Submission Worksheet

Submission Data

Course: IT114-450-M2025
Assignment: IT114 Milestone 2 - RPS
Student: Anthony L. (agl8)
Status: Submitted | **Worksheet Progress:** 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Started: 10/26/2025 4:42:53 PM
Updated: 11/2/2025 7:47:28 PM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-2-rps/grading/agl8
View Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-2-rps/view/agl8

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

Payload.java image

ConnectionPayload.java image

PointsPayload.java image

Saved: 11/2/2025 6:01:16 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. ConnectionPayload.java: Extends Payload for connect/room sync. Used for join,leave,sync, and handshake. PointsPyaload.java: Extends Pyaload to sync a players score. Adds int points plus toString and prints it. Used for broadcasting points.

Saved: 11/2/2025 6:01:16 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

```java
/** {@inheritDoc} */
//agl8 10-26-25
//when user joins syncs phase, ready, turn status.
@Override
protected void onClientAdded(ServerThread sp) {
    // sync GameRoom state to new client
    syncCurrentPhase(sp);
    syncReadyStatus(sp);
    syncTurnStatus(sp);
} <- #37-42 protected void onClientAdded(ServerThread sp)
```

onClientAdded

```java
/** {@inheritDoc} */
//agl8 10-26-25
//cleans up timers and turns
@Override
protected void onClientRemoved(ServerThread sp) {
    // added after Summer 2024 Demo
    // Stops the timers so room can clean up
    LoggerUtil.INSTANCE.info("Player Removed, remaining: " + clientsInRoom.size());
    long removedClient = sp.getClientId();
    turnOrder.removeIf(player -> player.getClientId() == sp.getClientId());
    if (clientsInRoom.isEmpty()) {
        resetReadyTimer();
        resetTurnTimer();
        resetRoundTimer();
        onSessionEnd();
    } else if (removedClient == currentTurnClientId) {
        onTurnStart();
    }
```

onClientRemoved

Saved: 11/2/2025 6:01:32 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:

onClientAdded: Syncs the new user with the room status. onClientREmoved: If room is empty, it stops timers and end sissions.

Saved: 11/2/2025 6:01:32 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()`

```
92      /** {@inheritDoc} */
93      // agl8 10-26-25
94      // sets phase to IN_PROGRESS and starts the first round
95      @Override
96      protected void onSessionStart() {
97          LoggerUtil.INSTANCE.info("onSessionStart() start");
98          changePhase(Phase.IN_PROGRESS);
99          currentTurnClientId = Constants.DEFAULT_CLIENT_ID;
100         setTurnOrder();
101         round = 0;
102         LoggerUtil.INSTANCE.info("onSessionStart() end");
103         onRoundStart();
104     } <- #96-104 protected void onSessionStart()
```

onSessionStart

Saved: 11/2/2025 6:02:18 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:

onSessionStart sets the phase to IN_PROGRESS, clears current turn, shuffle turns order for ready players, sets round to zero, and calls onRoundStart.

Saved: 11/2/2025 6:02:18 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()`

```java
/** {@inheritDoc} */
// agl8 10-26-25
// starts a new round and resets turn status
@Override
protected void onRoundStart() {
    LoggerUtil.INSTANCE.info("onRoundStart() start");
    resetRoundTimer();
    resetTurnStatus();
    round++;
    relay(null, String.format("Round %d has started", round));
    // startRoundTimer(); Round timers aren't needed for turns
    // if you do decide to use it, ensure it's reasonable and based on the number of
    // players
    LoggerUtil.INSTANCE.info("onRoundStart() end");
    onTurnStart();
} <-- #110-121 protected void onRoundStart()
```

onRoundStart

Saved: 11/2/2025 6:05:38 PM

Part 2:

Progress: 100%

Details:

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

Your Response:

onRoundStart clears previous picks and resets everyone's turn. It also increments rounds and announces which round it is. Then it calls onTurnSTart for the timer to begin.

Saved: 11/2/2025 6:05:38 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 ( )

```java
// Note: logic between Round Start and Round End is typically handled via timers
// and user interaction
/** {@inheritDoc} */
// aql8 10 26 25
// ends the round and decides next step (session end or another round)
@Override
protected void onRoundEnd() {
    LoggerUtil.INSTANCE.info("onRoundEnd() start");
    resetRoundTimer(); // reset timer if round ended without the time expiring

    LoggerUtil.INSTANCE.info("onRoundEnd() end");
    if (round >= 3) {
        onSessionEnd();
    } else {
        onRoundStart();
    }
} <-- #167-177 protected void onRoundEnd()
```

onRoundEnd

Saved: 10/27/2025 11:19:12 PM

Part 2:

Progress: 100%

Details:

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

Your Response:

onRoundEnd stops the timer and then handles the approach to the next step. If the round limit has been reached then it'll call onSessionEnd and if not it'll call onRoundStart and start the next round.

Saved: 10/27/2025 11:19:12 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()`

```java
/** {@inheritDoc} */
// agl8 10-26-25
// cleans up after session and returns to READY phase
@Override
protected void onSessionEnd() {
    LoggerUtil.INSTANCE.info("onSessionEnd() start");
    turnOrder.clear();
    currentTurnClientId = Constants.DEFAULT_CLIENT_ID;
    resetReadyStatus();
    resetTurnStatus();
    changePhase(Phase.READY);
    LoggerUtil.INSTANCE.info("onSessionEnd() end");
} <-- #183-191 protected void onSessionEnd()
// end lifecycle methods
```

onSessionEnd

Saved: 11/2/2025 6:06:32 PM

Part 2:

Progress: 100%

Details:

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

Your Response:

onSessionEnd clears the turn order, resets readies, switches phase to READY and gets ready for a new ready check.

Saved: 11/2/2025 6:06:32 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)

```
//agl8 10-26-25
//accepts picks and sends to server
else if (text.startsWith("pick")) {
    String arg = text.replaceFirst("^pick\\s*", "").trim();
    if (arg.isEmpty()) {
        LoggerUtil.INSTANCE.warning(TextFX.colorize("Usage: /pick <r|p|s>", Color.RED));
        return true;
    }
    String choice = arg.substring(0,1).toLowerCase();
    if (!choice.equals("r") && !choice.equals("p") && !choice.equals("s")) {
        LoggerUtil.INSTANCE.warning(TextFX.colorize("Invalid choice. Use r, p, or s.", Color.RED));
        return true;
    }
    sendPick(choice);
    LoggerUtil.INSTANCE.info(TextFX.colorize("You picked: " + choice, Color.GREEN));
    wasCommand = true;
} <- #224-238 else if (text.startsWith("pick"))
```

process client command

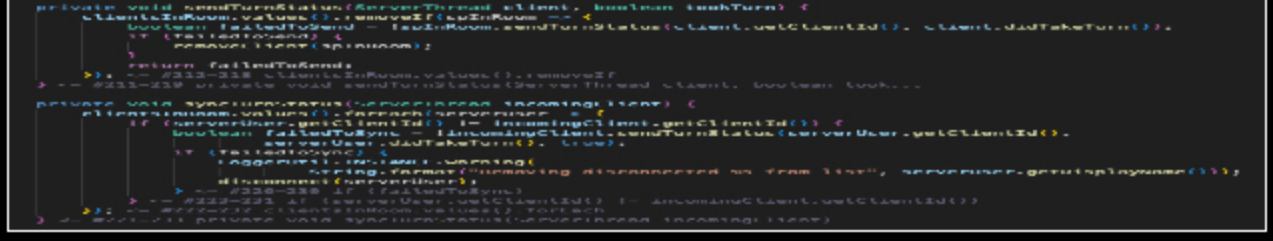
```
//agl8 10-26-25
//routes PICK payload to active Gameroom
case PICK:
    try {
        ((GameRoom) currentRoom).handlePick(this, incoming.getMessage());
    }
    catch (Exception e) {
        sendMessage(Constants.DEFAULT_CLIENT_ID, "You must be in a Gameroom to pick.");
    }
    break;
default:
    LoggerUtil.INSTANCE.warning(TextFX.colorize("Unknown payload type received", Color.RED));
    break;
```

process serverthread method

```
//agl8, 10-26-25
//handles rps choices
protected void handlePick(ServerThread currentUser, String choice) {
    try {
        checkPlayerInRoom(currentUser);
        checkCurrentPhase(currentUser, Phase.IN_PROGRESS);
        checkIsReady(currentUser);
        picks.put(currentUser.getClientId(), choice.toLowerCase());
        currentUser.setTookTurn(true);
        sendTurnStatus(currentUser, true);
        checkAllTookTurn();
    } <- #380-388 try
    catch(Exception e) {
        currentUser.sendMessage(Constants.DEFAULT_CLIENT_ID, "You can't pick right now.");
    }
} <- #379-392 protected void handlePick(ServerThread currentUser, String ch...
```

gameroom handle method

```
// send/sync data to ServerThread(s)
private void sendResetTurnStatus() {
    clientsInRoom.values().forEach(spInRoom -> {
        boolean failedToSend = !spInRoom.sendResetTurnStatus();
        if (failedToSend) {
            removeClient(spInRoom);
        }
    }); <- #781-787 clientsInRoom.values().forEach
} <- #780-788 private void sendResetTurnStatus()
```



send/sync method

```java
public boolean sendTurnStatus(long clientId, boolean didTakeTurn) {
    return sendTurnStatus(clientId, didTakeTurn, false);
}

public boolean sendTurnStatus(long clientId, boolean didTakeTurn, boolean quiet) {
    // NOTE for now using ReadyPayload as it has the necessary properties
    // An actual turn may include other data for your project
    ReadyPayload rp = new ReadyPayload();
    rp.setPayloadType(quiet ? PayloadType.SYNC_TURN : PayloadType.TURN);
    rp.setClientId(clientId);
    rp.setReady(didTakeTurn);
    return sendToClient(rp);
} <- #68-76 public boolean sendTurnStatus(long clientId, boolean didTakeT...
```

serverthread send method part 1 (turn sync)

```java
public boolean sendResetReady() {
    ReadyPayload rp = new ReadyPayload();
    rp.setPayloadType(PayloadType.RESET_READY);
    return sendToClient(rp);
} <- #85-89 public boolean sendResetReady()
```

serverthread send method part 2 (reset turn)

```java
public boolean sendCurrentPhase(Phase phase) {
    Payload p = new Payload();
    p.setPayloadType(PayloadType.PHASE);
    p.setMessage(phase.name());
    return sendToClient(p);
} <- #78-83 public boolean sendCurrentPhase(Phase phase)
```

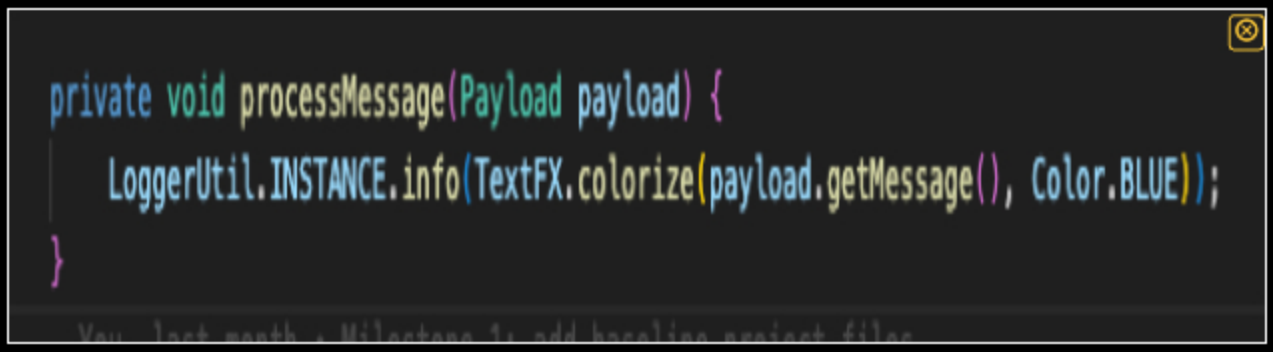
serverthread send method part 3 (phase)

```java
 */
protected boolean sendMessage(long clientId, String message) {
    Payload payload = new Payload();
    payload.setPayloadType(PayloadType.MESSAGE);
    payload.setMessage(message);
    payload.setClientId(clientId);
    return sendToClient(payload);
} <- #194-200 protected boolean sendMessage(long clientId, String message)
```

serverthread send method part 4 (message)

process method (switch)

process method (part 1 handlers)



process method (part 2 handlers)

Saved: 10/28/2025 9:09:12 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 picks and the client builds a pick payload with the choice and sends it. ServerThread: processPayload sends the pick to active GameRoom. GameRoom: handlePick validates room and readiness, records, marks turn, and broadcasts turn status. Client: receive TURN/SYNC_TURN and update the turn status display. GameRoom: once all ready players have picked, resolves rps,

sends the round result message, and ends the round. Client: receives the round outcome and prints it. if turn changes, they are notified.

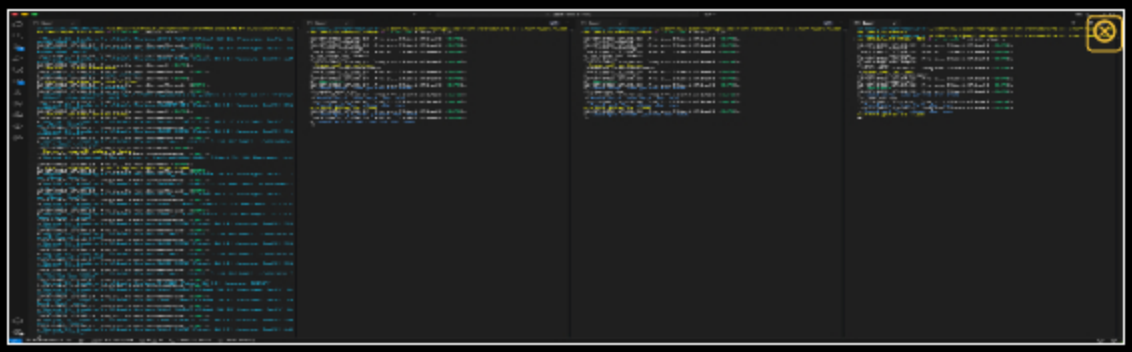
Saved: 10/28/2025 9:09:12 PM

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

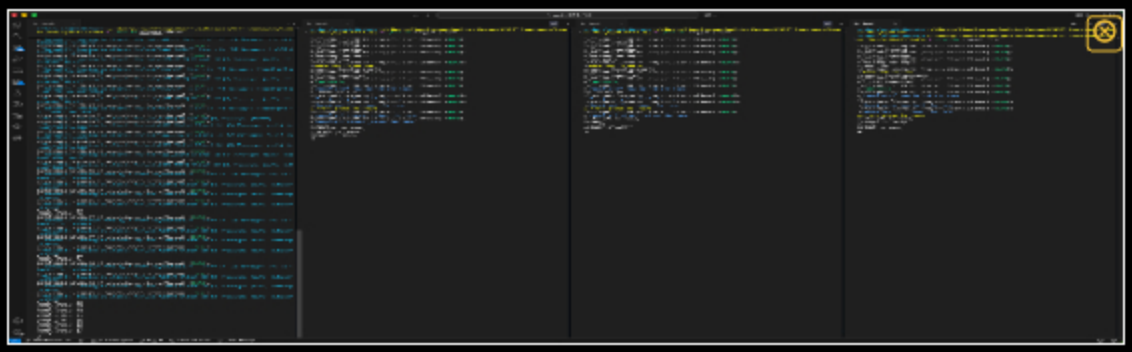
Progress: 100%

Details:

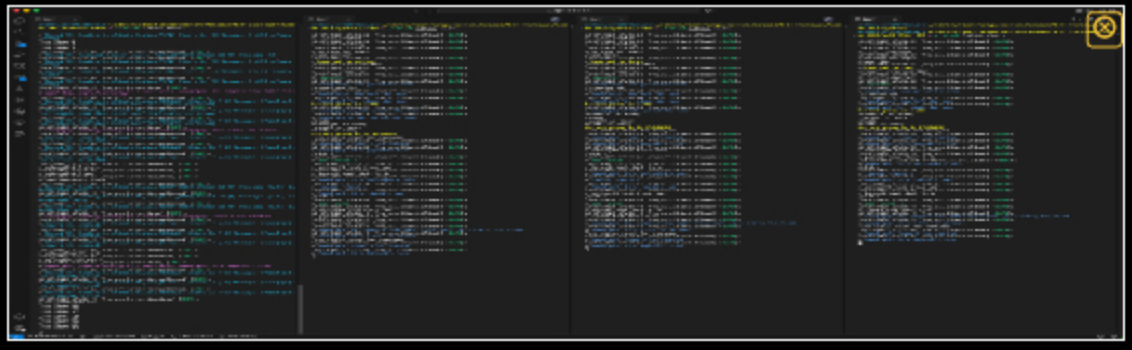
- Show examples from the terminal of a full session demonstrating each command and progress output
- This includes battle outcomes, scores and scoreboards, etc
- Ensure at least 3 Clients and the Server are shown
- Clearly caption screenshots



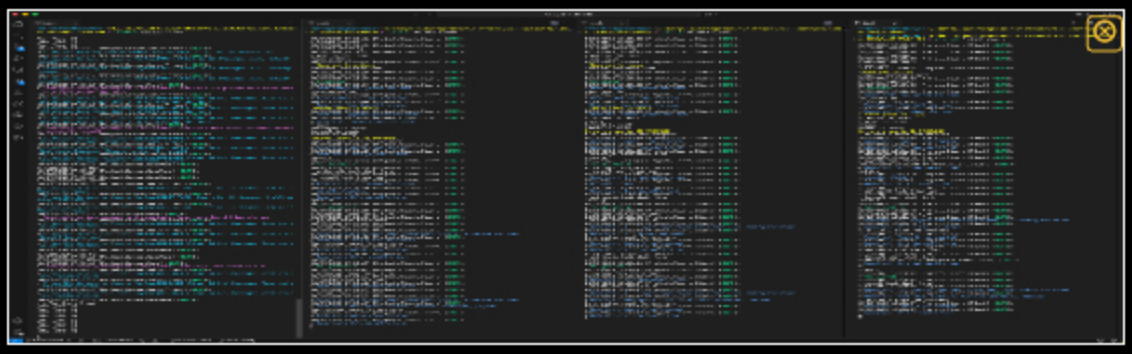
connect and join room



ready check



round 1 picks tie



round 2 multiple winners

round 3 session end

Saved: 10/27/2025 5:39:25 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

commits

Saved: 11/2/2025 7:47:28 PM

Part 2:

Progress: 100%

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

URL #1
https://github.com/agl8-2025/agl8-IT114-[illegible]/

URL
https://github.com/agl8-2025/agl

Saved: 11/2/2025 7:47:28 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 part 1

wakatime part 2

Saved: 11/2/2025 7:42:31 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 learned how communications work at a deeper level between clients and servers. I also learned how to make a server and client communicate via payloads as well as keeping multiple clients in sync at different phases.

Saved: 11/2/2025 7:13:49 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:

Easiest part about this milestone was setting up the loggers that print room status.

Saved: 11/2/2025 7:14:16 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:

Hardest part about this milestone was integrating the turn based component. For instance getting it on the right phase and keeping track of points for all players/determining a winner was tough.

Saved: 11/2/2025 7:15:10 PM