Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-450-M2024/it114-milestone-4-chatroom-2024-m24/grade/vk686

IT114-450-M2024 - [IT114] Milestone 4 Chatroom 2024 M24

Submissions:

Submission Selection

1 Submission [active] 7/26/2024 6:14:07 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

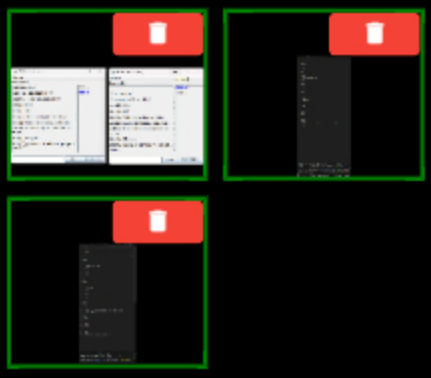
Task #1 - Points: 3

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 of exported chat history (include



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

#2) Show the code related to building the export data (where the

Caption (required) ✓
Describe/highlight what's being shown
Show the code related to building the export data

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The exportChatHistory method checks each component to see if it is a JEditorPane and then gathers all the messages from the chat area. The text from these components then goes into a StringBuilder. Once it has all the messages, it uses the current date and time to make a unique file name and, it use a BufferedWriter to save the chat history to a text file. If there are any errors during this process, they are logged.

#3) Show the UI interaction that will trigger an export

Caption (required) ✓
Describe/highlight what's being shown
Showing the UI interaction that triggers an export

Explanation (required) ✓
Explain where you put it any why
PREVIEW RESPONSE
The "Export Chat" button is added to the input panel at the bottom of the chat panel, alongside the text field and the "Send" button.
When I click this button("Export Chat"), it triggers the exportChatHistory method, which collects all the chat messages and saves them to a text file.

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 of

Caption (required) ✓
Describe/highlight what's being shown
Not working

#2) Show the code related to loading

Caption (required) ✓
Describe/highlight what's being shown
Not working

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
Not working

#3) Show the code related to saving the

Caption (required) ✓
Describe/highlight what's being shown
Not working

Explanation (required)
Explain in concise steps how this logically works
PREVIEW RESPONSE
Missing text

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

#2) Show a few

generates

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

Explanation (required)
Explain in concise steps how this logically works
PREVIEW RESPONSE
Missing text

examples of

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

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 users appear

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

#2) Show the code flow (client

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

Explanation (required)
Explain in concise steps how this logically works
PREVIEW RESPONSE

#3) Show the UI for Last person to

Caption (required) ✓
Describe/highlight what's being shown
Showing the UI for Last person to send a message gets highlighted

#4) Show the code flow (client

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

Missing text

highlighted

Explanation (required)

✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

When a message is received, the Client class then calls the lastUser method of ClientUI. ClientUI instructs UserListPanel to highlight the user who sent the last message. UserListPanel goes through its UserListItem components, finds the one that matches the sender's client ID, and changes its text color to blue using setForeground, while resetting all other items to black. This makes the last message sender stand out in the user list.

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/VK686NJ/vk686-IT114-450/pull/15

URL
https://github.com/VK686NJ/vk686-IT114-450/p

+ ADD ANOTHER URL

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

During this assignment, I successfully added a button to export the chat history and a feature to highlight the last person who sent a message.

While working on these features, I encountered some difficulties in handling the UI components and ensuring that the messages were collected correctly.

There were two features I couldn't complete: saving the mute list to a file and sending notifications when users are muted or unmuted. I tried to implement them many times but could not get them to work as expected. I am short on time and am afraid that I will not have time to pass another 202 class. So I am submitting the result as is for now, unfortunately. This is not an excuse, but rather an explanation.

Overall, this project was a valuable learning experience. I improved my skills in managing UI components and handling file operations in Java. The professor's lectures helped me a lot. Thanks

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

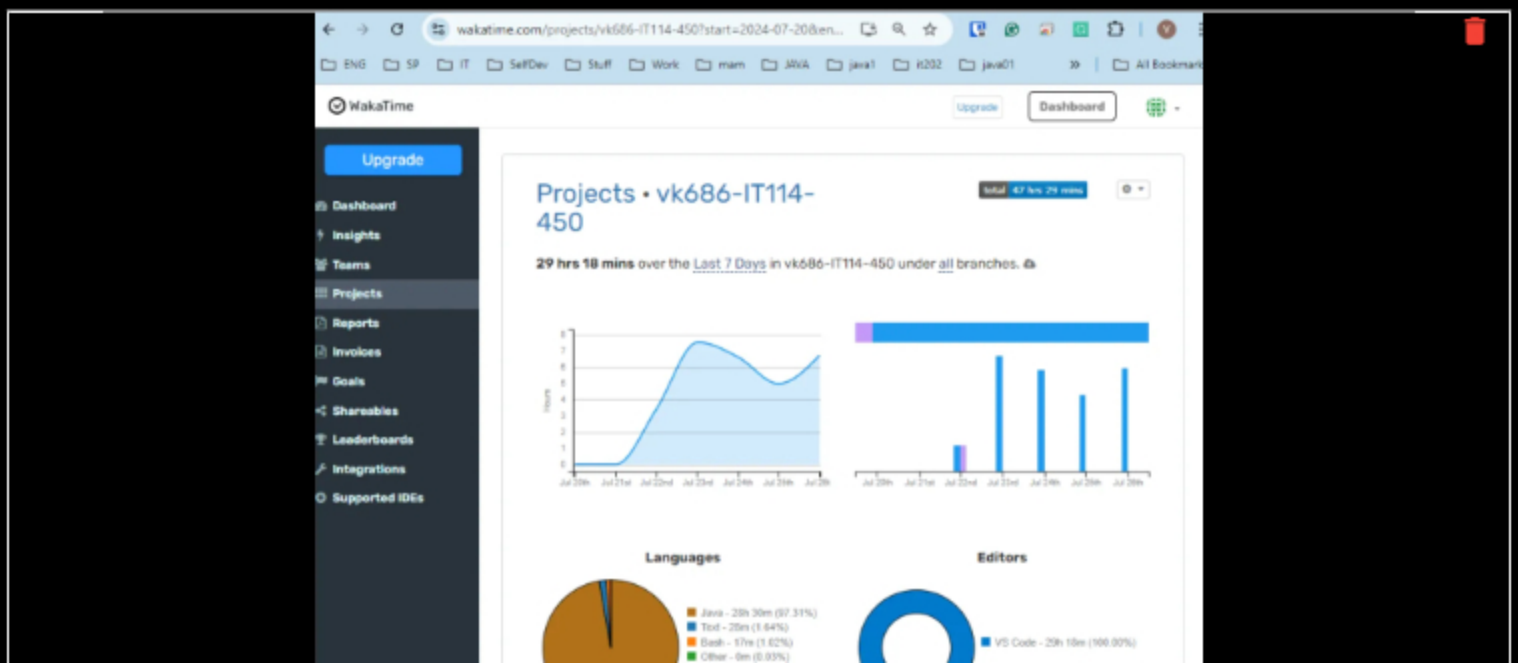
Details:

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

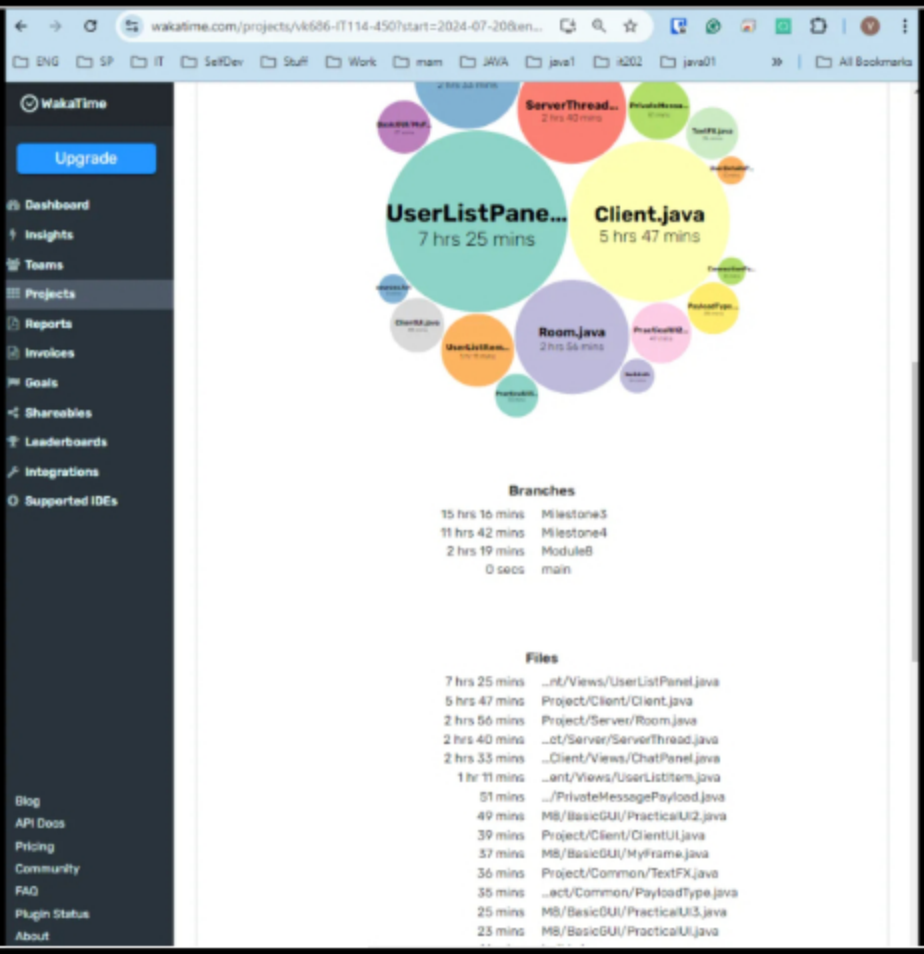
Task Screenshots:

Gallery Style: Large View

Small Medium Large



WakaTime



WakaTime

End of Assignment