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
Assigment: [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-QXP3pyQ7xN3lRxwmcwCc/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: 1

COLLAPSE

Task

100%

Group: Payloads
Task #1: Base Payload Class
Weight: ~50%
Points: ~0.50

COLLAPSE

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

Columns: 1

Sub-Task

100%

Group: Payloads
Task #1: Base Payload Class
Sub Task #1: Show screenshot of the code for this file

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

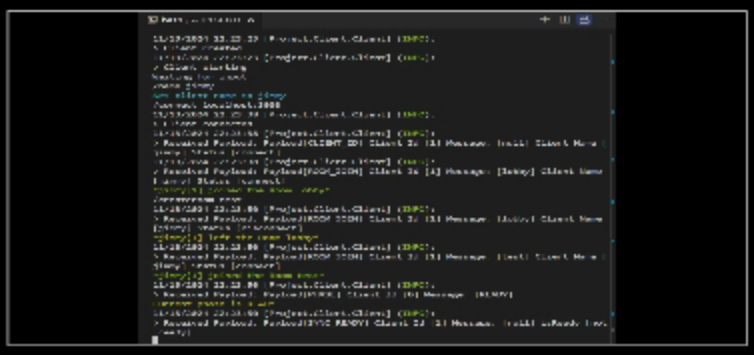
100%

Group: Payloads
Task #1: Base Payload Class
Sub Task #2: Show screenshot examples of the terminal output for this object

Task Screenshots

Gallery Style: 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

100%

Group: Payloads
Task #2: PointsPayload (or equivalent)
Weight: ~50%
Points: ~0.50

COLLAPSE

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

Columns: 1

Sub-Task

100%

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:

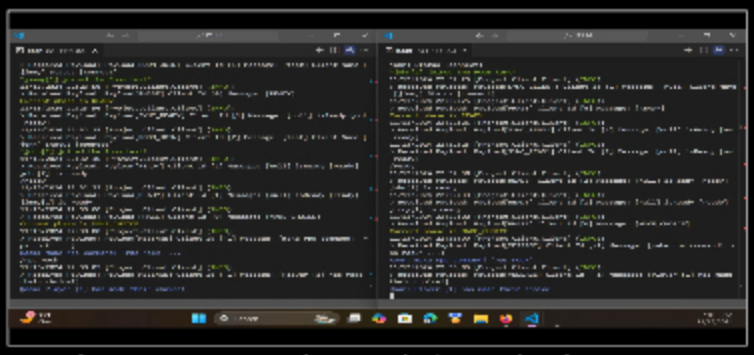
choice payload handles sending the users choice from client, converts the data from a string to a number for easier calculations

Sub-Task
100%
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
100%
Group: Session Start
Tasks: 1
Points: .5
COLLAPSE

Task
100%
Group: Session Start
Task #1: Trigger Round Start

Weight: ~100%
Points: ~0.50

COLLAPSE

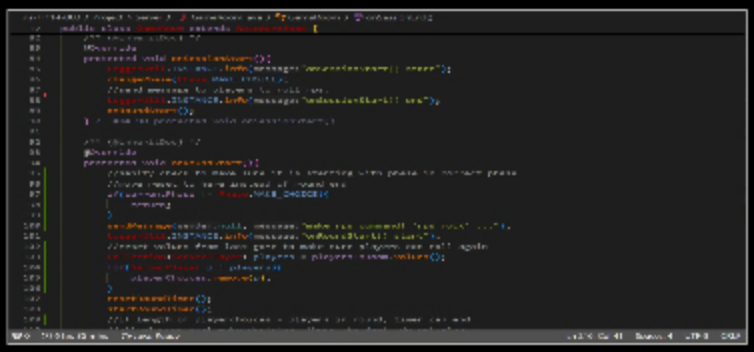
Details:
All code screenshots must have ucid/date visible.

Sub-Task
100%
Group: Session Start
Task #1: Trigger Round Start
Sub Task #1: Show the related code

Task Screenshots

Gallery Style: 2 Columns

4 2 1



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
94%
Group: Round Start
Tasks: 3
Points: 1.25

COLLAPSE

Task

100%

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

COLLAPSE

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

Sub-Task

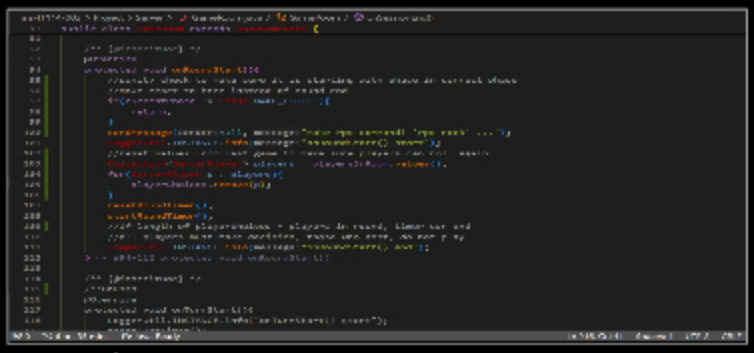
100%

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



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 ucid/date visible.

Columns: 1

Sub-Task

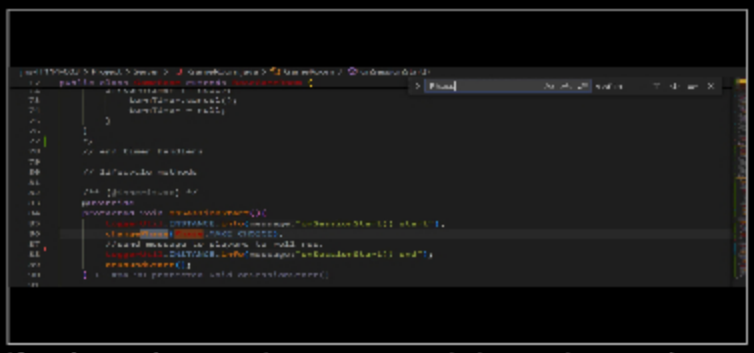
100%

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

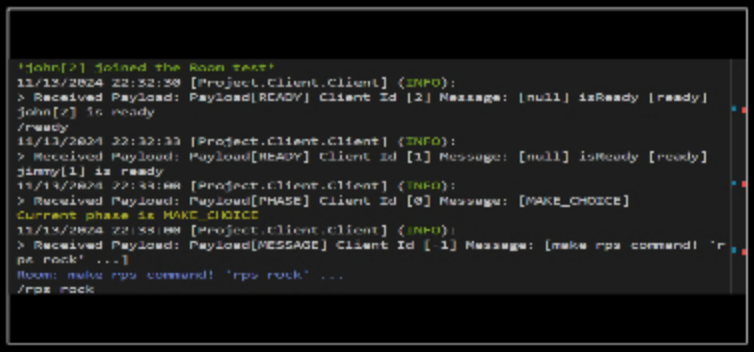
100%

Group: Round Start
Task #2: Choosing Phase (or similar)
Sub Task #2: Show terminal output of this change

Task Screenshots

Gallery Style: 2 Columns

4 2 1



phase change

Caption(s) (required) ✓

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

End of Task 2

Task

83%

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

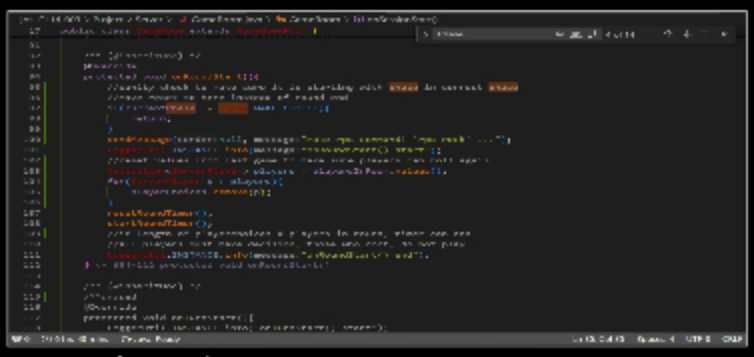
100%

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

4 2 1



reset and start timer

Caption(s) (required) ✓
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 50%
Group: Round Start
Task #3: Round Timer
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:

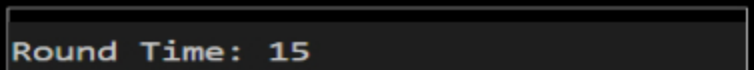
to be frank, i didnt implement a client side clock... but i need to submit the assignment by tonight.

Sub-Task 100%
Group: Round Start
Task #3: Round Timer
Sub Task #3: Show an example from the terminal

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
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

100%

Group: During Round
Tasks: 1
Points: 2.5

COLLAPSE

Task

100%

Group: During Round
Task #1: Choice Command
Weight: ~100%
Points: ~2.50

COLLAPSE

Details:

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

Columns: 1

Sub-Task

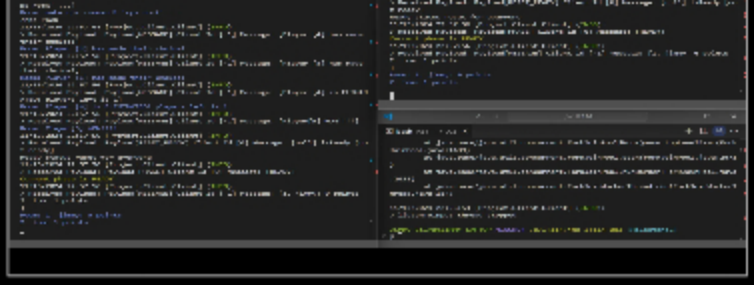
100%

Group: During Round
Task #1: Choice Command
Sub Task #1: Show the code that processes the /choice text

Task Screenshots

Gallery Style: 2 Columns

4 2 1



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 recieve that it was recieved.

Sub-Task

100%

Group: During Round

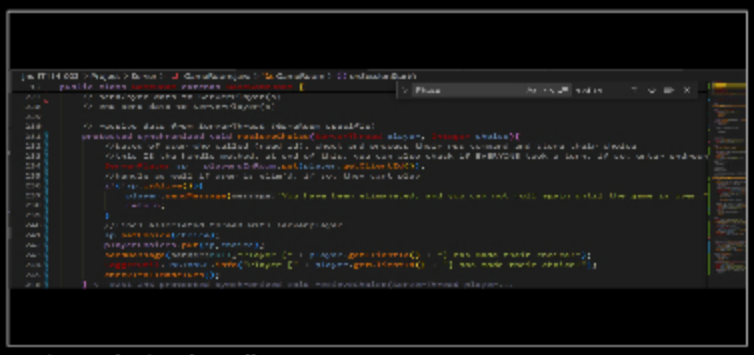
Task #1: Choice Command

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

4 2 1



recieveChoice 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

Task #1: Choice Command

100%

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

94%

Group: Round End

Tasks: 4

Points: 2.5

COLLAPSE

Task

100%

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

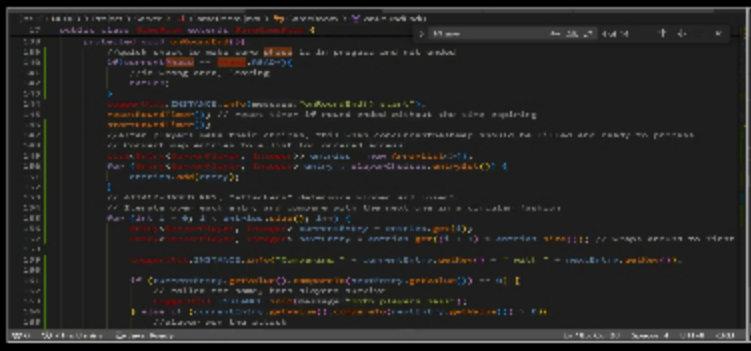
Sub-Task Group: Round End

Group: Round End
100%
Task #1: Round End Conditions
Sub Task #1: Show the code related to the round ending from round timer expiring

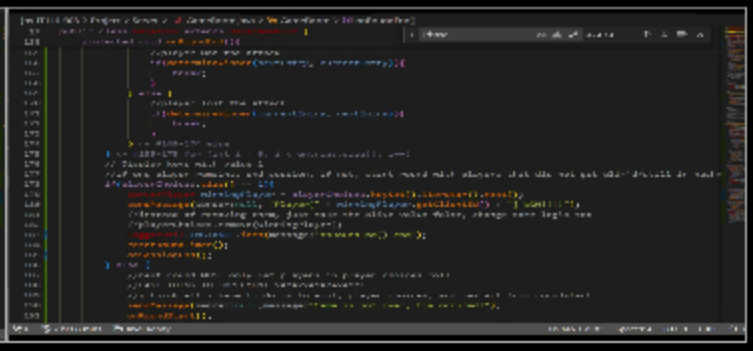
Task Screenshots

Gallery Style: 2 Columns

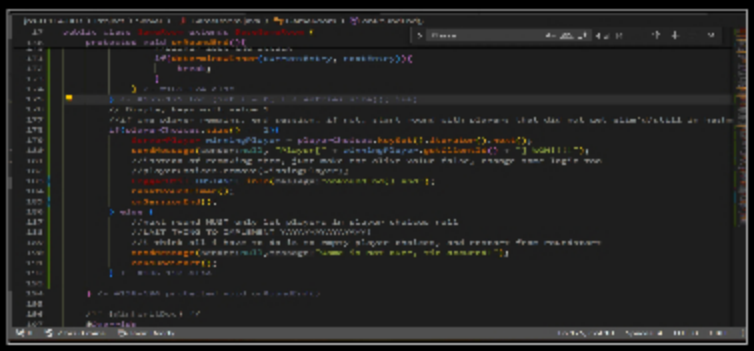
4 2 1



1/3



2/3



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:

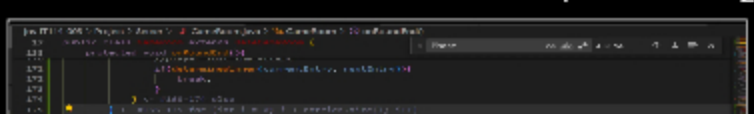
handles logic for comparing players enter choices, attacker based, first person attacks second in list. checks at the end of there is one person left alive, in which then their is a winner.

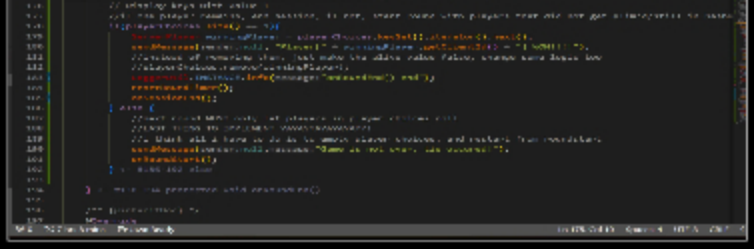
Sub-Task
100%
Group: Round End
Task #1: Round End Conditions
Sub Task #2: Show the code related to the round ending from all eligible Players marking their choices

Task Screenshots

Gallery Style: 2 Columns

4 2 1





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

93%

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

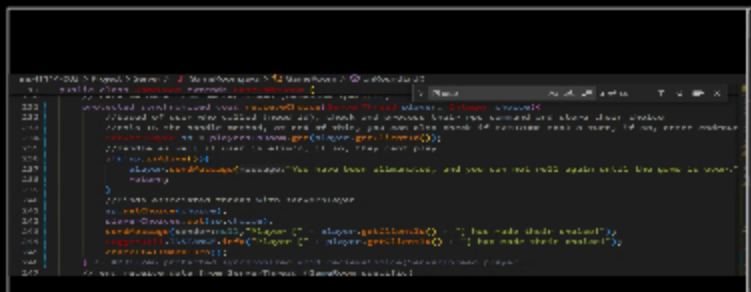
100%

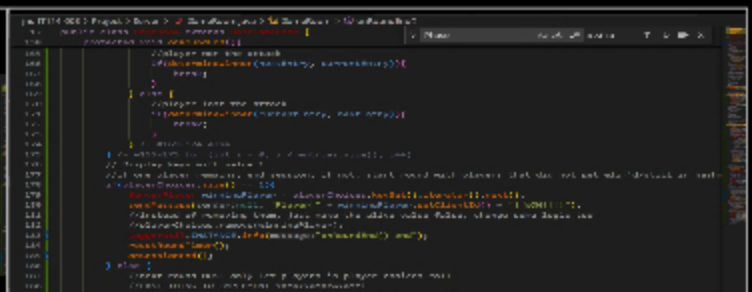
Group: Round End
Task #2: Battles
Sub Task #1: Show the code related to doing all the battles in round-robin

Task Screenshots

Gallery Style: 2 Columns

4 2 1





1/2

2/2

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

100%

Group: Round End

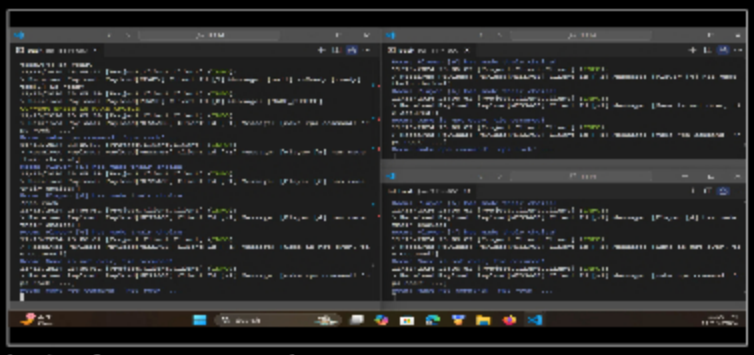
Task #2: Battles

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

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

100%

Group: Round End

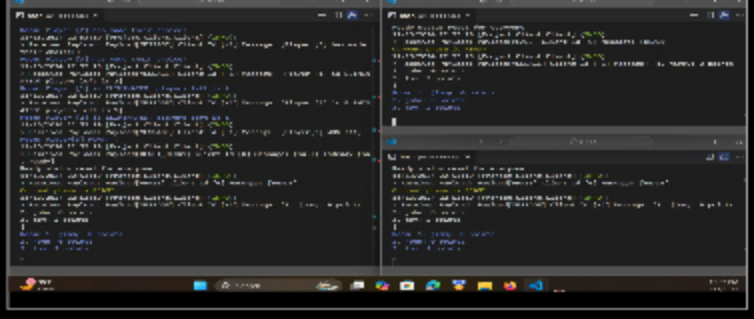
Task #2: Battles

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

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.

Sub-Task

75%

Group: Round End

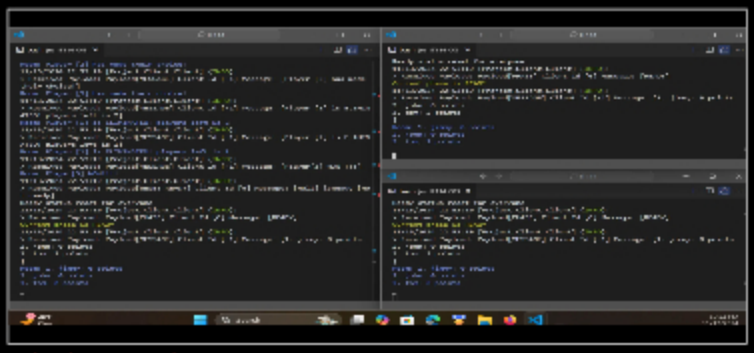
Task #2: Battles

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

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)

End of Task 2

Task

83%

Group: Round End

Task #3: Elimination

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.

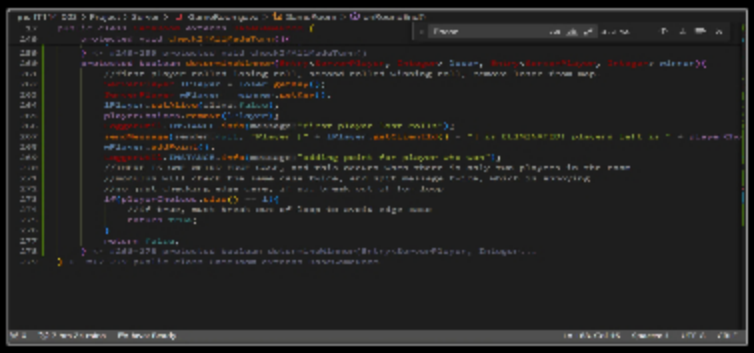
Columns: 1

Sub-Task
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



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
100%
Group: Round End
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
100%
Group: Round End
Task #4: Next Round or Session End
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.

Conditions:

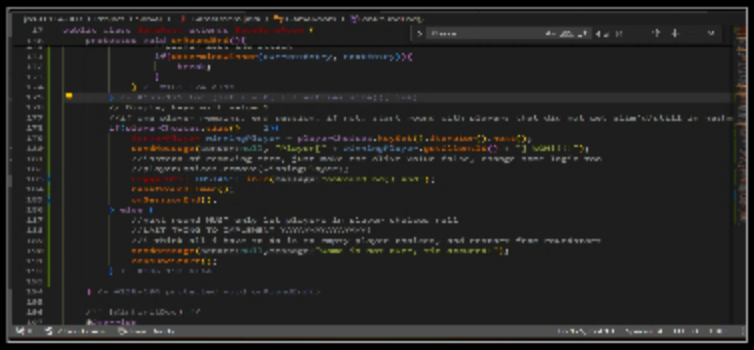
Columns: 1

Sub-Task
100%
Group: Round End
Task #4: Next Round or Session End
Sub Task #1: Show the code that checks how many eligible Players are left

Task Screenshots

Gallery Style: 2 Columns

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 if greater then one, if so, new round intails.

Sub-Task

100%

Group: Round End

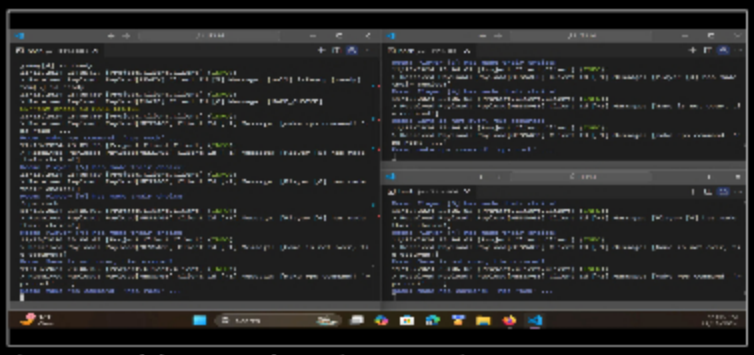
Task #4: Next Round or Session End

Sub Task #2: Show the relevant terminal output

Task Screenshots

Gallery Style: 2 Columns

4 2 1



tie occured, 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

100%

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

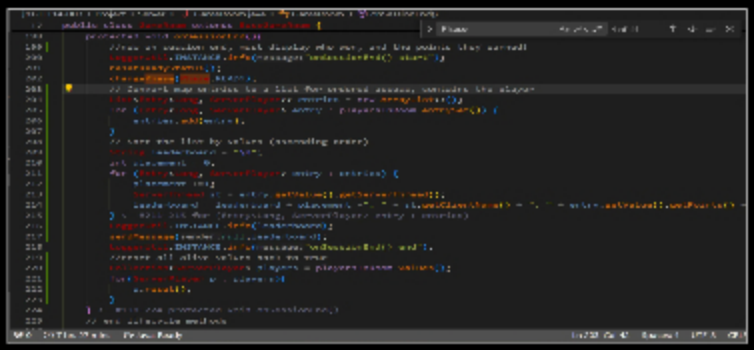
100%

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

4 2 1



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

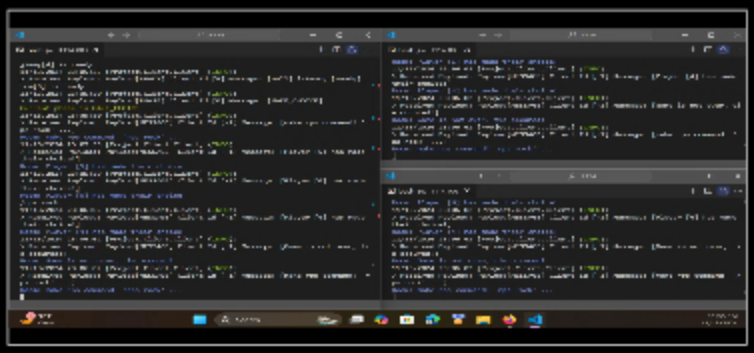
100%

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 occured, starts new round

End of Task 1

Task

100%

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

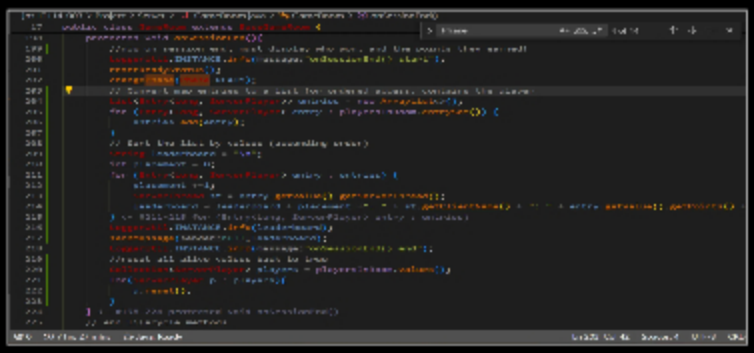
100%

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1



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

100%

Group: Session End

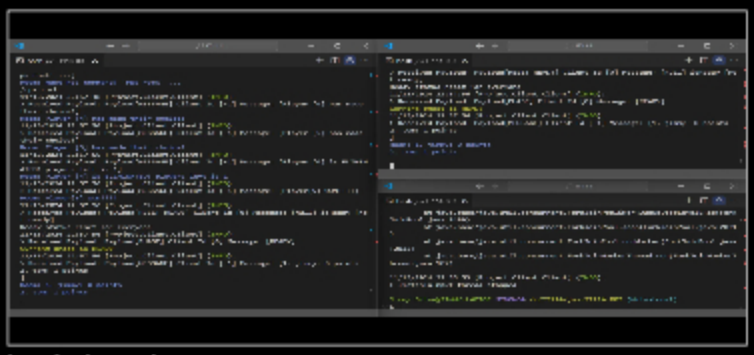
Task #2: Scoreboard

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1



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

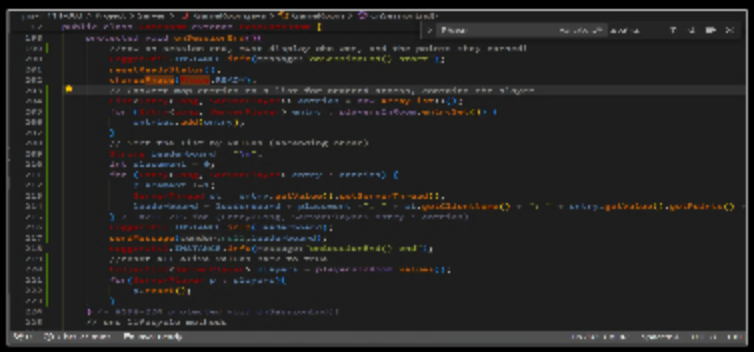
100%

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 1



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 snippit shown in previous boxes

Sub-Task

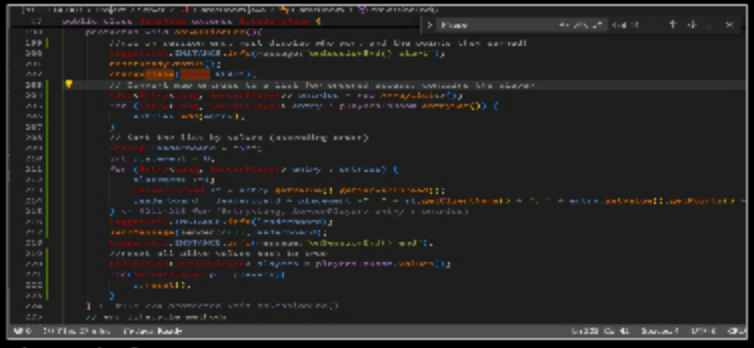
100%

Group: Session End
Task #3: Cleanup
Sub Task #2: Show any terminal evidence

Task Screenshots

Gallery Style: 2 Columns

4 2 1



shown before

Caption(s) (required) ✓

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

Sub-Task

100%

Group: Session End

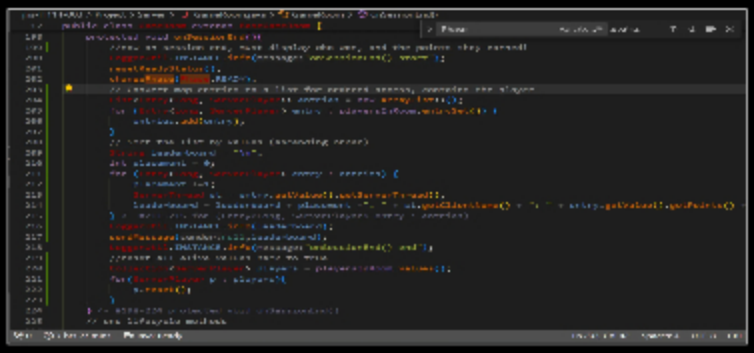
Task #3: Cleanup

Sub Task #3: Show the code related to shifting back to the ready Phase

Task Screenshots

Gallery Style: 2 Columns

4 2 1



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

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://learn.ethereallab.app/v

URL
https://learn.ethereallab.app/v

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:

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

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

im free.

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment