# Submission Worksheet

Course: IT114-002-S2025
Assignment: IT114 Milestone 2 - RPS
Student: Trinity C. (tdc28)
Status: Submitted | Worksheet Progress: 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-002-S2025/it114-milestone-2-rps/grading/tdc28

## Instructions

1. Refer to Milestone2 of Rock Paper Scissors
    1. Complete the features
2. Ensure all code snippets include your ucid, date, and a brief description of what the code does
3. Switch to the `Milestone2` branch
    1. `git checkout Milestone2`
    2. `git pull origin Milestone2`
4. Fill out the below worksheet as you test/demo with 3+ clients in the same session
5. Once finished, click "Submit and Export"
6. Locally add the generated PDF to a folder of your choosing inside your repository folder and move it to Github
    1. `git add .`
    2. `git commit -m "adding PDF"
    3. `git push origin Milestone2`
    4. On Github merge the pull request from `Milestone2` to `main`
7. Upload the same PDF to Canvas
8. Sync Local
    1. `git checkout main`
    2. `git pull origin main`

- Complete each section and task sequentially.
- Review the details and validation criteria for each task.
- Ensure subtasks are completed before the parent task.

---

100%
## Section #1: ( 1 pt.) Payloads

100%
### Task #1 ( 1 pt.) - Show Payload classes and subclasses

> **Combo Task:**

## Combo Task:

**Weight:** *100%*
**Objective:** *Show Payload classes and subclasses*

**Details:**

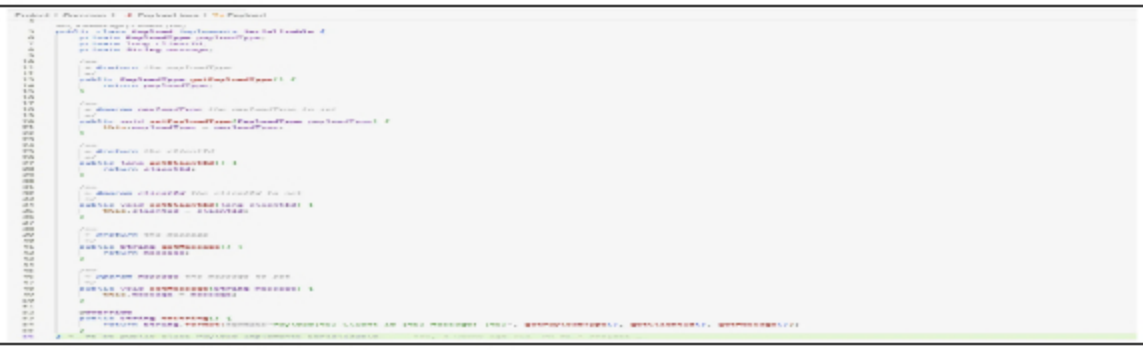
- Reqs from the document
  - Provided Payload for applicable items that only need client id, message, and type
  - PointsPayload for syncing points of players
  - Each payload will be presented by debug output (i.e, properly override the toString() method like the lesson examples)

## Image Prompt

**Weight:** *50%*

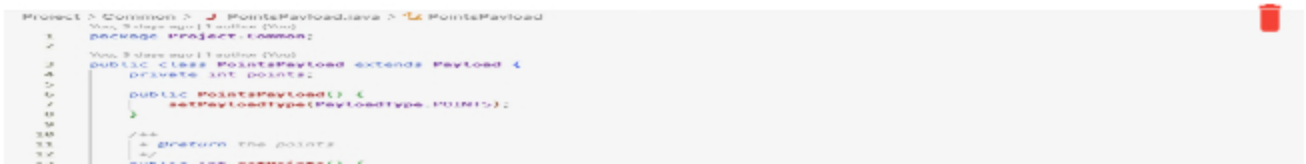
**Details:**

- Show the code related to your payloads (Payload, PointsPayload, and any new ones added)
- Each payload should have an overriden toString() method showing its internal data



public class Payload

payloadtype.java



pointspayload.java

timerpayload

roomresultpayload.java

readypayload.java

# Section #2: ( 4 pts.) Lifecycle Events

# Task #1 ( 0.80 pts.) - GameRoom Client Add/Remove

## Combo Task:

**Weight:** *20%*
**Objective:** *GameRoom Client Add/Remove*

### Image Prompt

**Weight:** *50%*

**Details:**

- Show the `onClientAdded()` code
- Show the `onClientRemoved()` code

OnClientAdded & OnClientRemoved

Saved: 5/9/2025 9:39:22 AM

### Text Prompt

**Weight:** *50%*

**Details:**

- Briefly note the actions that happen in `onClientAdded()` (app data should at least be synchronized to the joining user)
- Briefly note the actions that happen in `onClientRemoved()` (at least should handle logic for an empty session)

Your Response:

OnClientAdded is called when a new client joins. It has syncCurrentPhase, syncReadyStatus, syncTurnStatus, and syncPlayerPoints, which makes sure the player is synced with the game

OnClientRemoved is called when the client leaves. It has clientsInRoom.isEmpty, resetReadyTimer, resetTurnTimer, and resetRoundTimer, it logs the removal of the player and whoever remains.

Saved: 5/9/2025 9:39:22 AM

100%

# Task #2 ( 0.80 pts.) - GameRoom Session Start

## Combo Task:

**Weight:** *20%*
**Objective:** *GameRoom Session Start*

**Details:**

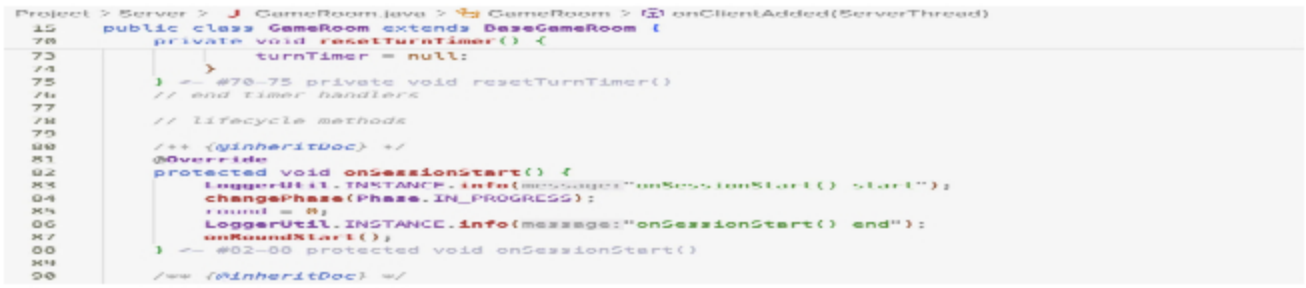
- Reqs from document
  - First round is triggered
- Reset/set initial state

## Image Prompt

**Weight:** *50%*

**Details:**

- Show the snippet of `onSessionStart()`




onSessionStart

Saved: 5/9/2025 9:48:41 AM

## Text Prompt

**Weight:** *50%*

**Details:**

- Briefly explain the logic that occurs here (i.e., setting up initial session state for your project) and next lifecycle trigger

Your Response:

onSessionStart sets up the game, it switches the phase to IN_PROGRESS, resets the round counter to 0. It also calls on onRoundStart() which begins the next round.

Saved: 5/9/2025 9:48:41 AM

100%

# Task #3 ( 0.80 pts.) - GameRoom Round Start

## Combo Task:

**Weight:** *20%*
**Objective:** *GameRoom Round Start*

**Details:**

- Reqs from Document
  - Initialize remaining Players' choices to null (not set)
  - Set Phase to "choosing"
  - GameRoom round timer begins

## Image Prompt

**Weight:** *50%*

**Details:**

- Show the snippet of `onRoundStart()`

onRoundStart

Saved: 5/9/2025 9:51:41 AM

## Text Prompt

**Weight:** *50%*

**Details:**

- Briefly explain the logic that occurs here (i.e., setting up the round for your project)

Your Response:

onRoundStart, logs the start of a new round and resets the player statuses & timers. It also broadcasts a message to all the players telling them its a start of a new round, and uses the round timer.

Saved: 5/9/2025 9:51:41 AM

100%

# Task #4 ( 0.80 pts.) - GameRoom Round End

## Combo Task:

**Weight:** *20%*
**Objective:** *GameRoom Round End*

**Details:**

- Reqs from Document
  - **Condition 1:** Round ends when round timer expires
  - **Condition 2:** Round ends when all active Players have made a choice
  - All Players who are not eliminated and haven't made a choice will be marked as eliminated
  - Process Battles:
    - Round-robin battles of eligible Players (i.e., Player 1 vs Player 2 vs Player 3 vs Player 1)
      - Determine if a Player loses if they lose the "attack" or if they lose the "defend" (since Player has two battles each round)
        - Give a point to the winning Player
        - Points will be stored on the Player/User object
        - Sync the points value of the Player to all Clients
    - Relay a message stating the Players that competed, their choices, and the

result of the battle
    - Losers get marked as eliminated (Eliminated Players stay as spectators but are skipped for choices and for win checks)
- Count the number of non-eliminated Players
    - If one, this is your winner (onSessionEnd())
    - If zero, it was a tie (onSessionEnd())
    - If more than one, do another round (onRoundStart())

## Image Prompt

**Weight:** *50%*

**Details:**

- Show the snippet of `onRoundEnd()`

onRoundEnd

ProcessBattles that is called in onRoundEnd

Saved: 5/9/2025 12:04:03 PM

## Text Prompt

**Weight:** *50%*

**Details:**

- Briefly explain the logic that occurs here (i.e., cleanup, end checks, and next lifecycle events)

Your Response:

onRoundEnd is called on when the round ends, it resets the timer and broadcasts the message that the round has ended. It also calls on processBattles, which updates scores at the end of a round/session, gives the winner a point & the other person none. it also updates the player on who won or not.

Saved: 5/9/2025 12:04:03 PM

100%

# Task #5 ( 0.80 pts.) - GameRoom Session End

## Combo Task:

**Weight:** *20%*
**Objective:** *GameRoom Session End*

**Details:**

- Reqs from Document
  - **Condition 1:** Session ends when one Player remains (they win)
  - **Condition 2:** Session ends when no Players remain (this is a tie)
  - Send the final scoreboard to all clients sorted by highest points to lowest (include a game over message)
  - Reset the player data for each client server-side and client-side (do not disconnect them or move them to the lobby)
  - A new ready check will be required to start a new session

## Image Prompt

**Weight:** *50%*

**Details:**

- Show the snippet of `onSessionEnd()`

onSessionEnd

endGame that is called in onSessionEnd

Saved: 5/9/2025 12:05:53 PM

## Text Prompt

**Weight:** *50%*

**Details:**

- Briefly explain the logic that occurs here (i.e., cleanup/reset, next lifecycle events)

Your Response:

onSessionEnd logs that the session is ended, clears & resets the ready statuses + the turn statuses of the players. It also resets the game phase and changes it back to READY. It calls on endGame, endGame changes the game phase to ENDED, creates a scoreboard to broadcast the scores and notifies the players that the game is ending and asks if they want to play again.

Saved: 5/9/2025 12:05:53 PM

# Section #3: ( 4 pts.) Gameroom User Action And State

## Task #1 ( 2 pts.) - Choice Logic

## Combo Task:

**Weight:** *50%*
**Objective:** *Choice Logic*

**Details:**

- Reqs from document
  - Command: `/pick <[r,p,s]>` (user picks one)
    - GameRoom will check if it's a valid option
    - GameRoom will record the choice for the respective Player
    - A message will be relayed saying that "X picked their choice"
    - If all Players have a choice the round ends

## Image Prompt

**Weight:** *50%*

**Details:**

- Show the code snippets of the following, and clearly caption each screenshot
- Show the Client processing of this command (process client command)
- Show the ServerThread processing of this command (process method)
- Show the GameRoom handling of this command (handle method)
- Show the sending/syncing of the results of this command to users (send/sync method)
- Show the ServerThread receiving this data (send method)
- Show the Client receiving this data (process method)

handlePick

serverThread Process Payload case Pick

processPick (client)

sendPick (in client)


syncing/sending result

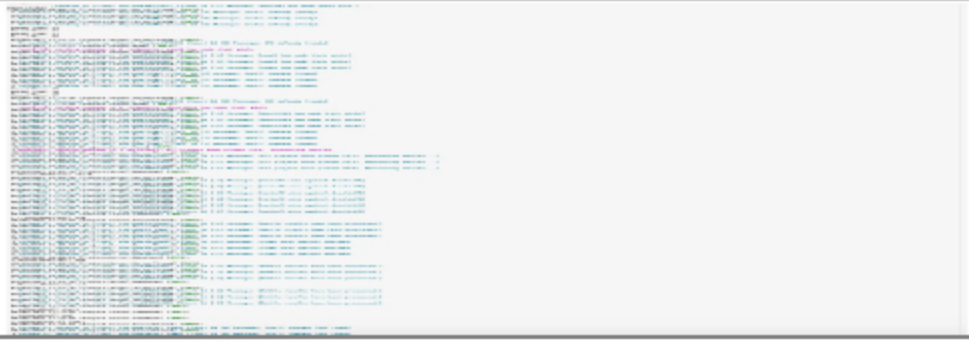
## Task #2 ( 2 pts.) - Game Cycle Demo

### Image Prompt

**Weight:** *50%*
**Objective:** *Game Cycle Demo*

**Details:**

- Show examples from the terminal of a full session demonstrating each command and progress output
- This includes battle outcomes, scores and scoreboards, etc
- Ensure at least 3 Clients and the Server are shown
- Clearly caption screenshots



server demonstration screenshot 1

# Section #4: ( 1 pt.) Misc

## Task #1 ( 0.33 pts.) - Github Details

### Combo Task:

**Weight:** *33.33%*
**Objective:** *Github Details*

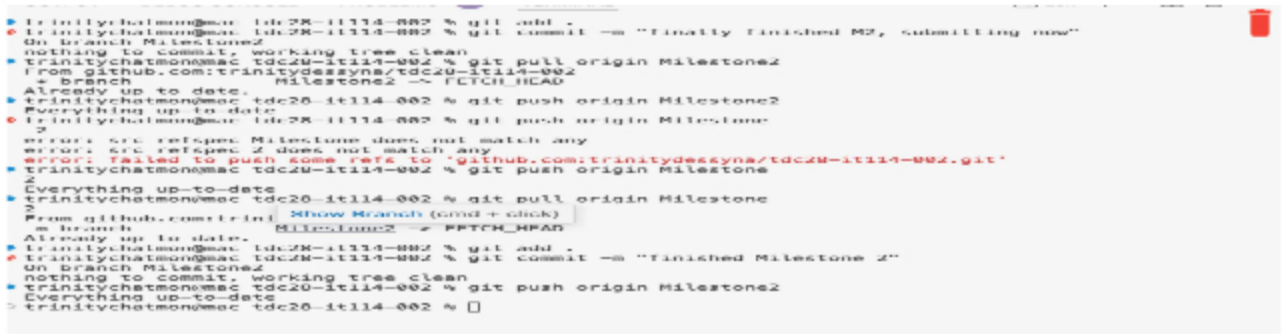
#### Image Prompt

**Weight:** *60%*

**Details:**

From the Commits tab of the Pull Request screenshot the commit history

commits tab (there should be more commits than this but for some reason the most recent ones arent showing up)



screenshot of me committing on terminal, but it wont show up on github

Saved: 5/9/2025 1:22:27 PM

## Url Prompt

**Weight:** *40%*

**Details:**

Include the link to the Pull Request (should end in `/pull/#`)

URL #1

## Task #2 ( 0.33 pts.) - WakaTime - Activity

## Image Prompt

**Weight:** *33.33%*

**Objective:** *WakaTime - Activity*

**Details:**

- Visit the WakaTime.com Dashboard
- Click `Projects` and find your repository
- Capture the overall time at the top that includes the repository name
- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary


screenshot of total time

## 100% Task #3 ( 0.33 pts.) - Reflection

**Weight:** *33.33%*
**Objective:** *Reflection*

## Sub-Tasks:

### 100% Task #1 ( 0.33 pts.) - What did you learn?

#### Text Prompt

**Weight:** *33.33%*
**Objective:** *What did you learn?*

**Details:**
Briefly answer the question (at least a few decent sentences)

Your Response:

I learned how to code more things than I knew before. I learned about different methods and how to code them. I also learned how all of the files connected or "Extended" one another.

Saved: 5/9/2025 1:27:30 PM

## Task #2 ( 0.33 pts.) - What was the easiest part of the assignr

### Text Prompt

**Weight:** *33.33%*
**Objective:** *What was the easiest part of the assignment?*

**Details:**
Briefly answer the question (at least a few decent sentences)

Your Response:

The easiest part of the assignment was testing out to see if the code worked. using ./build.sh and./sh, it was fun to see how it all came together.

Saved: 5/9/2025 1:28:26 PM

## Task #3 ( 0.33 pts.) - What was the hardest part of the assign

### Text Prompt

**Weight:** *33.33%*
**Objective:** *What was the hardest part of the assignment?*

**Details:**
Briefly answer the question (at least a few decent sentences)

Your Response:

The hardest part of the assignments was figuring out which part of each file i needed to add my own code to, and actually coding it

Saved: 5/9/2025 1:29:05 PM