Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-451-M2024/it114-milestone-4-chatroom-2024-m24/grade/sa2796

IT114-451-M2024 - [IT114] Milestone 4 Chatroom 2024 M24

Submissions:

Submission Selection

1 Submission [active] 7/22/2024 10:33:23 AM

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

Tasks: 7 Points: 10.00

Features (9 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 3

Text: Client can export chat history of their current session (client-side)

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.
A StringBuilder must be used for consolidation.
Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

#1) Show a few examples

Caption (required) ✓
Describe/highlight what's being shown
Displaying a few examples of exported chat history (include the filename showing that there are multiple copies)

#2) Show the code related to

Caption (required) ✓
Describe/highlight what's being shown
Displaying the code related to building the export data (where the messages are gathered from, the StringBuilder, and the file)

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
We first initiliaze stringbuilder using Stringbuilder chatHistory = new Stringbuilder(); and it creates a object that stores the chat history.

The text is collected that is looped through chatArea and appends the text from JEditorPane to

#3) Show the UI interaction

Caption (required) ✓
Describe/highlight what's being shown
Displaying the UI interaction that will trigger an export

Explanation (required)
✓
Explain where you put it any why
PREVIEW RESPONSE
Firstly, the menu gets initialized using JMenu = fileMenu = new JMenu("File"). This file menu has the menu items which now include the export chat option. There's also JMenuItem exportChatMenuItem = new JMenuItem("Export Chat") which creates the new menu item "Export Chat".

There's also an action listner so when the

Stringbuilder.

JFileChooser is created to save text files and bufferedwritter is used to write the collected text to file.

Once finished, a confirmation message is shown to indiciate a valid export.

menu item is clicked, it uses the exportchathistory method.

COLLAPSE

Task #2 - Points: 3

Text: Client's Mute List will persist across sessions (server-side)

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.

#1) Show multiple examples

Caption (required) ✓
Describe/highlight what's being shown
Displaying multiple examples of mutelist files and their content (their names should have/include the user's client name)

#2) Show the code related to

Caption (required) ✓
Describe/highlight what's being shown
Displaying the code related to loading the mutelist for a connecting client (and logic that handles if there's no file)

Explanation (required)

#3) Show the code related to

Caption (required) ✓
Describe/highlight what's being shown
Displaying the code related to saving the mutelist whenever the list changes for a client

Explanation (required)
✓
Explain in concise steps

✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The file is initialiazed and a file object is created called <clientname>mutelist.txt. The file is then checked if it exists and using BufferedReader, it reads the file. Each line represents a muted user and that muted user is added to the mutelist.

how this logically works

PREVIEW RESPONSE

Similarly to the load mute list, The file is initialiazed and a file object is created called <clientname>mutelist.txt. There is also the for loop, for(String mutedUsername:mutelist), it iterates over each username in mutelist. Then it writes each username to file and logs that mutelist is saved successfully.

COLLAPSE

Task #3 - Points: 1

Text: Clients will receive a message when they get muted/unmuted by another user

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 they're unmuted. Similarly for /unmute Bob

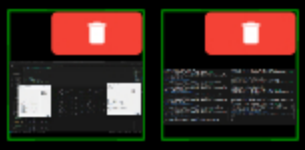
#1) Show the code that

Caption (required) ✓

Describe/highlight what's being shown

Displaying the code that generates the well formatted message only when the mute state changes.

#2) Show a few examples



Caption (required) ✓

Describe/highlight what's being shown

Doing /mute and /mute twice only the muter, in this case, Lamine, sees "Randy is already muted" and "Randy is not muted".

Explanation (required)

✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

For both handleMute and handleUnmute, it exracts target username and tries to find target client and if it exists, it will for handleMute check to see if client is already muted by sender and if it is, a message is sent to sender saying that the said user is already muted. They are then added to mute list and sends a confirmation message to sender and client about mute.

For handleUnmute its a similar process except if the target client is muted by sender, then it removes client from the mute list and sends confirmation message to both sender and client about unmute.

COLLAPSE

Task #4 - Points: 3

Text: The user list on the Client-side should update per the status of each user

Details:

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

#1) Show the UI for Muted

#2) Show the code flow (client

#3) Show the UI for Last person

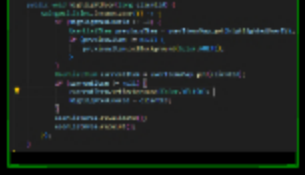
#4) Show the code flow (client

Caption (required) ✓
Describe/highlight what's being shown
Displaying UI for Muted users appear grayed out show some examples showing it updates correctly when changing from mute/unmute

Caption (required) ✓
Describe/highlight what's being shown
Displaying the code flow (client receiving -> UI) for Muted users appear grayed out (or similar indication of your choosing)

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
We first get the user list item using userlistitem item = useritemsmap.get(clientid).We then, check if the item exists and based on the user's status if they are muted, the text color witll be set to gray. If user isn't muted the text color will be black indicating that they are unmuted.

Caption (required) ✓
Describe/highlight what's being shown
Displaying the UI for Last person to send a message gets highlighted (or similar indication of your choosing)



Caption (required) ✓
Describe/highlight what's being shown
Displaying the code flow (client receiving -> UI) for Last person to send a message gets highlighted

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The highlight user method checks if there's a user highlighted by seeing if highteduserid isn't equal to -1 . If the user is highlighted, it gets userlistitem along with highlighteduserid from useritemsmap. If userlistitem exists, the background color changes to yellow because of new user and are now highlighted.

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/SHUAIB2796/sa2796-IT114-4513

https://github.com/SHUAIB2796/sa2796-IT114-4

+ ADD ANOTHER URL

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

The main issue I had during this assignment was particularly getting the grayed out option to appear for task #4 but I have provided the code along with a screenshot demonstration of the issue so Professor Toegel and Lucas can see where the issue could have been. Other than that, I was ok doing the other three tasks with no problem.

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

VSCode Wakatime

Files

3 hrs 21 mins	Project/Server/Room.java
1 hr 51 mins	...ct/Server/ServerThread.java
1 hr 33 mins	...nt/Views/UserListPanel.java
1 hr 28 mins	Project/Client/ClientUI.java
1 hr 10 mins	...Client/Views/ChatPanel.java
53 mins	Project/Client/Client.java
35 mins	...ent/Views/UserListItem.java
24 mins	...Views/UserDetailsPanel.java
15 mins	Project/Server/Server.java

Branches

12 hrs 11 mins	Milestone4
41 mins	main
17 mins	Unknown
8 mins	Milestone3

Files and Branch times

Projects • sa2796-IT114-451

13 hrs 19 mins over the Last 7 Days in sa2796-IT114-451 under all branches.

Overall time on Project

End of Assignment