# Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-451-M2024/it114-module-4-sockets-part-1-3/grade/st278

IT114-451-M2024 - [IT114] Module 4 Sockets Part 1-3

Submissions:

Submission Selection

1 Submission [active] 6/18/2024 10:26:54 PM

Instructions

COLLAPSE

Overview Video: https://youtu.be/5a5HL0n6jek

1. Create a new branch for this assignment
2. If you haven't, go through the socket lessons and get each part implemented (parts 1-3)
    1. You'll probably want to put them into their own separate folders/packages (i.e., Part1, Part2, Part3) These are for your reference
3. Part 3, below, is what's necessary for this HW
    3. https://github.com/MattToegel/IT114/tree/M24-Sockets-Part3
4. Create a new folder called Part3HW (copy of Part3)
5. Make sure you have all the necessary files from Part3 copied here and fix the package references at the top of each file
    1. Add/commit/push the branch
    2. Create a pull request to main and keep it open
6. Implement **two** of the following **server-side** activities for all connected clients (majority of the logic should be processed server-side and broadcasted/sent to all clients if/when applicable)
    1. Simple number guesser where all clients can attempt to guess while the game is active
        1. Have a /start command that activates the game allowing guesses to be interpreted
        2. Have a /stop command that deactivates the game, guesses will be treated as regular messages (i.e., guess messages are ignored)
        3. Have a /guess command that include a value that is processed to see if it matches the hidden number (i.e., */guess 5*)
            1. Guess should only be considered when the game is active
            2. The response should include who guessed, what they guessed, and whether or not it was correct (i.e., Bob guessed 5 but it was not correct)
            3. No need to implement complexities like strikes
    2. Coin toss command (random heads or tails)

    1. Command should be something logical like /flip or /toss or /coin or similar
    2. The result should mention *who* did *what* and got what *result* (i.e., Bob Flipped a coin and got heads)
3. Dice roller given a command and text format of "/roll #d#" (i.e., /roll 2d6)
    1. Command should be in the format of /roll #d# (i.e., /roll 1d10)
    2. The result should mention *who* did *what* and got what *result* (i.e., Bob rolled 1d10 and got 7)
4. Math game (server outputs a basic equation, first person to guess it correctly gets congratulated and a new equation is given)
    1. Have a /start command that activates the game allowing equaiton to be answered
    2. Have a /stop command that deactivates the game, answers will be treated as regular messages (i.e., any game related commands when stopped will be ignored)
    3. Have an answer command that include a value that is processed to see if it matches the hidden number (i.e., */answer 15*)
        1. The response should include who answered, what they answered, and whether or not it was correct (i.e., Bob answered 5 but it was not correct)
5. Private message (a client can send a message targetting another client where only the two can see the messages)
    1. Command can be /pm, /dm followed by the user's name or an @ preceding the users name (clearly note which)
    2. The server should properly check the target audience and send the response to the original sender and to the receiver (no one else should get the message)
    3. Alternatively (make note if you do this and show evidence) you can add support to private message multiple people at once. Evidence should show a larger number of clients than the target list of the private message to show it works. Note to grader: if this is accomplished add 0.5 to total final grade on Canvas
6. Message shuffler (randomizes the order of the characters of the given message)
    1. Command should be /shuffle or /randomize (clearly mention what you chose) followed by the message to shuffle (i.e., /shuffle hello everybody)
    2. The message should be sent to all clients showing it's from the user but randomized
        1. Example: Bob types /*command* hello and everyone recevies Bob: lleho
7. Fill in the below deliverables
8. Save the submission and generated output PDF
9. Add the PDF to the Part3HW folder (local)
10. Add/commit/push your changes
11. Merge the pull request
12. Upload the same PDF to Canvas

**Branch name:** M4-Sockets3-Homework

Tasks: 6 Points: 10.00

Baseline (2 pts.)

COLLAPSE

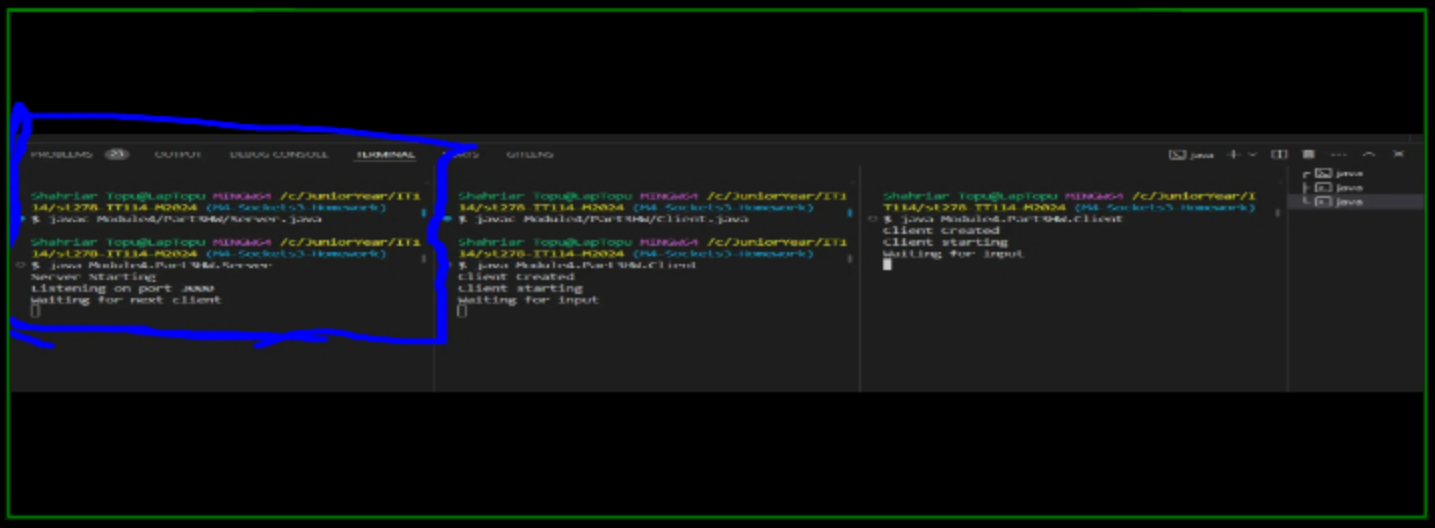
COLLAPSE

## Task #1 - Points: 1

**Text: Demonstrate Baseline Code Working**

**Details:**
This can be a single screenshot if everything fits, or can be multiple screenshots

#1) Show and clearly note which terminal is the Server

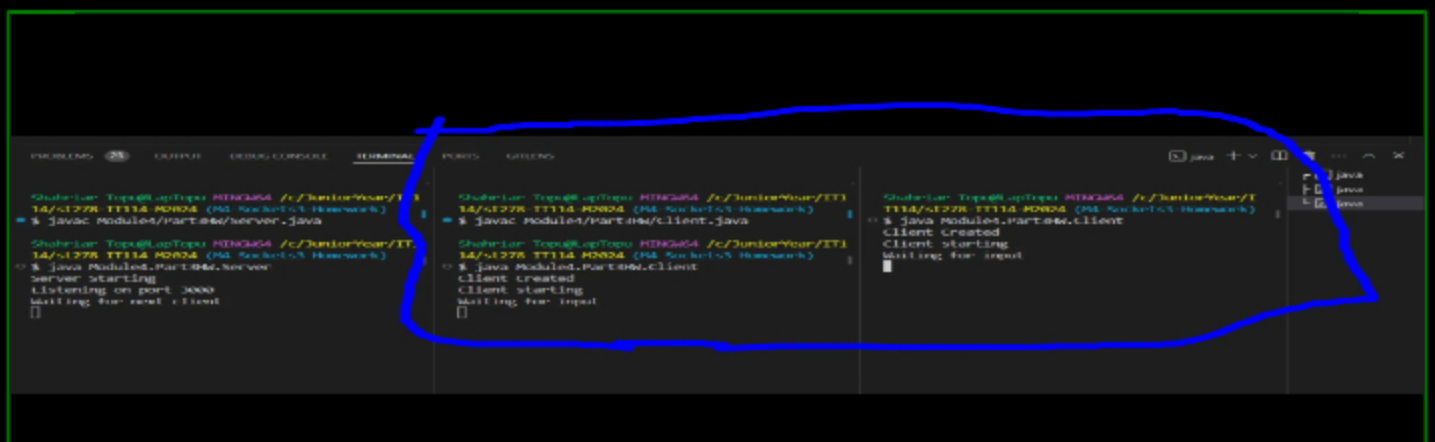



**Caption (required)** ✓

*Describe/highlight what's being shown*

The leftmost terminal is the server terminal

#2) Show and clearly note which terminals are the client

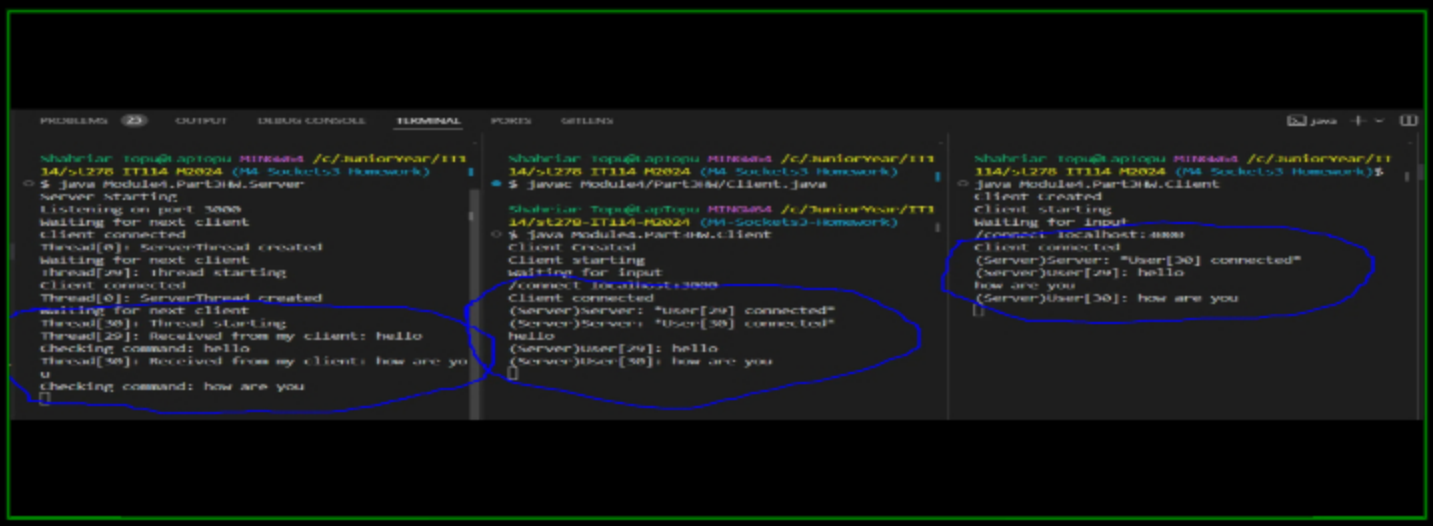


**Caption (required)** ✓

*Describe/highlight what's being shown*

The middle and rightmost terminals are the client terminals.

#3) Show all clients receiving the broadcasted/relayed messages

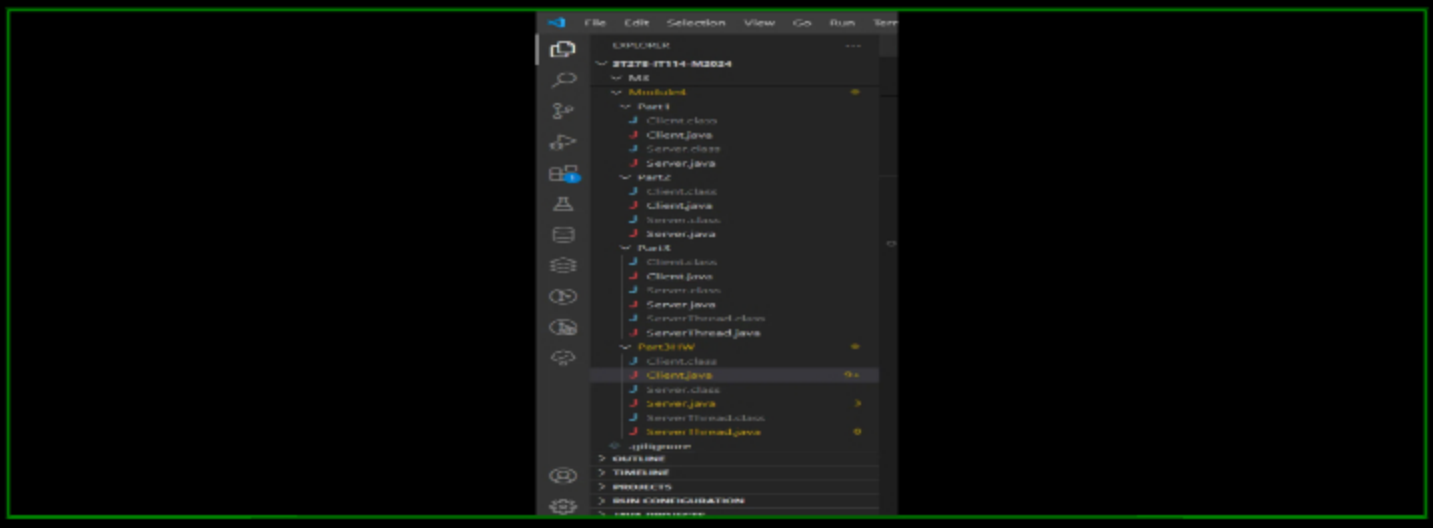



**Caption (required)** ✓

*Describe/highlight what's being shown*

The server and client communicating

#4) Include a screenshot showing you grabbed Parts 1-3 correctly and have them in your repository alongside Part3HW




**Caption (required)** ✓

*Describe/highlight what's being shown*

The files are shown in the same location.

Feature 1 (3 pts.)

COLLAPSE

COLLAPSE

## Task #1 - Points: 1

**Text: Solution**

#1) Show the code related to the feature (ucid and date must be present as a comment)

**Caption (required)** ✓

*Describe/highlight what's being shown*

Implementation 1 integrated and this is the code to show that.

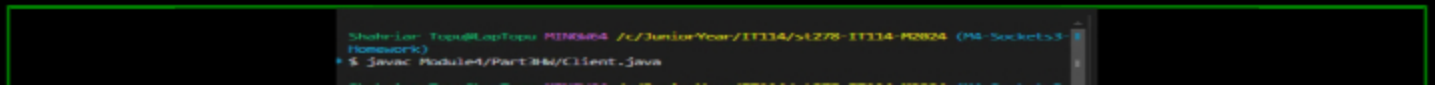
**Explanation (required)** ✓

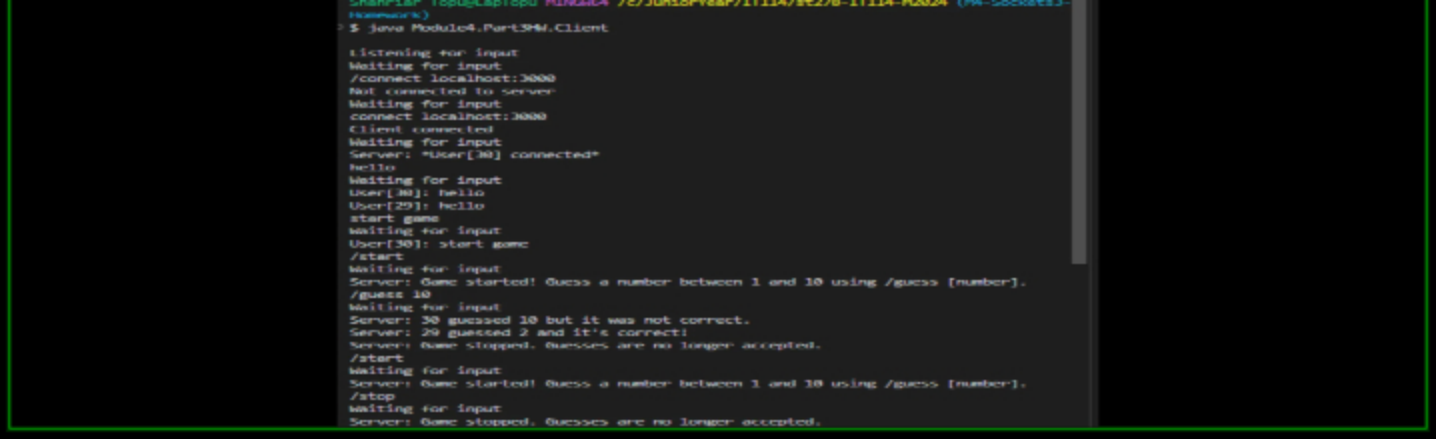
*Mention specific feature and explain sufficiently and concisely the implementation (should be aligned with code snippets)*

PREVIEW RESPONSE

The original Server class manages client connections, maintains a list of connected clients, and relays messages among clients. The number guessing game addition introduces game state management with isGameActive and hiddenNumber, and extends processCommand to handle /start, /stop, and /guess number commands. New methods startGame, stopGame, and handleGuess manage the game state and process client guesses. Integration ensures command processing checks the game state before handling /guess commands. This combines the original server functionality with the new features for a number guessing game, allowing clients to start, stop, and participate in the game through their guesses.

#2) Show the feature working (i.e., all terminals and their related output)






**Caption (required)** ✓

*Describe/highlight what's being shown*

The result/output of the first implemented feature.

Feature 2 (3 pts.)

^COLLAPSE^

^COLLAPSE^

Task #1 - Points: 1

**Text: Solution**

#1) Show the code related to the feature (ucid and date must be present as a comment)

```java
// st278 and 06/18/2024
/**      You, 24 minutes ago • Uncommitted changes
 * Simulates a coin flip and relays the result to all clients
 *
 * @param sender
 */
private void flipCoin(ServerThread sender) {
    String[] outcomes = {"heads", "tails"};
    String result = outcomes[(int) (Math.random() * 2)];
    relay(String.format(format:"%s flipped a coin and got %s", sender.getClientId(), result), sender:null);
}

Run | Debug | Run main | Debug main
public static void main(String[] args) {
    System.out.println(x:"Server Starting");
    Server server = new Server();
    int port = 3000;
    try {
        port = Integer.parseInt(args[0]);
    } catch (Exception e) {
        // can ignore, will either be index out of bounds or type mismatch
        // will default to the defined value prior to the try/catch
    }
    server.start(port);
    System.out.println(x:"Server Stopped");
}
```

**Caption (required)** ✓

*Describe/highlight what's being shown*

Implementation 2 integrated and this is the code to show that.

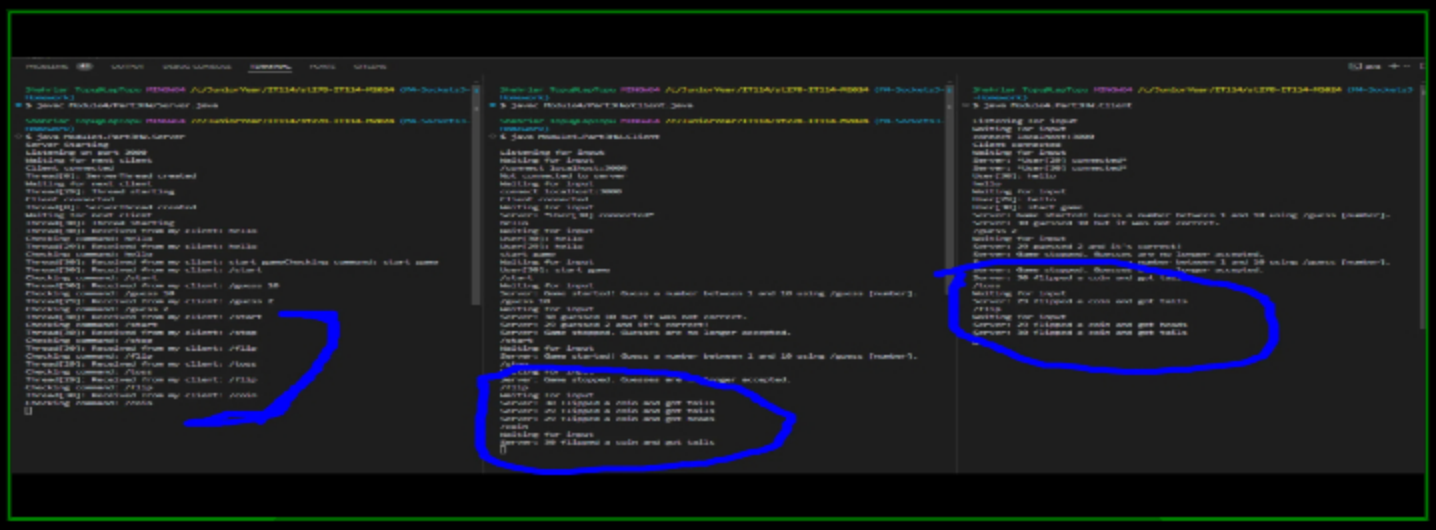
**Explanation (required)** ✓

*Mention specific feature and explain sufficiently and concisely the implementation (should be aligned with code snippets)s*

PREVIEW RESPONSE

The coin toss command is added to the processCommand method to recognize /flip, /toss, or /coin. When received from a client, these commands trigger the flipCoin method, which simulates a coin flip by randomly selecting "heads" or "tails" and then relays the result to all connected clients. This method is integrated into the existing command processing flow, allowing clients to request a coin toss, with the server broadcasting who performed the action and what the outcome was to all connected clients.

#2) Show the feature working (i.e., all terminals and their related output)



**Caption (required)** ✓

*Describe/highlight what's being shown*

The result/output of the second implemented feature.

Misc (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

**Text: Reflection**

#1) Learn anything new? Face any challenges? How did you overcome any issues?

**Explanation (required)** ✓

*Provide at least a few logical sentences*

PREVIEW RESPONSE

During this assignment, I learned how sockets help connect clients and servers over a network. I used ServerSocket to accept connections from clients and Socket to connect to the server. I discovered that threads are important for handling multiple clients at the same time, keeping the server running smoothly. I also learned to use ObjectInputStream and ObjectOutputStream to send and receive messages. With this knowledge, I was able to add features like a number guessing game and a coin toss command. This assignment taught me the basics of network

programming and real-time data exchange.

COLLAPSE

## Task #2 - Points: 1

**Text: Pull request link**

Details:
URL should end with /pull/# and be related to this assignment

URL #1
https://github.com/st278/st278-IT114-M2024/pull/7/

COLLAPSE

## Task #3 - Points: 1

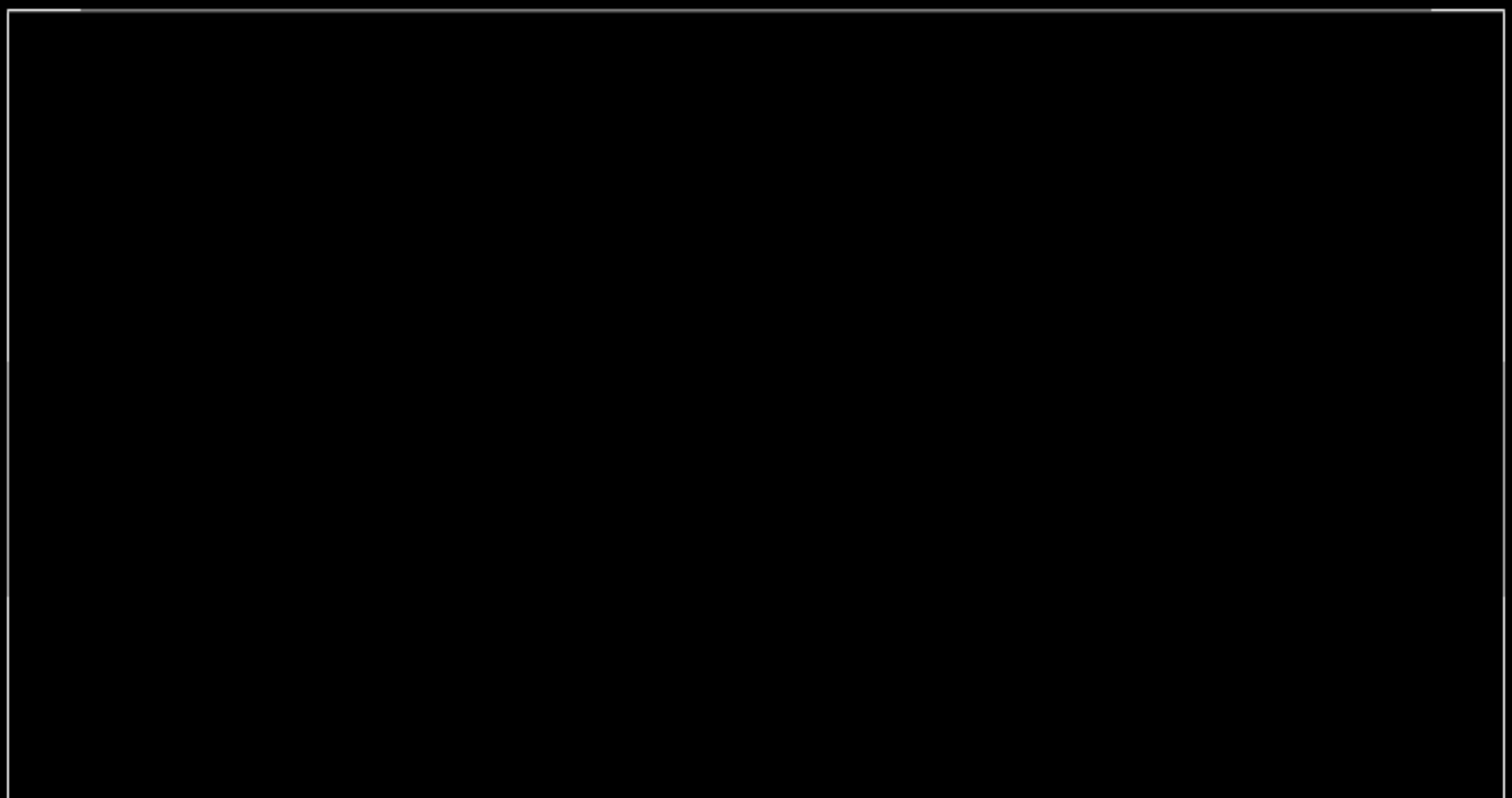
**Text: Waka Time (or related) Screenshot**

Details:
Screenshot clearly shows what files/project were being worked on (the duration of time doesn't correlated with the grade for this item)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Overall wakatime amount over the last 7 days.

Specifically how long on each file.

End of Assignment