Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-451-M2024/it114-milestone-3-chatroom-2024-m24/grade/sa2796

IT114-451-M2024 - [IT114] Milestone 3 Chatroom 2024 M24

Submissions:

Submission Selection

1 Submission [active] 7/11/2024 7:00:54 PM

Instructions

COLLAPSE

Implement the Milestone 3 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 Milestone3 branch Create a pull request from Milestone3 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: Milestone3

Tasks: 8 Points: 10.00

Basic UI (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: UI Panels

Details:

All code screenshots must include ucid/date.

App screenshots must have the UCID in the title bar like the lesson gave.

#1) Show the ConnectionPane

Caption (required) ✓
Describe/highlight what's being shown
Displaying the ConnectionPanel by running the app (should have host/port)

#2) Show the code related to

Caption (required) ✓
Describe/highlight what's being shown
Displaying code related to the ConnectionPanel

Explanation (required) ✓
Briefly explain how it works and how it's used
PREVIEW RESPONSE
For the host input, a label and text field for host adress hostValue is added and a label to display error messages is hidden which is hostError. it is the same case for port input except there is portValue and portError.

There is also a next button which when the port number is parsed and if it's incorrect an error message is dispalyed and button is blocked. If inputs are correct, the panel switches to next screen.

#3) show the UserDetailsPane

Caption (required) ✓
Describe/highlight what's being shown
Displaying the UserDetailsPanel by running the app (should have username)

#4) Show the code related to

Caption (required) ✓
Describe/highlight what's being shown
Displaying the code related to the UserDetailsPanel

Explanation (required) ✓
Briefly explain how it works and how it's used
PREVIEW RESPONSE
To start, for the username input field, a label is added and a text field to enter username. There also is a error label which is only prompted when a username is not given.

There is a previous and connect button added in which the previous button sends users back to the previous panel and connect button gets the username and makes sure it isn't empty and if it is correct, it sends to connection panel

#5) Show the ChatPanel

#6) Show the code related to

Caption (required) ✓
Describe/highlight what's being shown
Displaying the ChatPanel (there should be at least 3 users present and some example messages)

Caption (required) ✓
Describe/highlight what's being shown
Displaying the code related to the ChatPanel

Explanation (required) ✓
Briefly explain how it works and how it's used (note the important parts of the ChatPanel)

PREVIEW RESPONSE

The removeUserListItem removes a user from the user list by their client ID and clearUserList() clears the entire user list. For messages to display, addText(String text) is added so new messages can appear in chat area and it displays proper text formatting features.

To sum up, for adding messages, when a new message is recieved, addText is called to display the message to chat area and to manage users the methods removeUserListitem and clearUserList are used to update user list by the means of removing specific user

Build-up (3 pts.)

^COLLAPSE ^

^COLLAPSE ^

Task #1 - Points: 1

Text: Results of /flip and /roll appear in a different format than regular chat text

Details:
All code screenshots must include ucid/date.

App screenshots must have the UCID in the title bar like the lesson gave.

#1) Show examples of it

Caption (required) ✓
Describe/highlight what's being shown
Displaying examples of /roll and /flip

#2) Show the code on the Room

Caption (required) ✓
Describe/highlight what's being shown
Displaying the code on the Room side that changes this format

Explanation (required) ✓
Explain what you did and how it works
PREVIEW RESPONSE
for the handleFlip method, to determine the flip result, i used random.nextBoolean() to randomly determine result of either heads or tails and the message is formatted to include sender name and flip result. sendMessage(sender:null, message) sends formatted message to all clients in the room.

For handleRoll method, the dice information is retrieved from Dicenumber and Sidesnumber from payload and then I construct a Stringbuilder to build result message with sender's name and number of dice/sides. the dice roll generates random result for each die and appends it to resultMessage and a total is tracked. The message is sent similarly to the handleFlio method and is displayed to all clients in room.

^COLLAPSE ^

Task #2 - Points: 1

Text: Text Formatting appears correctly on the UI

Details:
All code screenshots must include ucid/date.

App screenshots must have the UCID in the title bar like the lesson gave.

#1) Show examples of bold

Caption (required) ✓
Describe/highlight what's being shown
Displaying examples of bold, italic, underline,

#2) Show the code changes

Caption (required) ✓
Describe/highlight what's being shown
Displaying the code changes necessary to get this to work

each color implemented and combination of bold, italic, underline, and one color

Explanation (required)

✓

Briefly explain what was necessary and how it works

PREVIEW RESPONSE

The code changes that were necessary took place in both room.java and chatpanel.java. For room.java, I had to implement the processTextEffects method which processes the message to apply formatting commands before it's sent to clients in the room. It replaces patterns in the text with HTML tags. Regular experssions are used to search for custom tags and it's replaced with HTML tags. It processes styles like bold, italics, and underline and once message is processed, it is returned and ready to be sent out to clients

In chatpanel.java, we simply create a JEditorPane with it set to text/html which allows HTML formatting.

New Features (4 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Private messages via @username

Details:

All code screenshots must include ucid/date.

App screenshots must have the UCID in the title bar like the lesson gave.

- **Note**: This will not be a slash command
- **Note**: The writer and the receiver are the only two that will receive the message from the server-side
- It's not valid to just hide it on the client-side (i.e., data must not be sent from the server-side)
- The Client-side will capture the message/target, find the appropriate client id, and send that along with the original message to the server-side
  - If a client id isn't found for the target, a message will be shown to the Client stating so and will not cause a payload to be sent to the server-side
- The ServerThread will receive this payload and pass the id and message to the Room
- The Room will match the id to the respective target and send the message to the sender and target (receiver)

#1) Show a few examples

Caption (required) ✓

Describe/highlight what's being shown

Displaying a few examples across different clients (there should be at least 3 clients in the Room)

#2) Show the client-side code

Caption (required) ✓

Describe/highlight what's being shown

Displaying the client-side code that processes the text per the requirement

Explanation (required)

✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

For the sendPrivateMessage method, it creates a new payload and the payload type is set to MESSAGE which has the message content, target username and isPrivate set. The payload is then sent using the send(p) method.

#3) Show the ServerThread

Caption (required) ✓

Describe/highlight what's being shown

Displaying the ServerThread code receiving the payload and passing it to Room

Explanation (required)

✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

For sendMessage method, it sends a message to client which includes sender's ID and if the message is private or not.

For processPayload, the method processes incoming payloads from client and uses switch statemnt to handle

#4) Show the Room code that

Caption (required) ✓

Describe/highlight what's being shown

Displaying the Room code that verifies the id and sends the message to both the sender and receiver

Explanation (required)

✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The target username is acquired from payload using payload.getTargetUsername() and the message is also acquired by using payload.getMessage(). for the text effects, the method that was used

The processClientCommand method first checks to see if text starts with @ to detect private messages and it splits the username and message. If the username and message is formatted properly, it calls sendPrivateMessage and if it is incorrect it prints an error message

different payload types. In this case, if it's a pricate message it calls the currentRoom.handlePrivate payload).

In short, serverthread recieves payload from client and processPayload method is called and checks the payloade type via switch. For

was processTextEffects(message). if a targetclient is found, then a private message includes a whisper and targetclient.sendmessage is used to send a private message to target client. sender.sendmessage sends the private message back to sender for confirmation.

COLLAPSE

Task #2 - Points: 1

Text: Mute and Unmute

Details:

All code screenshots must include ucid/date.
App screenshots must have the UCID in the title bar like the lesson gave.

- Client-side will implement a /mute and /unmute command (i.e., /mute Bob or /unmute Bob)
  - Client side grabs the target and finds the client id related to the target
    - If no target found, an appropriate message will be displayed and no message will be sent
    - If a target is found, the id will be sent in a payload to the server-side with the appropriate action
- ServerThread will receive the payload and extract the data, then pass it to a Room method
- The Room will confirm the id against the list of clients
  - If found, it'll record the client's name on a list of the sender's ServerThread for a mute, otherwise it'll remove the name from the list
    - **Note**: This list must be unique and must not be directly exposed, the ServerThread must provide method accessors like add()/remove()
  - Upon success mute/unmute, the sender should receive a confirmation of the action clearly stating what happened
- Any time a message would be received (i.e., normal messages or private messages) the sender's name will be compared against the receiver's mute list
  - **Note**: The mute list won't be exposed directly, there should be a method on the ServerThread that takes the name and returns a boolean about whether or not the person is muted
  - If the user is muted, the receive must not be sent the message (i.e., they get skipped)
    - You must log in the terminal that the message was skipped due to being muted, but no message should be sent in this regard

#1) Show a few examples

#2) Show the client-side code

#3) Show the ServerThread

#4) Show the Room code that

Caption (required) ✓
Describe/highlight what's being shown
mute a user muted user sends a msg All except A get the msg A unmute the user previously muted user sends msg all users sees msg

Caption (required) ✓
Describe/highlight what's being shown
Displaying the client-side code that processes the text per the requirement

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The client checks if text starts with /mute or/mute and if it is detected it initiates the commands.

for the mute command, the text after /mute gets trimmed and is done by text.replace ("/mute", "").trim().
It is similar for the unmute command as well.
The sendMuteCommand is called with target username and it also goes for the unmute. They create a payload object and is set to the payload types mute or unmute and is sent to server

Caption (required) ✓
Describe/highlight what's being shown
Displaying the ServerThread code receiving the payload and passing it to Room

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
For the processPayload, when mute/unmute payload is recieved the method gets the data and calls handleMute or handleUnmute in Room class.

handleMute or handleUnmute methods update the mute list which is in serverthread that is a set of client ID's that are muted. Prior to a message being sent to a client, it is checked that the sender is in the mute list of the reciever and if the sender's muted, the message isn't sent to reciever.

Caption (required) ✓
Describe/highlight what's being shown
Displaying the Room code that verifies the id and add/removes the muted name to/from the ServerThread's list

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The methods handleMute and handleUnmute extract target username from payload using payload.getTargetUsername().

for handleMute, if client is found, the targer client is added to sender's mute list via sender.addtomutelist(targetClien
The similar process is done for handleunmute but instead it is sender.removefrommutelist.

Regardless of the method, they both send messages back to sender to let them know if it was successful or not for the mute/unmute.

#5) Show the Room code that

#6) Show terminal supplemental

Caption (required) ✓
Describe/highlight what's being shown
Displaying the Room code that checks the mute list during send message. private message, and any other relevant location

Caption (required) ✓
Describe/highlight what's being shown
Displaying terminal supplemental evidence per the requirements (refer to the details of this task

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

PREVIEW RESPONSE

There are two sendMessage methods that are overloaded with the first one calling the second one that has an extra parameter "isPrivate" that is set to false.

The second sendMessage method starts checking if room is running and if it isn't it blocks action and returns.

For each client, it checks if sender's muted by client using client.isMuted(senderId).If sender is muted by client, the message is skipped and if sender is not muted by client, the message is sent to client by calling client.sendMessage(senderId, messageTosend[0].

isPriv

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-

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

+ ADD ANOTHER URL

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

I had a couple of issues being that I couldn't get the client up and running at first but realized I had couple of errors left for my files such as client.java, room.java, and more that got resolved from getting the correct imports in and by tweaking the code for the files. I also had some trouble getting the actual UI to show up properly but I managed to tweak the files like clientui.java and client.java which had resolved the problems. I had a lot of guidance from Professor Toegel along the way that had helped me throughout this milestone 3.

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:

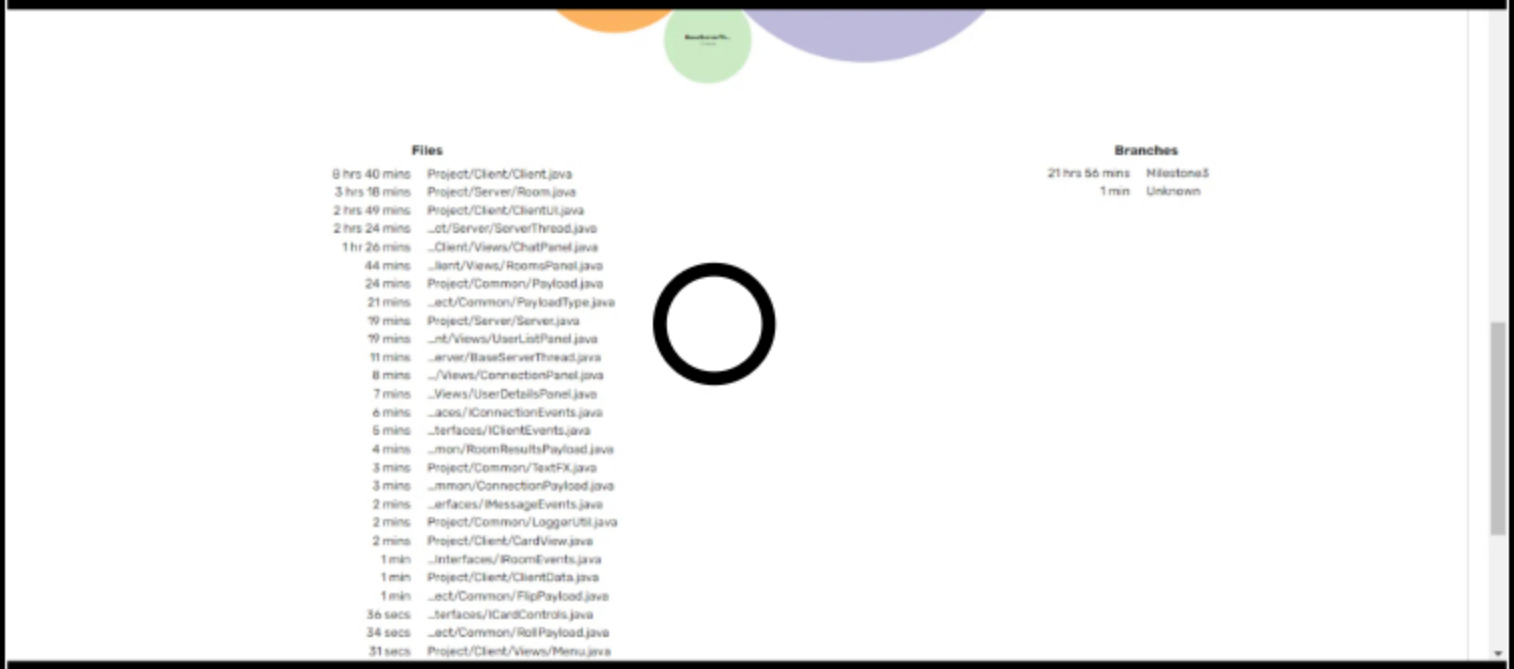
Gallery Style: Large View

Small Medium Large

Projects • sa2796-IT114-451

21 hrs 57 mins over the Last 7 Days in sa2796-IT114-451 under all branches.

VSCode time



Files

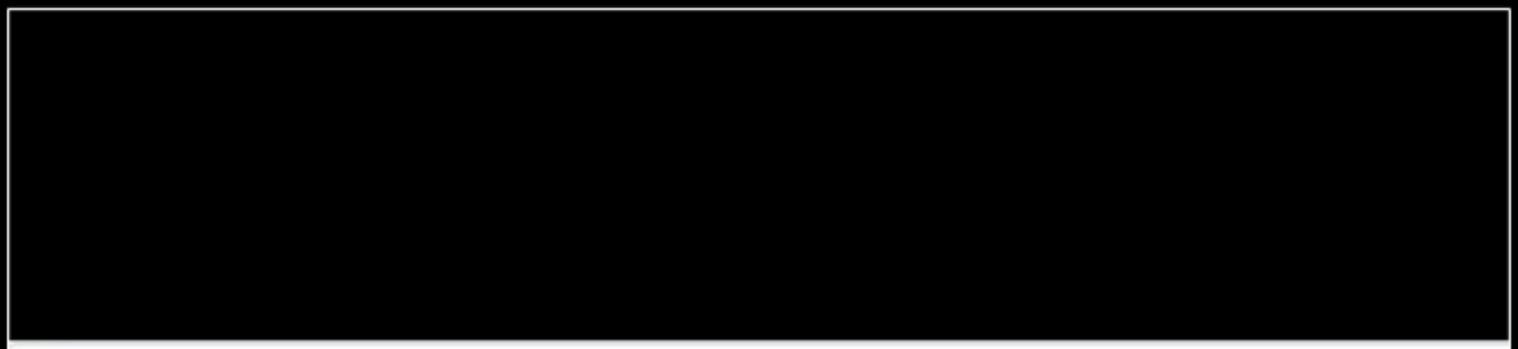
8 hrs 40 mins	Project/Client/Client.java
3 hrs 18 mins	Project/Server/Room.java
2 hrs 49 mins	Project/Client/ClientUI.java
2 hrs 24 mins	...ct/Server/ServerThread.java
1 hr 26 mins	.../Client/Views/ChatPanel.java
44 mins	.../ient/Views/RoomsPanel.java
24 mins	Project/Common/Payload.java
21 mins	...ect/Common/PayloadType.java
19 mins	Project/Server/Server.java
19 mins	...nt/Views/UserListPanel.java
11 mins	...erver/BaseServerThread.java
8 mins	.../Views/ConnectionPanel.java
7 mins	.../Views/UserDetailsPanel.java
6 mins	...aces/IConnectionEvents.java
6 mins	...terfaces/IClientEvents.java
4 mins	...mon/RoomResultsPayload.java
3 mins	Project/Common/TextFX.java
3 mins	...mmon/ConnectionPayload.java
2 mins	...erfaces/IMessageEvents.java
2 mins	Project/Common/LoggerUtil.java
2 mins	Project/Client/CardView.java
1 min	...Interfaces/IRoomEvents.java
1 min	Project/Client/ClientData.java
1 min	...ect/Common/FlipPayload.java
36 secs	...terfaces/ICardControls.java
34 secs	...ect/Common/RollPayload.java
31 secs	Project/Client/Views/Menu.java

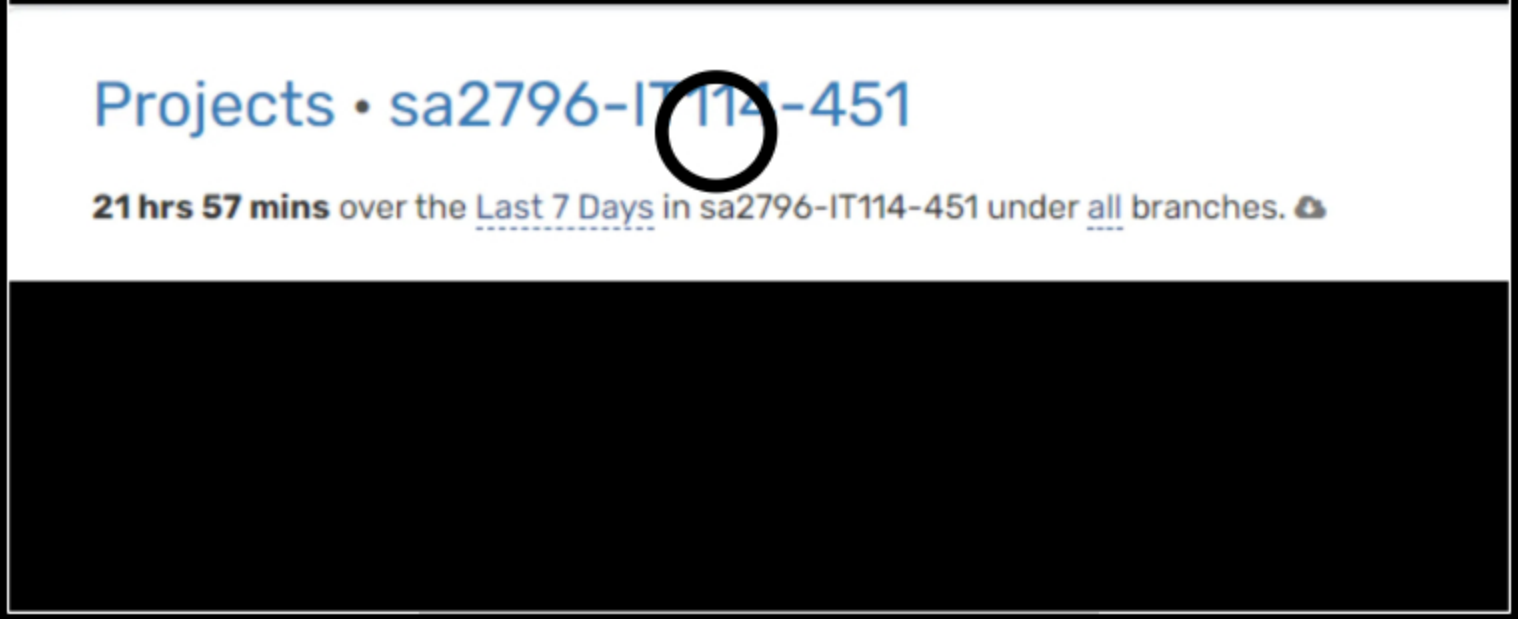
Branches

21 hrs 56 mins	Milestone3
1 min	Unknown

all files and branch times

all files and branch times





Overall time in all projects

End of Assignment