ClientUI.java > ClientUI > ClientUI(String)

```
28   public class ClientUI extends JFrame implements IClientEvents, ICardControls {
179      }
180
181      @Override
182      public void onMessageReceive(long clientId, String message) {
183          if (currentCard.ordinal() >= CardView.CHAT.ordinal()) {
184              String clientName = Client.INSTANCE.getClientNameFromId(clientId);
185              chatPanel.addText(String.format(format:"%s: %s", clientName, message));
186          //UCID - ob75 - April 22, 2024
187              chatPanel.highlightLastMessage(clientId);
188          }
189      }
190      //UCID - ob75 - April 24, 2024
191      public void onReceiveMute(long clientId){
192          chatPanel.colorClients(clientId, isMuted:true);
193      }
194
195      //UCID - ob75 - April 24, 2024
196      public void onReceiveUnmute(long clientId){
197          chatPanel.colorClients(clientId, isMuted:false);
198      }
```

Both highlighting last message and onReceive mute and unmute included in screenshot, new code for highlighting last message is under UCID comment line 187 and for graying out muted clients is under next UCID comment from 191-198 is relevant code. In client UI side.

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

Project > Client > Views > ChatPanel.java > ChatPanel > ChatPanel(ICardControls)

```
34  public class ChatPanel extends JPanel {
178     //UCID - ab75 - April 22, 2824
179     public void highlightLastMessage(long clientId) {
180         userListPanel.highlightLastMessage(clientId);
181     }
182
183     //UCID - ab75 - April 24, 2824
184     public void colorClients(long clientId, boolean isMuted) {
185         userListPanel.colorClients(clientId, isMuted);
186     }
187
```

Both highlighting last message and graying muted clients code is included here, under UCID comment from lines 179 - 186 is relevant code. In ChatPanel class.

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

Project > Client > Views > UserListItem.java > UserListItem

```
13  public class UserListItem extends JPanel {
41      //UCID - ob75 - April 22, 2824
42      public void setLastMessage(long clientId) {
43          if (getName().equals(Long.toString(clientId))) {
44              lastMessage.setVisible(aFlag:true);
45              lastMessage.setBackground(Color.GREEN);
46          } else {
47              lastMessage.setVisible(aFlag:false);
48          }
49      }
50
51      //UCID - ob75 - April 22, 2824
52      public void setClientColor(long clientId, boolean isMuted){
53      System.out.print(String.format(format:" %s and %s", clientId, isMuted));
54      if (getName().equals(Long.toString(clientId))) {
55          if(isMuted){
56              colorClient.setVisible(aFlag:true);
57              colorClient.setBackground(Color.GRAY);
58          }
59          else{
60              colorClient.setVisible(aFlag:false);
61          }
62          colorClient.revalidate();
63          colorClient.repaint();
64      }
65  }
66
67  }       Omar Barrera, 5 days ago • Working on Milestone4 pt2
```

Both highlight last message sent and graying out muted users code included, in UserListItem class under UCID comment lines 42 - 65 is relevant code.

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

Project > Client > Views > UserListPanel.java > ...

```
17  public class UserListPanel extends JPanel {
91      //UCID - ob75 - April 22, 2824
92      protected void highlightLastMessage(long clientId) {
93          Component[] cs = userListArea.getComponents();
94          for (Component c : cs) {
95              if (c instanceof UserListItem) {
96                  UserListItem uli = (UserListItem) c;
97                  uli.setLastMessage(clientId);
98              }
99          }
100     }
```

```java
101     //UCID - ob75 - April 24, 2024
102     protected void colorClients(long clientId, boolean isMuted){
103         Component[] cs = userListArea.getComponents();
104         for (Component c : cs) {
105             if (c instanceof UserListItem) {
106                 UserListItem uli = (UserListItem) c;
107                 uli.setClientColor(clientId, isMuted);
108             }
109         }
110     }
111 }
112
```

Both highlight last message sent and graying out muted users code included, in UserListPanel class under UCID comment lines 92 - 110 is relevant code.

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

Project > Client > Views > UserListItem.java > UserListItem > setClientColor(long, boolean)

Omar Barrera, 3 days ago | 1 author (Omar Barrera)

```java
13  public class UserListItem extends JPanel {
14      //UCID - ob75 - April 24, 2024
15      private JEditorPane usernameContainer;
16      private JPanel lastMessage;
17      private JPanel colorClient;
18
19      public UserListItem(long clientId, String clientName) {
20          // Set a layout that will organize the children vertically
21          this.setLayout(new GridLayout(rows:1, cols:3));
22          this.setName(Long.toString(clientId));
23          lastMessage = new JPanel();
24          colorClient = new JPanel();
25
26          this.lastMessage.setVisible(aFlag:false);
27          this.colorClient.setVisible(aFlag:true);
28
29          this.add(lastMessage);
30          this.add(colorClient);
31
32          JEditorPane textContainer = new JEditorPane(type:"text/plain", clientName);
33          this.usernameContainer = textContainer;
34          textContainer.setEditable(b:false);
35          textContainer.setName(Long.toString(clientId));
36
37          ClientUtils.clearBackground(textContainer);
38          this.add(textContainer);
39      }
40
```

adding JPanels for color to appear next to client for muted clients which is Gray or who sent a message last which is green. Under UCID comment lines 16-17 and 26-29

green. Under UCID comment line 16-17, and 26-30.

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

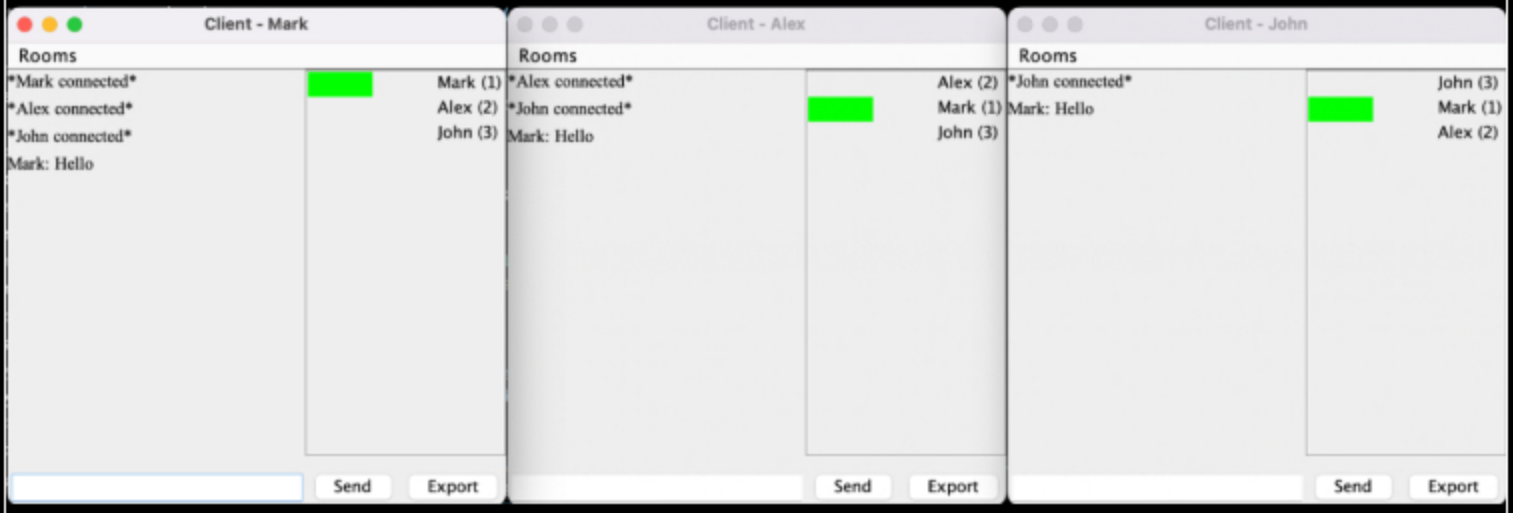
*The checkboxes are for your own tracking

#	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

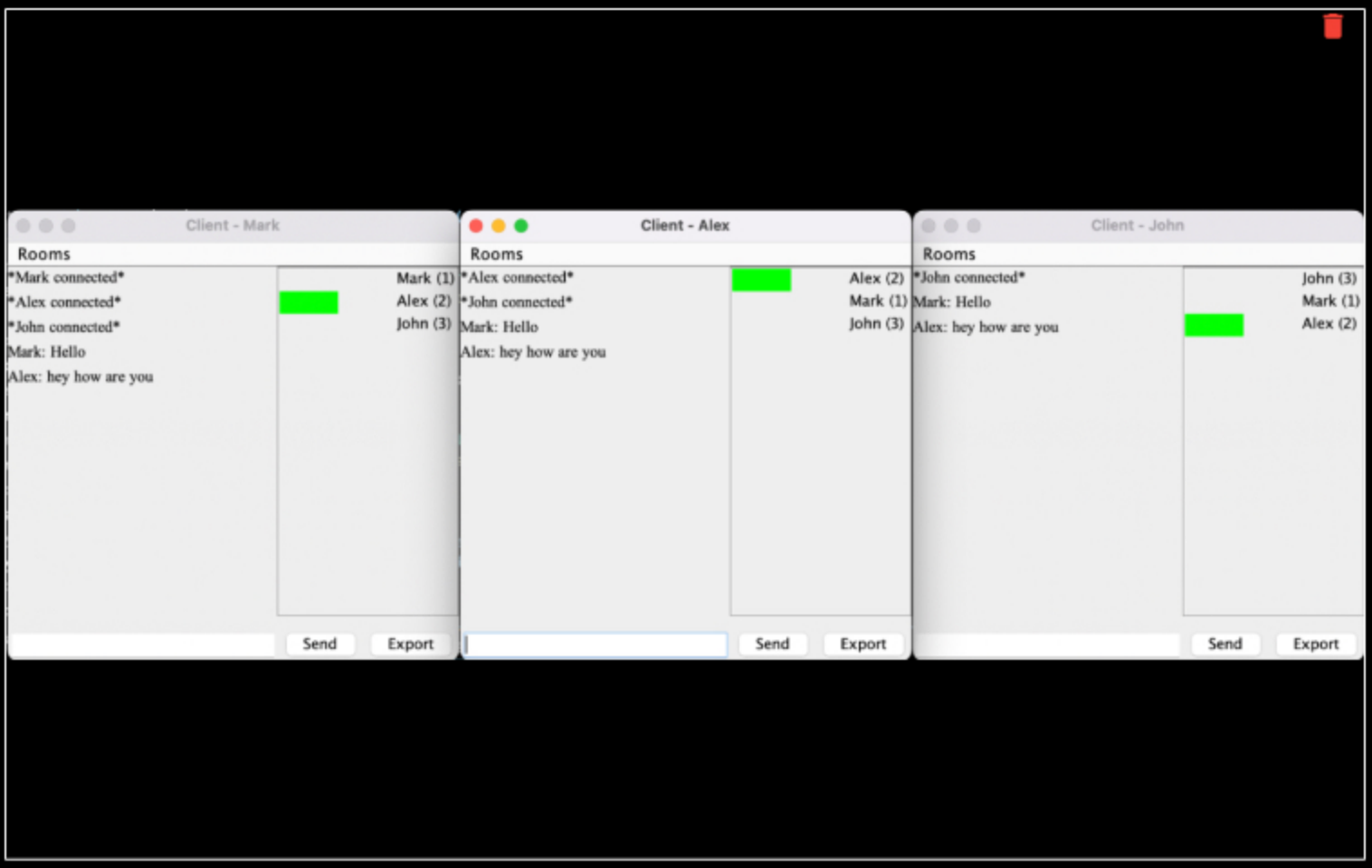


Mark is last client to send message so he is highlighted green. (left client window is Mark)

Checklist Items (2)

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

#3 Each screenshot should be clearly captioned

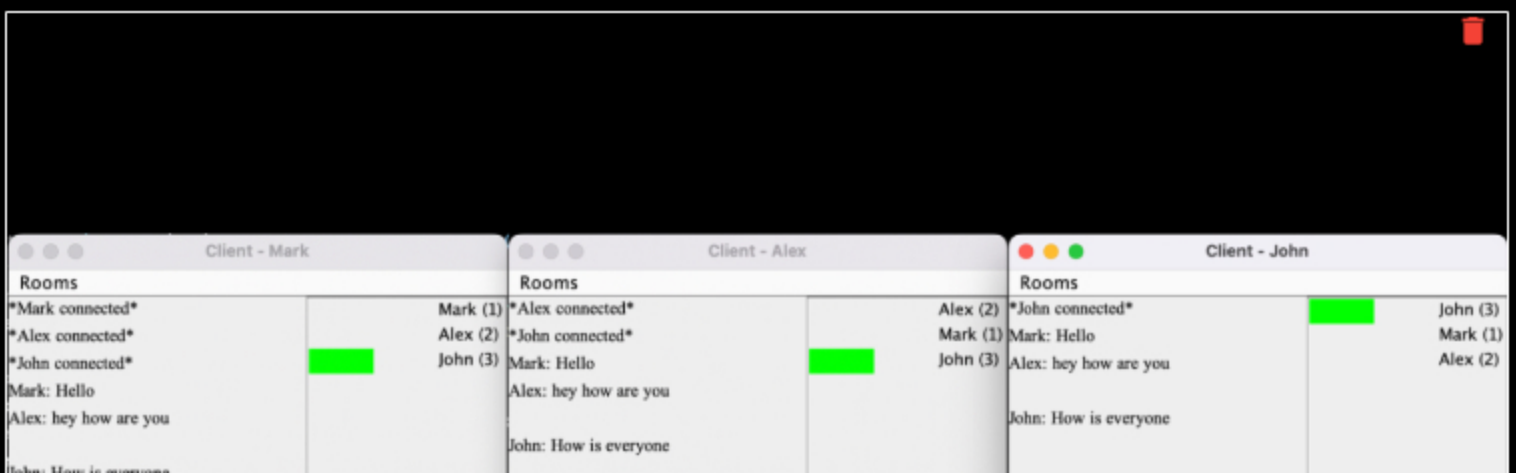


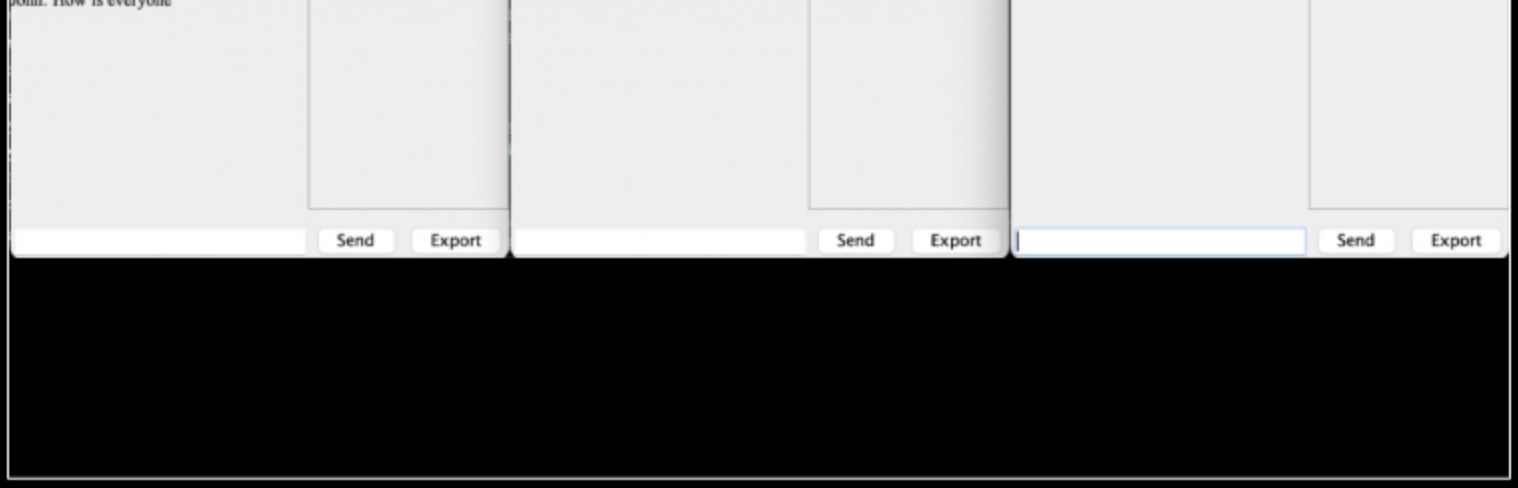
Alex is new last person to send message so he is highlighted green. (middle client window is Alex)

Checklist Items (2)

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

#3 Each screenshot should be clearly captioned



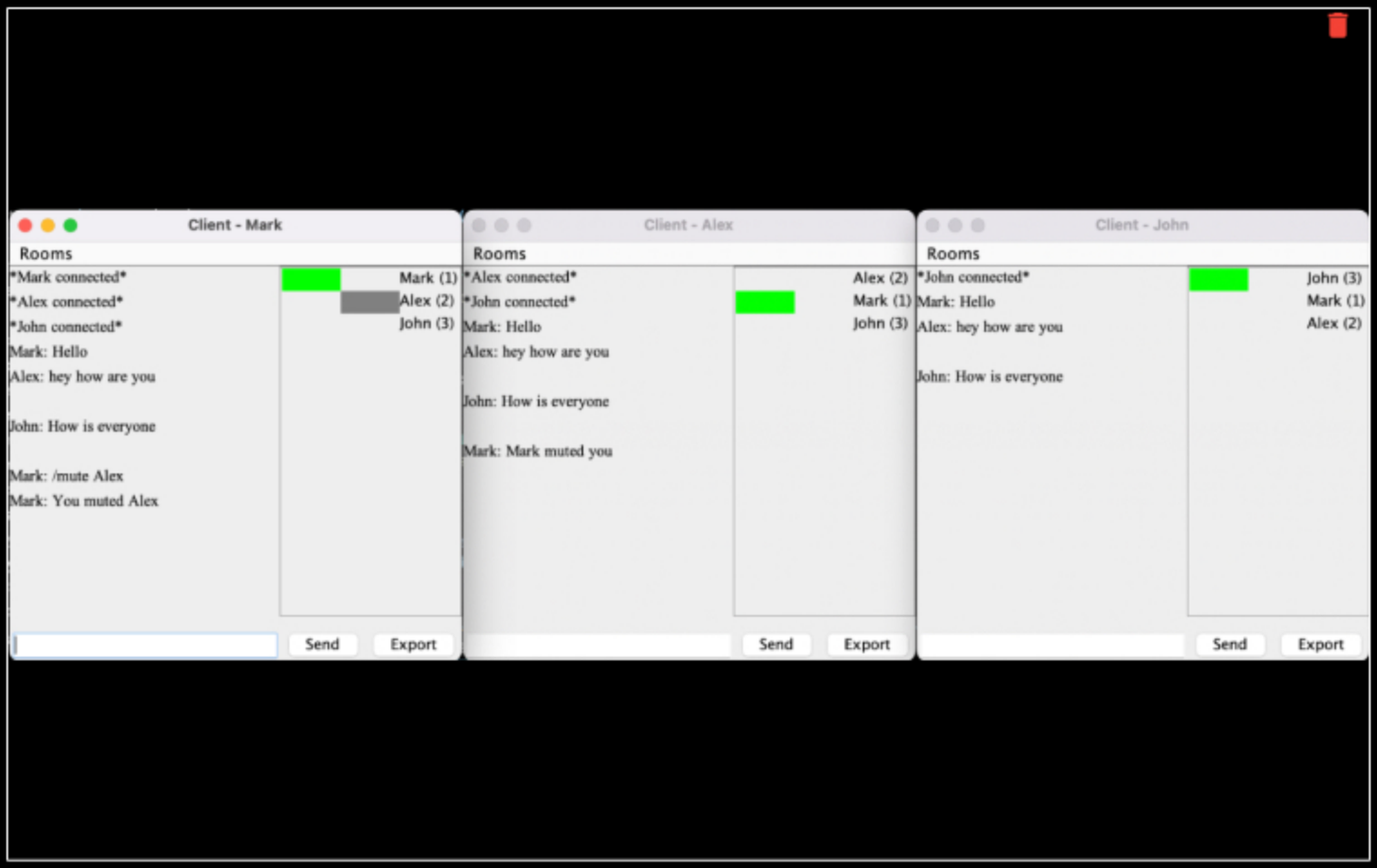


John now sent last message and is the new highlighted green client. (right client window is John)

Checklist Items (2)

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

#3 Each screenshot should be clearly captioned

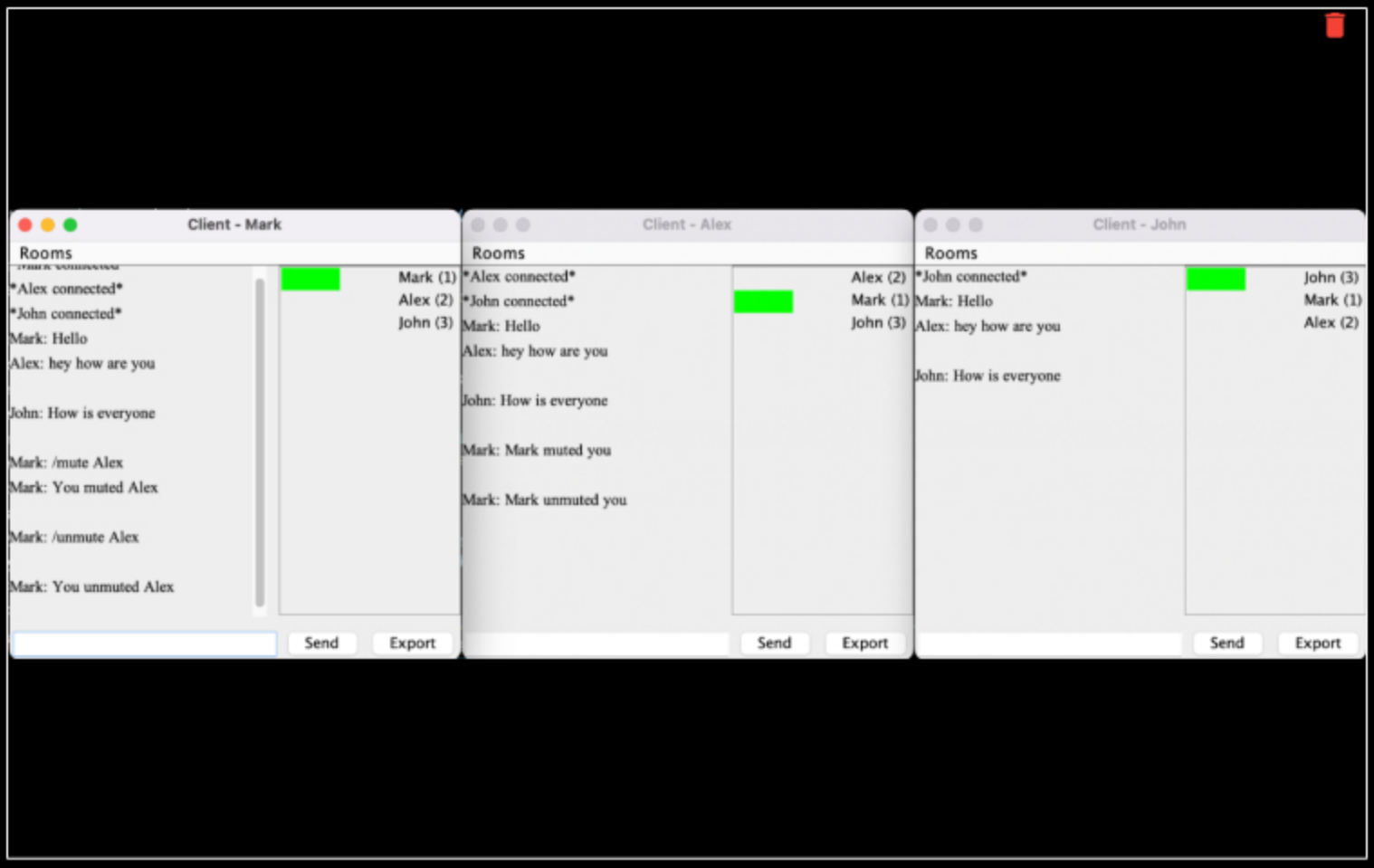


Client Mark mutes Alex and shows on his client window that Alex is now greyed out. (left client window mutes middle client window)

Checklist Items (2)

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

#3 Each screenshot should be clearly captioned



Client Mark unmutes client Alex and now grey box disappears on Mark's client window. (left client window unmutes middle client window)

Checklist Items (2)

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

#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 graying out muted clients, first piece of code that was done was creating new send methods in the Server Thread side name sendMuteClient and sendUnmuteClient in which would send over a payload over to client side once called in Room side. In the Room side, the send methods were called in the setMute and setUnmute methods after the loop checking for the list was done and would send over either the muted client'sID or the unmuted client's ID over to the client side. The client side would then receive the MUTE payload or UNMUTE payload and process them inside the

processPayload method with their respective case (MUTE or UNMUTE). Inside the case, there is a call to IClientEvents where MUTE calls the onReceiveMute and UNMUTE calls the onReceiveUnmute and these two methods are initialized first in IClientEvents and take in the clients ID and then is moved on to ClientUI. In ClientUI, there are two more methods create named the same onReceiveMute and onReceiveUnmute and call the method colorClients to the ChatPanel and for mute it take the muted client ID and set to true for isMuted and unmute is the opposite where it takes the unmuted client ID and isMuted is set to false. In ChatPanel, the method colorClients is made there with the arguments of client ID and a boolean isMuted for the true or false of mute or unmute. This in turn moves to the UserListPanel class and in there is created the method to get the components of userListArea and to cycle for an instance when the component c is of UserListItem and then sets the color by calling the method setClientColor with the client ID and isMuted boolean. UserListItem is finally implement and imported in Milestone4 and does the setColorClient method checking for who the client ID is first and when the boolean isMuted is set to true and if it is then it will have muted client and put a gray border box next to their name. If it is false which is unmuted, then the client will have that gray border box taken away from being next to their name by doing setVisible to false and a JPanel named colorClient was made and added to create the gray box and setVisible was set to true originally but would change depending on state of mute or unmute. For highlighting the last message sent, an edit was done in the ClientUI side by adding inside the onMessageReceive, a call to ChatPanel to do the method highlightLastMessage and takes the client ID only. Then from here is following the same process as the gray box for muted clients in which ChatPanel calls userListPanel to do method highlightLastMessage and in the UserListPanel class with the highlightLastMessage method, it gets the userListArea components setting it to component cs and checking when its an instance of UserListItem and sets it to UserListItem uli once its an instance of and calls the UserListItem method. Lastly, the UserListItem class contains the last method named setLastMessage which takes the client ID and gets the name and gives the client a green box to highlight that they were the last person to send a message and if they were not, then the box visibility is set to false so it is not seen. A JPanel was also created for this and was named lastMessage and was added and setVisible was set to false for this but would change depending on who sent a message last.

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/OmarBarrera1/ob75-IT114-002/pull/11

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

Some learnings that was obtained from this assignment was learning how to extract text specifically from a chat once a user goes to export their messages into a file. This proved to be helpful since only the messages would be exported

a user once to export their messages into a file. This proved to be helpful since only the messages would be exported with messages and not everything else like connection panels with the host and port numbers or trying to include the names of buttons and such that would make the file unreadable for viewing the messages. Also learning about highlighting messages or graying out specific clients proved to be able to clearly show who sent a message last or who is muted and unmuted to the client by displaying different types of colors to whatever state they are in like green for last message sent or gray for someone who is muted to the client. This assignment has been able to better my understanding on grabbing certain text or coloring in certain users for certain states.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

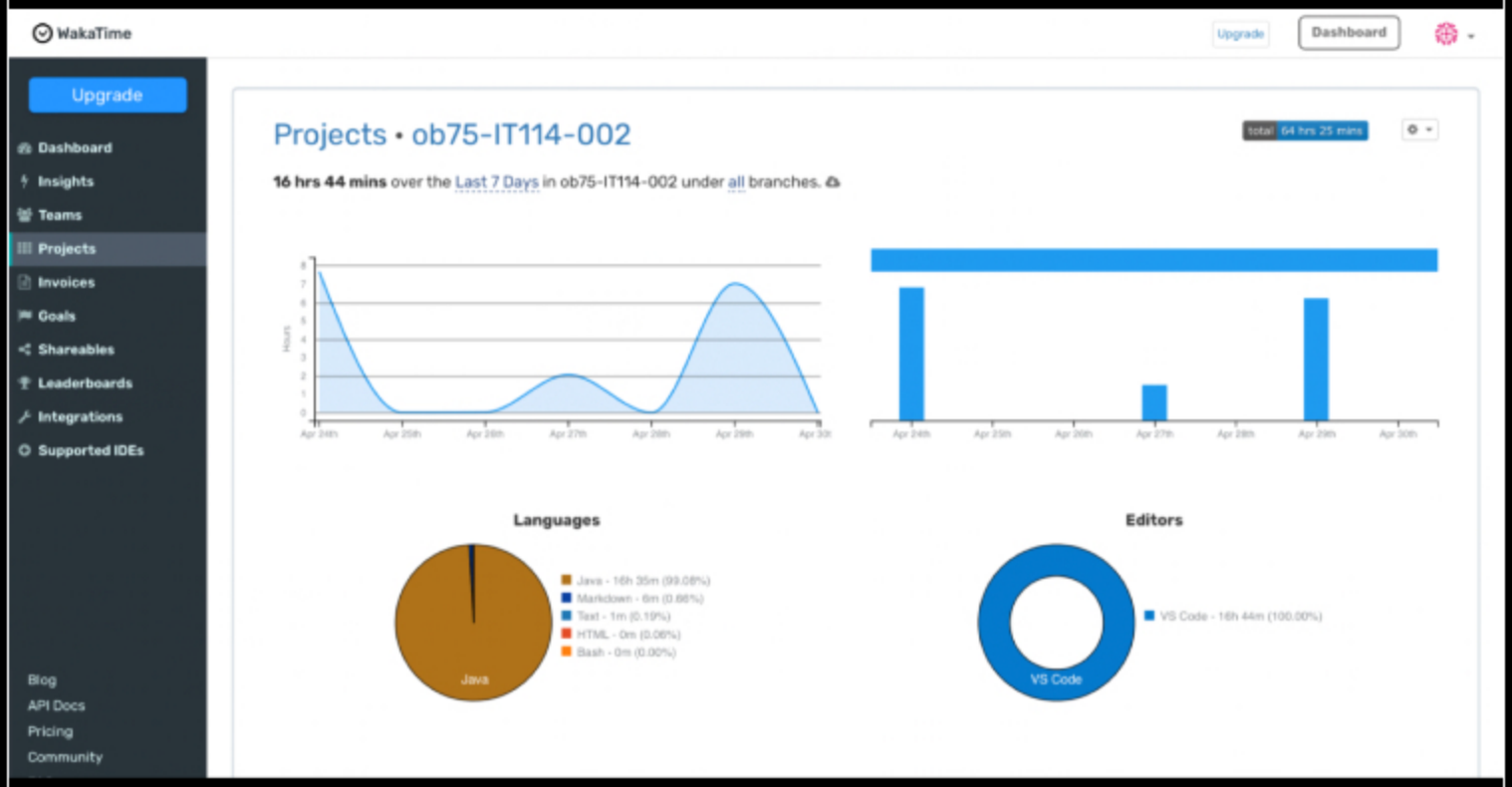
Details:

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

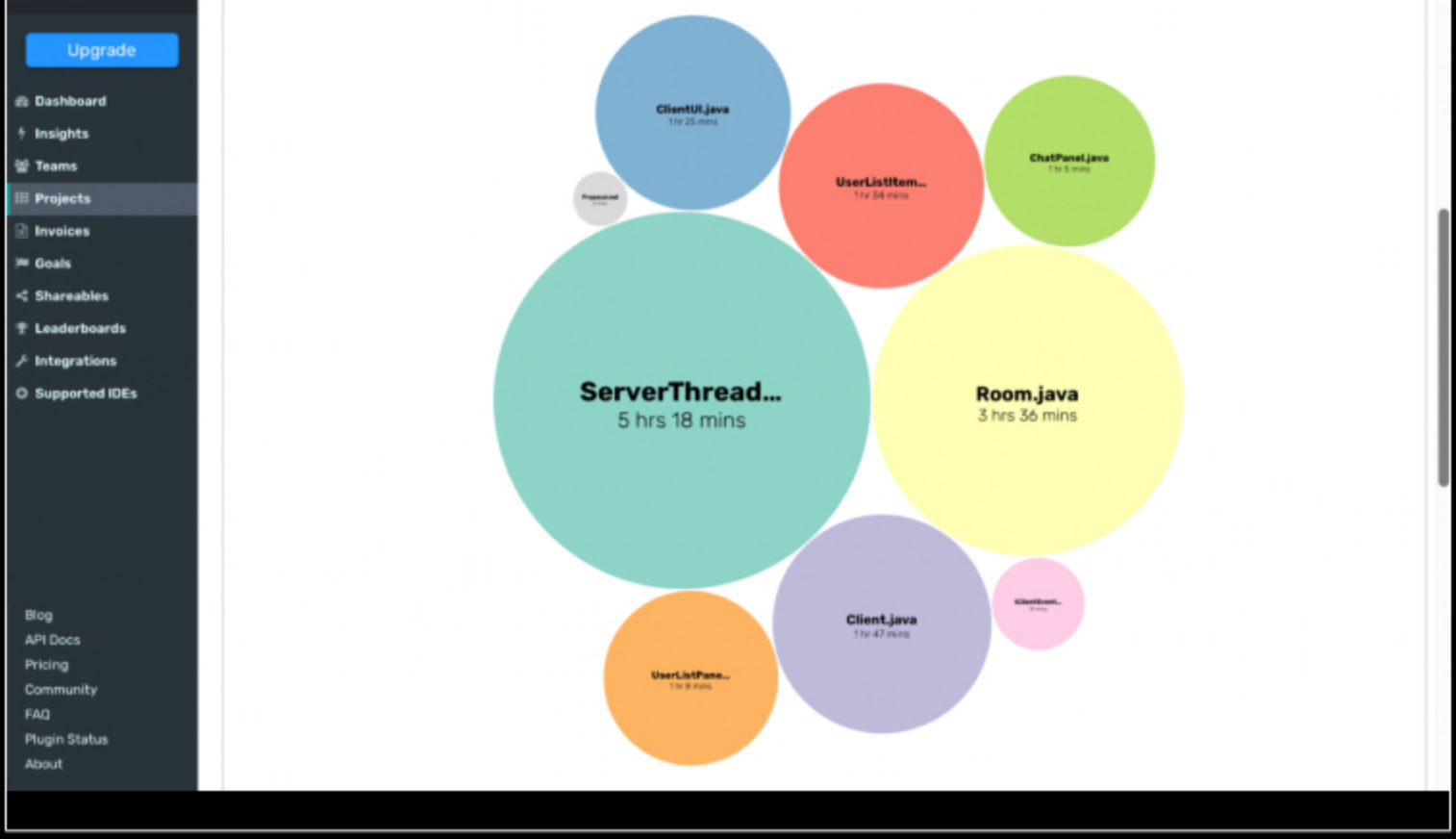
Task Screenshots:

Gallery Style: Large View

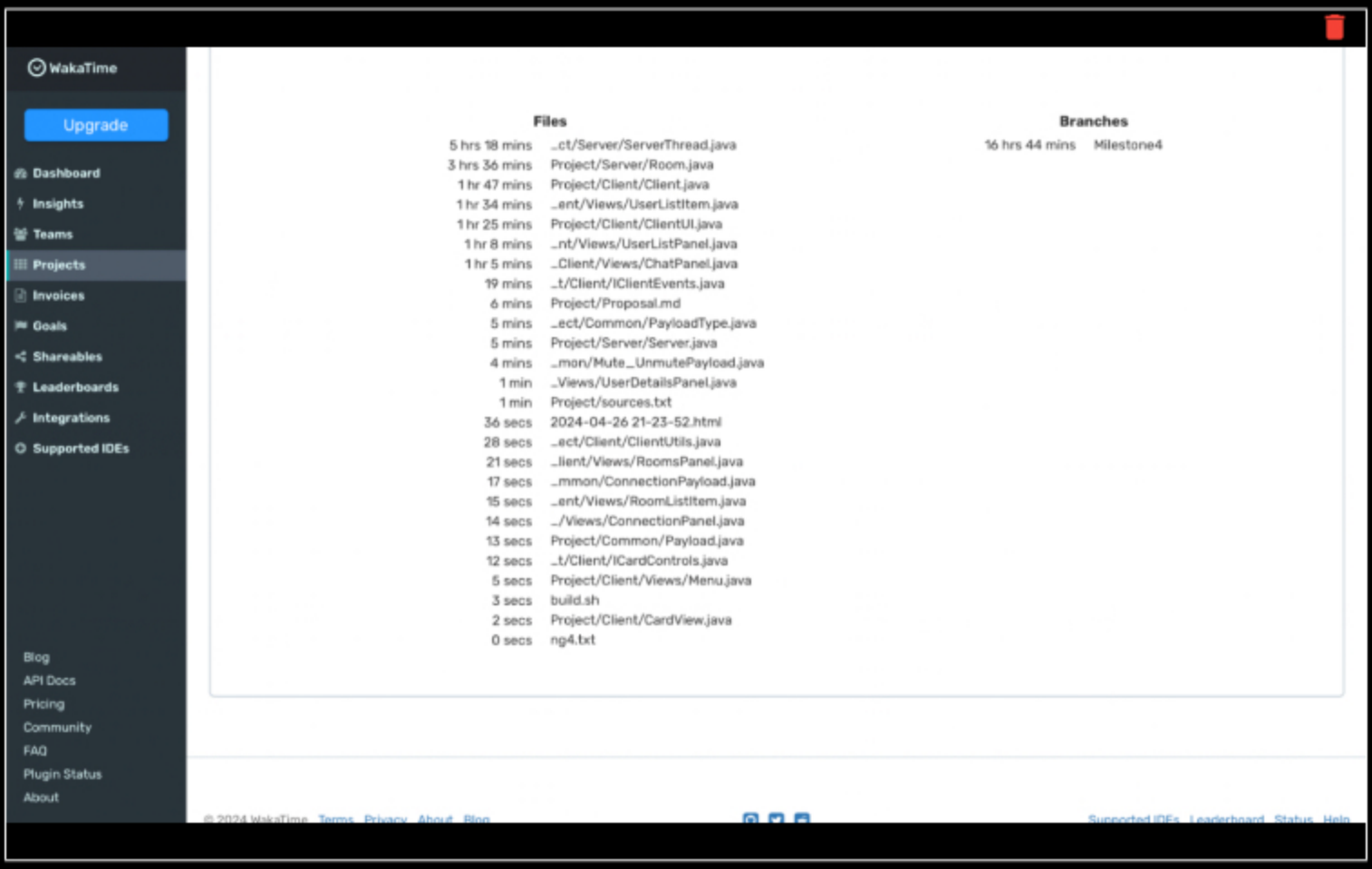
Small Medium Large



Wakatime top page showing project folder and total time worked on.



Wakatime middle page showing classes worked on in Milestone 4 with time.



Wakatime bottom page showing files worked on in Milestone 4 and total time in Milestone4 branch.

End of Assignment