

# Submission Worksheet

## Submission Data

**Course:** IT114-450-M2025

**Assignment:** IT114 Milestone 2 - RPS

**Student:** Graham B. (gb373)

**Status:** Submitted | **Worksheet Progress:** 100%

**Potential Grade:** 10.00/10.00 (100.00%)

**Received Grade:** 0.00/10.00 (0.00%)

**Started:** 7/9/2025 3:56:31 PM

**Updated:** 7/10/2025 7:12:03 PM

**Grading Link:** <https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-2-rps/grading/gb373>

**View Link:** <https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-2-rps/view/gb373>

## 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`

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

Progress: 100%

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

Progress: 100%

#### Details:

- Reqs from the document
  - Provided Payload for applicable items that only need client id, message, and type



### Details:

- Briefly explain the purpose of each payload shown in the screenshots and their properties

**Your Response:**

The first one is the first payload, which is the base for all payloads that are exchanged between the client and the server. It contains encapsulated properties, such as the payload type message and client ID. The `toString()` helps make it readable to summarize the property that we need. The second payload is for the points, also known as `PointsPayload`, which is used for transferring point values to the players who are in the game. The last screenshot is for the newly added payload types for points and the points updated.



Saved: 7/9/2025 4:27:03 PM

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

Progress: 100%

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

Progress: 100%

## Part 1:

Progress: 100%

### Details:

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

[illegible]

### onClientAdded and on ClientRemoved



Saved: 7/9/2025 4:55:40 PM

## ⇒ Part 2:

Progress: 100%

### 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:

For On Client Added, it first syncs with the current game phase for any new clients that have joined. Then it syncs the client's ready status to know who is ready. Then it syncs the turn status for the progress of the player's turns. Finally, it syncs the current points of all players to the new clients to keep the score up to date and correct. For On Client Removed, it first removes the client that is leaving from the turn order and also the player list. Then, it will see that if the session is empty, the timers will reset, and the session ends. Finally, if the player was the one in turn, then it will go to the next player in turn.



Saved: 7/9/2025 4:55:40 PM

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

Progress: 100%

#### Details:

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

#### 📄 Part 1:

Progress: 100%

#### Details:

- Show the snippet of `onSessionStart()`

```
// UUID: 6b575
// DATE: 7/7/2025
// Summary: Handles the start of a session, initializing the game state and notifying players.
// @author: [Your Name]
protected void onSessionStart() {
    LoggerUtil.info(TAG, "onSessionStart() Start");
    changePhase(Phase.IN_PROGRESS);
    currentTurnClientId = Constants.DEFAULT_CLIENT_ID;
    setTurnOrder();
    round = 0;
    LoggerUtil.info(TAG, "onSessionStart() End");
    onRoundStart();
}
```

Code for onSessionStart()



Saved: 7/9/2025 5:07:05 PM

## Part 2:

Progress: 100%

### Details:

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

### Your Response:

For onSessionStart, it sets up the initial session state for a new game about to be played. It resets the game before and resets and sets the initial state to the correct one. The phase will be changed to in progress as the game has started. Then the turn for the clientid is reset to get rid of any data from a previous game. It also generates a turn order and resets the counter, and begins the round by triggering onRoundStart().



Saved: 7/9/2025 5:07:05 PM

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

Progress: 100%

### Details:

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

### Part 1:

Progress: 100%

### Details:

- Show the snippet of `onRoundStart()`

```
// UUID: 6b578
// Date: 07/09/2025
// Summary: Handles the start of a round, resetting timers and notifying players.
// This method is called at the start of each round.
@Override
protected void onRoundStart() {
    LoggerUtil.INSTANCE.info(message:"onRoundStart() start");

    resetRoundTimer();
    resetRoundStatus();
    resetPlayerChoices();
    changePhase(Phase.Choosing);
    startRoundTimer();

    round++;
    delaySenderNull, String.Format(Format:"Round {0} has started", round);
    LoggerUtil.INSTANCE.info(message:"onRoundStart() end");

    onTurnStart();
}
// 0119 128 protected void onRoundStart()
```

Code for onRoundStart()

## ≡ Part 2:

Progress: 100%

### Details:

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

### Your Response:

The logic that occurs is in `onRoundStart`, which sets up a new round for the clients to play. It first resets any timers from the previous round and any choices made by setting, which is set to null by code in the `gameroom.java`. It also changes the Phase to Choosing and starts the game round timer. A round counter adds one to itself, and a message is announced saying the round has started, and it also triggers the `onTurnStart` event to start a client's turn.

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

Progress: 100%

### 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 each 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()`)

## ≡ Part 1:

Progress: 100%

**Details:**

- Show the snippet of `onRoundEnd()`

- Details:**
- Show the snippet of `onRoundEnd()`

[illegible]

Code for onRoundEnd() part 1

Code for onRoundEnd() part 2



 Saved: 7/10/2025 5:54:29 PM

⇒ Part 2:

⇒ Part 2:

**Details:**

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

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

**Your Response:**

For `onRoundEnd()`, cleanup happens with the round timer being reset and players who didn't choose that round are marked as eliminated. How the battle works is that the player chooses either r, p, or s to attack another player with. If you win an attack, you get a point; if you lose, you get nothing and are marked as eliminated. If a defender loses, then they are marked as eliminated and can't make a choice for the next round. All scores are synced to each client to keep them up to date. At the end, the logic will count how many players are active. If one player is remaining, that player is the winner. If no players are remaining, it is considered a draw, and the session ends for both outcomes. If there is more than one player, then there is a new round starts and the game continues.



## Progress: 100%

- 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

## Progress: 100%

- Show the snippet of `onSessionEnd()`


### Code for onSessionEnd()

## Progress: 100%

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

For `onSessionEnd()`, it will first figure out what the outcome was whether it is a tie or a win. It will then relay the correct message for that certain outcome. After, it resets player data like points,



 Saved: 7/10/2025 1:41:15 PM

## Progress: 100%

## Progress: 100%

- 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

## Progress: 100%

- 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)

### Client processing the command



Code for client receiving the data by processing the turn



Saved: 7/10/2025 6:29:11 PM

## Part 2:

Progress: 100%

### Details:

- Briefly explain/list in order the whole flow of this command being handled from the client-side to the server-side and back

### Your Response:

For the /pick command, for the client they handle it first by seeing what they put in wheather it being r, p, or s. In the processClientCommand() it checks for if a user puts in /pick. Then, it makes sure the input is either r, p, or s. Then it sends the pick choice to the tuen payload and to the server. The client sends the payload to the server with sendPickChoice(). The server thread then processes it with how it processes all payloads, which it sees its a turn type, which is handled by the game room. The gameroom then handles the command by seeing that they have taken their turn and going to the next player. If all have picked, then it goes to the scoring and round end. Then the server thread gets the turn status, which they send to the client, where they handle it by processing the turn and processing the points, and updating the score for all clients.



Saved: 7/10/2025 6:29:11 PM

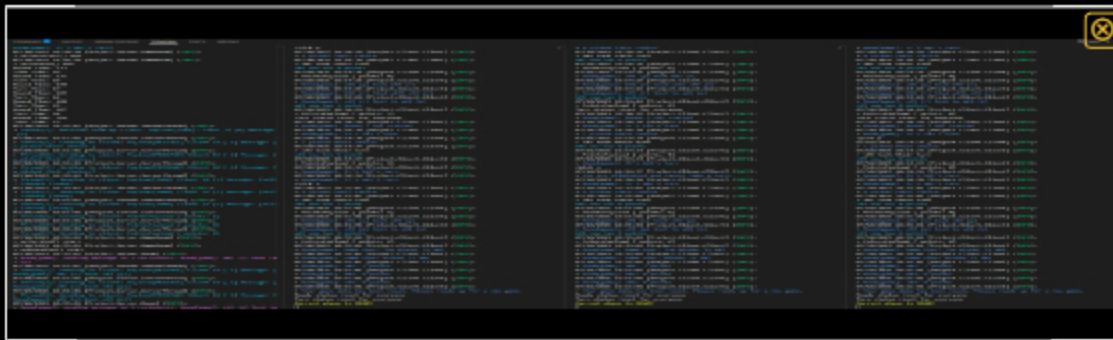
## Task #2 ( 2 pts.) - Game Cycle Demo

Progress: 100%

### 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

Example 1 of output Tie game

A screenshot of a terminal window displaying a game log. The log shows a series of moves and scores for two players, resulting in a tie game. The text is color-coded, with green for moves and red for scores. The terminal has a dark background and a yellow close button in the top right corner.

Example 2 of output One Player Remaining

A screenshot of a terminal window displaying a game log. The log shows a series of moves and scores for two players, resulting in one player remaining. The text is color-coded, with green for moves and red for scores. The terminal has a dark background and a yellow close button in the top right corner.

Example 3 of output All choose same so game continues

 Saved: 7/10/2025 6:58:46 PM

## Section #4: ( 1 pt.) Misc

Progress: 100%

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

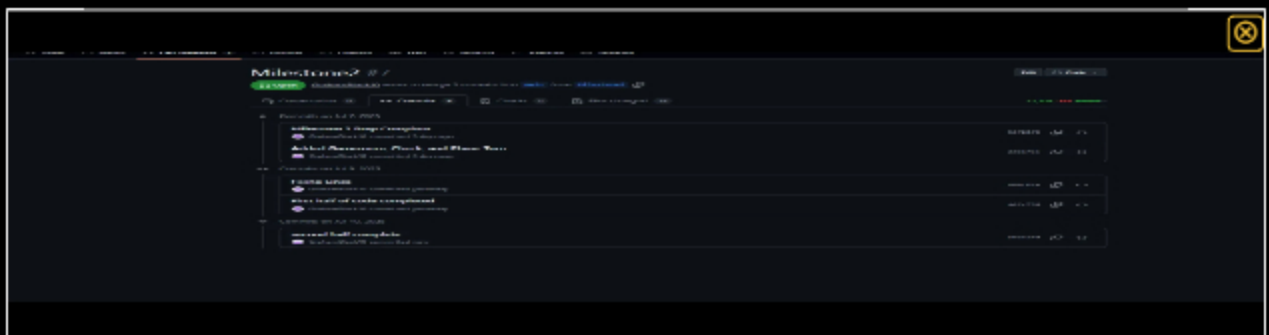
Progress: 100%

#### Part 1:

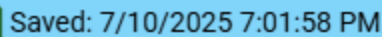
Progress: 100%

##### Details:

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

A screenshot of a GitHub Pull Request page. The page shows the commit history of the pull request, with a table of commits. The table has columns for commit message, author, and date. The commits are listed in descending order of date. The page has a dark background and a yellow close button in the top right corner.

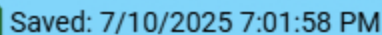
Pull History from commit from Milestone 2 branch



Progress: 100%

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

<https://github.com/GrahamBlack1>



Progress: 100%

[illegible]



Saved: 7/10/2025 7:03:33 PM

## ≡ Task #3 ( 0.33 pts.) - Reflection

Progress: 100%

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

Progress: 100%

**Details:**

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

**Your Response:**

I learn a lot more about clients and server. I learn about how to add commands so clients can use them for different tasks. I added the pick command for the rps and had to figure out the logic that went behind it so I did a lot of research on what I would need to change and how to get it so that clients could play rps in a gameroom. I also learn about creating a new payload that would be in charge of keeping track of the points for the players, and I had to incorporate it so that the client and server use this payload correctly.



Saved: 7/10/2025 7:05:38 PM

### ⇒ Task #2 ( 0.33 pts.) - What was the easiest part of the assignment?

Progress: 100%

**Details:**

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

**Your Response:**

The easiest part of the assignment was understanding the rounds start and session start and end. With some of these not needing to much change from the tutorial the professor had. I was able to understand it a lot from the videos and only had to add some new changes that would help the game run smoothly. during the session start and end, it was easy to understand that it would need to make a turn order and also start the round while the end would clean everything up for the next session. The start of the round would be setting the timers and having the phase change so the players can make a pick.





Saved: 7/10/2025 7:08:41 PM

## ⇒ Task #3 ( 0.33 pts.) - What was the hardest part of the assignment?

Progress: 100%

### Details:

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

### Your Response:

The hardest part was at the end for the round ending and the /pick command. For the round end, I had difficulty with the logic understanding what methods I needed to add for the round to end right and to keep the right data and show the clients score and other things. I also found an issue where players who were eliminated were still able to pick, but I now made it so that if they try to, they get a message saying they are eliminated, and when their turn ends, it goes to the next player. As for the command, it was a lot of logic that went into making it work with the whole process going through the client to the server and back, which made it difficult to make so many methods for processing data and sending it, and receiving it.



Saved: 7/10/2025 7:12:03 PM