# Submission Worksheet

## Submission Data

**Course:** IT114-007-F2025
**Assignment:** IT114 Milestone 2 - RPS
**Student:** Lukas P. (lap5)
**Status:** Submitted | **Worksheet Progress:** 100%
**Potential Grade:** 10.00/10.00 (100.00%)
**Received Grade:** 0.00/10.00 (0.00%)
**Started:** 11/24/2025 10:44:23 PM
**Updated:** 11/24/2025 11:56:57 PM
**Grading Link:** https://learn.ethereallab.app/assignment/v3/IT114-007-F2025/it114-milestone-2-rps/grading/lap5
**View Link:** https://learn.ethereallab.app/assignment/v3/IT114-007-F2025/it114-milestone-2-rps/view/lap5

## 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

```
public class Payload implements Serializable {
    private PayloadType payloadType;
    private long clientId;
    private String message;

    // Getters and Setters...

    @Override
    public String toString() {
        return String.format("Payload[%s] Client Id [%s] Message: [%s]",
getPayloadType(), getClientId(), getMessage());
    }
}
```

payload

```
public class PointsPayload extends Payload {
    private long clientId;
    private int points;

    public PointsPayload() {
        setPayloadType(PayloadType.SYNC_PAYLOAD);
    }

    // Getters and Setters...

    @Override
    public String toString() {
        return String.format("PointsPayload[clientId=%d, points=%d]",
clientId, points);
    }
}
```

payload

Saved: 11/24/2025 10:52:44 PM

## Part 2:

Progress: 100%

**Details:**

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

Your Response:

The base class for all communication. It contains the payloadType (to identify the action), clientId (sender/target), and a generic message. The toString() method is overridden to provide clear debug output.

A specialized payload for syncing player scores. It extends Payload and adds a points field. It sets

its type to SYNC_PAYLOAD automatically.

Saved: 11/24/2025 10:52:44 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

```
protected synchronized void addClient(ServerThread client) {
        if (!isRunning) {
            return;
        }
        if (clientsInRoom.containsKey(client.getClientId())) {
            info("Attempting to add a client that already exists in the
room");
            return;
        }
        clientsInRoom.put(client.getClientId(), client);
        client.setCurrentRoom(this);
        client.sendResetUserList();
        syncExistingClients(client); // Synchronize app data to joining
user
        // notify clients of someone joining
        joinStatusRelay(client, true);
    }
```

onClient

```
protected synchronized void removeClient(ServerThread client) {
        if (!isRunning) {
            return;
        }
        if (!clientsInRoom.containsKey(client.getClientId())) {
            info("Attempting to remove a client that doesn't exist in the
room");
            return;
        }
        ServerThread removedClient =
clientsInRoom.get(client.getClientId());
        if (removedClient != null) {
            // notify clients of someone leaving
            joinStatusRelay(removedClient, false);
            clientsInRoom.remove(client.getClientId());
            autoCleanup(); // Handle logic for empty session
        }
    }
```

onClient

Saved: 11/24/2025 10:57:04 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:

When a client joins, they are added to the clientsInRoom map. Crucially, syncExistingClients(client) is called to send the current room state (existing users) to the new client, ensuring synchronization.

When a client leaves, they are removed from the map, and other clients are notified. autoCleanup() is called to check if the room is empty and should be closed (handling empty session logic).

Saved: 11/24/2025 10:57:04 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()`

```
private void startGame() {
        gameState = Constants.GAME_STATE_READY;
        relay(null, "All players ready! Game starting...");
        // Reset points and elimination for initial state
        clientsInRoom.values().forEach(client -> {
            client.getUser().setPoints(0);
            client.getUser().setEliminated(false);
            client.getUser().setCurrentChoice(null);
        });
        startRound(); // Trigger next lifecycle event
    }
```

sessionstart

Saved: 11/24/2025 11:08:56 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:

This method initializes the session. It resets all player data (points to 0, elimination status to false) to ensure a fresh game.

It immediately calls startRound() to begin the first round of the game.

Saved: 11/24/2025 11:08:56 PM

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

Progress: 100%

**Details:**

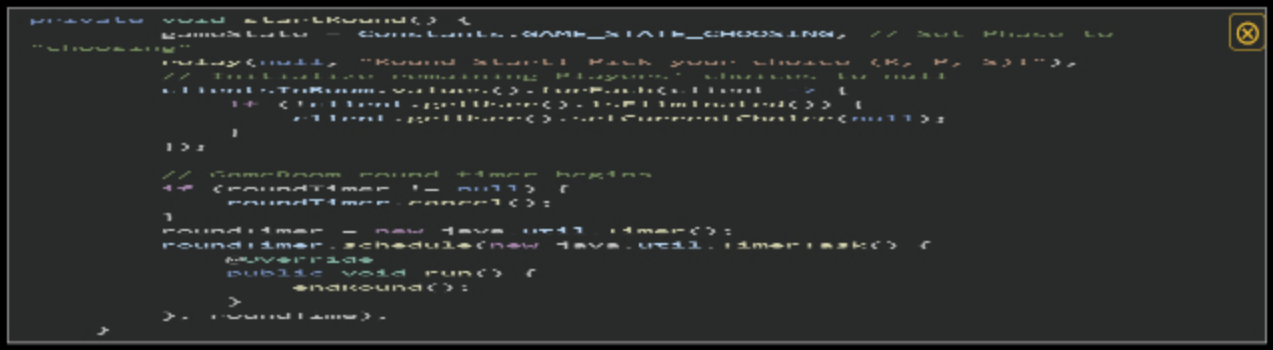
- Reqs from Document
  - Initialize remaining Players' choices to null (not set)
  - Set Phase to "choosing"
  - GameRoom round timer begins

### Part 1:

Progress: 100%

**Details:**

- Show the snippet of `onRoundStart()`



roundtsart

Saved: 11/24/2025 11:09:54 PM

### Part 2:

Progress: 100%

**Details:**

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

Your Response:

Sets the game state to CHOOSING. Clears any previous choices for active players. Starts a 30-second timer that will trigger endRound() if time expires.

Saved: 11/24/2025 11:09:54 PM

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

Progress: 100%

**Details:**

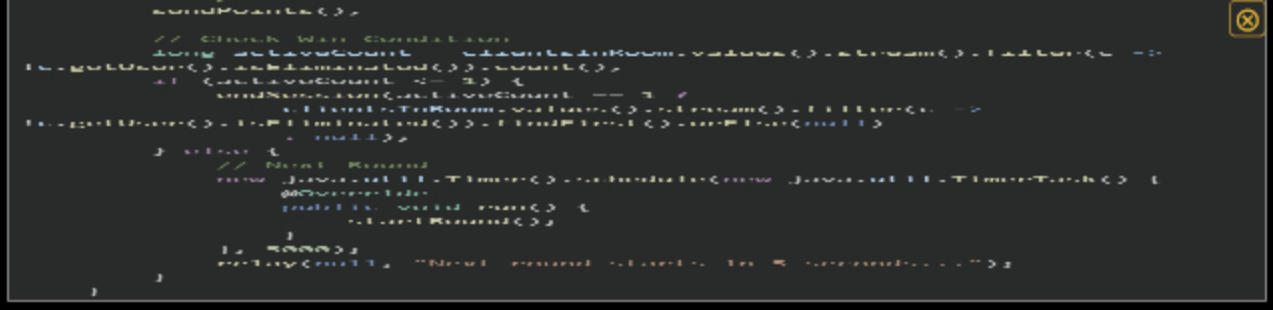
- Reqs from Document
  - **Condition 1:** Round ends when round timer expires
  - **Condition 2:** Round ends when all active Players have made a choice
  - All Players who are not eliminated and haven't made a choice will be marked as eliminated
  - Process Battles:
    - Round-robin battles of eligible 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()`



end

end

end

Saved: 11/24/2025 11:12:52 PM

## Part 2:

Progress: 100%

**Details:**

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

Your Response:

Stops the timer. Eliminates players who didn't pick. Executes round-robin battles (P1 vs P2, P2 vs P3, etc.). Awards points to winners and marks losers for elimination. Syncs points to all clients. Checks if 0 or 1 player remains to end the session, otherwise starts the next round.

Saved: 11/24/2025 11:12:52 PM

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

Progress: 100%

**Details:**

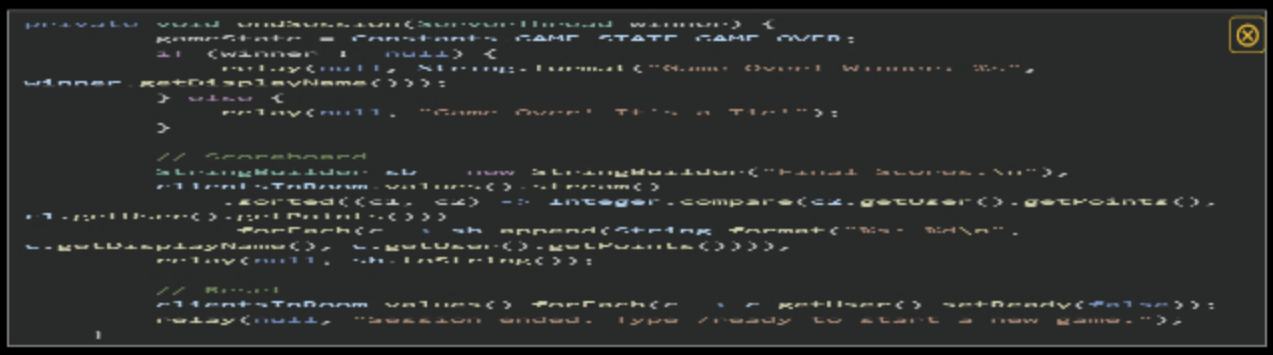
- Reqs from Document
  - **Condition 1:** Session ends when one Player remains (they win)
  - **Condition 2:** Session ends when no Players remain (this is a tie)
  - Send the final scoreboard to all clients sorted by highest points to lowest (include a game over message)
  - Reset the player data for 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()`



sessionend

Saved: 11/24/2025 11:42:04 PM

## Part 2:

Progress: 100%

**Details:**

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

Your Response:

Declares the winner or tie. Displays the final scoreboard sorted by points. Resets the isReady status for all clients so a new session can be started with a fresh ready check

Saved: 11/24/2025 11:42:04 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)

```
case PICK:
                currentRoom.handlePick(this, incoming.getMessage());
                break;
```

code

```
private void sendPick(String choice) throws IOException {
        Payload payload = new Payload();
        payload.setPayloadType(PayloadType.PICK);
```

```
        payload.setMessage(choice);
        sendToServer(payload);
    }
```

code

```
} else if (text.startsWith("pick")) {
                String choice = text.replace("pick", "").trim();
                // ... validation ...
                sendPick(choice);
                wasCommand = true;
            }
```

code

```
private void processMessage(Payload payload) {
        System.out.println(TextFX.colorize(payload.getMessage(),
Color.BLUE));
    }
```

code

```
protected synchronized void handlePick(ServerThread sender, String choice)
{
        // ... phase and elimination checks ...
        sender.getUser().setCurrentChoice(choice);
        relay(null, String.format("%s made a choice",
sender.getDisplayName()));

        // Check if all active players have picked
        if (allPicked) {
            processEndRound();
        }
    }
```

code

Saved: 11/24/2025 11:49:43 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:

Client: User types /pick R. processClientCommand parses it and calls sendPick. Client: sendPick creates a PICK payload and sends it to the serverServer: ServerThread receives the payload and callscurrentRoom.handlePick. Room: handlePick validates the choice, updates the user's state, and relays a confirmation message ("X made a choice"). It then checks if all players have picked to potentially end the round.Client: Receives the confirmation message viaprocessMessage and displays it.

Saved: 11/24/2025 11:49:43 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

terminal

terminal

terminal

terminal

Saved: 11/24/2025 11:51:08 PM

# Section #4: ( 1 pt.) Misc

Progress: 100%

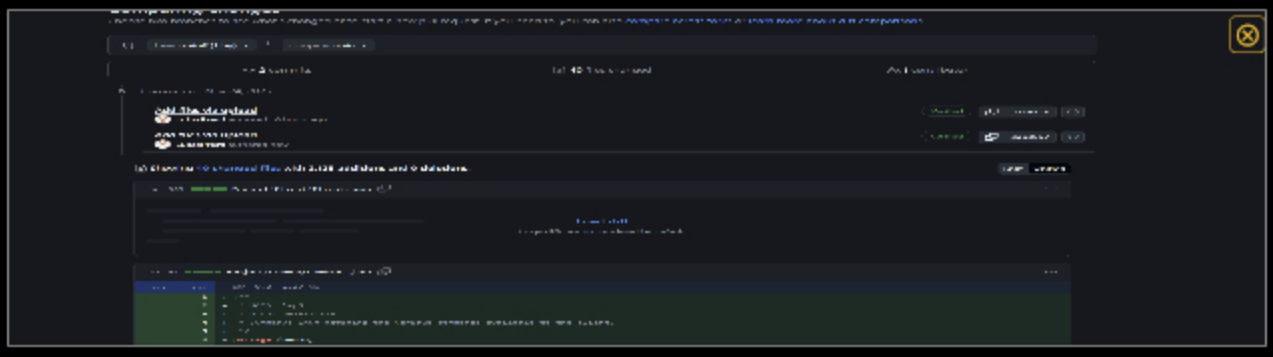
## Task #1 ( 0.33 pts.) - Github Details

Progress: 100%

### Part 1:

Progress: 100%

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



something about me not having everything from repo on local machine so it woudlnt let me push so I dropped file in from browser

Saved: 11/24/2025 11:55:20 PM

### Part 2:

Progress: 100%

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

URL #1
https://github.com/LukasPresti/lap5-it1114-007/compare/main%40%7B1day%7D...main

URL
https://github.com/LukasPresti/la

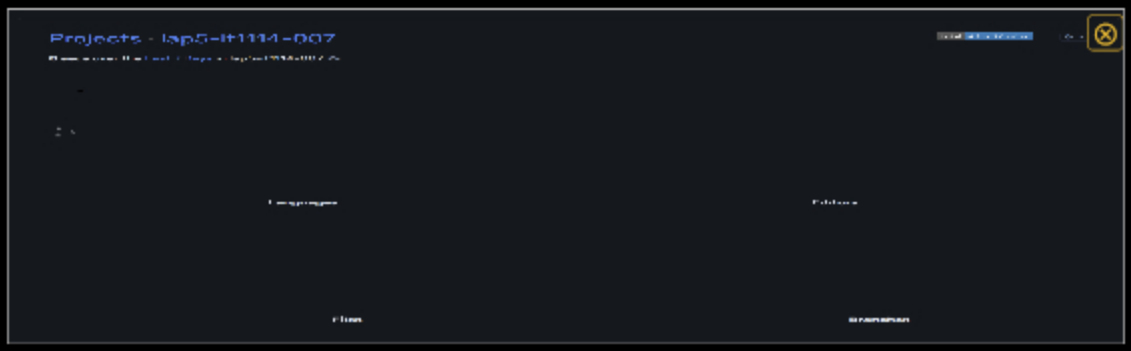
Saved: 11/24/2025 11:55:20 PM

## 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




wakatime not capturing time spent working for some reason

Saved: 11/24/2025 11:56:26 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 still really dislike coding

Saved: 11/24/2025 11:56:37 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:

taking all the pictures

Saved: 11/24/2025 11:56:48 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:

everything else

Saved: 11/24/2025 11:56:57 PM