Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-milestone-4-rps-2024-m24/grade/jns

Course: IT114-003-F2024
Assigment: [IT114] Milestone 4 RPS 2024 M24
Student: Jimmy S. (jns)

Submissions:

Submission Selection

1 Submission [submitted] 12/10/2024 7:55:37 PM

Instructions

COLLAPSE

- Implement the Milestone 4 features from the project's proposal document: https://docs.google.com/document/d/11SRMo7JkLAMM-PuuiGwl_Z-QXP3pyQ7xN3lRxwmcwCc/view
- Make sure you add your ucid/date as code comments where code changes are done
- All code changes should reach the Milestone4 branch
- Create a pull request from Milestone4 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: Milestone4

Group

Group: Away
Tasks: 1
Points: 2.5

COLLAPSE

Task

100%

Group: Away
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game
Weight: ~100%
Points: ~2.50

COLLAPSE

Columns: 1

Sub-Task

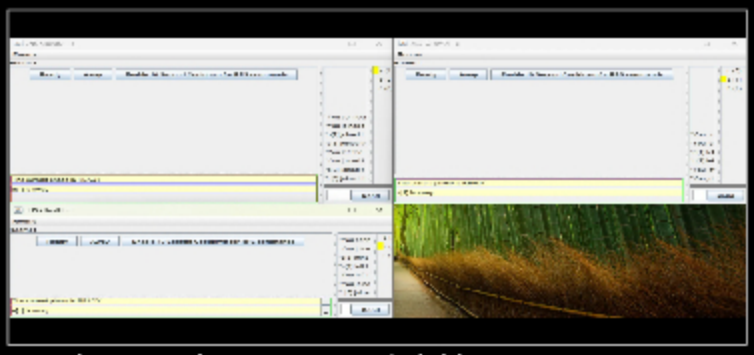
100%

Group: Away
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game
Sub Task #1: Screenshots of editors must have the frame title visible with your ucid and the client name. Code screenshots must have ucid/data comments.

Task Screenshots

Gallery Style: 2 Columns

4 2 1



two players and one spectator in lobby

Caption(s) (required) ✓

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

Sub-Task

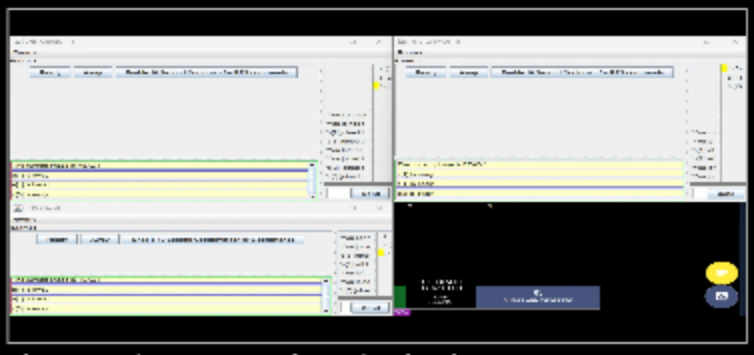
100%

Group: Away
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game
Sub Task #2: Show a few examples of the Game Events Panel showing away/not away status in a clear message

Task Screenshots

Gallery Style: 2 Columns

4 2 1



players going away and coming back

players going away and coming back

Caption(s) (required) ✓

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

Sub-Task 100%

Group: Away

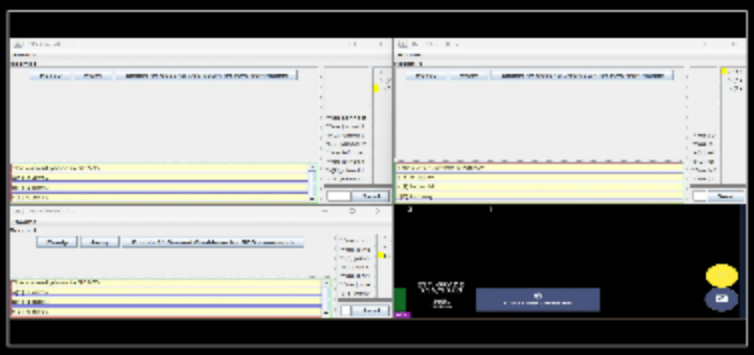
Task #1: Away: Client can mark themselves "away" to be skipped in the turn flow but still be in the game

Sub Task #3: Show a few examples of the User List panel showing a Client away (demonstrate that this changes when the status is toggled)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



the yellow boxes represent if the user is away

Caption(s) (required) ✓

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

Sub-Task 100%

Group: Away

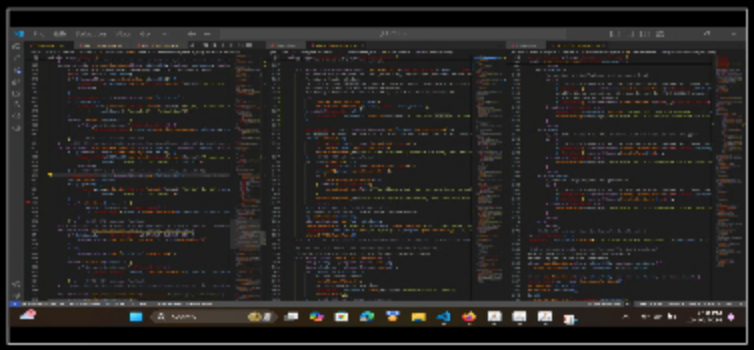
Task #1: Away: Client can mark themselves "away" to be skipped in the turn flow but still be in the game

Sub Task #4: Show the code flow that handles the away toggle (from Client interaction to updating server-side state)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



crazy to show full code flow, it goes through a lot of files, this shows some of the main logic in client, game, and servthread

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

readypanel: ui recieves click event from user on away button, tries to send away in client client: creates away payload, and sends to serverthread serverthread: processes payload, and handles it in gameroom gameroom: markes player as away, and has blockers created in recieve choice to stop the user from send rps commands while they are away serverthread: gameroom tells other rooms that user is away, and sends to other clients clients: recieve away payload for specific users, and updates the ui to show yellow symbol that the user is away

Sub-Task

100%

Group: Away

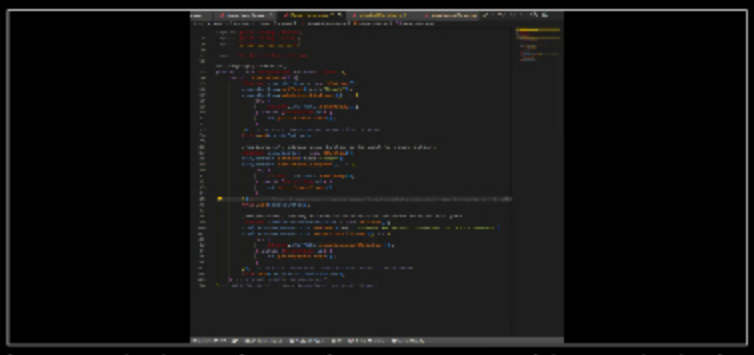
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game

Sub Task #5: Show the code that handles the Game Events Panel message (from the server-side state changing to sending the payload)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



button exist in ready panel, not game panel, just so its in the phase before game phase (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:

event for away button, when clicked, it will try to sendAway in the client.

Sub-Task

100%

Group: Away

Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game

Sub Task #6: Show the code that handles updating the User List Panel (from Client receiving -> the UI change)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

userlistpanel ui, recieveAway handles incoming user information about clients away statuses.

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

when client recieves a away payload and processes it, it will send to recieveAway here, and mark user with yellow set turn.

Sub-Task

100%

Group: Away

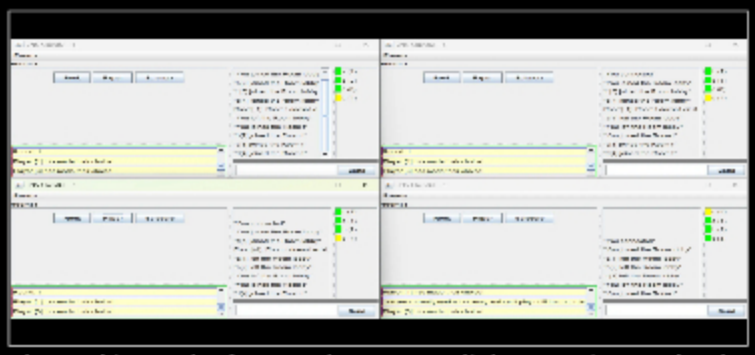
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game

Sub Task #7: Show the project logic that skips/ignores away players (away players can't take a turn) and include a UI screenshot of an applicable message if someone tries the action while away

Task Screenshots

Gallery Style: 2 Columns

4 2 1



player d is marked away, they try to click an option and only they get a message that they cant play

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

away players can still try to send rps commands, but if they are marked away, gameroom checks it and stops the

choice from being determined. in future thoughts, would probably utilize the client ui instead to stop the gameroom from dealing with this.

End of Task 1

End of Group: Away
Task Status: 1/1

Group

75%

Group: Spectator
Tasks: 1
Points: 2.5

COLLAPSE

Task

75%

Group: Spectator
Task #1: Spectator: Client can join as spectator
Weight: ~100%
Points: ~2.50

COLLAPSE

Details:
Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.
Spectator control/access logic must be handled in the GameRoom.
They can see all chat but are ignored from turns and can't send messages

Columns: 1

Sub-Task

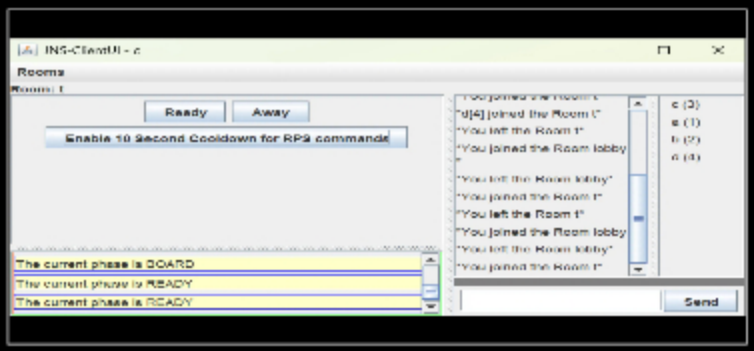
100%

Group: Spectator
Task #1: Spectator: Client can join as spectator
Sub Task #1: Show the spectator UI and demonstrate how they're blocked/ignored from game actions (turns and messages)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



client c is a spectator, was rushing implementing this so

client c is a spectator, was rushing implementing this so they is nothing displayed in ui but they cant send commands

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

Sub-Task 0%
Group: Spectator
Task #1: Spectator: Client can join as spectator
Sub Task #2: Show the User List Panel representation of Spectators (must differ from active Players)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Missing Caption

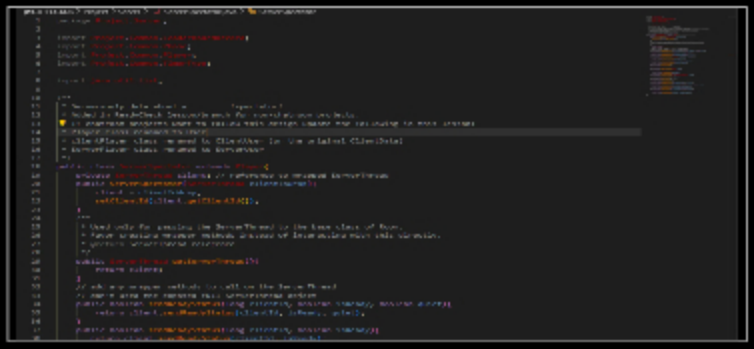
Caption(s) (required)
Caption Hint: *Describe/highlight what's being shown*
Missing caption(s)

Sub-Task 100%
Group: Spectator
Task #1: Spectator: Client can join as spectator
Sub Task #3: Show the code logic that blocks/ignores spectators from game actions (turns and messages)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



different serverthread wrapper for spectators that comes with limited features that make them not able to play

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

a few things are utilized so that spectators cant send inputs: serverSpectator: wrapper class of serverthread, limited things it can do basegameroom: when client joins a game room, they have a different onClientAdded that moves them into a hashmap of spectators, therefore ignoring them in any game logic involving playersInRoom gameroom: conditional checks for spectators trying to send commands, checks them so that room does not throw a bunch of errors (defo could have handled this better but uh this due tmr)

Sub-Task

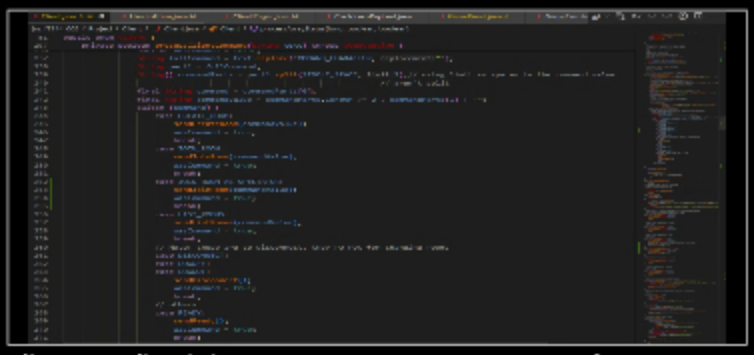
100%

Group: Spectator
Task #1: Spectator: Client can join as spectator
Sub Task #4: Show the code flow of the server-side sending the spectator status to Clients and having the User List Panel updated accordingly

Task Screenshots

Gallery Style: 2 Columns

4 2 1



client sending join room as spectator command

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

roomspanel: new button to join as spectator, event for when click, in which it will sendJoinRoomAsSpectator client method serverthread: processes it and sends to basegameroom basegmrm: handles in onSpectatorAdded, put user in spectator hash map

not fully implement, doesnt send anything to other user ui's, no real visual showing that someone is a spectator

End of Task 1

End of Group: Spectator
Task Status: 0/1

Group

50%

Group: Project Specific
Tasks: 2
Points: 4

COLLAPSE

Task

0%

Group: Project Specific
Task #1: Implement extra options beyond Rock Paper and Scissors
Weight: ~50%
Points: ~2.00

COLLAPSE

Details:
Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

- Optionally only activate this at different stages of the game (i.e., last 3 people)

Columns: 1

Sub-Task

0%

Group: Project Specific
Task #1: Implement extra options beyond Rock Paper and Scissors
Sub Task #1: Show the pre-game screen (can be ready panel) that toggles the extra option mode

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

Note the extra options you went with and how the work
Response:
Missing Response

Sub-Task Group: Project Specific

0%

Task #1: Implement extra options beyond Rock Paper and Scissors
Sub Task #2: Show the code related to handling these extra options and where it's used/set (on session start or towards the last few rounds, note this as well)

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:
Missing Response

End of Task 1

Task

100%

Group: Project Specific
Task #2: Implement a cooldown on an option (i.e., same option can't be picked twice in a row by the same player)
Weight: ~50%
Points: ~2.00

COLLAPSE

Details:
Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

Columns: 1

Sub-Task

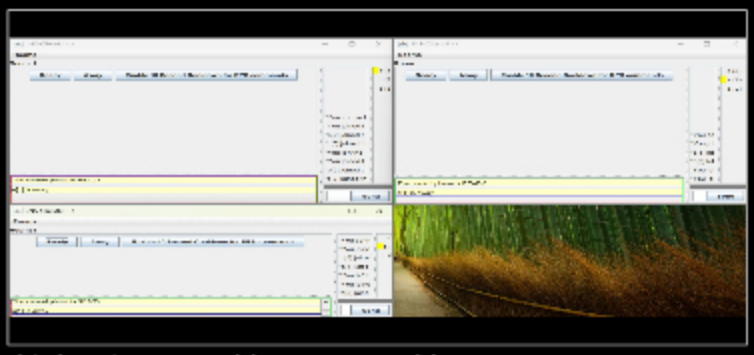
100%

Group: Project Specific
Task #2: Implement a cooldown on an option (i.e., same option can't be picked twice in a row by the same player)
Sub Task #1: Show the pre-game screen (can be ready panel) that toggles this feature

Task Screenshots

Gallery Style: 2 Columns

4 2 1



third option to enable 10 sec cooldown games

Caption(s) (required) ✓

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

Sub-Task

100%

Group: Project Specific

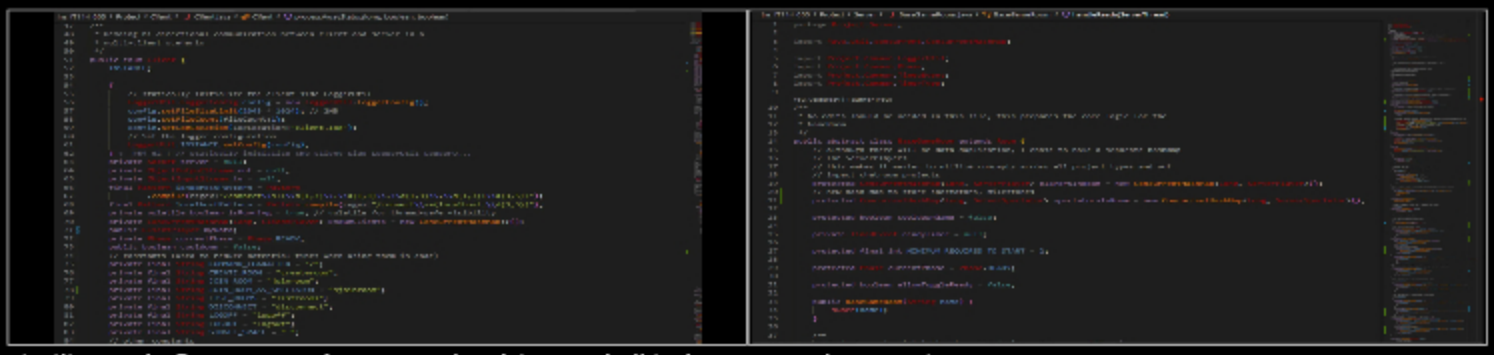
Task #2: Implement a cooldown on an option (i.e., same option can't be picked twice in a row by the same player)

Sub Task #2: Show the code related to handling this on the server-side and client-side

Task Screenshots

Gallery Style: 2 Columns

4 2 1



similiar code flow as send away payload, instead all it does see other caption is update cooldown boolean value for both the clients and gameroom

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works and explain how you handle the "prevent twice in a row" logic

Response:

code flow follows the same logic as away functionality, but now instead it is just send this boolean value if the game is a cooldown game or not, and the clients ui handles it, so if it is a 10 sec game, in the client ui it will create a 10 sec timer after clicking a button, send a message that there is a ten second cooldown, and then send the command.

Sub-Task

100%

Group: Project Specific

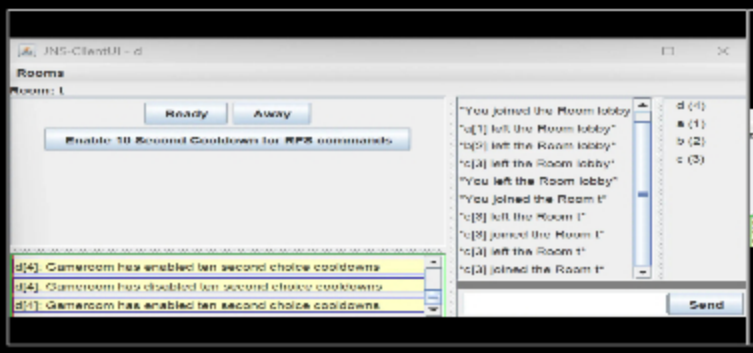
Task #2: Implement a cooldown on an option (i.e., same option can't be picked twice in a row by

the same player)
Sub Task #3: Demonstrate from the UI this feature in effect including any potential related Game Event messages

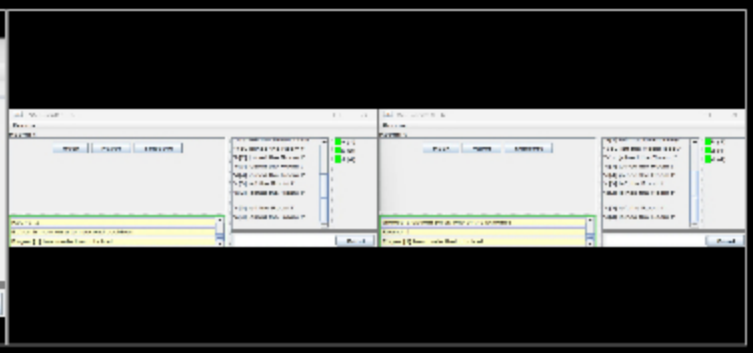
Task Screenshots

Gallery Style: 2 Columns

4 2 1



player using button



only player who clicked rps command will see additional msg that they are now on a ten second cooldown

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

Task Response Prompt

Explain in concise steps how this logically works
Response:

most game logic actually occurs in the user ui, in which when a button is clicked, a timer is started on the clients side before they can send other button command

End of Task 2

End of Group: Project Specific
Task Status: 1/2

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/jnsnjit/jns-IT114-003/pull/17

URL
https://github.com/jnsnjit/jns-IT114-003/pull/17

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:

well i think the biggest problem with me for this project was time management and keeping track of changes. I now really understand the importance of good commenting and follow procedures for implementing similiar things. by the end i was getting overwhelmed by the amount of code that acculumalated, i think if i was to do this class again, I could do it a lot better.

End of Task 2

Task

100%

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

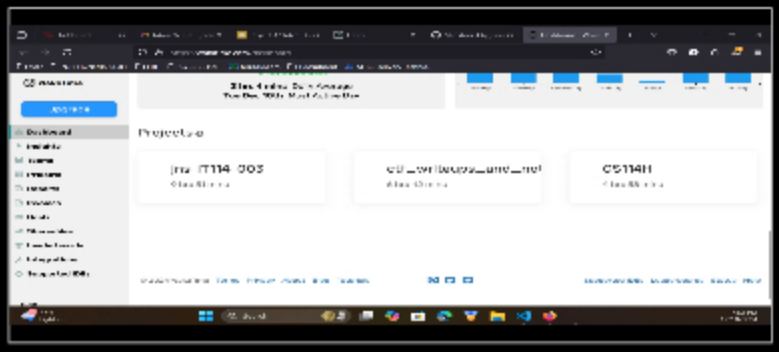
COLLAPSE

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



yea 21 hours of vscode in one week, finals week really beating me up lol

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment