Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-milestone-2-chatroom-2024/grade/oha2

IT114-006-S2024 - [IT114] Milestone 2 Chatroom 2024

Submissions:

Submission Selection

1 Submission [active] 4/3/2024 11:09:32 PM

Instructions

COLLAPSE

Implement the Milestone 2 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 Milestone2 branch
Create a pull request from Milestone2 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: Milestone2

Tasks: 12 Points: 10.00

Demonstrate Usage of Payloads (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of your Payload class and subclasses and PayloadType

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Payload, equivalent of RollPayload, and any others
#2	1	Screenshots should include ucid and date comment
#3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```java
package Project.Common;
//oha2 april 1st 2024
public class RollPayload extends Payload {

    public RollPayload(int numDice, int numSides) {
        setPayloadType(PayloadType.ROLL);
        this.numDice = numDice;
        this.numSides = numSides;
    }

    private int numDice;
    private int numSides;

    public int getNumDice() {
        return numDice;
    }

    public void setNumDice(int numDice) {
        this.numDice = numDice;
    }

    public int getNumSides() {
        return numSides;
    }

    public void setNumSides(int numSides) {
        this.numSides = numSides;
    }

    @Override
    public String toString() {
        return super.toString() + ", Number of Dice: " + getNumDice() + ", Number of Sides: " + getNumSides();
    }
}
```

roll payload

Checklist Items (0)

```java
package Project.Common;
//oha2 april 1st 2024
public class FlipPayload extends Payload {

    private String result; // Store 'heads' or 'tails'

    public FlipPayload(String result) {
        setPayloadType(PayloadType.FLIP);
        this.result = result;
    }

    public String getResult() {
        return result;
    }

    public void setResult(String result) {
        this.result = result;
    }
```

```java
    }

    @Override
    public String toString() {
        return super.toString() + ", Result: " + getResult();
    }
}
```

flip payload

Checklist Items (0)

^COLLAPSE^

Task #2 - Points: 1

Text: Screenshots of the payloads being debugged/output to the terminal

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Demonstrate flip
☐ #2	1	Demonstrate roll (both versions)
☐ #3	1	Demonstrate formatted message along with any others
☐ #4	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large

Missing Caption

COLLAPSE

Task #3 - Points: 1

Text: Explain the purpose of payloads and how your flip/roll payloads were made

Response:

I made the payloads extend payload. Flip will store the result from the code in room where it assigns it heads or tails.

Roll is more complicated. It saves the number of dice rolled as well as the number of sides. It also stores the result and then prints it.

Demonstrate Roll Command (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the following items

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Client code that captures the command and converts it to a RollPayload (or equivalent) for both scenarios /roll # and /roll #d#
☐ #2	1	ServerThread code receiving the payload and passing it to the Room
■ #3	1	Room handling the roll action correctly for both scenarios (/roll # and /roll #d#) including the message going back out to all clients
☐ #4	1	Code screenshots should include ucid and date comment
☐ #5	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
    }
    //oha2 4/3/2024
private boolean processCommand(String message) {
    logger.info("Checking command: " + message);

    // Check if the message is a command
    if (message.startsWith(prefix:"/")) {
        String[] parts = message.split(regex:" ");
        String command = parts[0].toLowerCase(); // Extract the command

        switch (command) {
            case "/roll":
                if (parts.length == 2) {
```

```java
            String argument = parts[1].toLowerCase();
            if (argument.matches(regex:"\\d+d\\d+")) { // Check if it's in the format "NdN"
                // Process the roll command with NdN format
                String[] rollParams = argument.split(regex:"d");
                int numDice = Integer.parseInt(rollParams[0]);
                int numSides = Integer.parseInt(rollParams[1]);
                RollPayload rollPayload = new RollPayload(numDice, numSides);
                broadcast("Rolled " + numDice + "d" + numSides + ": " + rollPayload.toString());
            } else if (argument.matches(regex:"\\d+")) { // Check if it's in the format "0-X" or "1-X"
                int startValue = Integer.parseInt(argument);
                broadcast("Starting at: " + startValue);
            } else {
                // Invalid roll format
                logger.info("Invalid roll format: " + argument);
            }
        } else {
            // Invalid number of arguments for /roll command
            logger.info(msg:"Invalid number of arguments for /roll command");
        }
```

roll code

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Explain the logic in how the two different roll formats are handled and how the message flows from the client, to the Room, and shared with all other users

Response:

I could not figure out the client and server thread stuff because i switched which project I am doing last second. Here is code that first checks if /roll is typed and then looks at the formatting afterwards. If it is just /roll then a number it sets the number of dice to 1 and rolls the dice. if it is /roll number d number then it takes the first number as number of dice and then secodn number the sides and does math.random on number of sides and puts it together where ti is gathered by the payload to print the correct thing. It also parses the integers

Demonstrate Flip Command (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the following items

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Client code that captures the command and converts it to a payload
☐ #2	1	ServerThread receiving the payload and passing it to the Room
☐ #3	1	Room handling the flip action correctly
☐ #4	1	Code screenshots should include ucid and date comment
☐ #5	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large

Missing Caption

COLLAPSE

Task #2 - Points: 1

Text: Explain the logic in how the flip command is handled and processed and how the message flows from the client, to the Room, and shared with all other users

Response:

Missing Response

Demonstrate Formatted Messages (4 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of Room how the following formatting is processed from a message

Details:

Note: this processing is server-side

Slash commands are not valid solutions for this and will receive 0 credit

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Room code processing for bold
☐ #2	1	Room code processing for italic
☐ #3	1	Room code processing for underline
☐ #4	1	Room code processing for color (at least R, G, B or support for hex codes)
☐ #5	1	Show each one working individually and one showing a combination of all of the formats and 1 color from the terminal
☐ #6	1	Must not rely on the user typing html characters, but the output can be html characters
☐ #7	1	Code screenshots should include ucid and date comment
☐ #8	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
    // Process text formatting commands
    if (message.contains(s:"*") || message.contains(s:"!") || message.contains(s:"_") || message.contains(s:"#")) {
        // Apply formatting based on symbols
        message = applyFormatting(message);

        broadcast(message);

        return true;
    }
}
return false;


private String applyFormatting(String message) {
    // Replace * with <b> and </b> tags for bold
    message = message.replaceAll(regex:"\\*(.*?)\\*", replacement:"<b>$1</b>");

    // Replace ! with <i> and </i> tags for italic
    message = message.replaceAll(regex:"!(.*?)!", replacement:"<i>$1</i>");

    // Replace _ with <u> and </u> tags for underline
    message = message.replaceAll(regex:"_(.*?)_", replacement:"<u>$1</u>");

    //the color ones
    message = message.replaceAll(regex:"#r(.*?)r#", replacement:"< color='red'>$1</color>");

    message = message.replaceAll(regex:"#b(.*?)b#", replacement:"< color='blue'>$1</color>");

    message = message.replaceAll(regex:"#g(.*?)g#", replacement:"< color='green'>$1</color>");

    return message;
}
//oha2 april 1, 2024
```

it processing the formatting symbols

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Explain the following

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Which special characters translate to the desired effect
☐ #2	1	How the logic works that converts the message to its final format

Response:

makes it bold ! makes it italic and _ makes it underlined. #r r# #b b# and #g g# change the colors to red blue and green respectively. The processer checks every message if they have any of these symbols and runs it through if there are 2 of any then it replaces the symbols with html tags.

COLLAPSE

Misc (1 pt.)

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/Jomarrrrr/Oha2-it114-006/pull/8

URL #2
https://github.com/Jomarrrrr/Oha2-it114-006/pull/7

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

I have a lot of troubles with errors with my payloads and enums. I was doing rps at first but that felt actually impossible past making the rps logic for me. I am looking to fix everything this weekend but I just want to have something in. I accidentally merged it before i added the pdf so there is a second link.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

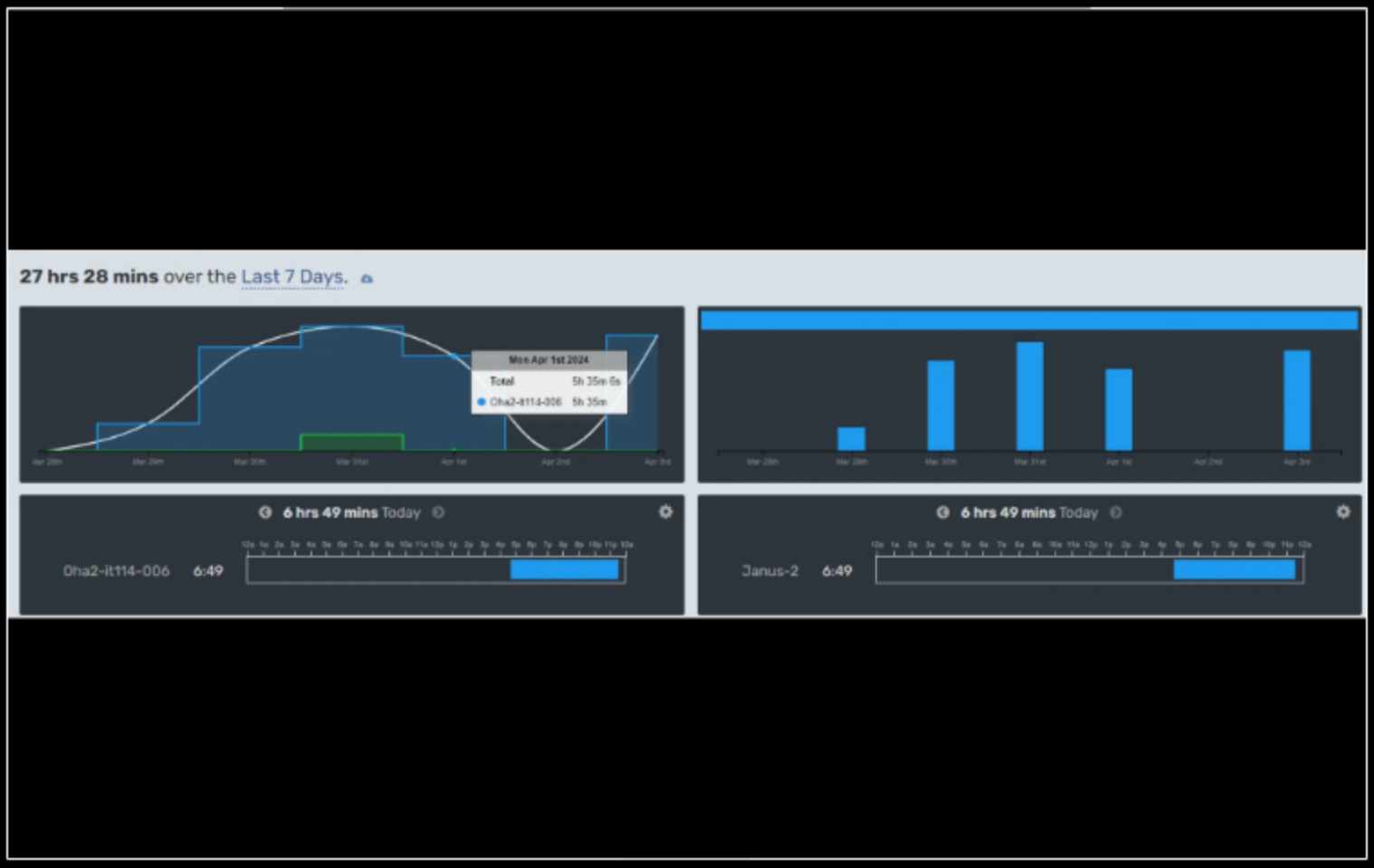
Details:

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



waka time screenshot I spent alot of time because i switched off rock paper scissors last second

End of Assignment