Submission Worksheet

Submission Data

Course: IT114-003-F2025
Assignment: IT114 - Milestone 3 - 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: 12/11/2025 12:42:38 PM
Updated: 12/11/2025 9:05:54 PM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-003-F2025/it114-milestone-3-rps/grading/bs768
View Link: https://learn.ethereallab.app/assignment/v3/IT114-003-F2025/it114-milestone-3-rps/view/bs768

Instructions

1. Refer to Milestone3 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 `Milestone3` branch
    1. `git checkout Milestone3`
    2. `git pull origin Milestone3`
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 Milestone3`
    4. On Github merge the pull request from `Milestone3` 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.) Core Ui

Progress: 100%

Task #1 (0.50 pts.) - Connection/Details Panels

Progress: 100%

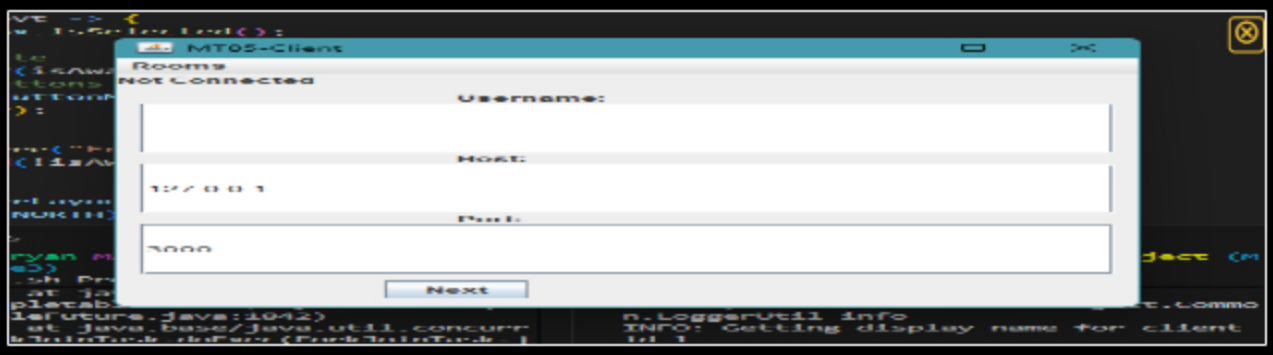
Part 1:

Progress: 100%

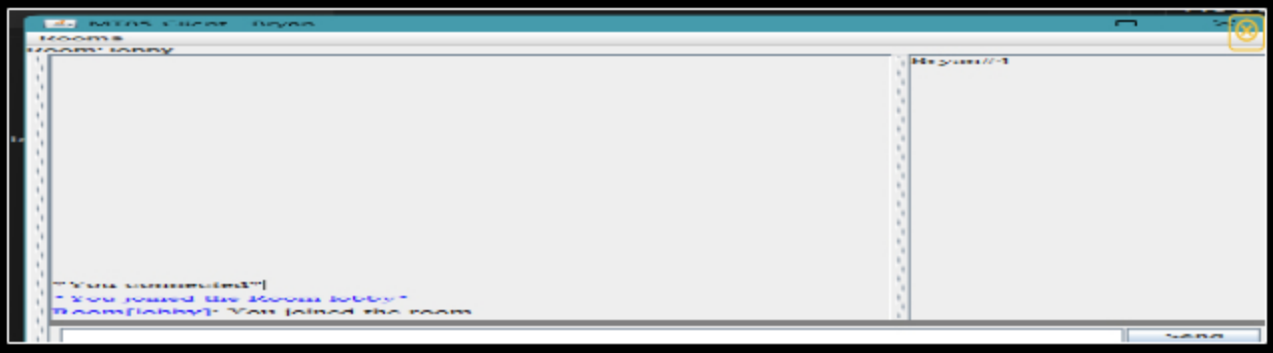
Details:

- Show the connection panel with valid data

- Show the user details panel with valid data



ConnectionPanel



UserDetails

Saved: 12/11/2025 12:48:33 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code flow from recording/capturing these details and passing them through the connection process

Your Response:

User enter information Infomation is then validated If valid then it triggers the connection Server Assigns ID Client Receives ID Send Username Server Broadcasts Join All Clients Update User List

Saved: 12/11/2025 12:48:33 PM

Task #2 (0.50 pts.) - Ready Panel

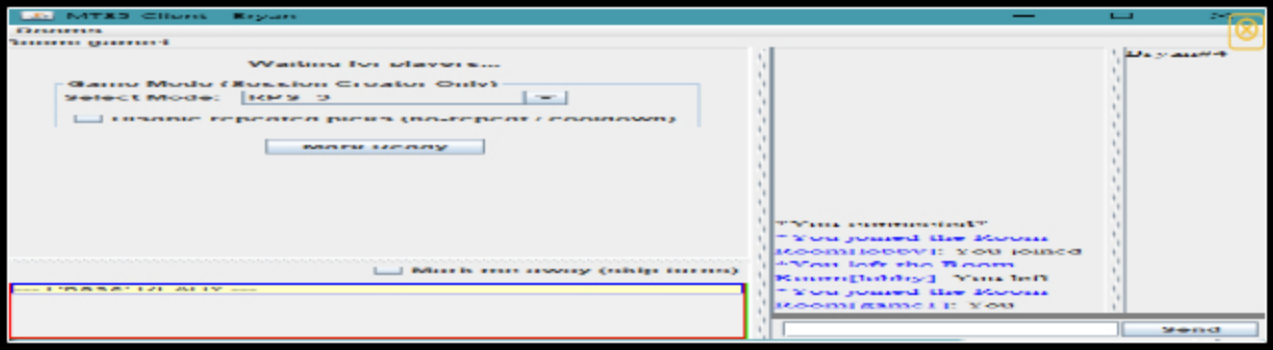
Progress: 100%

Part 1:

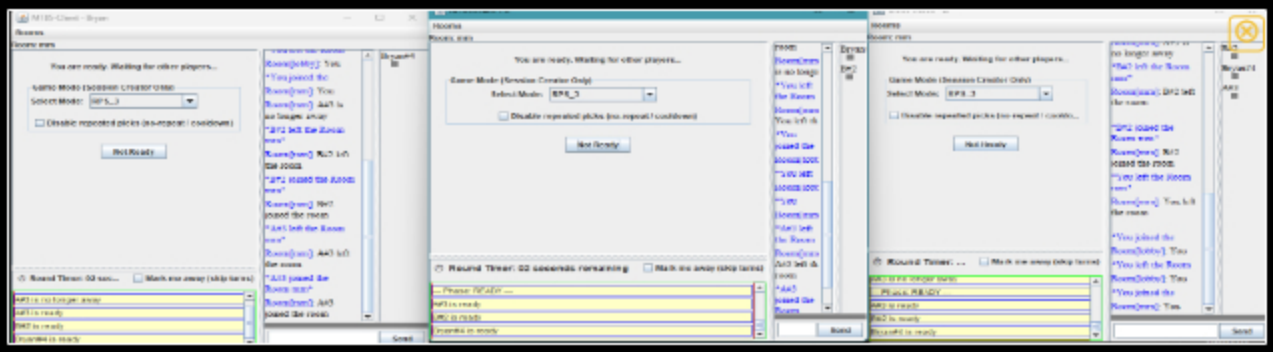
Progress: 100%

Details:

- Show the button used to mark ready
- Show a few variations of indicators of clients being ready (3+ clients)



Ready Button



Clients being ready 3+

Saved: 12/11/2025 1:12:28 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code flow for marking READY from the UI
- Briefly explain the code flow from receiving READY data and updating the UI

Your Response:

UI Button Click Updates Local UI Sends to Server
Server Updates State Tells All Clients That Client is Ready and is waiting for Others Each Client Udates User List Visual Indicator Appears Next to Clients Name

Saved: 12/11/2025 1:12:28 PM

Section #2: (2 pts.) Project Ui

Progress: 100%

Task #1 (0.67 pts.) - User List Panel

Progress: 100%

Details:

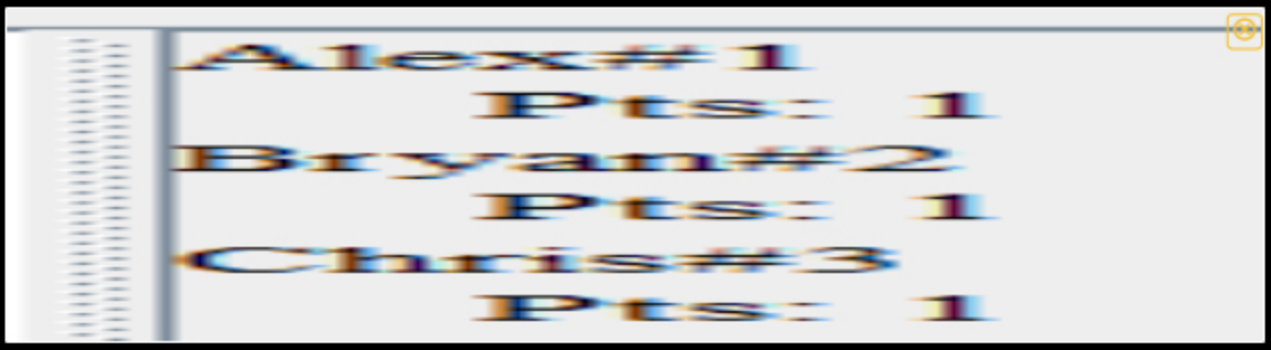
- Show the username and id of each user
- Show the current points of each user
- Users should appear in score order, sub-sort by name when ties occur
- Pending-to-pick users should be marked accordingly
- Eliminated users should be marked accordingly

Part 1:

Progress: 100%

Details:

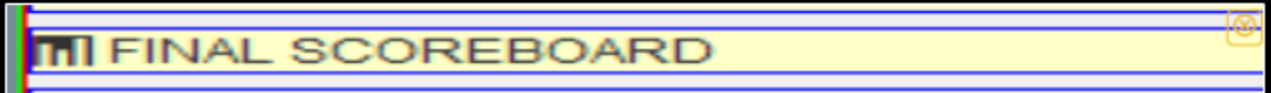
- Show various examples of points (3+ clients visible)
  - Include code snippets showing the code flow for this from server-side to UI
- Show that the sorting is maintained across clients
  - Include code snippets showing the code that handles this
- Show various examples of the pending-to-pick indicators
  - Include code snippets showing the code flow for this from server-side to UI
- Show various examples of elimination indicators
  - Include code snippets showing the code flow for this from server-side to UI

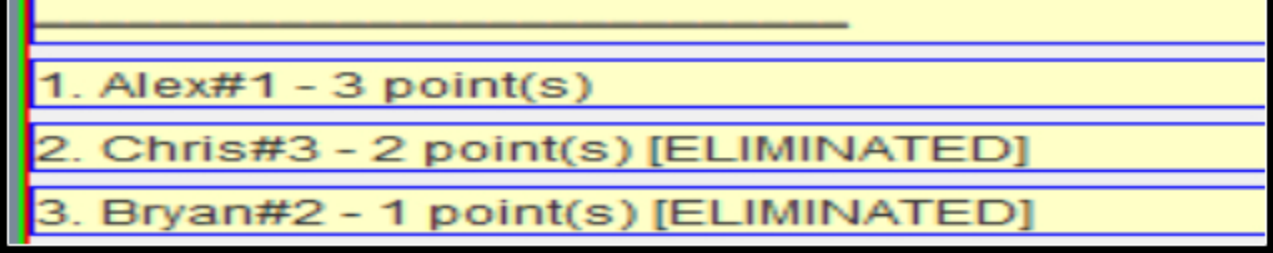


Sorting of players

```
for (ServerThread winner : roundWinners) {
    winner.changePoints(1);
    sendPlayerPoints(winner);
    sendGameEvent(String.format("%s wins the round and gets 1 point!
(Total: %d)",
            winner.getDisplayName(), winner.getPoints()));
}
```

Awarding points

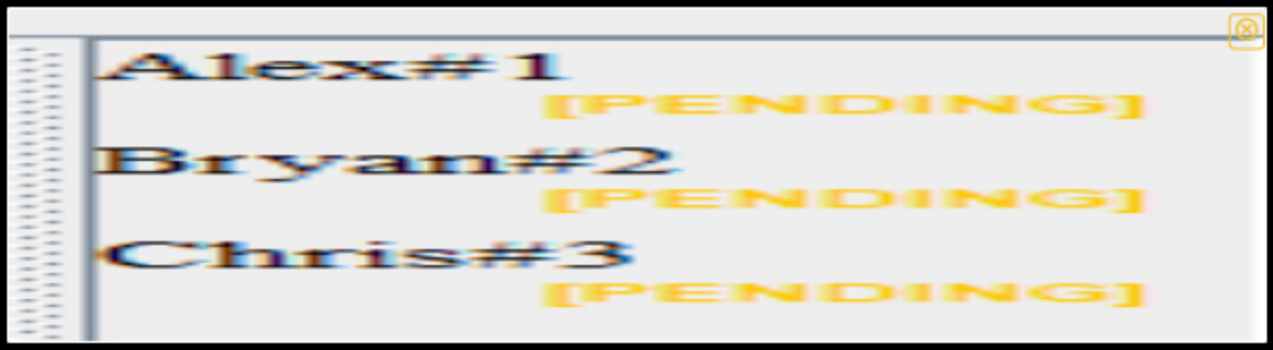




examples of points

```
private void sendPlayerPoints(ServerThread player) {
    PointsPayload pp = new PointsPayload();
    pp.setClientId(player.getClientId());
    pp.setPoints(player.getPoints());
    clientsInRoom.values().forEach(client -> {
        client.sendToClient(pp);
    });
}
```

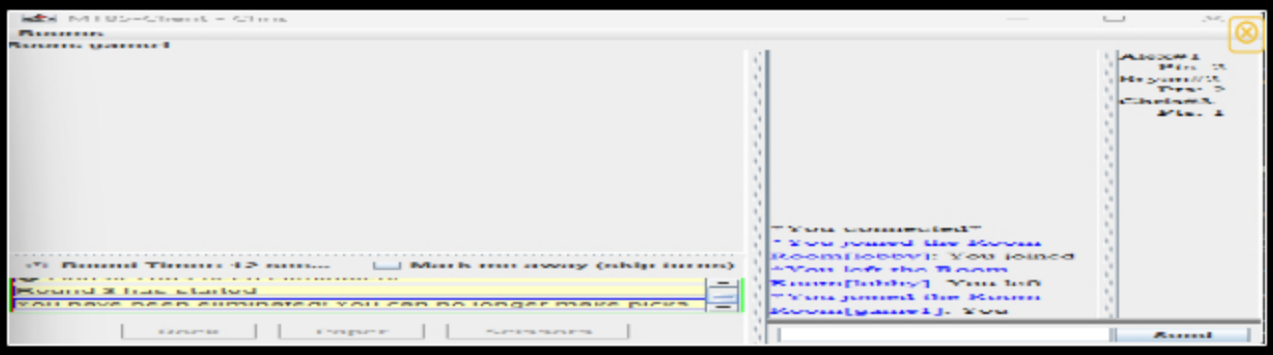
sending points to other clients(UI)



pending indicators

```
private void resortUserList() {
    SwingUtilities.invokeLater(() -> {
        LoggerUtil.INSTANCE.info("Resorting user list");
        try {
            userListArea.removeAll();

            java.util.List<UserListItem> sortedItems = userItemsMap.values().stream()
                    .sorted((a, b) -> {
                        int pointCompare = Integer.compare(b.getPoints(), a.getPoints());
                        if (pointCompare != 0) {
                            return pointCompare;
                        }
                        return a.getDisplayName().compareTo(b.getDisplayName());
                    })
                    .collect(java.util.stream.Collectors.toList());
```

sorting code



Elimination indicator in feed and on user list

Elimination indicator in feed and on user list

```
activePlayers.stream()
        .filter(p -> wins.get(p) == 0)
        .forEach(loser -> {
            loser.setEliminated(true);
            sendEliminationStatus(loser, true);
            sendGameEvent(String.format("%s has been eliminated!",
                    loser.getDisplayName()));
        });
```

elimination indicator

Saved: 12/11/2025 6:34:16 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code flow for points updates from server-side to the UI
- Briefly explain the code flow for user list sorting
- Briefly explain the code flow for server-side to UI of pending-to-pick indicators
- Briefly explain the code flow for server-side to UI of elimination indicators

Your Response:

1. when points get awarded it sends the PointsPayload to all clients clients then recieve payload and updates and resorts list of users
2. When resortUserList() is called, it sorts all players by points then alphabetically by name as a tiebreaker. The sorted items are then re-added to the UI panel in sequential order. 3.The server resets all turn statuses when a round starts, causing clients to mark all players with pending badges. As each player makes their choice, the server sends turn updates that clear the pending badge for that specific player. 4.When a player loses all battles in a round, the server sets them as eliminated and broadcasts an EliminationPayload to all clients. The client's UserListItem then displays a strike-through name.

Saved: 12/11/2025 6:34:16 PM

Task #2 (0.67 pts.) - Game Events Panel

Progress: 100%

Details:

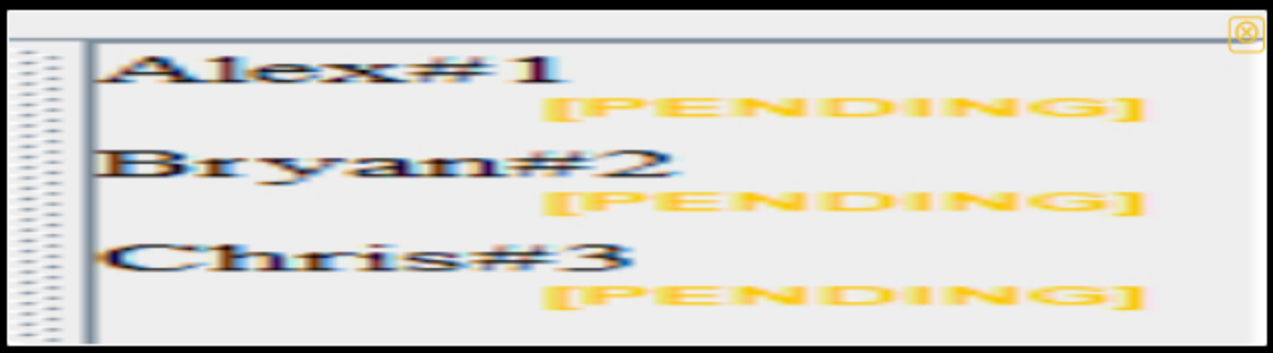
- Show the status of users picking choices
- Show the battle resolution messages from Milestone 2
  - Include messages about elimination
- Show the countdown timer for the round

Part 1:

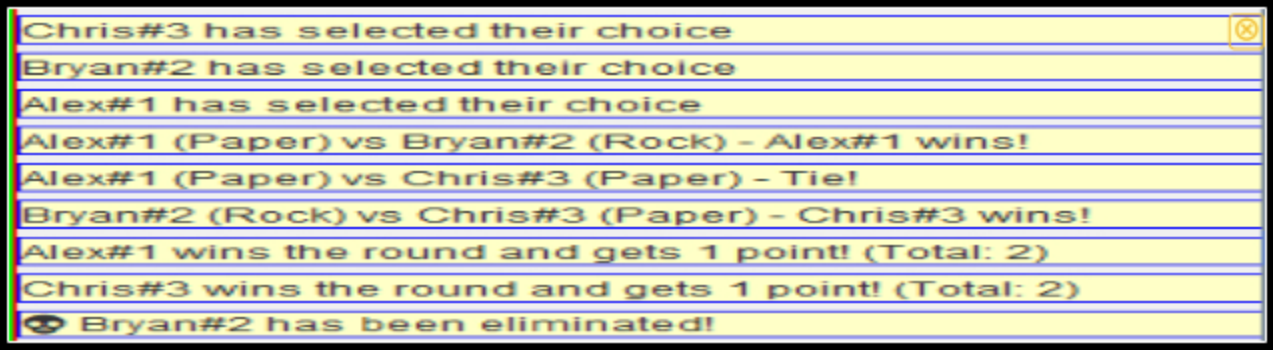
Progress: 100%

Details:

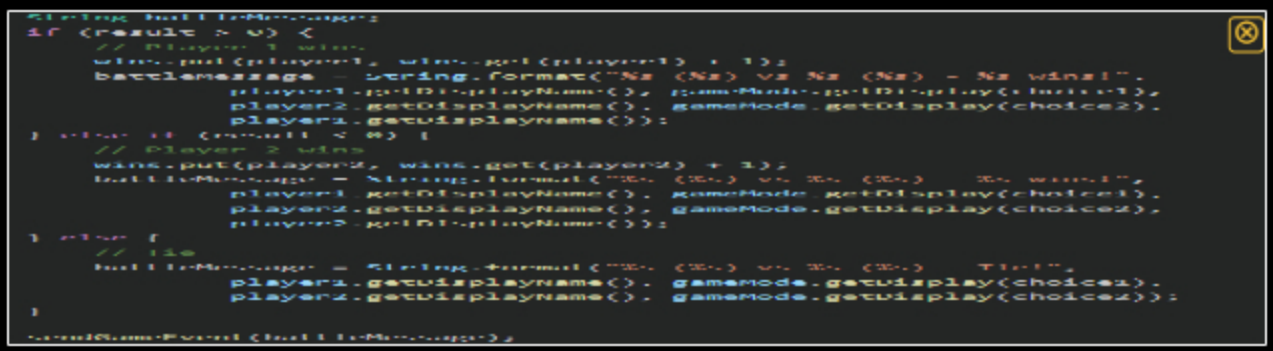
- Show various examples of each of the messages/visuals
- Show code snippets related to these messages from server-side to UI



status of users picking choices



elimination messages/users picks



code

```
activePlayers.stream()
        .filter(p -> wins.get(p) == 0)
        .forEach(loser -> {
            loser.setEliminated(true);
            sendEliminationStatus(loser, true);
            sendGameEvent(String.format("%s has been eliminated!",
                    loser.getDisplayName()));
        });
```

elimination code

```
private void startRoundTimer() {
    final int duration = 15;
    roundTimer = new Timer();
    roundTimer.scheduleAtFixedRate(new TimerTask() {
        int timeLeft = duration;
        @Override
        public void run() {
            sendTimerUpdate(TimerType.ROUND, timeLeft);
            if (timeLeft <= 0) {
                cancelRoundTimer();
                onRoundEnd();
            }
            timeLeft--;
        }
    }, 0, 1000);
}
```

code for timer

Saved: 12/11/2025 6:43:25 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code flow for generating these messages and getting them onto the UI

Your Response:

The server's GameRoom.java generates messages based on game events and broadcasts them to all clients via payload objects or the sendGameEvent() method. The client's Client.java receives these payloads, processes them, and triggers UI callbacks that update the appropriate view components.

Saved: 12/11/2025 6:43:25 PM

Task #3 (0.67 pts.) - Game Area

Progress: 100%

Details:

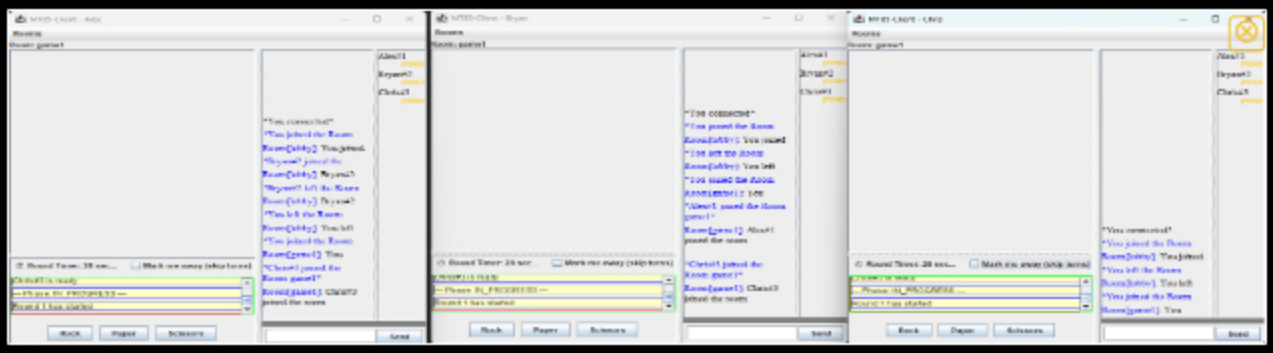
- UI should have components to allow the user to select their choice

Part 1:

Progress: 100%

Details:

- Show various examples of selections across clients (3+ clients visible)
- Show the code related to sending choices upon selection
- Show the code related to showing visually what was selected



selections across clients

```java
@Override
public void onMessageReceive(long id, String message) {
    if (id == Constants.GAME_EVENT_CHANNEL) {
        addText(message);
    }
}
```

Show the code related to showing visually what was selected

```java
private void onPickChoice(String choice) {
    try {
        if (isEliminated) {
            LoggerUtil.INSTANCE.warning("Cannot pick - player is eliminated");
            return;
        }

        LoggerUtil.INSTANCE.info("Player chose: " + choice.toUpperCase());
        if (cooldownEnabled && lastLocalChoice != null && lastLocalChoice.equals(choice)) {
            LoggerUtil.INSTANCE.warning("Choice on cooldown: " + choice);
            return;
        }

        Client.INSTANCE.sendPick(choice);
        lastLocalChoice = choice;
    } catch (IOException e) {
        LoggerUtil.INSTANCE.severe("Error sending pick", e);
    }
}
```

code related to sending choices upon selection

Saved: 12/11/2025 6:56:27 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code flow for selecting a choice and having it reach the server-side
- Briefly explain the code flow for receiving the selection for the current player to update the UI

Your Response:

1.When a player clicks a choice button, GameEventsView.onPickChoice() validates the selection and calls Client.INSTANCE.sendPick(). The client creates a PLAYER_PICK payload containing the choice string and sends it to the server, where GameRoom.handlePlayerPick() receives it, stores

the choice on the player's ServerThread object, and broadcasts a selection message to all clients. 2.When the server processes a pick, it calls syncTurnStatus() which sends a ReadyPayload with tookTurn=true to all clients. The client's Client.processTurn() receives this payload and triggers UserListView.onTookTurn(), which clears the pending badge for that player by calling setPendingPick(false) on their UserListItem.

Saved: 12/11/2025 6:56:27 PM

Section #3: (4 pts.) Project Extra Features

Progress: 100%

Task #1 (2 pts.) - Extra Choices

Progress: 100%

Details:

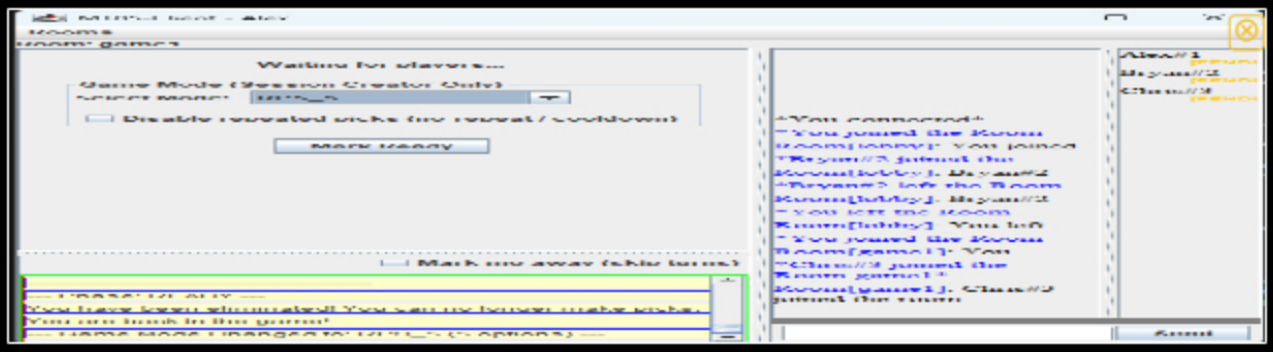
- Setting should be toggleable during Ready Check by session creator
  - (Option 1) Extra choices are available during the full session
  - (Option 2) Only activate extra options at different stages (i.e., last 3 players remaining)
- There should be at least 2 extra options for rps-5

Part 1:

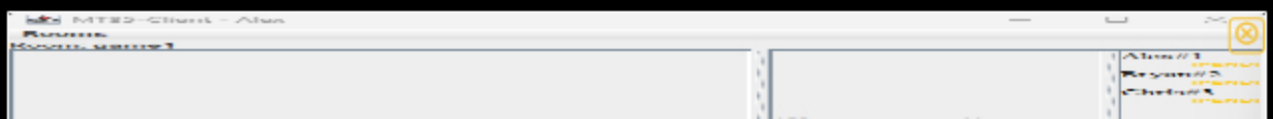
Progress: 100%

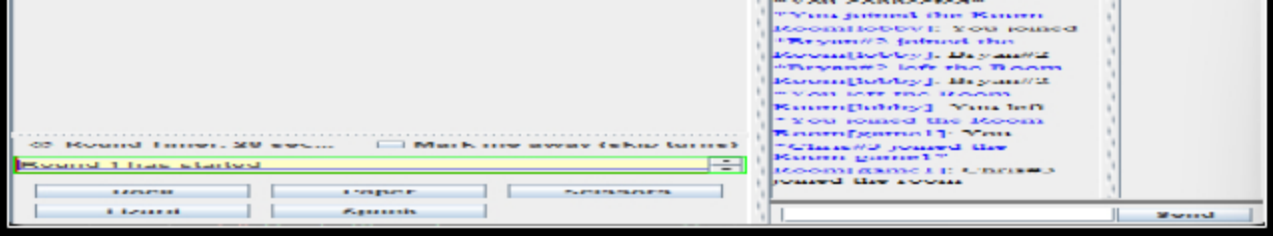
Details:

- Show the Ready Check screen with the option for the host (3+ clients must be visible)
  - Show the related code that makes this interactable only for the host
- Show the play screen with the extra options available
  - Show the related code for the UI and handling of these extra options (including battle logic)

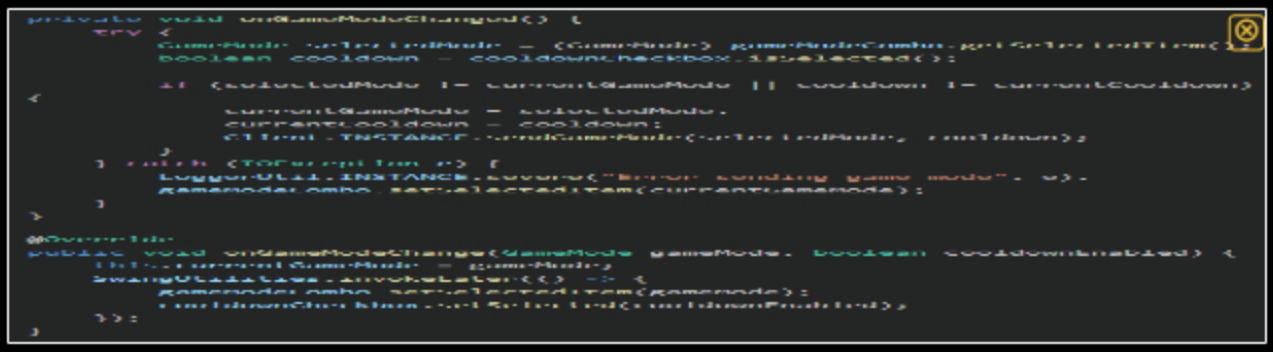


Ready Check screen with the option



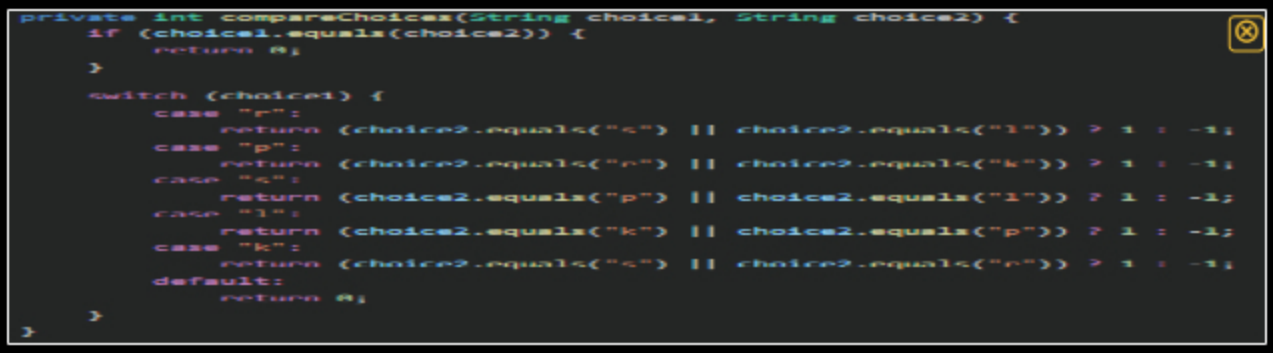


extra options



code that makes this interactable only for the host

UI and handling of these extra options



battle logic

Saved: 12/11/2025 7:11:27 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code for the host's option to toggle this feature
- Briefly explain the code related to handling these options including how it's handled during

the battle logic
- Note which option you went with in terms of activating the choices

Your Response:

1.The host uses a dropdown in ReadyView.java to select between RPS-3 and RPS-5. When the selection changes, onGameModeChanged() sends the chosen mode to the server via Client.INSTANCE.sendGameMode(), and the server broadcasts this to all clients so their UI syncs automatically. Only the host can initiate the change, and all other clients' dropdowns update to match the host's selection. 2.When a game mode is selected, regenerateButtons() in GameEventsView.java dynamically creates buttons based on gameMode.getDisplays() - for RPS-5, generates 5 buttons. The battle logic in GameRoom.compareChoices() uses a switch statement where each choice checks if it beats the opponent's choice. This single method handles both RPS-3 and RPS-5.

Saved: 12/11/2025 7:11:27 PM

Task #2 (2 pts.) - Choice cooldown

Progress: 100%

Details:

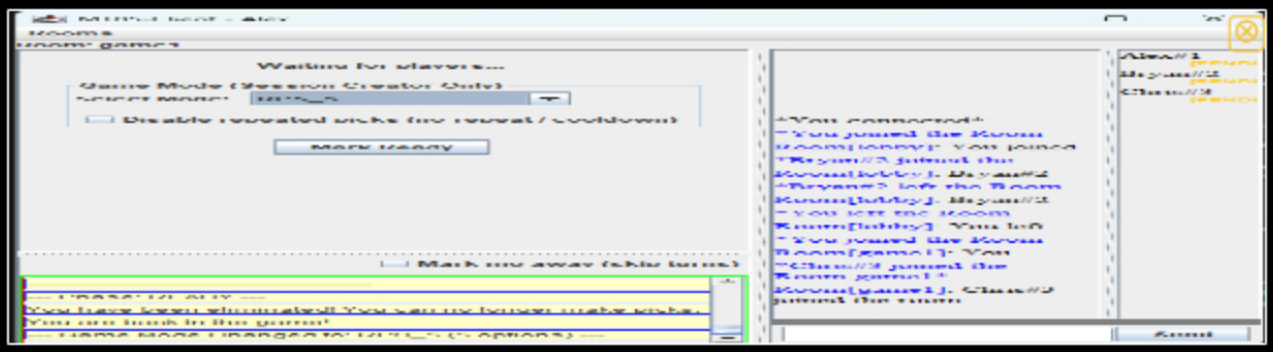
- Setting should be toggleable during Ready Check by session creator
- The choice on cooldown must be disable on the UI for the User

Part 1:

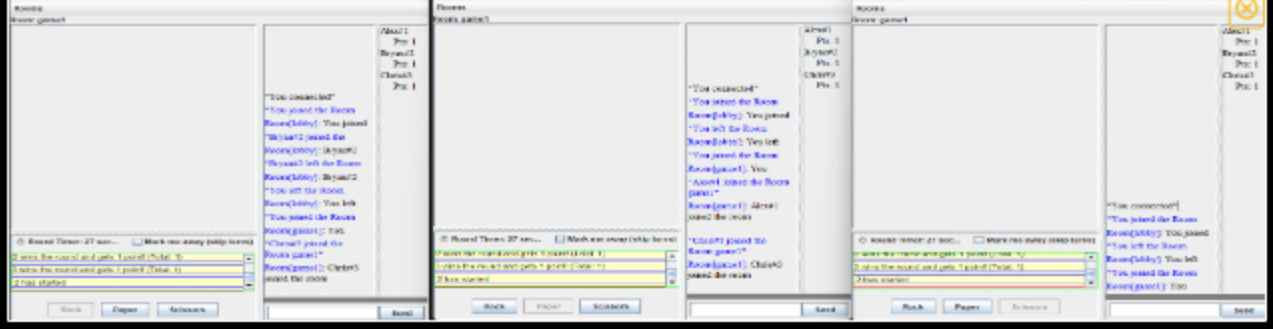
Progress: 100%

Details:

- Show the Ready Check screen with the option for the host (3+ clients must be visible)
  - Show the related code that makes this interactable only for the host
- Show a few examples of the play screen with the choice on cooldown
  - Show the related code for the UI and handling of the cooldown and server-side enforcing it



Ready Check screen with the option



cooldown

```
private void onGameModeChanged() {
    try {
        GameMode selectedMode = (GameMode) gameModeCombo.getSelectedItem();
        boolean cooldown = cooldownCheckbox.isSelected();

        if (selectedMode != currentGameMode || cooldown != currentCooldown)
{
            currentGameMode = selectedMode;
            currentCooldown = cooldown;
            Client.INSTANCE.sendGameMode(selectedMode, cooldown);
        }
    } catch (IOException e) {
        LoggerUtil.INSTANCE.severe("Error sending game mode", e);
        gameModeCombo.setSelectedItem(currentGameMode);
    }
}
```

cooldown code

```
private void updateChoiceButtons(GameMode gameMode) {
    buttonPanel.removeAll();
    String[] displays = gameMode.getDisplays();
    String[] choices = gameMode.getChoices();

    for (int i = 0; i < displays.length; i++) {
        String display = displays[i];
        String choice = choices[i];

        JButton choiceButton = new JButton(display);
        choiceButton.addActionListener(_ -> onPickChoice(choice));

        if (cooldownEnabled && lastLocalChoice != null &&
lastLocalChoice.equals(choice)) {
            choiceButton.setEnabled(false);
        }

        choiceButtonMap.put(choice, choiceButton);
        buttonPanel.add(choiceButton);
    }

    buttonPanel.revalidate();
    buttonPanel.repaint();
}
```

ui code

```
        if (cooldownEnabled && lastLocalChoice != null &&
lastLocalChoice.equals(choice)) {
            LoggerUtil.INSTANCE.warning("Choice on cooldown: " + choice);
            return;
        }
```

server side

Saved: 12/11/2025 7:19:08 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code for the host's option to toggle this feature
- Briefly explain the code related to handling and enforcing the cooldown period (include how this is recorded per user and reset when applicable)

Your Response:

1.The host uses a checkbox in ReadyView.java that sends the cooldown setting to the server via Client.INSTANCE.sendGameMode(). The server stores it in cooldownEnabled and broadcasts to all clients so their UI syncs. 2.Each player's last choice is stored in lastLocalChoice, and when cooldown is enabled, regenerateButtons() disables the button matching that choice for the next round. The lastLocalChoice resets to null when a new game starts (Phase.READY), allowing all choices again.

Saved: 12/11/2025 7:19:08 PM

Section #4: (2 pts.) Project General Requirements

Progress: 100%

Task #1 (1 pt.) - Away Status

Progress: 100%

Details:

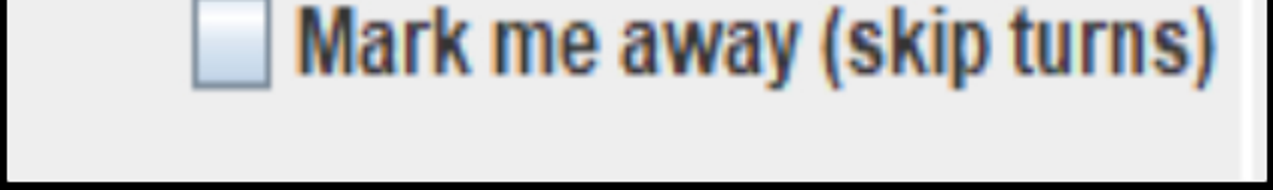
- Clients can mark themselves away and be skipped in turn flow but still part of the game
- The status should be visible to all participants
- A message should be relayed to the Game Events Panel (i.e., Bob is away or Bob is no longer away)
- The user list should have a visual representation (i.e., grayed out or similar)

Part 1:

Progress: 100%

Details:

- Show the UI button to toggle away
- Show the related code flow from UI to server-side back to UI for showing the status
- Show the related code flow for sending the message to Game Events Panel
- Show various examples across 3+ clients of away status (including Game Events Panel messages)
- Show the code that ignores an away user from turn/round logic



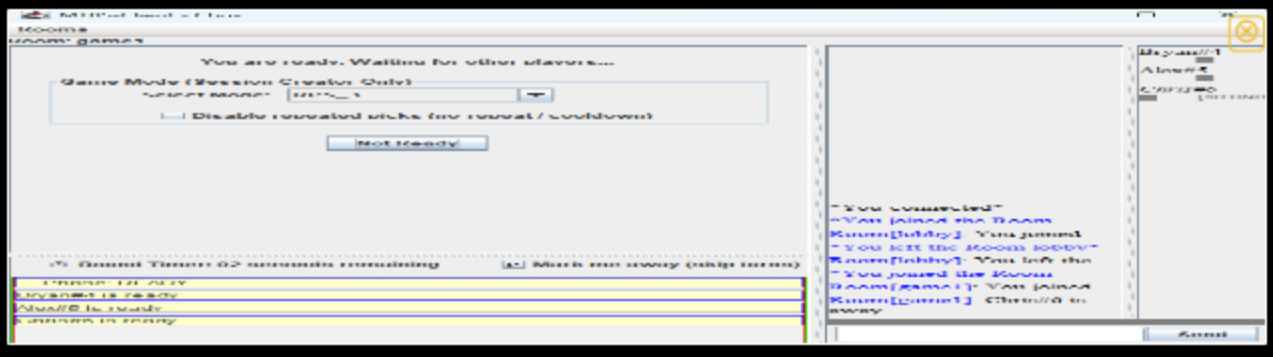
UI button to toggle away

```
protected synchronized void handleAway(ServerThread sender,
Project.Common.AwayPayload payload) {
    try {
        sender.setAway(payload.isAway());

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

        String display = sender.getDisplayName();
        String msg = String.format("%s is %s", display, payload.isAway() ?
"away" : "no longer away");
        relay(null, msg);
    } catch (Exception e) {
        LoggerUtil.INSTANCE.severe("handleAway exception", e);
    }
}
```

code

```
String display = sender.getDisplayName();
String msg = String.format("%s is %s", display, payload.isAway() ? "away" :
"no longer away");
relay(null, msg);
```

code flow for sending the message to Game Events



clients of away status

```
List<ServerThread> activePlayers = clientsInRoom.values().stream()
        .filter(p -> !p.isEliminated() && !p.isAway())
        .collect(Collectors.toList());
```

code that ignores an away user

Saved: 12/11/2025 7:29:55 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code flow for the away action from UI to server-side and back to UI
- Briefly explain how the server-side ignores the user from turn/round logic

Your Response:

1.Player clicks away checkbox, sends AwayPayload to server via Client.sendAway(), server updates status and broadcasts to all clients. All clients update their UI with gray italic text, SITTING OUT badge, and display the away message in Game Events panel. 2.The server uses .filter(!p.isAway()) in stream operations to exclude away players from active gameplay. Away players are skipped when counting ready players, determining active players for battles, and selecting session winners.

Saved: 12/11/2025 7:29:55 PM

Task #2 (1 pt.) - Spectators

Progress: 100%

Details:

- Spectators are users who didn't mark themselves ready
  - Optionally you can include a toggle on the Ready Check page
- The can see all chat but are ignored from turn/round actions and can't send messages
- Spectators will have a visual representation in the user list to distinguish them from other players
- A message should be relayed to the Game Events Panel that a spectator joined (i.e., during an in-progress session)

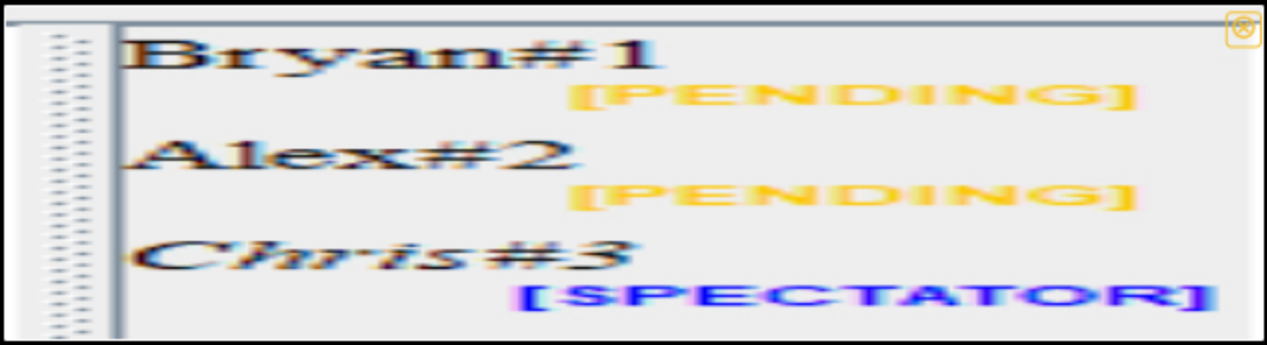
Part 1:

Progress: 100%

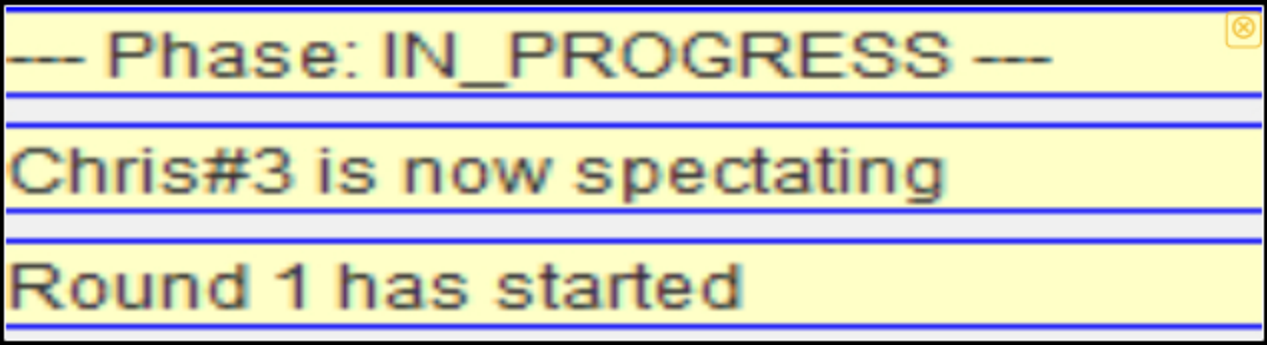
Details:

- Show the UI indicator of a spectator (visual and message)
- Show the related code flow from UI to server-side back to UI for showing the status
- Show the related code flow for sending the message to Game Events Panel
- Show various examples across 3+ clients of spectator status (including Game Events Panel messages)
- Show the code that ignores a spectator from turn/round logic
- Show the code that prevents spectators from sending messages (server-side)
- Show the spectator's view of the session

- Show the code related to the spectator seeing the session data (including things participants won't see)



UI indicator of a spectator



UI indicator of a spectator

```
protected void onSessionStart() {
    clientsInRoom.values().forEach(p -> {
        if (!p.isReady()) {
            p.setSpectator(true);

            Project.Common.ConnectionPayload cp = new
Project.Common.ConnectionPayload();
            cp.setClientId(p.getClientId());
            cp.setClientName(p.getDisplayName());
            cp.setSpectator(true);
            cp.setPayloadType(Project.Common.PayloadType.SYNC_CLIENT);
            clientsInRoom.values().forEach(client ->
client.sendToClient(cp));

            sendGameEvent(String.format("%s is now spectating",
p.getDisplayName()));
        }
    });
}
```

UI to server-side back to UI

```
sendGameEvent(String.format("%s is now spectating", p.getDisplayName()));

private void sendGameEvent(String message) {
    sendMessage(Constants.GAME_EVENT_CHANNEL, message);
}
```

Game events

```
List<ServerThread> activePlayers = clientsInRoom.values().stream()
        .filter(p -> !p.isEliminated() && !p.isAway() && !p.isSpectator())
```

```
.filter(p -> !p.isEliminated() && !p.isAway() && !p.isSpectator())
.collect(Collectors.toList());
```

code showing spectator is ignored

Saved: 12/11/2025 9:05:54 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code flow for the spectator logic from server-side and to UI
- Briefly explain how the server-side ignores the user from turn/round logic
- Briefly explain the logic that prevents spectators from sending a message
- Briefly explain the logic that shares extra details to the spectator (information normal participants won't see)

Your Response:

1. When game starts, server marks non-ready players as spectators, broadcasts ConnectionPayload with isSpectator=true to all clients, and sends Game Events message. 2.Server Ignoring Spectators from Game Logic: Server uses .filter(p -> !p.isSpectator()) in stream operations when collecting active players for battles, counting ready players, and determining session winners. Spectators remain in the room but are completely excluded from all game logic calculations. 3.Server's Room.handleMessage() checks if sender.isSpectator() and immediately returns with error message "Spectators cannot send messages" before processing the message. 4.Spectators receive the exact same payloads as participants so they see all session data including battle results and player stats.

Saved: 12/11/2025 9:05:54 PM

Section #5: (1 pt.) Misc

Progress: 100%

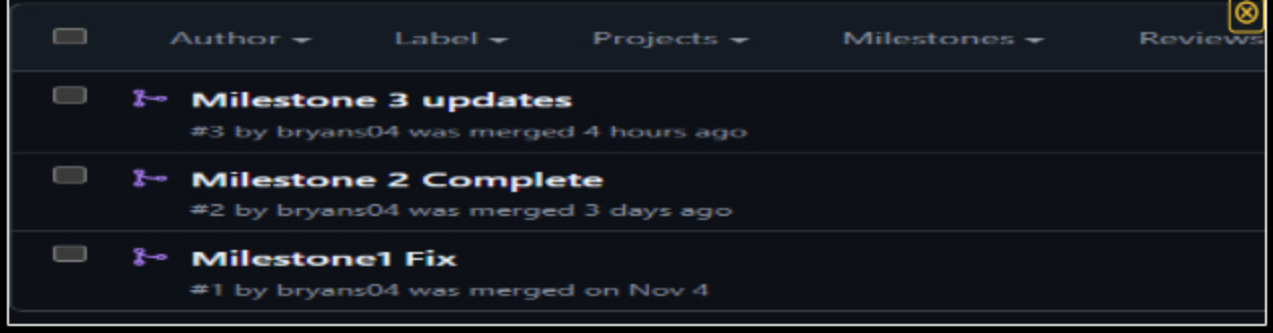
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

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



commit history - I also made a separate commit from main as I had to switch computers mid way but forgot to switch back branches

Saved: 12/11/2025 8:57:00 PM

Part 2:

Progress: 100%

Details:
Include the link to the Pull Request for Milestone3 to main (should end in `/pull/#`)

URL #1
https://github.com/bryans04/it114-project/pull/3

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

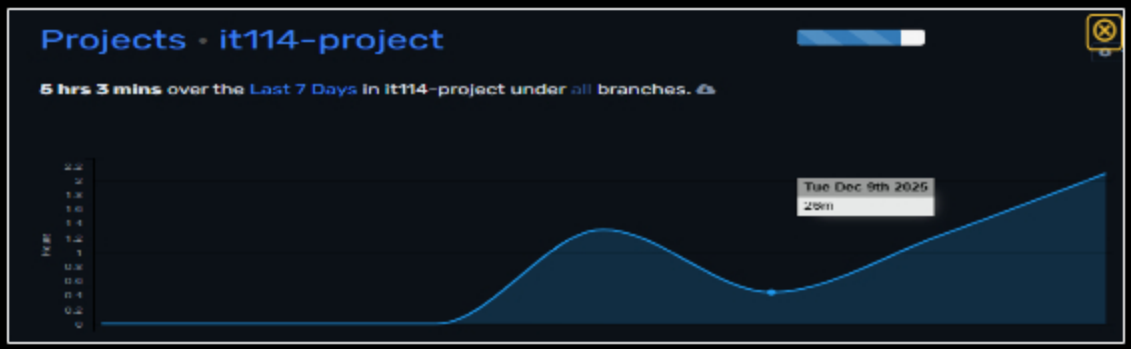
Saved: 12/11/2025 8:57:00 PM

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



waka time

Saved: 12/11/2025 8:43:14 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:

This was probably one of the most difficult projects I have worked on. Nothing was going right for me but, it did teach me a lot. For example I have never worked with commands before so getting to set my own up was actually very fun.Also setting up the UI and seeing my project come to life was my favortite part of this project.

Saved: 12/11/2025 8:23:22 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 round robin sounded the most intimidating to me but it turned out to be one of the most easiest. Even for the 5 option game, it was very straight foward

Saved: 12/11/2025 8:25:03 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:

Setting up the spectator was the most difficult part for me. I wasbt fully able to set it up right away for example the buttons kept showing up for the spectator then, it still counted the spectator as a player and it was just back and forth trying to set it up. Which in the end I just couldnt figure out how to fully implement it.

Saved: 12/11/2025 8:57:40 PM