

# Submission Worksheet

**CLICK TO GRADE**

<https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-module-2-rps-2024-m24/grade/jns>

Course: IT114-003-F2024

Assignment: [IT114] Module 2 RPS 2024 (M24)

Student: Jimmy S. (jns)

## Submissions:

Submission Selection

1 Submission [submitted] 11/13/2024 11:48:17 PM ▾

## Instructions

▲ COLLAPSE ▾

1. Implement the Milestone 2 features from the project's proposal document: [https://docs.google.com/document/d/11SRMo7JkLAMM-PuuiGwl\\_Z-QXP3pyQ7xN3IRxwmcwCc/view](https://docs.google.com/document/d/11SRMo7JkLAMM-PuuiGwl_Z-QXP3pyQ7xN3IRxwmcwCc/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



Group: Payloads

Tasks: 2

Points: 1

▲ COLLAPSE ▾

## Task

Group: Payloads

Task #1: Base Payload Class

Weight: ~50%

Points: ~0.50

100%

[▲ COLLAPSE ▾](#)

### ⓘ Details:

All code screenshots must have ucid/date visible.



Columns: 1

#### Sub-Task

Group: Payloads

Task #1: Base Payload Class

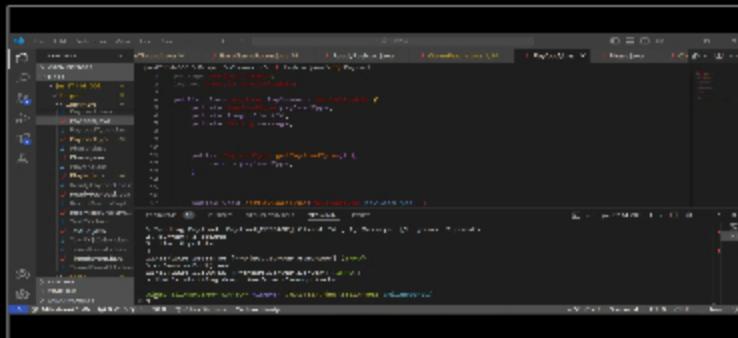
Sub Task #1: Show screenshot of the code for this file

100%

## ☒ Task Screenshots

Gallery Style: 2 Columns

4      2      1



base payload

Caption(s) (required) ✓

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

## ≡ Task Response Prompt

*Explain in concise steps how this logically works*

Response:

extended by all other payload, deals with generic payload types like create/join room, message, ... utilized for readyPayload and choicePayload.

#### Sub-Task

Group: Payloads

Task #1: Base Payload Class

Sub Task #2: Show screenshot examples of the terminal output for this object

100%

## ☒ Task Screenshots

Columns: 2 Columns

4 2 1

## /createroom command using payload

**Caption(s) (required)** ✓

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

End of Task 1

## Task

## Group: Payloads

## Task #2: PointsPayload (or equivalent)

Weight: ~50%

Points: ~0.50

[COLLAPSE](#)

## ● Details:

All code screenshots must have ucid/date visible.

四

Columns: 1

### Sub-Task

## Group: Payloads

#### Task #2: PointsPayload (or equivalent)

Sub Task #1: Show screenshot of the code for this file

## Task Screenshots

## Gallery Style: 2 Columns

4            2            1

choice payload

**Caption(s) (required)** ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

choice payload handles sending the users choice from client, converts the data from a string to a number for easier calculations

**Sub-Task**

Group: Payloads



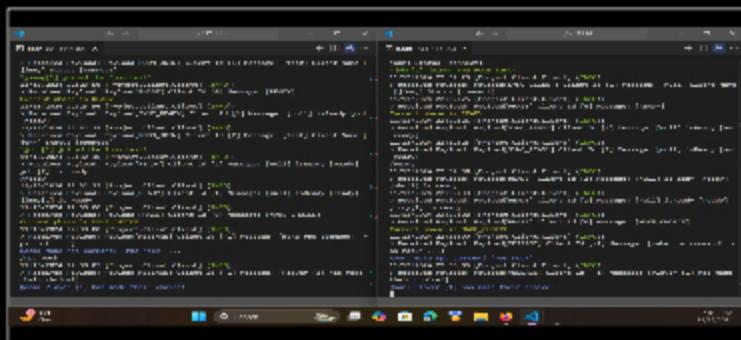
Task #2: PointsPayload (or equivalent)

Sub Task #2: Show screenshot examples of the terminal output for this object

## Task Screenshots

Gallery Style: 2 Columns

4      2      1



user does rps command, uses choicePayload

**Caption(s) (required)** ✓

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

End of Task 2

End of Group: Payloads

Task Status: 2/2

**Group**

Group: Session Start

Tasks: 1

Points: .5

 COLLAPSE

**Task**

Group: Session Start

Task #1: Trigger Round Start



100%

Weight: ~100%

Points: ~0.50

▲ COLLAPSE ▾

**ⓘ Details:**

All code screenshots must have ucid/date visible.

**Sub-Task**

Group: Session Start

100%

Task #1: Trigger Round Start

Sub Task #1: Show the related code

**☒ Task Screenshots**

Gallery Style: 2 Columns

4      2      1

```

2023-04-12 19:01:12 [INFO] [Session Start] Session 2 triggered by 27080000000000000000000000000000
public static void onSessionStart() {
    if (playerCount >= 2) {
        // Check if all players have passed their /ready check
        for (Player player : players) {
            if (!player.isReady()) {
                continue;
            }
            player.sendMessage("You are now ready!");
        }
        // Broadcast a message to all players
        broadcastMessage("All players are now ready!");
        // Set the session start time
        sessionStartTime = System.currentTimeMillis();
        // Set the session start date
        sessionStartDate = LocalDate.now();
    }
}

```

on sessionstart

**Caption(s) (required)** ✓Caption Hint: *Describe/highlight what's being shown***≡ Task Response Prompt***Explain in concise steps how this logically works*

Response:

sessionstart is activated after at least two player uses /ready command and passes ready check

End of Task 1

End of Group: Session Start

Task Status: 1/1

**Group**

Group: Round Start

Tasks: 3

Points: 1.25

94%

[▲ COLLAPSE ▲](#)

### Task



Group: Round Start  
Task #1: Reset Choices  
Weight: ~33%  
Points: ~0.42

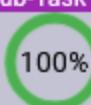
[▲ COLLAPSE ▲](#)

#### ⓘ Details:

All code screenshots must have ucid/date visible.



#### Sub-Task



Group: Round Start  
Task #1: Reset Choices  
Sub Task #1: Show the code that resets the remaining Player's choices to null

## ☒ Task Screenshots

Gallery Style: 2 Columns

4      2      1

```

import java.util.List;
import java.util.ArrayList;
import java.util.Collections;

public void onRoundStart() {
    for (PlayerChoice choice : choices) {
        choice.set(null);
        System.out.println("Resetting choice " + choice);
    }
}

```

onRoundStart

Caption(s) (required) ✓

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

## ≡ Task Response Prompt

*Explain in concise steps how this logically works*

Response:

on roundstart just resets the list of player choices so that players can make new choices if their is multiple rounds in a session (tie occurs).

End of Task 1

### Task

Group: Round Start

100%

## Task #2: Choosing Phase (or similar)

Weight: ~33%

Points: ~0.42

COLLAPSE

## ⓘ Details:

All code screenshots must have uid/date visible.



Columns: 1

### Sub-Task

## Group: Round Start

## Task #2: Choosing Phase (or similar)

Sub Task #1: Show the code that changes to Phase to the choosing phase

## Task Screenshots

## Gallery Style: 2 Columns

4 2 1

if code reaches sessionstart, game is in session, so phase is changed to make\_choice

**Caption(s) (required)** ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

#### **Response:**

when session starts, phase is changed to make choice, where the rps command is able to work, and should only work in a "active" game room.

### Sub-Task

### Group: Round Start

#### **Task #2: Choosing Phase (or similar)**

**Sub Task #2:** Show terminal output of this change

## Task Screenshots

## Gallery Style: 2 Columns

```
[John@2] joined the Room test
11/15/2024 22:32:38 [Project.Client.Client] (INFO):
> Received Payload: Payload[READY] Client Id [2] Message: [null] isReady [ready]
john@[2] is ready
/ready
11/15/2024 22:32:39 [Project.Client.Client] (INFO):
> Received Payload: Payload[READY] Client Id [1] Message: [null] isReady [ready]
jimmy@[1] is ready
11/15/2024 22:44:08 [Project.Client.Client] (INFO):
> Received Payload: Payload[PHASE] Client Id [0] Message: [MAKE_CHOICE]
Current phase as MAKE_CHOICE
11/15/2024 22:44:08 [Project.Client.Client] (INFO):
> Received Payload: Payload[MESSAGE] Client Id [1] Message: [make rps command] 'r
ps rock' ...
Room: make rps command! 'rps rock' ...
/rps rock
```

phase change

**Caption(s) (required)** ✓

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

End of Task 2

## Task



## Group: Round Start

### Task #3: Round Timer

Weight: ~33%

Points: ~0.42

COLLAPSE

## ● Details:

All code screenshots must have ucid/date visible.  
Any terminal output should have at least 3 clients involved.

Columns: 1

### Sub-Task



### Group: Round Start

### Task #3: Round Timer

**Sub Task #1:** Show the code that triggers the round timer (including the expiry callback)

## Task Screenshots

## Gallery Style: 2 Columns

```
16-7:14:00.0 [Info] [C:\Users\Kev\IdeaProjects\Comprehensive\src\main\java\com\kev\comprehensive\controller\BookController.java:101] (main)
17- updateBook: BookController [BookController]
18- [BookController]
19- [BookController]
20- [BookController]
21- [BookController]
22- [BookController]
23- [BookController]
24- [BookController]
25- [BookController]
26- [BookController]
27- [BookController]
28- [BookController]
29- [BookController]
30- [BookController]
31- [BookController]
32- [BookController]
33- [BookController]
34- [BookController]
35- [BookController]
36- [BookController]
37- [BookController]
38- [BookController]
39- [BookController]
40- [BookController]
41- [BookController]
42- [BookController]
43- [BookController]
44- [BookController]
45- [BookController]
46- [BookController]
47- [BookController]
48- [BookController]
49- [BookController]
50- [BookController]
51- [BookController]
52- [BookController]
53- [BookController]
54- [BookController]
55- [BookController]
56- [BookController]
57- [BookController]
58- [BookController]
59- [BookController]
60- [BookController]
61- [BookController]
62- [BookController]
63- [BookController]
64- [BookController]
65- [BookController]
66- [BookController]
67- [BookController]
68- [BookController]
69- [BookController]
70- [BookController]
71- [BookController]
72- [BookController]
73- [BookController]
74- [BookController]
75- [BookController]
76- [BookController]
77- [BookController]
78- [BookController]
79- [BookController]
80- [BookController]
81- [BookController]
82- [BookController]
83- [BookController]
84- [BookController]
85- [BookController]
86- [BookController]
87- [BookController]
88- [BookController]
89- [BookController]
90- [BookController]
91- [BookController]
92- [BookController]
93- [BookController]
94- [BookController]
95- [BookController]
96- [BookController]
97- [BookController]
98- [BookController]
99- [BookController]
```

reset and start timer

Reset and start timer ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

in start round, it resets timer after safety check and works so if multiple rounds are called, timer still works.

Sub-Task

Group: Round Start

Task #3: Round Timer

50%

Sub Task #2: Show how the clients are informed of the time (i.e., synced or stopwatch-like)

## Task Screenshots

Gallery Style: 2 Columns

4

2

1



Missing Caption

**Caption(s) (required)**

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

Missing caption(s)

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

to be frank, i didnt implement a client side clock... but i need to submit the assignment by tonight.

Sub-Task

Group: Round Start

Task #3: Round Timer

100%

Sub Task #3: Show an example from the terminal

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

Round Time: 15

```
Round Time: 14
Round Time: 13
Round Time: 12
Round Time: 11
Round Time: 10
Round Time: 9
Round Time: 8
```

server side timer message

**Caption(s) (required)** ✓

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

End of Task 3

End of Group: Round Start

Task Status: 2/3

Group

Group: During Round

Tasks: 1

Points: 2.5

100%

▲ COLLAPSE ▲

Task

Group: During Round

Task #1: Choice Command

Weight: ~100%

Points: ~2.50

100%

▲ COLLAPSE ▲

### ⓘ Details:

All code screenshots must have ucid/date visible.

Any terminal output should have at least 3 clients involved.



Columns: 1

Sub-Task

Group: During Round

Task #1: Choice Command

Sub Task #1: Show the code that processes the /choice text

100%

## 🖼 Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
public void receiveChoice(String choice) {
    if (choice.equals("A")) {
        // handle choice A logic
    } else if (choice.equals("B")) {
        // handle choice B logic
    }
}
```

code decides to not work anymore :c. works for two people still

#### Caption(s) (required) ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

player can send choice command only if they are "alive"/not eliminated, in which you get a message from the receiver that it was received.

#### Sub-Task

Group: During Round

100%

Task #1: Choice Command

Sub Task #2: Show the GameRoom code that handles the choice (stores it on the Player and sends a message) (show any checks to ensure only valid Players are handled)

## Task Screenshots

Gallery Style: 2 Columns

```
private void receiveChoice(String choice) {
    if (choice.equals("A")) {
        // handle choice A logic
    } else if (choice.equals("B")) {
        // handle choice B logic
    }
}
```

receiveChoice handler

#### Caption(s) (required) ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

this method handles the received choice, and makes changes to player accordingly, with safety checks as well to make they are "alive".

#### Sub-Task

Group: During Round

100%

Task #1: Choice Command

100%

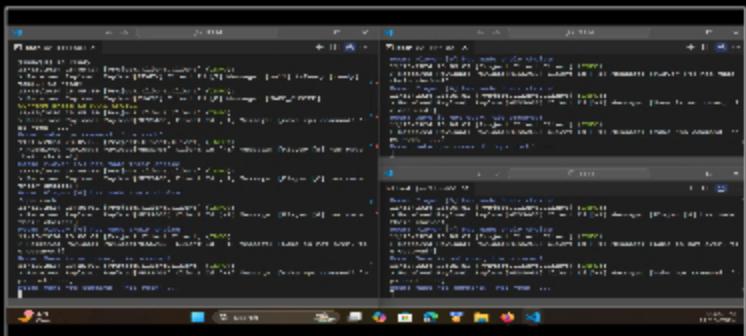
Task #1: Choice Command

Sub Task #3: Show 3 examples of terminal output showing Players picking their choice (opponents shouldn't see what was chosen)

## Task Screenshots

Gallery Style: 2 Columns

4      2      1



picking choice, other players see they made a choice, but not what they choose

**Caption(s) (required) ✓**

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

End of Task 1

End of Group: During Round

Task Status: 1/1

Group



Group: Round End

Tasks: 4

Points: 2.5

**▲ COLLAPSE ▲**

Task



Group: Round End

Task #1: Round End Conditions

Weight: ~25%

Points: ~0.63

**▲ COLLAPSE ▲**

### ⓘ Details:

All code screenshots must have ucid/date visible.  
Any terminal output should have at least 3 clients involved.



Columns: 1

Sub-Task

Group: Round End

Sub Task

Group: Round End

Task #1: Round End Conditions

Sub Task #1: Show the code related to the round ending from round timer expiring

100%

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
public void handleRoundEnd() {
    // Check if all players are dead
    if (allPlayersDead()) {
        // Set the round winner
        setRoundWinner();
        // Stop the round
        stopRound();
    }
}
```

1/3

```
public void handleRoundEnd() {
    // Check if all players are dead
    if (allPlayersDead()) {
        // Set the round winner
        setRoundWinner();
        // Stop the round
        stopRound();
    }
}
```

2/3

```
public void handleRoundEnd() {
    // Check if all players are dead
    if (allPlayersDead()) {
        // Set the round winner
        setRoundWinner();
        // Stop the round
        stopRound();
    }
}
```

3/3

Caption(s) (required) ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

handles logic for comparing players enter choices, attacker based, first person attacks second in list. checks at the end of there is one person left alive, in which then their is a winner.

Sub-Task

Group: Round End

Task #1: Round End Conditions

Sub Task #2: Show the code related to the round ending from all eligible Players marking their choices

100%

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
public void handleRoundEnd() {
    // Check if all players are dead
    if (allPlayersDead()) {
        // Set the round winner
        setRoundWinner();
        // Stop the round
        stopRound();
    }
}
```

```
// handle logic for win condition
if (player1.isPlayerAlive() && player2.isPlayerAlive()) {
    System.out.println("Both players are alive!");
} else if (player1.isPlayerAlive() & !player2.isPlayerAlive()) {
    System.out.println("Player 1 has won!");
    player1.setWinCount(player1.getWinCount() + 1);
    player2.setLoseCount(player2.getLoseCount() + 1);
    player1.setScore(player1.getScore() + 1);
}
else if (!player1.isPlayerAlive() & player2.isPlayerAlive()) {
    System.out.println("Player 2 has won!");
    player2.setWinCount(player2.getWinCount() + 1);
    player1.setLoseCount(player1.getLoseCount() + 1);
    player2.setScore(player2.getScore() + 1);
}
else if (!player1.isPlayerAlive() & !player2.isPlayerAlive()) {
    System.out.println("Both players have lost!");
}
```

this handles win condition

### Caption(s) (required) ✓

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

## → Task Response Prompt

*Explain in concise steps how this logically works*

Response:

when one player is remaining, send message of who won.

## End of Task 1

### Task



Group: Round End  
Task #2: Battles  
Weight: ~25%  
Points: ~0.63

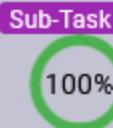
▲ COLLAPSE ▲

### ⓘ Details:

All code screenshots must have ucid/date visible.  
Any terminal output should have at least 3 clients involved.



Columns: 1



Sub-Task  
Group: Round End  
Task #2: Battles  
Sub Task #1: Show the code related to doing all the battles in round-robin

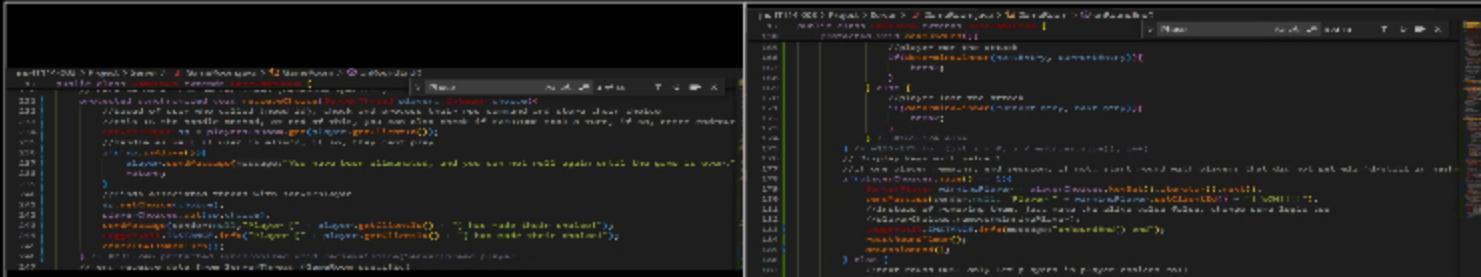
## 🖼 Task Screenshots

Gallery Style: 2 Columns

4

2

1



**Caption(s) (required) ✓**

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

**Task Response Prompt**

*Explain in concise steps how this logically works*

Response:

.compareTo function checks with integer version of rps, and depending on >=, process differently.

**Sub-Task**

Group: Round End

Task #2: Battles

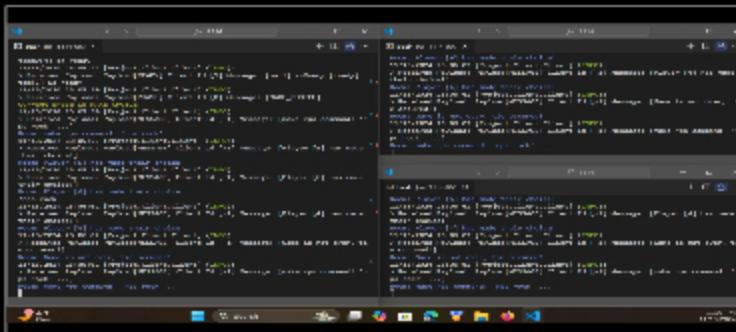
Sub Task #2: Show the code related to relaying the appropriate outcome messages

100%

**Task Screenshots**

Gallery Style: 2 Columns

4      2      1



logic of post commands

**Caption(s) (required) ✓**

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

**Task Response Prompt**

*Explain in concise steps how this logically works*

Response:

in this case, all players draw'd, continue rounds.

**Sub-Task**

Group: Round End

Task #2: Battles

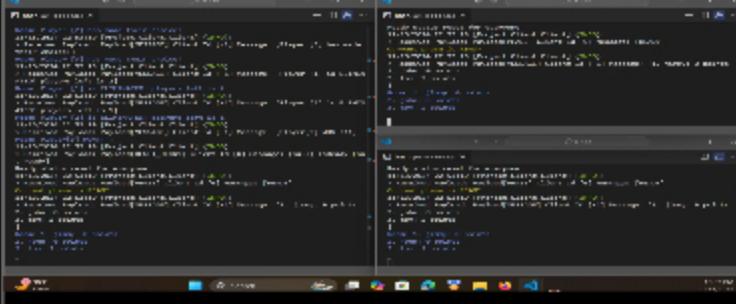
Sub Task #3: Show the code related to winning Players getting points (and syncing to clients)

100%

**Task Screenshots**

Gallery Style: 2 Columns

4      2      1



displaying leaderboard

**Caption(s) (required)** ✓

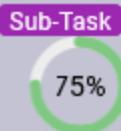
**Caption Hint:** *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works.*

### **Response:**

after game ends, leaderboard is shown with player points.



### Group: Round End

## Task #2: Battles

Sub Task #4: Show terminal output of the battle outcomes

## Task Screenshots

## Gallery Style: 2 Columns



```
4

class Program
{
    static void Main(string[] args)
    {
        Print();
        Print();
    }

    static void Print()
    {
        Console.WriteLine("Hello World!");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Print();
        Print();
    }

    static void Print()
    {
        Console.WriteLine("Hello World!");
    }
}
```

## Missing Caption

**Caption(s) (required)**

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

End of Task 2



### Group: Round End

### Task #3: Elimination

Weight: ~25%

Points: ~0.63

COLLAPSE

### 1 Details:

All code screenshots must have ucid/date visible.  
Any terminal output should have at least 3 clients involved.



Columns: 1

#### Sub-Task

100%

Group: Round End

Task #3: Elimination

Sub Task #1: Show the code related to Players being eliminated during a battle (subset of the battle code above likely)

## Task Screenshots

Gallery Style: 2 Columns

4 2 1

```
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
```

determines winner and loser

Caption(s) (required) ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

this segment runs in roundend, in which loser is marked no longer alive, and winner earns a point

#### Sub-Task

50%

Group: Round End

Task #3: Elimination

Sub Task #2: Show the code that eliminates Players who didn't pick a choice in time

## Task Screenshots

Gallery Style: 2 Columns

4 2 1





Missing Caption

**Caption(s) (required)**

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

Missing caption(s)

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

not implemented :((

**Sub-Task**

Group: Round End

100%

Task #3: Elimination

Sub Task #3: During a battle, what determines a Player gets eliminated, is it only if they're attacking, only if they're defending, or either?

## Task Response Prompt

*Explain*

Response:

attacking players are eliminated if they lose the attack.

End of Task 3

**Task**

Group: Round End

100%

Task #4: Next Round or Session End

Weight: ~25%

Points: ~0.63

**▲ COLLAPSE ▲**

**① Details:**

All code screenshots must have ucid/date visible.  
Any terminal output should have at least 3 clients involved.

**Conditions:**

Columns: 1

**Sub-Task**

Group: Round End

100%

Task #4: Next Round or Session End

Sub Task #1: Show the code that checks how many eligible Players are left

## Task Screenshots

4

2

1

## checks

**Caption(s) (required)** ✓

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

### Task Response Prompt

*Explain in concise steps how this logically works*

### **Response:**

checks if size is greater than one, if so, new round entails.

### **Sub-Task**

### Group: Round End

#### Task #4: Next Round or Session End

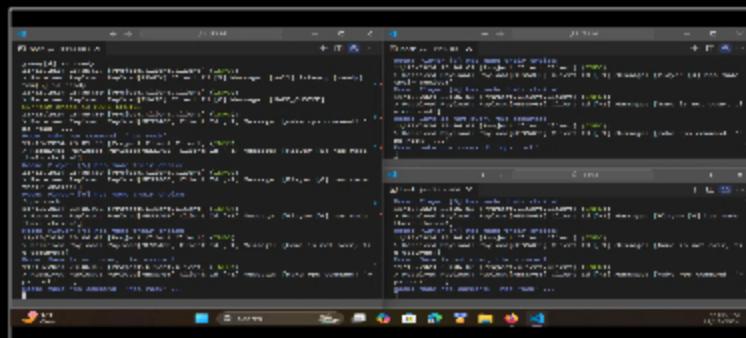
**Sub Task #2: Show the relevant terminal output**

## Task Screenshots

4

2

1



tie occurred, but round continues again

**Caption(s) (required)** ✓

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

End of Task 4

#### **End of Group: Round End**

Task Status: 2/4

## Group

100%

Group: Session End

### Tasks: 3

Points: 1.25

COLLAPSE

### Task

Group: Session End

## Task #1: Session End Conditions

Weight: ~33%

Points: ~0.42

COLLAPSE

### Details:

All code screenshots must have ucid/date visible.



Columns: 1

### Sub-Task

### Group: Session End

## Task #1: Session End Conditions

**Sub Task #1:** Show the code that ends a session with a winner (likely similar to a previous task)

## Task Screenshots

## Gallery Style: 2 Columns

resets values, creates leaderboard

**Caption(s) (required) ✓**

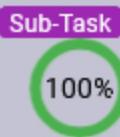
**Caption Hint:** *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

#### **Response:**

resets all player values and creates leaderboard with a string.

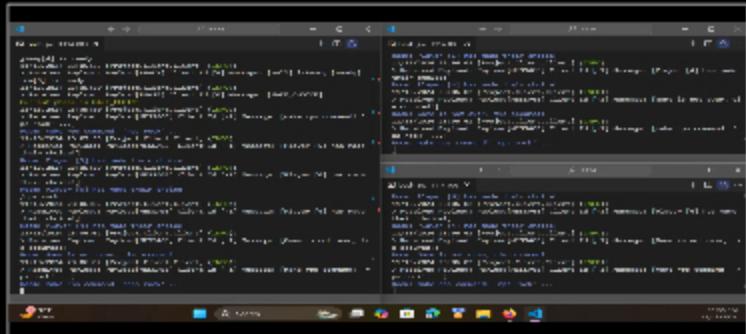


Sub-Task  
Group: Session End  
Task #1: Session End Conditions  
Sub Task #2: Show the code that ends a session with a tie (likely similar to a previous task)

## Task Screenshots

Gallery Style: 2 Columns

4      2      1



tie

Caption(s) (required) ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

tie occurred, starts new round

End of Task 1

### Task



Group: Session End  
Task #2: Scoreboard  
Weight: ~33%  
Points: ~0.42

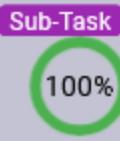
[▲ COLLAPSE ▲](#)

### Details:

All code screenshots must have ucid/date visible.  
Any terminal output should have at least 3 clients involved.



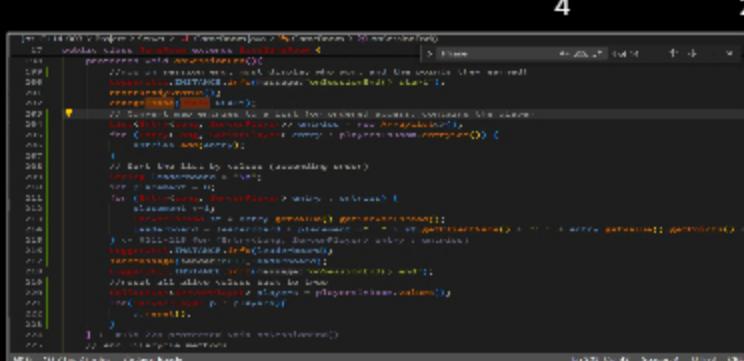
Columns: 1



Sub-Task  
Group: Session End  
Task #2: Scoreboard  
Sub Task #1: Show the code related to the final scoreboard

## Task Screenshots

Gallery Style: 2 Columns



```
for player in sorted_players:
    # Create a string for each player's name and score
    leaderboard += f'{player} {score}\n'

# Print the final leaderboard
print(leaderboard)
```

creates leaderboard string

**Caption(s) (required)** ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

creates leaderboard string with player scores

Sub-Task



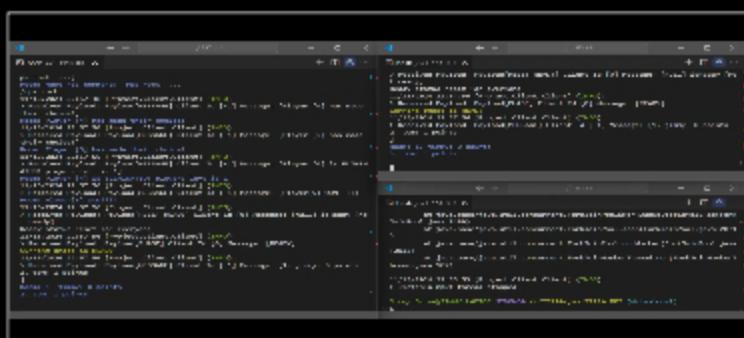
Group: Session End

Task #2: Scoreboard

Sub Task #2: Show the final scoreboard from the terminal

## Task Screenshots

Gallery Style: 2 Columns



```
root@kali:~/Desktop/Pygame/Project 1# python3 scoreboard.py
Player Score
John 100
Jane 90
Mike 80
Sarah 70
David 60
Emily 50
Katie 40
Chris 30
Jordan 20
Olivia 10
```

leaderboard

**Caption(s) (required)** ✓

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

End of Task 2

Task

Group: Session End

100%

**Task #3: Cleanup**  
Weight: ~33%  
Points: ~0.42

COLLAPSE

### Details:

All code screenshots must have ucid/date visible.  
Any terminal output should have at least 3 clients involved.



Columns: 1

### Sub-Task

### Group: Session End

### Task #3: Cleanup

**Sub Task #1:** Show the code related to resetting the Player data client-side and server-side (do not disconnect them or move them to the lobby).

## Task Screenshots

## Gallery Style: 2 Columns

4 2

deals with resetting player values for each serverplayer

**Caption(s) (required)** ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

### **Response:**

**code snippet shown in previous boxes**

## Sub-Task

Group: Session End

### Task #3: Cleanup

Sub Task #2: Show any terminal evidence

## Task Screenshots

## Gallery Style: 2 Columns

4 2

4 2

4 2

shown before

**Caption(s) (required)** ✓

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

<b>Sub-Task</b>	Group: Session End
 100%	Task #3: Cleanup
	Sub Task #3: Show the code related to shifting back to the ready Phase

## Task Screenshots

## Gallery Style: 2 Columns

4 2 1

```
197     private void handleConnection(Communication communication) {
198         if (communication instanceof Broadcast) {
199             handleBroadcast((Broadcast) communication);
200         } else if (communication instanceof Message) {
201             handleMessage((Message) communication);
202         }
203     }
204
205     private void handleBroadcast(Broadcast broadcast) {
206         String broadcastType = broadcast.getBroadcastType();
207         if (broadcastType.equals("player")) {
208             handlePlayerBroadcast(broadcast);
209         } else if (broadcastType.equals("team")) {
210             handleTeamBroadcast(broadcast);
211         } else if (broadcastType.equals("arena")) {
212             handleArenaBroadcast(broadcast);
213         } else if (broadcastType.equals("global")) {
214             handleGlobalBroadcast(broadcast);
215         }
216     }
217
218     private void handlePlayerBroadcast(Broadcast broadcast) {
219         String broadcastContent = broadcast.getBroadcastContent();
220         String broadcastTitle = broadcast.getBroadcastTitle();
221         String broadcastAuthor = broadcast.getBroadcastAuthor();
222         String broadcastType = broadcast.getBroadcastType();
223         String broadcastTime = broadcast.getBroadcastTime();
224
225         Player broadcastPlayer = broadcast.getPlayer();
226         if (broadcastPlayer != null) {
227             broadcastPlayer.sendMessage(broadcastContent);
228             broadcastPlayer.setBroadcastTitle(broadcastTitle);
229             broadcastPlayer.setBroadcastAuthor(broadcastAuthor);
230             broadcastPlayer.setBroadcastType(broadcastType);
231             broadcastPlayer.setBroadcastTime(broadcastTime);
232         }
233     }
234
235     private void handleTeamBroadcast(Broadcast broadcast) {
236         String broadcastContent = broadcast.getBroadcastContent();
237         String broadcastTitle = broadcast.getBroadcastTitle();
238         String broadcastAuthor = broadcast.getBroadcastAuthor();
239         String broadcastType = broadcast.getBroadcastType();
240         String broadcastTime = broadcast.getBroadcastTime();
241
242         Team broadcastTeam = broadcast.getTeam();
243         if (broadcastTeam != null) {
244             broadcastTeam.sendMessage(broadcastContent);
245             broadcastTeam.setBroadcastTitle(broadcastTitle);
246             broadcastTeam.setBroadcastAuthor(broadcastAuthor);
247             broadcastTeam.setBroadcastType(broadcastType);
248             broadcastTeam.setBroadcastTime(broadcastTime);
249         }
250     }
251
252     private void handleArenaBroadcast(Broadcast broadcast) {
253         String broadcastContent = broadcast.getBroadcastContent();
254         String broadcastTitle = broadcast.getBroadcastTitle();
255         String broadcastAuthor = broadcast.getBroadcastAuthor();
256         String broadcastType = broadcast.getBroadcastType();
257         String broadcastTime = broadcast.getBroadcastTime();
258
259         Arena broadcastArena = broadcast.getArena();
260         if (broadcastArena != null) {
261             broadcastArena.sendMessage(broadcastContent);
262             broadcastArena.setBroadcastTitle(broadcastTitle);
263             broadcastArena.setBroadcastAuthor(broadcastAuthor);
264             broadcastArena.setBroadcastType(broadcastType);
265             broadcastArena.setBroadcastTime(broadcastTime);
266         }
267     }
268
269     private void handleGlobalBroadcast(Broadcast broadcast) {
270         String broadcastContent = broadcast.getBroadcastContent();
271         String broadcastTitle = broadcast.getBroadcastTitle();
272         String broadcastAuthor = broadcast.getBroadcastAuthor();
273         String broadcastType = broadcast.getBroadcastType();
274         String broadcastTime = broadcast.getBroadcastTime();
275
276         Global broadcastGlobal = broadcast.getGlobal();
277         if (broadcastGlobal != null) {
278             broadcastGlobal.sendMessage(broadcastContent);
279             broadcastGlobal.setBroadcastTitle(broadcastTitle);
280             broadcastGlobal.setBroadcastAuthor(broadcastAuthor);
281             broadcastGlobal.setBroadcastType(broadcastType);
282             broadcastGlobal.setBroadcastTime(broadcastTime);
283         }
284     }
285 }
```

changes phase to ready

**Caption(s) (required)** ✓

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

## Task Response Prompt

*Explain in concise steps how this logically works*

### Response:

in session end, changes phase back to ready

End of Task 3

End of Group: Session End

Task Status: 3/3

### 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://learn.ethereallab.app/v>

URL

<https://learn.ethereallab.app/v>

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:

so many issues. so many struggles, but preserved. even thought about dropping the class ngl. and this milestone was a mess but i did not accurately schelude my time, so i spent the last three days cramming this milestone, but for the next one, I will make to do better and not leave it for too late.

End of Task 2

### Task



Group: Misc

Task #3: WakaTime Screenshot

Weight: ~33%

Points: ~0.33

[▲ COLLAPSE ▲](#)

## 1 Details:

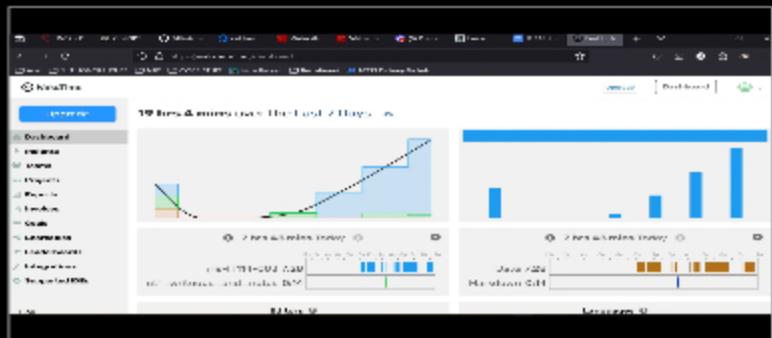
Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved



## Task Screenshots

Gallery Style: 2 Columns

4      2      1



im free.

End of Task 3

End of Group: Misc

Task Status: 3/3

End of Assignment