Submission Worksheet

Submission Data

Course: IT114-003-F2025
Assignment: IT114 Milestone 2 - RPS
Student: Bryan S. (bs768)
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 2:12:35 PM
Updated: 11/24/2025 8:02:29 PM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-003-F2025/it114-milestone-2-rps/grading/bs768
View Link: https://learn.ethereallab.app/assignment/v3/IT114-003-F2025/it114-milestone-2-rps/view/bs768

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

```
    //bs768, 11/24/2025, toString method for Payload
    @Override
    public String toString() {
        return String.format("Payload[%s] Client Id [%s] Message: [%s]", getPayloadType(), getClientId(), getMessage());
    }
}
```

Payload.java

```
    //bs768, 11/24/2025, toString method for PointsPayload
    @Override
    public String toString() {
        return String.format("PointsPayload[ClientId=%d, Points=%d]", getClientId(), points);
    }
}
Ctrl+I for Command, Ctrl+L for Agent
```

PointsPayload.java

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

    public PointsPayload(long clientId, int points) {
        setClientId(clientId);
        setPoints(points);
        setPayloadType(PayloadType.POINTS_UPDATE);
        setMessage("Points updated");
    }

    public int getPoints() {
        return points;
    }

    public void setPoints(int points) {
        this.points = points;
    }

    //bs768, 11/24/2025, toString method for PointsPayload
    @Override
    public String toString() {
        return String.format("PointsPayload[ClientId %d, Points %d]", getClientId(),
```

PointsPayload.java

Saved: 11/24/2025 6:40:43 PM

Part 2:

Progress: 100%

Details:

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

Your Response:

Payload.java - Base payload class for general communication containing client ID, message, and payload type. PointsPayload.java - Specialized payload for synchronizing player points across all clients.

Saved: 11/24/2025 6:40:43 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

```
//bs768, 11/24/2025, addClient method for Room
protected synchronized void addClient(ServerThread client) {
    if (!isRunning) { // block action if Room isn't running
        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);
    // notify clients of someone joining
    joinStatusRelay(client, true);
}
```

addClient()

```
//bs768, 11/24/2025, removeClient method for Room
protected synchronized void removeClient(ServerThread client) {
    if (!isRunning) { // block action if Room isn't running
        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 joining
        joinStatusRelay(removedClient, false);
        clientsInRoom.remove(client.getClientId());
        readyPlayers.remove(client.getClientId());
        autoCleanup();
    }
}
```

removeClient()

Saved: 11/24/2025 6:47:28 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:

Adds:Checks if room is running and client doesn't already exist Adds client to clientsInRoom map Sets the current room reference on the client Clears the joining client's user list Synchronizes all existing clients' data Notifies all clients that a new player has joined

Removes: Checks if room is running and client exists Notifies all remaining clients that a player has left Removes client from clientsInRoom map Removes client from readyPlayers set Checks if room is empty

Saved: 11/24/2025 6:47:28 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()`

```java
//bs768, 11/24/2025, sessionStart method for Room
private void sessionStart() {
    isSessionActive = true;
    relay(null, "All players ready! Session starting.");
    // Reset all players
    for (ServerThread client : clientsInRoom.values()) {
        client.setPoints(0);
        client.setEliminated(false);
        client.setChoice(null);
        syncPlayerPoints(client);
    }
```

```
        roundStart();
    }
```

sessionStart()

Saved: 11/24/2025 6:52:29 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:

Sets isSessionActive = true to indicate game is in progress Sends start message to all players Sets all players points to 0 Marks all players as not eliminated Clears all player choices Calls roundStart() to begin the first round

Saved: 11/24/2025 6:52:29 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()`

```
//bs768, 11/24/2025, roundStart method for Room
private void roundStart() {
    if (!isSessionActive)
        return;

    // Reset choices for active players
    for (ServerThread client : clientsInRoom.values()) {
        if (!client.isEliminated()) {
            client.setChoice(null);
        }
    }

    Payload p = new Payload();
    p.setPayloadType(PayloadType.ROUND_START);
    p.setMessage("Round Started! Type /pick [r|p|s]");
    broadcast(p);
```

```
}
```

roundStart()

Saved: 11/24/2025 6:56:37 PM

Part 2:

Progress: 100%

Details:

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

Your Response:

Verifies session is active before proceeding Makes all non-eliminated players choices to null Sends ROUND_START payload with instructions startRoundTimer() starts a 30 second timer

Saved: 11/24/2025 6:56:37 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:

Part 1:

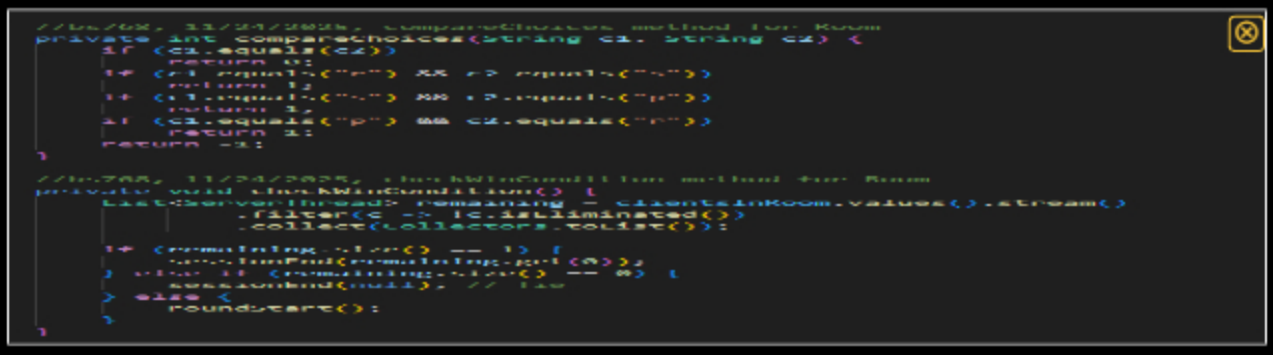
Progress: 100%

Details:

- Show the snippet of `onRoundEnd()`

Part 1

Part2



Part3

Saved: 11/24/2025 7:06:16 PM

Part 2:

Progress: 100%

Details:

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

Your Response:

Sets roundActive = false to prevent duplicate processing Cancels the round timer if still active Marks any active player who didn't make a choice as eliminated Calls processBattles() to determine round outcomes Each player battles their neighbor Uses compareChoices() to determine winner Winner receives 1 point updatePlayerPoints() automatically syncs points to all clients Losers are marked as eliminated Battle results relayed to all clients Calls checkWinCondition() to determine if game continues

Saved: 11/24/2025 7:06:16 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 7:10:13 PM

Part 2:

Progress: 100%

Details:

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

Your Response:

Sets isSessionActive = false to end the game Clears readyPlayers set for next session Displays winner or tie message Sends SESSION_END payload with game over message and scoreboard Resets points to 0 Clears elimination status Clears choices Syncs reset points to all clients Prompts players to use /ready for new session

Saved: 11/24/2025 7:10:13 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)

```
    // bs768, 11/24/2025, Client Processing
} else if (text.startsWith("pick")) {
    text = text.replace("pick", "").trim();
```

```
    if (text.length() == 0) {
        LoggerUtil.INSTANCE.warning(TextFX.colorize("Usage: /pick <r|p|s>", Color.RED));
    } else {
        sendPick(text);
    }
    wasCommand = true;
} else if (text.equalsIgnoreCase("ready")) {
    sendReady();
    wasCommand = true;
}
```

ClientProcessing

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

ServerThread

```
//bs768, 11/24/2025, handlePick method for Room
public synchronized void handlePick(ServerThread client, String choice) {
    if (!roundActive || client.isEliminated()) {
        return;
    }
    if (choice == null
            || (!choice.equalsIgnoreCase("r") && !choice.equalsIgnoreCase("p") && !choice.equalsIgnoreCase("s"))) {
        client.sendMessage(Constants.DEFAULT_CLIENT_ID, "Invalid choice. Use r, p, or s.");
        return;
    }

    client.setChoice(choice.toLowerCase());
    relay(null, client.getDisplayName() + " picked their choice.");

    checkRoundEnd();
}
```

GameRoomHandling

```
//bs768, 11/24/2025, relay method for Room
protected synchronized void relay(ServerThread sender, String message) {
    if (!isRunning) {
        return;
    }
    String senderString = sender == null ? String.format("Room[%s]", getName())
            : sender.getDisplayName();
    final long senderId = sender == null ? Constants.DEFAULT_CLIENT_ID : sender.getClientId();
    // Note: formattedMessage must be final (or effectively final) since outside
    // scope can't be changed inside a callback function (see removeIf() below)
    final String formattedMessage = String.format("%s: %s", senderString, message);

    // loop over clients and send out the message; remove client if message failed
    // to be sent
    // Note: this uses a lambda expression per each item in the values() collection,
    // it's one way we can safely remove items during iteration
    info(String.format("sending message to %s recipients: %s", clientsInRoom.size(), formattedMessage));

    clientsInRoom.values().removeIf(serverThread -> {
        boolean failedToSend = !serverThread.sendMessage(senderId, formattedMessage);
        if (failedToSend) {
            LoggerUtil.INSTANCE.warning(
                    String.format("Removing disconnected %s from list", serverThread.getDisplayName()));
            disconnect(serverThread);
        }
        return failedToSend;
    });
}
```

Send/Sync Method

```
//bs768, 11/24/2025, sendMessage method for ServerThread
protected boolean sendMessage(long clientId, String message) {
    Payload payload = new Payload();
    payload.setPayloadType(PayloadType.MESSAGE);
    payload.setMessage(message);
    payload.setClientId(clientId);
    return sendToClient(payload);
}
```

ServerThread

ServerThread

```java
//bs768, 11/24/2025, processMessage method for Client
private void processMessage(Payload payload) {
    LoggerUtil.INSTANCE.info(TextFX.colorize(payload.getMessage(), Color.BLUE));
}
```

ProcessMethod

Saved: 11/24/2025 7:29:42 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:

User types /pick r in terminal sendPick("r") creates Payload with type PLAYER_PICK and message "r" Payload sent to server via sendToServer() processPayload() receives PLAYER_PICK payload Calls currentRoom.handlePick(this, "r") handlePick() validates choice and round state Stores choice of player Calls relay() to broadcast pick notification

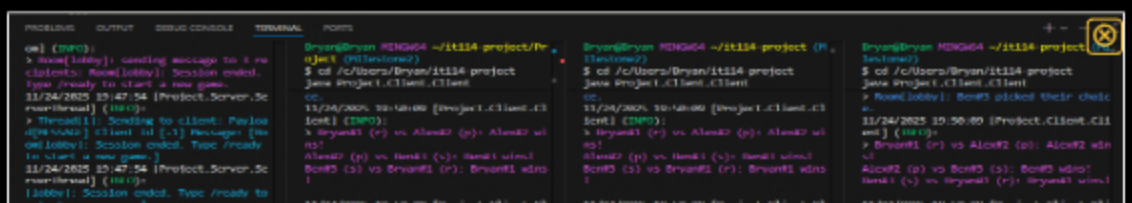
Saved: 11/24/2025 7:29:42 PM

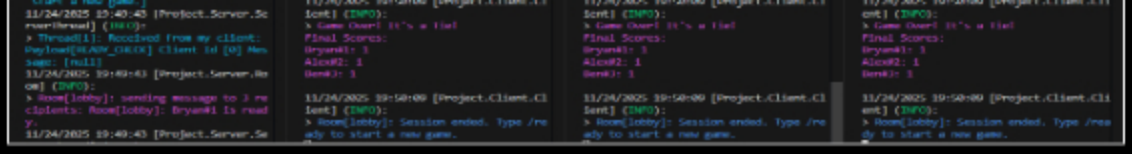
Task #2 (2 pts.) - Game Cycle Demo

Progress: 100%

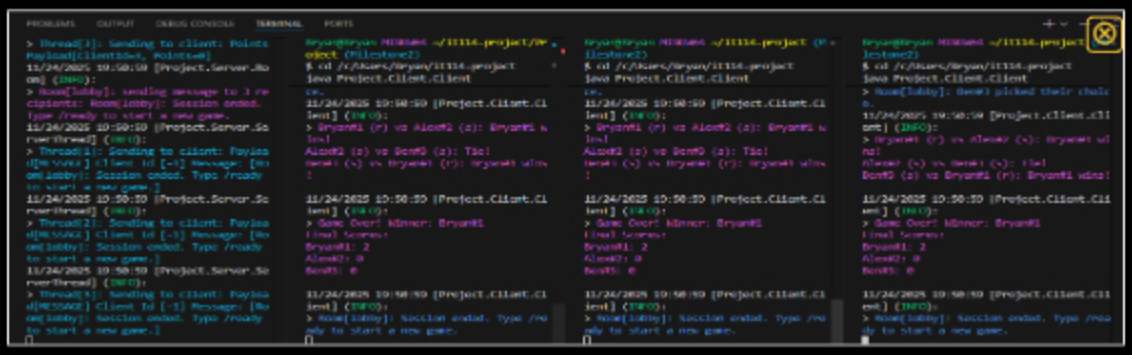
Details:

- 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





Game 1



Game 2

Saved: 11/24/2025 7:51:35 PM

Section #4: (1 pt.) Misc

Progress: 100%

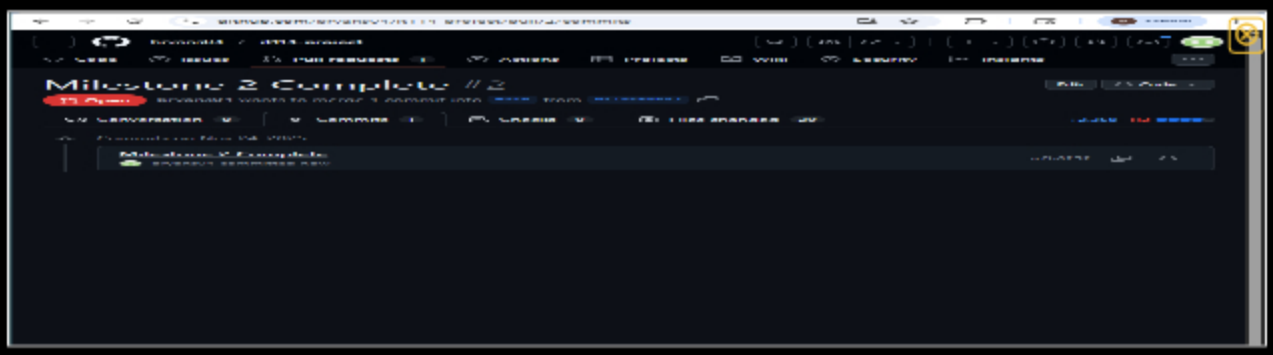
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

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



Github

Saved: 11/24/2025 7:54:56 PM

Part 2:

Progress: 100%

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

URL #1
https://github.com/bryans04/it114-project/pull/2/commits

URL
https://github.com/bryans04/it11

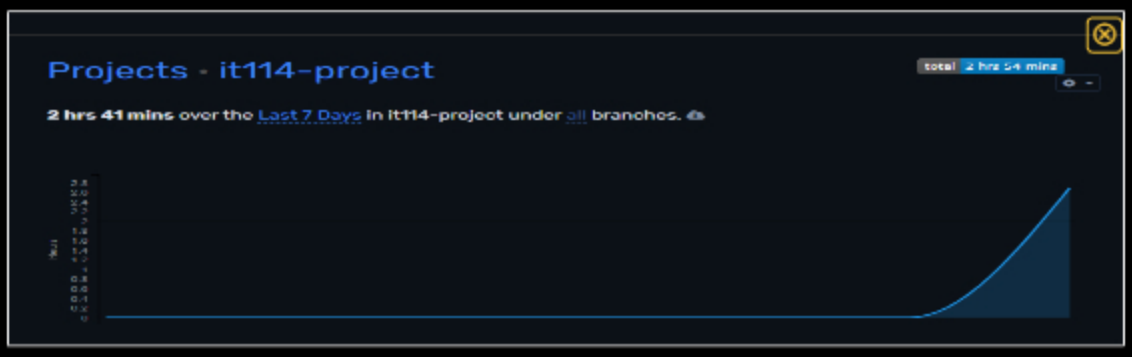
Saved: 11/24/2025 7:54:56 PM

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



wkatime

Saved: 11/24/2025 7:56:27 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:

Paykoads are tricky but also, very helpful once you learn how to use them correctly.

Saved: 11/24/2025 7:57:13 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:

Setting up the scoreboard was one of the easiest things for me. As well as suprisingly setting up the pick r/p/s since i usually tend to always seem to find a way to mess up commands in past HWs.

Saved: 11/24/2025 8:02:29 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 for me was setting up the round robin part, because you had to make every player verse eachother once and keep track of scores at the same time.

Saved: 11/24/2025 8:01:32 PM