Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-milestone-2-chatroom-2024-m24/grade/rev

Course: IT114-003-F2024
Assigment: [IT114] Milestone 2 Chatroom 2024 (M24)
Student: Ricardo V. (rev)

Submissions:

Submission Selection

1 Submission [submitted] 11/16/2024 8:27:53 PM

Instructions

COLLAPSE

1. Implement the Milestone 2 features from the project's proposal document: https://docs.google.com/document/d/1ONmvEvel97GTFPGfVwwQC96xSsobbSbk56145XizQG4/view
2. Make sure you add your ucid/date as code comments where code changes are done
3. All code changes should reach the Milestone2 branch
4. Create a pull request from Milestone2 to main and keep it open until you get the output PDF from this assignment.
5. Gather the evidence of feature completion based on the below tasks.
6. Once finished, get the output PDF and copy/move it to your repository folder on your local machine.
7. Run the necessary git add, commit, and push steps to move it to GitHub
8. Complete the pull request that was opened earlier
9. Upload the same output PDF to Canvas

Branch name: Milestone2

Group

100%

Group: Payloads
Tasks: 2
Points: 2

COLLAPSE

Task

100%

Group: Payloads
Task #1: Base Payload Class
Weight: ~50%
Points: ~1.00

COLLAPSE

Details:
All code screenshots must have ucid/date visible.

Columns: 4

Sub-Task

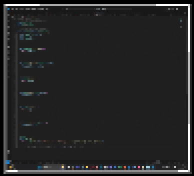
100%

Group: Payloads
Task #1: Base Payload Class
Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show screenshot of the Payload.java

Caption(s) (required) ✓

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

Task Response Prompt

Briefly explain the purpose of each property and serialization

Response:

Sub-Task

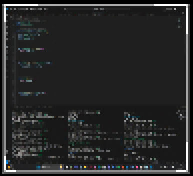
100%

Group: Payloads
Task #1: Base Payload Class
Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show screenshot examples of the terminal output for base Payload objects

Caption(s) (required) ✓

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

The purpose of each property in Payload.java is to define and serialize essential communication types (e.g., client connect, room list), allowing the application to distinguish between various payload actions effectively.

End of Task 1

Task

100%

Group: Payloads
Task #2: RollPayload Class
Weight: ~50%
Points: ~1.00

COLLAPSE

Details:
All code screenshots must have ucid/date visible.

Columns: 4

Sub-Task

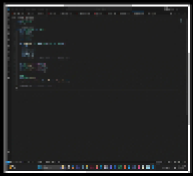
100%

Group: Payloads
Task #2: RollPayload Class
Sub Task #1: Show

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show screenshot of the RollPayload.java

Sub-Task

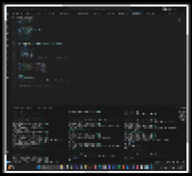
100%

Group: Payloads
Task #2: RollPayload Class
Sub Task #2: Show

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show screenshot examples of the terminal output for

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

base
RollPayload
objects

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

Task Response Prompt

Briefly explain the purpose of each property

Response:

The properties in RollPayload.java represent the roll result details, enabling the server to send a structured response to clients based on specific roll commands.

End of Task 2

End of Group: Payloads
Task Status: 2/2

Group

100%

Group: Client Commands
Tasks: 2
Points: 4

COLLAPSE

Task

100%

Group: Client Commands
Task #1: Roll Command
Weight: ~50%
Points: ~2.00

COLLAPSE

Details:
All code screenshots must have ucid/date visible.
Any output screenshots must have at least 3 connected clients able to see the output.
All commands must show who triggered it, what they did (specifically) and what the outcome was.

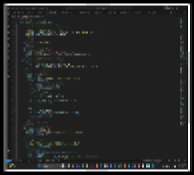
Columns: 4

Sub-Task
100%
Group: Client Commands
Task #1: Roll Command
Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show the client side code for handling /roll #

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

Task Response Prompt

Briefly explain the logic

Response:

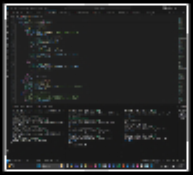
The client code for /roll # and /roll #d# takes user inputs for different dice rolls, sends a RollPayload with the outcome to the server, and broadcasts results to connected clients.

Sub-Task
100%
Group: Client Commands
Task #1: Roll Command
Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show the output of a few examples of /roll # (related payload output should be visible)

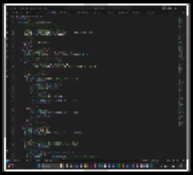
Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

Sub-Task
100%
Group: Client Commands
Task #1: Roll Command
Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show the client side code for handling /roll #d#

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

Task Response Prompt

Briefly explain the logic

Response:

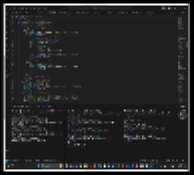
The client code for /roll # and /roll #d# takes user inputs for different dice rolls, sends a RollPayload with the outcome to the server, and broadcasts results to connected clients.

Sub-Task
100%
Group: Client Commands
Task #1: Roll Command
Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show the output of a few examples of /roll #d#

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

Sub-Task
100%
Group: Client Commands
Task #1: Roll Command

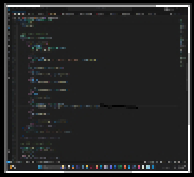
Sub-Task
100%
Group: Client Commands
Task #1: Roll Command

Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show the ServerThread code receiving the RollPayload

Caption(s) (required) ✓

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

Task Response Prompt

Briefly explain the logic

Response:

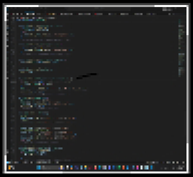
The client code for /roll # and /roll #d# takes user inputs for different dice rolls, sends a RollPayload with the outcome to the server, and broadcasts results to connected clients.

Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show the Room code that processes both Rolls and sends the response

Caption(s) (required) ✓

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

Task Response Prompt

Briefly explain the logic

Response:

The client code for /roll # and /roll #d# takes user inputs for different dice rolls, sends a RollPayload with the outcome to the server, and broadcasts results to connected clients.

End of Task 1

Task

100%

Group: Client Commands
Task #2: Flip Command
Weight: ~50%
Points: ~2.00

COLLAPSE

Columns: 4

Sub-Task

100%

Group: Client Commands

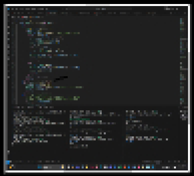
Task #2: Flip Command

Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show the client side code for handling /flip

Caption(s) (required) ✓

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

Sub-Task

100%

Group: Client Commands

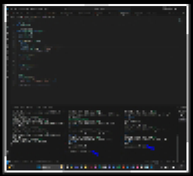
Task #2: Flip Command

Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show the output of a few examples of /flip

Caption(s) (required) ✓

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

Task Response Prompt

Briefly explain the logic

Response:

The client /flip command sends a flip payload to the server, which broadcasts a randomized result (e.g., heads or tails) to all connected clients.

End of Task 2

End of Group: Client Commands
Task Status: 2/2

Group

Group: Text Formatting
Tasks: 1

100%

Points: 3

COLLAPSE

Task

100%

Group: Text Formatting
Task #1: Text Formatting
Weight: ~100%
Points: ~3.00

COLLAPSE

Details:
All code screenshots must have ucid/date visible.
Any output screenshots must have at least 3 connected clients able to see the output.
Note: Having the user type out html tags is not valid for this feature, instead treat it like WhatsApp, Discord, Markdown, etc

Columns: 4

Sub-Task

100%

Group: Text Formatting
Task #1: Text Formatting
Sub Task

Sub-Task

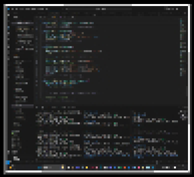
100%

Group: Text Formatting
Task #1: Text Formatting
Sub Task

Task Screenshots

Gallery Style: 2 Columns

4 2 1



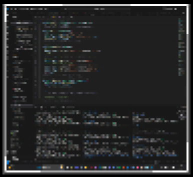
Show the code related to processing the special characters for bold, italic, underline, and colors, and converting them to other

Caption(s) (required) ✓
Caption Hint:

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Show examples of each: bold, italic, underline, colors (red, green, blue), and combination of bold, italic, underline and a colo

Describe/highlight what's being shown

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown

Task Response Prompt

Briefly explain how it works and the choices of the placeholder characters and the result characters

Response:

In Room.java, special characters for bold, italic, underline, and color formats are processed by matching placeholders (e.g., *, _, ~) and converting them into stylized text for display across all clients.

End of Task 1

End of Group: Text Formatting
Task Status: 1/1

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/RicardoVas9991/Rev-IT-114-003

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

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 started the Trivia Project first but I couldn't understand it; So I asked the profesor if I could change it to Chatroom and he said yes but by the grace period of this Wednesday. Again I am sorry for asking this but I could go further with trivia I got stumped.

End of Task 2

Task

100%

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

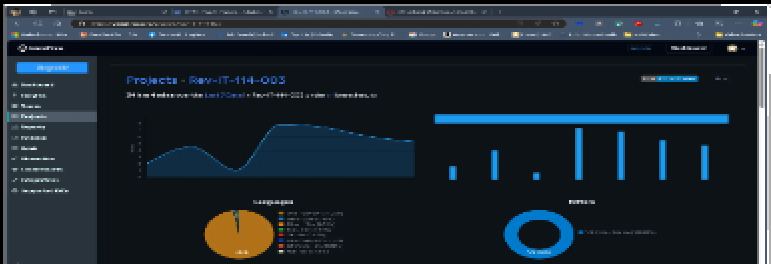
COLLAPSE

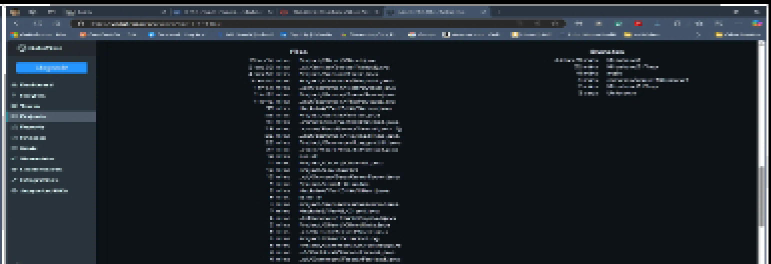
Details:
Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

Task Screenshots

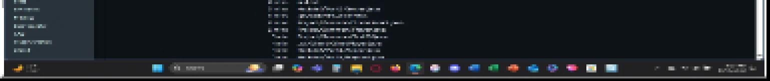
Gallery Style: 2 Columns

4 2 1





WakaTime Screenshot


WakaTime Screenshot

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment