

# Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-milestone-3-chatroom-2024-m24/grade/rev>

Course: IT114-003-F2024

Assignment: [IT114] Milestone 3 Chatroom 2024 M24

Student: Ricardo V. (rev)

## Submissions:

Submission Selection

1 Submission [submitted] 12/5/2024 11:46:20 AM

## Instructions

^ COLLAPSE ^

Implement the Milestone 3 features from the project's proposal document:

<https://docs.google.com/document/d/1ONmvEvel97GTFPGfVwwQC96xSsobbSbk56145XizOG4/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

Group

100%

Group: Basic UI  
Tasks: 1  
Points: 2

^ COLLAPSE ^

Task

100%





Group: Basic UI  
Task #1: UI Panels  
Weight: ~100%  
Points: ~2.00

## Details:

All code screenshots must include ucid/date.

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

Columns: 4

Columns: 4			
<div> <div>Sub-Task</div> <div> <div>100%</div> </div> <div> Group: Basic UI Task #1: UI Panels Sub Task #1: Show the </div> </div>	<div> <div>Sub-Task</div> <div> <div>100%</div> </div> <div> Group: Basic UI Task #1: UI Panels Sub Task #2: Show the code </div> </div>	<div> <div>Sub-Task</div> <div> <div>100%</div> </div> <div> Group: Basic UI Task #1: UI Panels Sub Task #3: show the </div> </div>	<div> <div>Sub-Task</div> <div> <div>100%</div> </div> <div> Group: Basic UI Task #1: UI Panels Sub Task #4: Show the code </div> </div>
<div> <div>Task</div> <div>Screenshots</div> <div>Gallery Style: 2 Columns</div> <div>4 2 1</div> <div>  </div> </div>	<div> <div>Task</div> <div>Screenshots</div> <div>Gallery Style: 2 Columns</div> <div>4 2 1</div> <div>  </div> </div>	<div> <div>Task</div> <div>Screenshots</div> <div>Gallery Style: 2 Columns</div> <div>4 2 1</div> <div>  </div> </div>	<div> <div>Task</div> <div>Screenshots</div> <div>Gallery Style: 2 Columns</div> <div>4 2 1</div> <div>  </div> </div>
<div>Show the ConnectionPanel by running the app</div>	<div>Show the code related to the ConnectionPanel by running the app</div>	<div>Show the code related to the UserDetailsPanel by running the app</div>	<div>Show the code related to the UserDetailsPanel by running the app</div>
<div> <div>Caption(s) (required) ✓</div> <div>Caption Hint:</div> <div>Describe/highlight what's being shown</div> </div>	<div> <div>Caption(s) (required) ✓</div> <div>Caption Hint:</div> <div>Describe/highlight what's being shown</div> </div>	<div> <div>Caption(s) (required) ✓</div> <div>Caption Hint:</div> <div>Describe/highlight what's being shown</div> </div>	<div> <div>Caption(s) (required) ✓</div> <div>Caption Hint:</div> <div>Describe/highlight what's being shown</div> </div>
<div> <div>Task</div> <div>Response</div> <div>Prompt</div> <div>Briefly explain how it works and how it's used</div> <div>Response:</div> <div> <div>The ConnectionPanel code defines a user interface component with input fields for host, port, and username. Event handlers validate inputs and trigger the connection logic to the server when a button or</div> </div> </div>	<div> <div>Task</div> <div>Response</div> <div>Prompt</div> <div>Briefly explain how it works and how it's used</div> <div>Response:</div> <div> <div>The UserDetailsPanel code initializes a form with a username input field and submit button, ensuring that the username adheres to predefined constraints and integrates seamlessly with the connection</div> </div> </div>	<div> <div>Task</div> <div>Response</div> <div>Prompt</div> <div>Briefly explain how it works and how it's used</div> <div>Response:</div> <div> <div>The UserDetailsPanel code initializes a form with a username input field and submit button, ensuring that the username adheres to predefined constraints and integrates seamlessly with the connection</div> </div> </div>	<div> <div>Task</div> <div>Response</div> <div>Prompt</div> <div>Briefly explain how it works and how it's used</div> <div>Response:</div> <div> <div>The UserDetailsPanel code initializes a form with a username input field and submit button, ensuring that the username adheres to predefined constraints and integrates seamlessly with the connection</div> </div> </div>

server when a button or  
enter key is pressed.

with the connection  
establishment logic.

#### Sub-Task

100%

Group:  
Basic UI  
Task #1: UI  
Panels  
Sub Task  
#5: Show  
the

#### Sub-Task

100%

Group:  
Basic UI  
Task #1: UI  
Panels  
Sub Task  
#6: Show  
the code



## Task

### Screenshots

Gallery Style: 2 Columns

4 2 1



Show the  
ChatPanel

#### Caption(s) (required) ✓

Caption Hint:

*Describe/highlight what's  
being shown*



## Task

### Screenshots

Gallery Style: 2 Columns

4 2 1

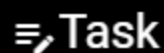


Show the codeShow the code  
related to the related to the  
ChatPanel ChatPanel

#### Caption(s) (required) ✓

Caption Hint:

*Describe/highlight what's  
being shown*



## Task

### Response

### Prompt

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

Response:

The ChatPanel code  
contains methods for  
dynamically updating the  
user list and chat history,  
managing user inputs for  
sending or executing  
commands, and rendering  
formatted messages in  
the UI with styling options  
like bold, italic, or colors.

End of Task 1

End of Group: Basic UI

Task Status: 1/1

Group



Group: Build-up

Tasks: 2

Points: 3

^ COLLAPSE ^

Task



Group: Build-up

Task #1: Results of /flip and /roll appear in a different format than regular chat text

Weight: ~50%

Points: ~1.50

^ COLLAPSE ^

Details:

All code screenshots must include ucid/date.

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



Columns: 4

Sub-Task



Group:  
Build-up  
Task #1:  
Results of  
/flip and  
/roll  
appear in a

Sub-Task



Group:  
Build-up  
Task #1:  
Results of  
/flip and  
/roll  
appear in a

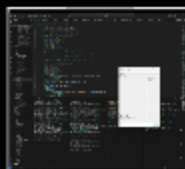


Task

Screenshots

Gallery Style: 2 Columns

4 2 1



Show

examples of it  
printing on  
screen

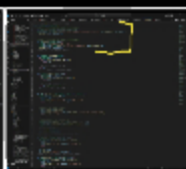


Task

Screenshots

Gallery Style: 2 Columns

4 2 1



Show

examples of it  
on the Room  
side that  
changes this

format

**Caption(s) (required)** ✓

Caption Hint:

*Describe/highlight what's  
being shown*

**Caption(s) (required)** ✓

Caption Hint:

*Describe/highlight what's  
being shown*

≡ Task

Response

Prompt

*Explain what you did and how  
it works*

Response:

The server-side Room  
code formats /flip and /or  
roll results by identifying  
these commands within  
received messages,  
applying a unique style,  
and broadcasting the  
formatted message to all  
clients.

End of Task 1

Task



Group: Build-up

Task #2: Text Formatting appears correctly on the UI

Weight: ~50%

Points: ~1.50

^ COLLAPSE ^

**i** Details:

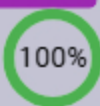
All code screenshots must include ucid/date.

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



Columns: 4

Sub-Task



Group:  
Build-up  
Task #2:  
Text  
Formatting  
appears  
correctly

Sub-Task



Group:  
Build-up  
Task #2:  
Text  
Formatting  
appears  
correctly

Task

Task

## Screenshots

Gallery Style: 2 Columns

4 2 1



Show examples of bold, italic, underline, each color implemented and a combination of bold, italic, underline, and one color in

**Caption(s) (required)** ✓

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

## Screenshots

Gallery Style: 2 Columns

4 2 1



Show the code changes necessary to get this to work

**Caption(s) (required)** ✓

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

≡ Task

## Response

### Prompt

*Briefly explain what was necessary and how it works*

Response:

The formatting code changes involve adding methods to identify and process formatting tags within messages, which are then converted into appropriate styled text for rendering on the chat interface.

End of Task 2

End of Group: Build-up

Task Status: 2/2

Group

100%

Group: New Features

Tasks: 2

Points: 4

⬆️ COLLAPSE ⬆️

## Task



Group: New Features

Task #1: Private messages via @username

Weight: ~50%

Points: ~2.00

^ COLLAPSE ^

## Details:

All code screenshots must include ucid/date.

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



Columns: 4

### Sub-Task



Group:  
New  
Features  
Task #1:  
Private  
messages  
via

### Sub-Task



Group:  
New  
Features  
Task #1:  
Private  
messages  
via

### Sub-Task



Group:  
New  
Features  
Task #1:  
Private  
messages  
via

### Sub-Task



Group:  
New  
Features  
Task #1:  
Private  
messages  
via



## Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show a few examples across different clients

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown



## Task Screenshots

Gallery Style: 2 Columns

4 2 1



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

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown

⇒ Task  
Response  
Prompt

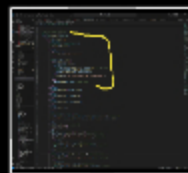
Explain in concise steps how



## Task Screenshots

Gallery Style: 2 Columns

4 2 1



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

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown

⇒ Task  
Response  
Prompt

Explain in concise steps how



## Task Screenshots

Gallery Style: 2 Columns

4 2 1



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

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown

⇒ Task  
Response



this logically works

Response:

The client-side private messaging code processes @username prefixes by validating the target username, extracting the corresponding client ID, and sending a payload containing the ID and message to the server only if the user is found.

this logically works

Response:

The ServerThread code receives private message payloads, extracts the sender and target client IDs, and passes them to the Room to handle private message delivery between only the two users.

Response

Prompt

Explain in concise steps how this logically works

Response:

The Room code for private messaging verifies the sender and receiver IDs, checks both are valid and sends the message exclusively to the two users while excluding everyone else.

## End of Task 1

### Task

100%

Group: New Features

Task #2: Mute and Unmute

Weight: ~50%

Points: ~2.00

^ COLLAPSE ^

### 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)

Columns: 4

#### Sub-Task

100%

Group:  
New  
Features  
Task #2:  
Mute and  
Unmute  
Sub Task

#### Sub-Task

100%

Group:  
New  
Features  
Task #2:  
Mute and  
Unmute  
Sub Task

#### Sub-Task

100%

Group:  
New  
Features  
Task #2:  
Mute and  
Unmute  
Sub Task

#### Sub-Task

100%

Group:  
New  
Features  
Task #2:  
Mute and  
Unmute  
Sub Task



Task  
Screenshots

Gallery Style: 2 Columns

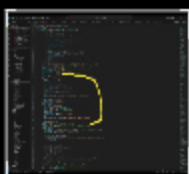
4 2 1



Task  
Screenshots

Gallery Style: 2 Columns

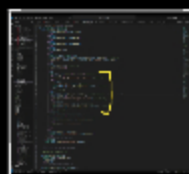
4 2 1



Task  
Screenshots

Gallery Style: 2 Columns

4 2 1



Task  
Screenshots

Gallery Style: 2 Columns

4 2 1





Show a few examples across different clients

**Caption(s) (required)** ✓

Caption Hint:

*Describe/highlight what's being shown*

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

**Caption(s) (required)** ✓

Caption Hint:

*Describe/highlight what's being shown*

≡ Task

Response

Prompt

*Explain in concise steps how this logically works*

Response:

The client-side mute/unmute code ~~processes~~ username and unmute username commands, validates the target user and sends the appropriate action and target ID to the server if the user exists.

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

**Caption(s) (required)** ✓

Caption Hint:

*Describe/highlight what's being shown*

≡ Task

Response

Prompt

*Explain in concise steps how this logically works*

Response:

The ServerThread code processes mute/unmute payloads by verifying the target user and invoking Room methods to add or remove the target user from the muter's unique mute list.

Show the Room code that verifies the id and add/removes the muted name to/from the ServerThread's list

**Caption(s) (required)** ✓

Caption Hint:

*Describe/highlight what's being shown*

≡ Task

Response

Prompt

*Explain in concise steps how this logically works*

Response:

The Room code verifies the target user's validity for mute/unmute operations, then delegates these actions to the muter's ServerThread by updating the mute list through accessor methods.

**Sub-Task**

100%

Group:  
New Features  
Task #2:  
Mute and Unmute  
Sub Task

**Sub-Task**

100%

Group:  
New Features  
Task #2:  
Mute and Unmute  
Sub Task



Task  
Screenshots

Gallery Style: 2 Columns

4 2 1



Task  
Screenshots

Gallery Style: 2 Columns

4 2 1



Show the Room code that checks the mute list during send message. private message, and any other relevant location

Show terminal supplemental evidence per the requirements

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

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

## Task Response Prompt

*Explain in concise steps how this logically works*  
Response:

The Room's message handling logic checks the sender's name against the receiver's mute list using a ServerThread method, ensuring messages from muted users are skipped while logging the action in the terminal.

End of Task 2

End of Group: New Features  
Task Status: 2/2

Group



Group: Misc  
Tasks: 3  
Points: 1

^ COLLAPSE ^

Task



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/RicardoVas9991/Rev-IT-114-0/pull/35>

URL

<https://github.com/RicardoVas9991/Rev-IT-114-0>

End of Task 1

#### Task



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 a lot of issues from trying to get the features to work and connecting to the rooms and back. This has been a tricky one than the previous ones. But I tried my best to do it correctly.

End of Task 2

#### Task



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

^ COLLAPSE ^

#### Details:

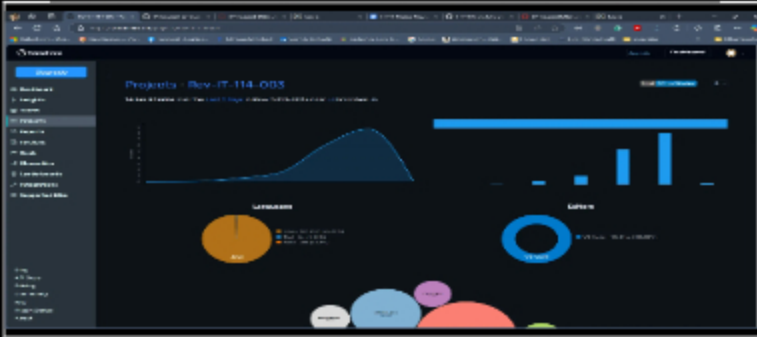
Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved



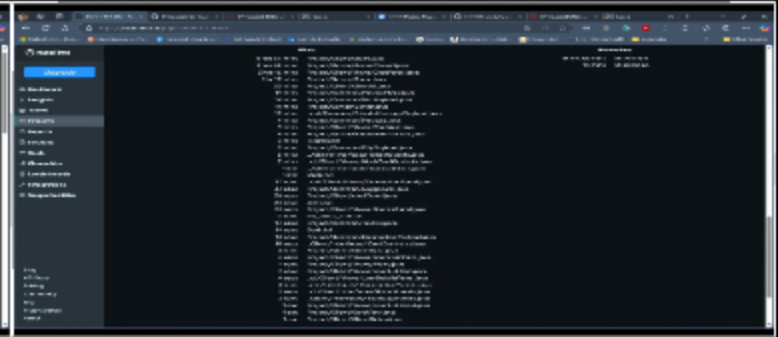
# Task Screenshots

Gallery Style: 2 Columns

4 2 1



WakaTime Screenshot



WakaTime Screenshot

End of Task 3

End of Group: Misc  
Task Status: 3/3

End of Assignment