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: 7/22/2025 5:41:49 PM
Updated: 7/22/2025 5:41:49 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

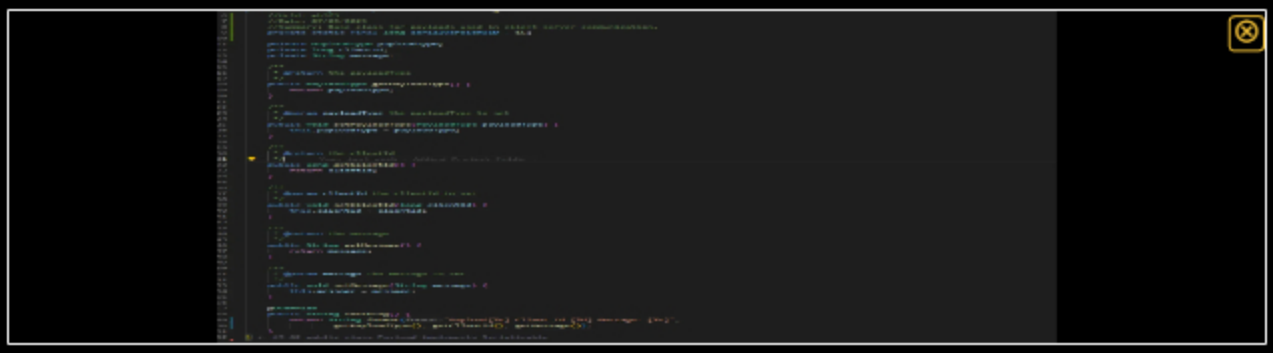
- PointsPayload for syncing points of players
- Each payload will be presented by debug output (i.e, properly override the toString() method like the lesson examples)

Part 1:

Progress: 100%

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



Code for updated Payload.java

Code for PointsPayload.java

Added new payload types

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

Part 2:

Progress: 100%

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 bae 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

```
// UCID: ph373
// Date: 07/05/2025
// Summary: Handles the addition of a new client to the GameRoom.
// onClientRemoved is called when a client disconnects or leaves the room.
// onClientAdded is called when a new client joins the room.
@Override
protected void onClientAdded(ServerThread sp) {
    syncCurrentPhase(sp);
    syncReadyStatus(sp);
    if (currentPhase != Phase.READY) {
        syncTurnStatus(sp);
        syncPlayerPoints(sp);
    }
} <-- #44-41 protected void onClientAdded(ServerThread sp)

@Override
protected void onClientRemoved(ServerThread sp) {
    clientsInRoom.remove(sp.getClientId());
    if (clientsInRoom.isEmpty()) {
        resetRoundTimer();
        onSessionEnd();
    }
} <-- #44-50 protected void onClientRemoved(ServerThread sp)
```

onClientAdded and on ClientRemoved

Saved: 7/22/2025 4:46:31 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. 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 room and also from the player list. Then, it will see that if the room is empty, the timers will reset, and the session ends.

Saved: 7/22/2025 4:46:31 PM

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

Progress: 100%

Details:

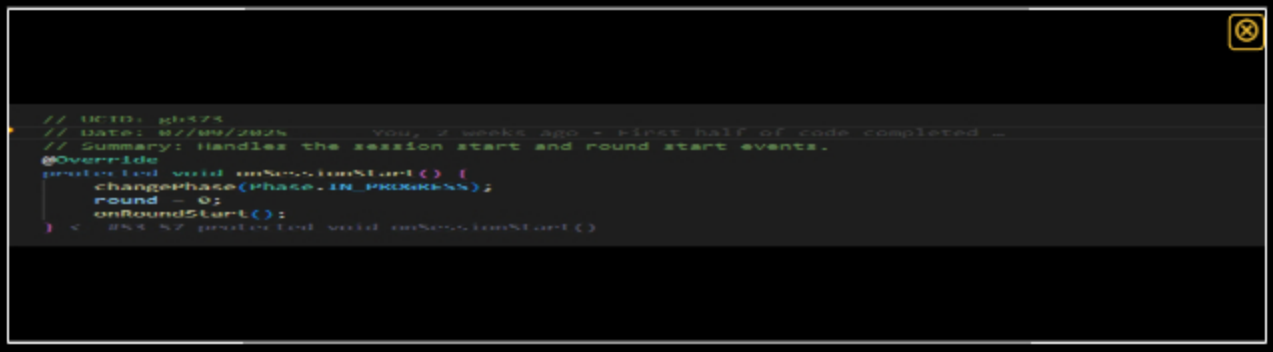
- Reqs from document
  - First round is triggered
- Reset/set initial state

Part 1:

Progress: 100%

Details:

- Show the snippet of `onSessionStart()`



Code for onSessionStart()

Saved: 7/22/2025 4:47:19 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 it sets the rounds to 0 and also starts the round.

Saved: 7/22/2025 4:47:19 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()`

Code for onRoundStart()

Saved: 7/22/2025 4:51:58 PM

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 round tiemrs and Player choices and changes the phase to in progress. Then, it adds 1 to the round and sends a message to all users that the round has started, and the user must pick r, p, or s. It will then make the round timer and start it for 30 seconds, and tick down as the round progresses. All of the payload will be sent to all players.

Saved: 7/22/2025 4:51:58 PM

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()`

Code for onRoundEnd() part 1

Code for onRoundEnd() part 2

Saved: 7/22/2025 4:59:55 PM

Part 2:

Progress: 100%

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. It resets the round timer as well and begins to start the evaluation phase. To start, a list of clients is collected, and any that are eliminated are set as eliminated and can't play for the next round. The battle works as everyone picks either r,p, or s at the same time. Then a round robin kind of order occurs where each user will attack another user while they defend. If a player loses on defence, they are marked as eliminated, and the attacker gets a point. You also get a point if you succeed in defending. If one survivor is remaining, then that player wins, and if no players are remaining, then it is considered a tie.

Saved: 7/22/2025 4:59:55 PM

Task #5 (0.80 pts) - GameRoom Session End

Task #3 (3.00 pts.) - GameRoom Session End

Progress: 100%

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

Part 1:

Progress: 100%

Details:

- Show the snippet of `onSessionEnd()`

Code for onSessionEnd()

Saved: 7/22/2025 5:12:18 PM

Part 2:

Progress: 100%

Details:

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

Your Response:

For onSessionEnd(), it first prints the final score and the outcome of the round. It does this by also sorting them by their points in descending order. After it cleans everything by setting the points to 0, setting the choice to null, setting tookturn to false, setting eliminated status to false, setting ready status to false, and it sends a message saying the game has ended and to /ready if you want to play again. Finally, it changes the phase to READY.

Saved: 7/22/2025 5:12:18 PM

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

Progress: 100%

Task #1 (2 pts.) - Choice Logic

Progress: 100%

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

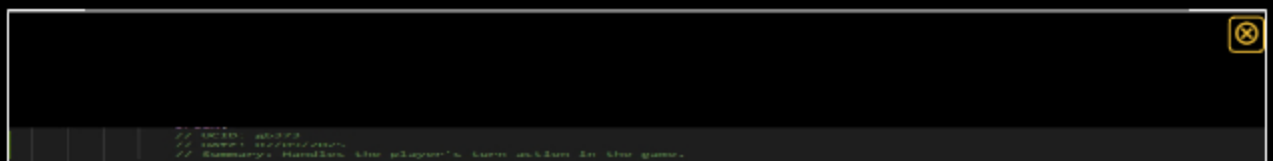
Part 1:

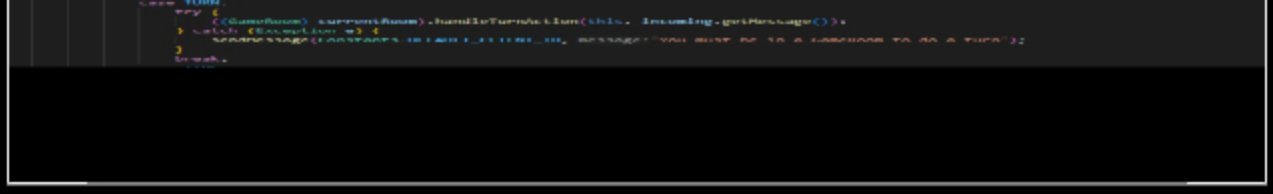
Progress: 100%

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)

Client processing the command





Code for serverthread processing a turn

code in gameroom for handling the command

code for the send/sync of the results of the command

Code for the send method in server thread

Code for client receving the data by processing the turn

Saved: 7/22/2025 5:32:42 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, the client handles it first by seeing what they put in, whether it is r, p, or s. In the processClientCommand(), it checks if a user puts in /pick. Then, it makes sure the input is either r, p, or s. The server thread will then receive the payload called TURN, which is processed and forwarded to the game room. The game room will then handle the turn actions and broadcast what a player picked. This will all happen, and by the end, the data will be processed and sent back to the client from the server side.

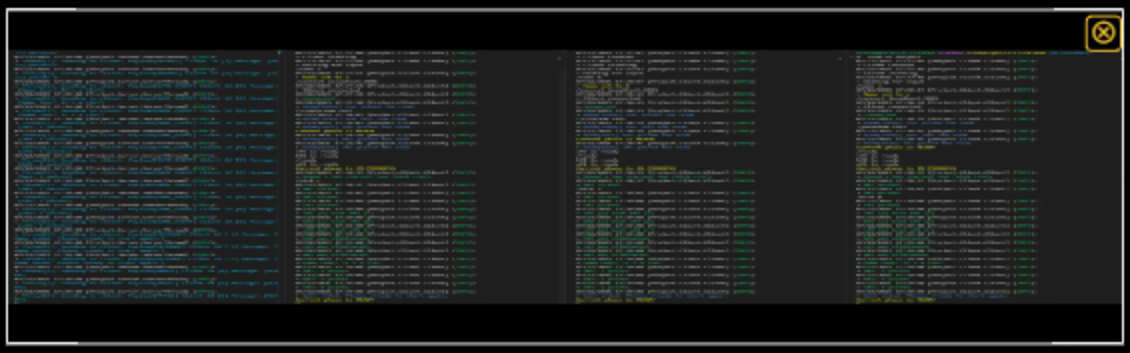
Saved: 7/22/2025 5:32:42 PM

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

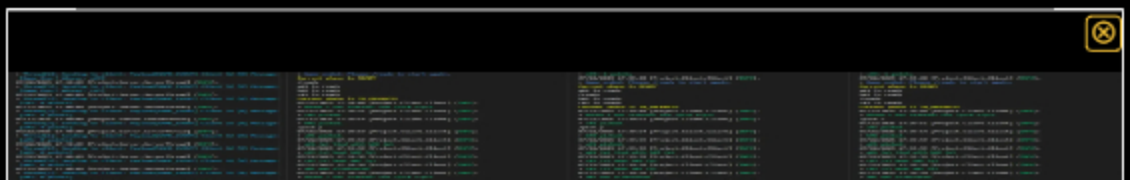
Progress: 100%

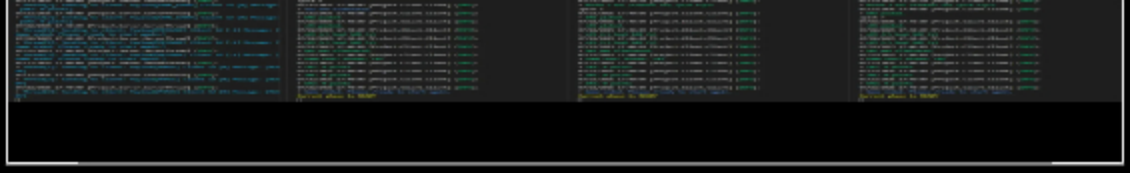
Details:

- 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

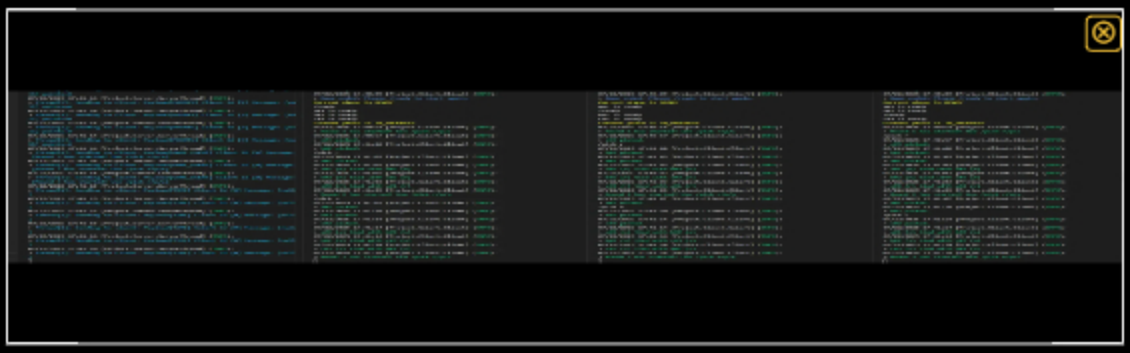


Example 1 of output Tie game





Example 2 of output One Player Remaining



Example 3 of output All choose same so game continues

Saved: 7/22/2025 5:41:49 PM

Section #4: (1 pt.) Misc

Progress: 100%

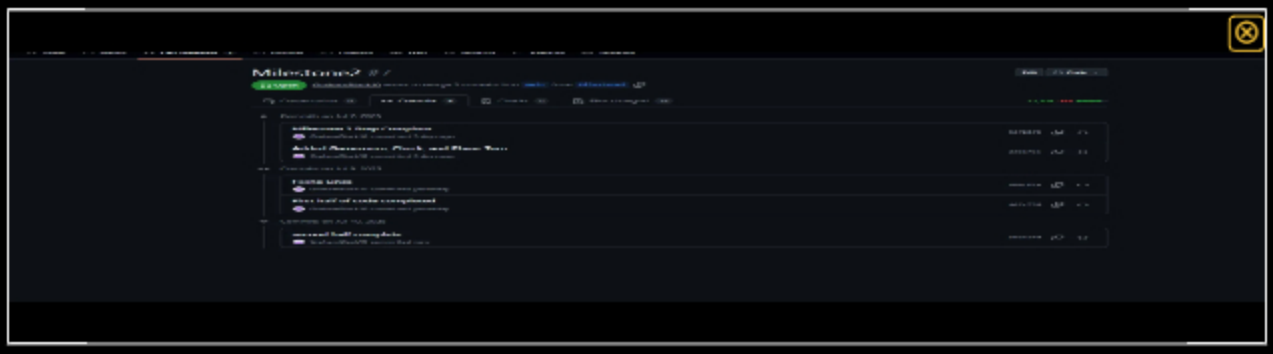
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

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



Pull History from commit from Milestone 2 branch

Saved: 7/10/2025 7:01:58 PM

Part 2:

Progress: 100%

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

URL #1
https://github.com/GrahamBlack10/gb373-IT114-450/pull/7

URL
https://github.com/GrahamBlack1

Saved: 7/10/2025 7:01:58 PM

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

Progress: 100%

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

Overall Time

Individual Time

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 servers. 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 game room. 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: 7/22/2025 5:33:38 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 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 all players choose r,p, or s and have them start it at the same time 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 players can make a pick.

Saved: 7/22/2025 5:35:17 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 of the round, the 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 client's 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 would get a message and not be able to pick, and not be counted in point giving. 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: 7/22/2025 5:36:19 PM