Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-005-F2024/it114-milestone-4-chatroom-2024-m24/grade/kh465

Course: IT114-005-F2024
Assigment: [IT114] Milestone 4 Chatroom 2024 M24
Student: Keven H. (kh465)

Submissions:

Submission Selection

1 Submission [submitted] 12/11/2024 5:22:31 PM

Instructions

COLLAPSE

- Implement the Milestone 4 features from the project's proposal document: https://docs.google.com/document/d/1ONmvEvel97GTFPGfVwwQC96xSsobbSbk56145XizQG4/view
- 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

Group

81%

Group: Features
Tasks: 4
Points: 9

COLLAPSE

Task

100%

Group: Features
Task #1: Client can export chat history of their current session (client-side)
Weight: ~0%
Points: ~0.01

COLLAPSE

Details:
For this requirement it's not valid to have another list keep track of messages. The goal is to utilize the location where messages are already present.

This must be a client-side implementation.

Columns: 1

Sub-Task

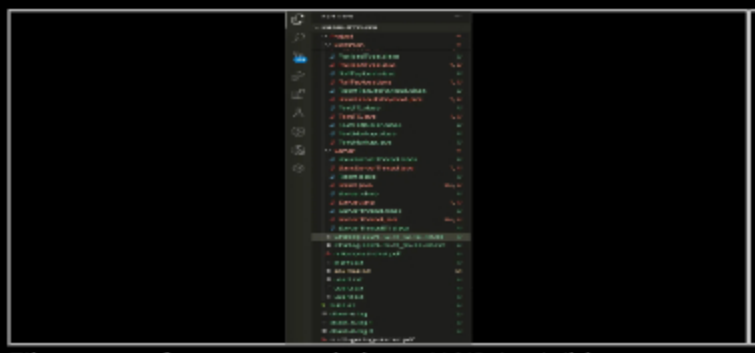
100%

Group: Features
Task #1: Client can export chat history of their current session (client-side)
Sub Task #1: Show a few examples of exported chat history (include the filename showing that there are multiple copies)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Filenames of two exported chats. UCID is visible

Chatlog exported on 12/11 at 12:10

Chatlog exported on 12/11 at 16:20

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

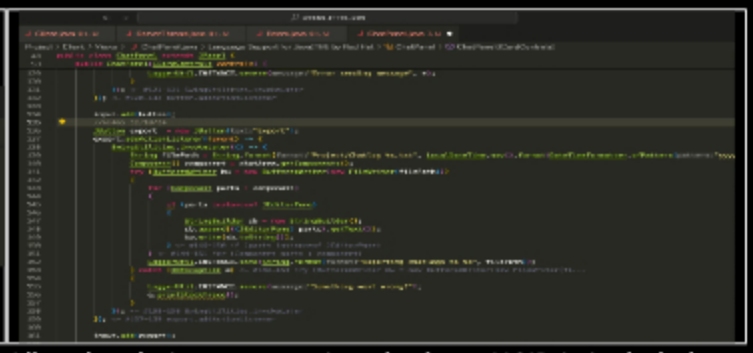
Group: Features
Task #1: Client can export chat history of their current session (client-side)
Sub Task #2: Show the code related to building the export data (where the messages are gathered from, the StringBuilder, and the file generation)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

textContainer is added to chatArea, it contains user messages. UCID is included



All code relating to exporting chatlogs. UCID is included

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

An export button was added beside the send message button. When this button is triggered, a component array gets all the components from chatArea and a BufferedWriter is created using a FileWriter saving to filePath (which takes the current date and time and formats it for easier reading). A for loop checks the components inside the components array, if there's an instanceof JEditorPane (which textContainer is), a StringBuilder is created, we get the text from textContainer, append it to the StringBuilder, then use BufferedWriter to write from StringBuilder using its toString() method.

Sub-Task

100%

Group: Features

Task #1: Client can export chat history of their current session (client-side)

Sub Task #3: Show the UI interaction that will trigger an export

Task Screenshots

Gallery Style: 2 Columns

4 2 1

ClientUI showing the export button. UCID is included

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain where you put it any why

Response:

I put the export button right beside the send button because I felt it was the easiest place to put it. Users may want to export their chats at any time and having a button right beside the send button encourages its use.

End of Task 1

Task

100%

Group: Features
Task #2: Client's Mute List will persist across sessions (server-side)
Weight: ~0%
Points: ~0.01

COLLAPSE

Details:
This must be a server-side implementation.

Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

Columns: 1

Sub-Task

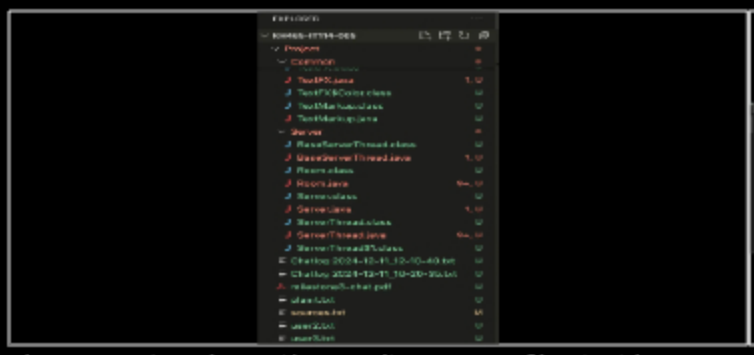
100%

Group: Features
Task #2: Client's Mute List will persist across sessions (server-side)
Sub Task #1: Show multiple examples of mutelist files and their content (their names should have/include the user's client name)

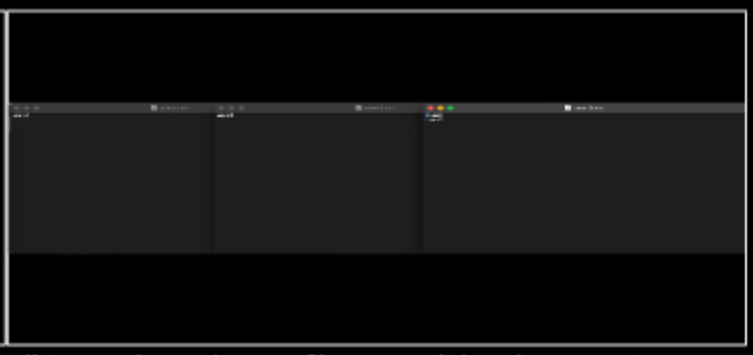
Task Screenshots

Gallery Style: 2 Columns

4 2 1



plamt, user2 and user3's mutelists as .txt files inside Project folder. UCID is included



All 3 users' mutelist .txt files consolidated to one pic. .txt files are named after users

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Features

Task #2: Client's Mute List will persist across sessions (server-side)

Sub Task #2: Show the code related to loading the mutelist for a connecting client (and logic that handles if there's no file)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Code in ServerThread for loading a mutelist. UCID is included

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

Upon receiving a CLIENT_CONNECT payload, ServerThread looks for the users' mutelist .txt file in the Project folder. A BufferedReader is created from the filePath, and a String is created to hold the current user. BufferedReader reads each line, and while it's not null it uses the current String held in user to add it to the muteList. If a user does not have a mutelist .txt file, a non-fatal error is thrown informing that it does not exist. This is because the file is created upon disconnect

Sub-Task

100%

Group: Features

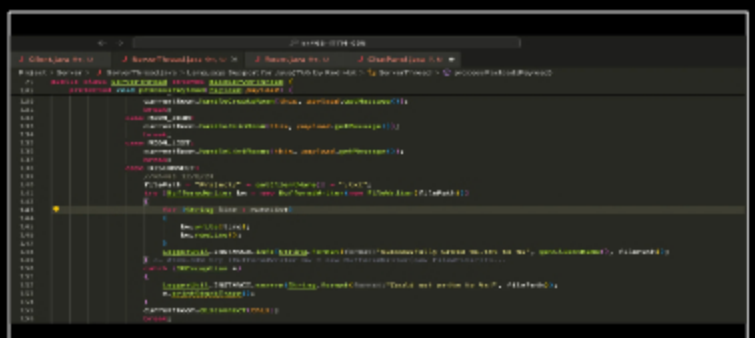
Task #2: Client's Mute List will persist across sessions (server-side)

Sub Task #3: Show the code related to saving the mutelist whenever the list changes for a client

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Mutelist updating upon client disconnect. UCID is included

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

I did not have the mutelist save every time there is a change in the mutelist. Instead, when the client disconnects, their mutelist is updated via BufferedWriter to the very same filePath used when a client connects. This overwrites the previous file, if the user has not changed their mutelist, nothing happens. If a user does change their mutelist during that session, it is tracked the next time they disconnect.

End of Task 2

Task

100%

Group: Features

Task #3: Clients will receive a message when they get muted/unmuted by another user

Weight: ~0%

Points: ~0.00

COLLAPSE

Details:

Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

I.e., /mute Bob followed by a /mute Bob should only send one message because Bob can only be muted once until

Columns: 1

Sub-Task

100%

Group: Features

Task #3: Clients will receive a message when they get muted/unmuted by another user

Sub Task #1: Show the code that generates the well formatted message only when the mute state changes (see notes in the details above)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Code that handles informing a user of being

muted/unmuted. UCID is included

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

A boolean exists in ServerThread called isMuted. Inside both the mute and unmute method there are checks to see if the user is muted when trying to mute a user, and vice versa for unmuting. If this returns true, we return and no action is performed. If false, sendMessage informs the muted user that they've been muted and by whom.

Sub-Task

100%

Group: Features

Task #3: Clients will receive a message when they get muted/unmuted by another user

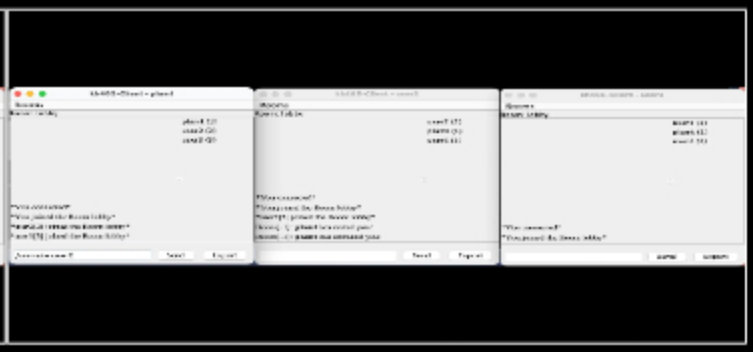
Sub Task #2: Show a few examples of this occurring and demonstrate that two mutes of the same user in a row generate only one message, do the same for unmute)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

plamt muting user2. Text was added back in for clarity. Sending this message results in nothing



plamt unmuting user2. Text was added back in for clarity. Sending this messages results in nothing

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 3

Task

25%

Group: Features

Task #4: The user list on the Client-side should update per the status of each user

Weight: ~0%

Points: ~0.01

COLLAPSE

Details:

Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

Columns: 1

Sub-Task

0%

Group: Features
Task #4: The user list on the Client-side should update per the status of each user
Sub Task #1: Show the UI for Muted users appear grayed out (or similar indication of your choosing) include a few examples showing it updates correctly when changing from mute/unmute and back

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Missing Caption

Caption(s) (required)
Caption Hint: *Describe/highlight what's being shown*
Missing caption(s)

Sub-Task

50%

Group: Features
Task #4: The user list on the Client-side should update per the status of each user
Sub Task #2: Show the code flow (client receiving -> UI) for Muted users appear grayed out (or similar indication of your choosing)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Missing Caption

Caption(s) (required)
Caption Hint: *Describe/highlight what's being shown*
Missing caption(s)

Task Response Prompt

Explain in concise steps how this logically works

Response:

I did not attempt this task since I have run out of time, and I am not quite sure how I would approach this problem.

Sub-Task

0%

Group: Features

Task #4: The user list on the Client-side should update per the status of each user

Sub Task #3: Show the UI for Last person to send a message gets highlighted (or similar indication of your choosing)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Missing Caption

Caption(s) (required)

Caption Hint: *Describe/highlight what's being shown*

Missing caption(s)

Sub-Task

50%

Group: Features

Task #4: The user list on the Client-side should update per the status of each user

Sub Task #4: Show the code flow (client receiving -> UI) for Last person to send a message gets highlighted (or similar indication of your choosing)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Missing Caption

Caption(s) (required)

Caption Hint: *Describe/highlight what's being shown*

Missing caption(s)

Task Response Prompt

Explain in concise steps how this logically works

Response:

I did not attempt this task as I have run out of time, but I believe I understand how it would be done. In UserListPanel, there would need to be a method that listens for messages in ChatPanel. When a message is received in ChatPanel, it would check who sent the message potentially as a boolean. UserListPanel would then highlight the user and update everyone else when this happens so all users are not highlighted after sending a message

End of Task 4

End of Group: Features
Task Status: 3/4

Group

100%

Group: Misc
Tasks: 3
Points: 1

COLLAPSE

Task

100%

Group: Misc
Task #1: Add the pull request link for the branch
Weight: ~33%
Points: ~0.33

COLLAPSE

Details:
Note: the link should end with /pull/#

Task URLs

URL #1
https://github.com/kh465/kh465-IT114-005/pull/15

URL
https://github.com/kh465/kh465-IT114-005/pull/

End of Task 1

Task

100%

Group: Misc
Task #2: Talk about any issues or learnings during this assignment
Weight: ~33%
Points: ~0.33

COLLAPSE

Task Response Prompt

Response:

I had issues with the highlighting of users when they send messages or mute/unmute users. Unfortunately I have run out of time to work through these issues, but I would like to learn about how this would work.

End of Task 2

Task

100%

Group: Misc
Task #3: WakaTime Screenshot
Weight: ~33%
Points: ~0.33

COLLAPSE

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: 2 Columns

4 2 1

WakaTime showing the last 7 days

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment