Submission Worksheet

Submission Data

Course: IT114-450-M2025
Assignment: IT114 - Milestone 3 - 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 11:29:58 AM
Updated: 7/28/2025 8:09:23 PM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-3-rps/grading/gb373
View Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-3-rps/view/gb373

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

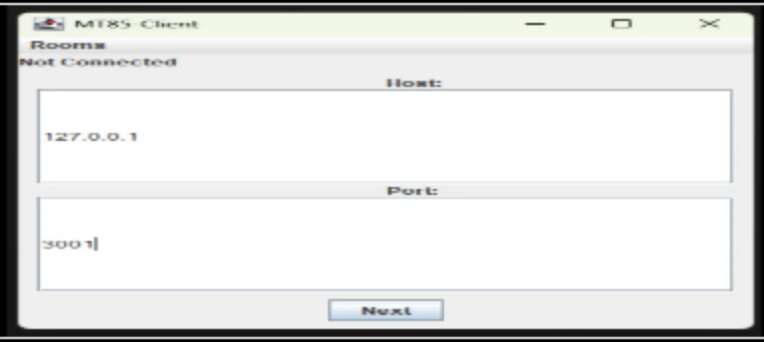
Progress: 100%

Part 1:

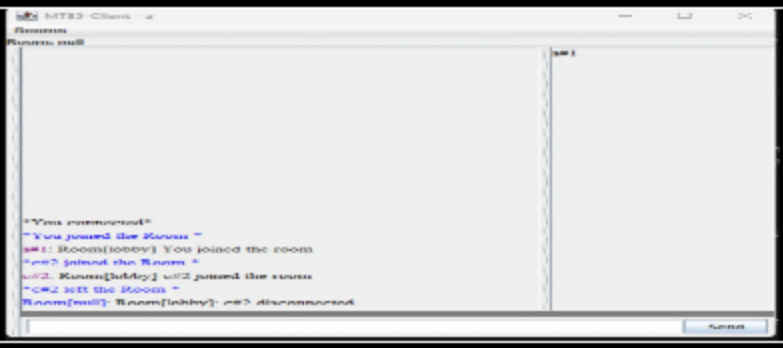
Progress: 100%

Details:

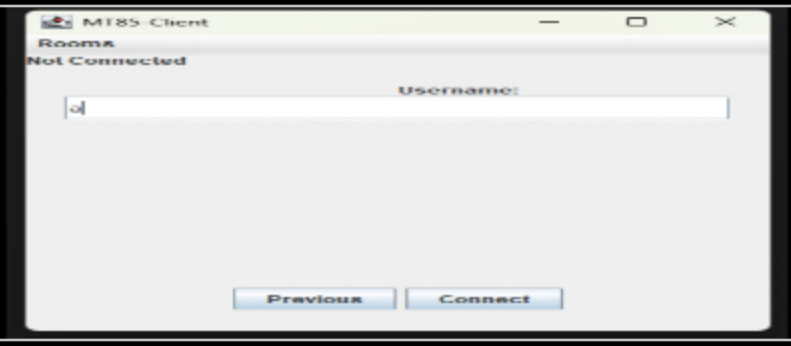
- Show the connection panel with valid data
- Show the user details panel with valid data



Connection Panel showing Port and Host



User Panel showing user a has join and is connected to the server



Panel for not connected clients making a username

Saved: 7/22/2025 12:01:32 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:

First, when the client is connecting, it first sees a panel showing that it needs to connect to a

certain host and port. This is the server thread being created, and it is displayed in the UI with the current view. After that, the client must enter a name for the server. This is what the payload is designed for, allowing a client to request joining using a specific name. This is then shown in the UI by clientui receiving a client ID, changing the view to the game screen, and adding it to the list view. After that, the user is connected to the first room, where it says it has been connected to the server, and the client is able to see the main user panel. This shows the user list and game screen to show that the user is connected to the server.

Saved: 7/22/2025 12:01:32 PM

Task #2 (0.50 pts.) - Ready Panel

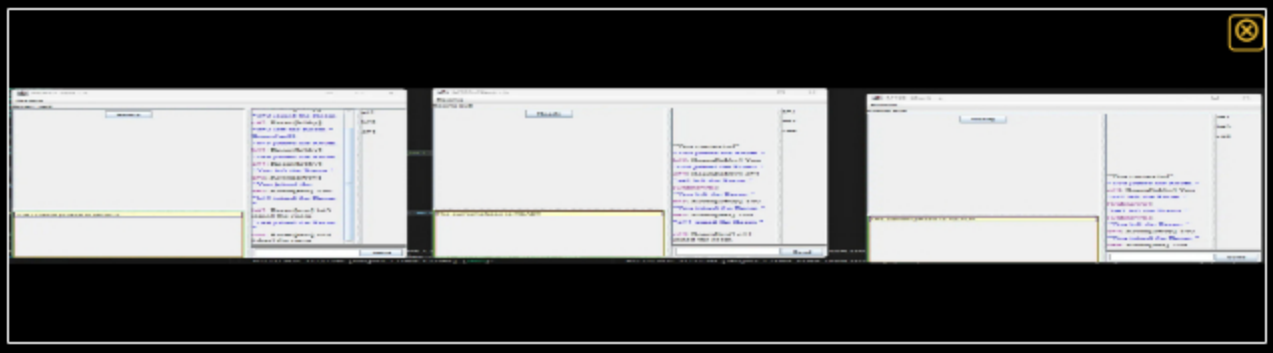
Progress: 100%

Part 1:

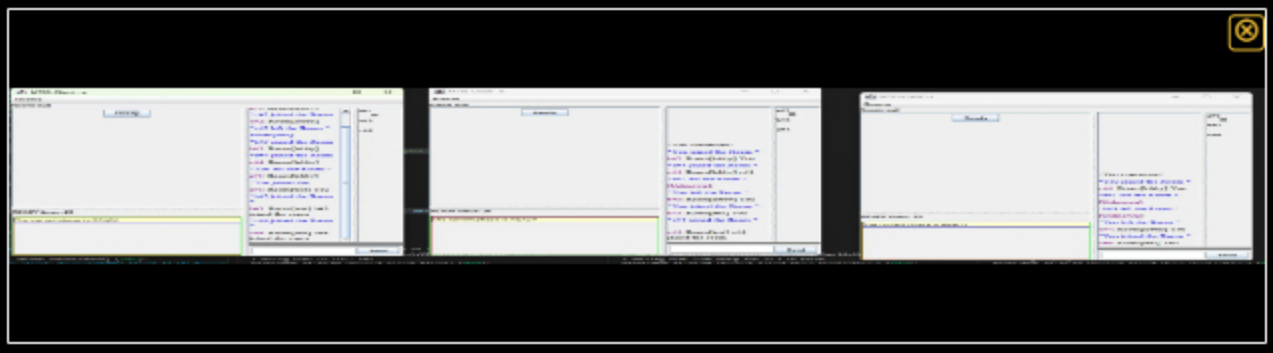
Progress: 100%

Details:

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

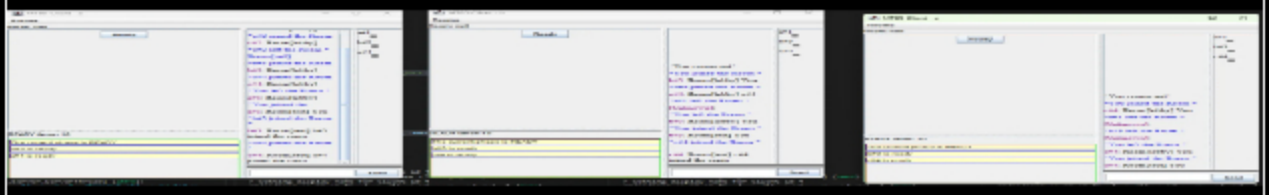


All users in room none are ready but ready button is therr



One user is ready

Two users are ready



All users are marked as ready

Saved: 7/22/2025 12:25:40 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:

The flow for ready from the ui is when a client joins or creates a room, they are considered not ready by making the item ready to false in the list view. For marking ready, the user clicks the ready button which is in the chat game view. It will then send a payload saying it's ready and also get the display name for the client ID that readied up and change their item ready to true, marking that they are ready in the ui. For code flow from receiving READY data and updating the UI, it first recognizes the type of payload being READY. The server then makes the game room control the actions that manage the game logic, like starting the ready timer, and this is shown in the UI with the help of the server thread. The UI will also display to the other clients that a user is ready, and they will also see the ready timer if they are in the same room.

Saved: 7/22/2025 12:25:40 PM

Section #2: (2 pts.) Project Ui

Progress: 100%

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

Progress: 100%

Details:

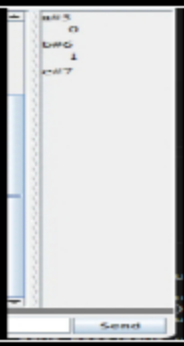
- Show the username and id of each user
- Show the current points of each user
- Users should appear in score order, sub-sort by name when ties occur
- Pending-to-pick users should be marked accordingly
- Eliminated users should be marked accordingly

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



score output example 1

score output after end of game

score output example 2 after game

```
// UCID: gh371
// Date: 07/22/2025
// Summary: Sends points to the client, which can be used for scoring or other purposes.
public Boolean sendPoints(long clientId, int points) {
    PointsPayload pp = new PointsPayload();
    pp.setPayloadType(PayloadType.POINTS);
    pp.setClientId(clientId);
    pp.setPoints(points);
    return sendToClient(pp);
} x   815A 182 public boolean sendPoints(long clientId, int points)
```

code for sending points in server

```
//UCID: gh371
// Date: 07/22/2025
// Summary: Closes the server connection and cleans up resources. Sends points update to UI.
// Not process*() methods
private void processPoints(Payload payload) {
    if (!(payload instanceof PointsPayload)) {
        error(message:"Invalid payload subclass for processPoints");
        return;
    }
    PointsPayload pp = (PointsPayload) payload;
    long targetId = pp.getClientId();
    int points = pp.getPoints();
    if (targetId == Constants.DEFAULT_CLIENT_ID) {
        // error 811
        knownClients.values().forEach(cp -> cp.setPoints(-1));

        passToUiCallback(new PointsEvent(this, p -> p.onPointsUpdate(Constants.DEFAULT_CLIENT_ID, -1)));
    } else if (knownClients.containsKey(targetId)) {
        knownClients.get(targetId).setPoints(points);

        passToUiCallback(new PointsEvent(this, p -> p.onPointsUpdate(targetId, points)));
    } // 8085-087 else if (knownClients.containsKey(targetId))
} x   8580 888 private void processPoints(Payload payload)
```

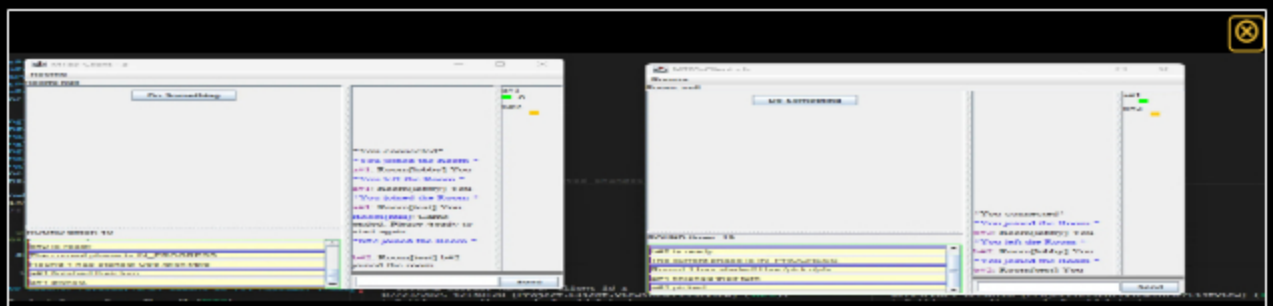
code for client processing points

```
// UCID: gh371
// Date: 07/22/2025
// Summary: Updates the points for a user in the user list.
// This method is called when the server sends a points update.
@Override
public void onPointsUpdate(long clientId, int points) {
    if (clientId == Constants.DEFAULT_CLIENT_ID) {
        SwingUtilities.invokeLater(() -> {
            try {
                userItemsMap.values().forEach(u -> u.setPoints( 1));// reset all
            } catch (Exception e) {
                LoggerUtil.INSTANCE.severe(message:"Error resetting user items", e);
            }
        }); <- #172-179 SwingUtilities.invokeLater
    } else if (userItemsMap.containsKey(clientId)) {
        SwingUtilities.invokeLater(() -> {
            try {
                userItemsMap.get(clientId).setPoints(points);
            } catch (Exception e) {
                LoggerUtil.INSTANCE.severe(message:"Error setting user item", e);
            }
        }); <- #193-199 SwingUtilities.invokeLater
    } <- #190-192 else if (userItemsMap.containsKey(clientId))
} x   8171 188 public void onPointsUpdate(long clientId, int points)
```

code for UI getting points and updating them for the user

Output for points sorted

code for sorting players by points and name if there is a tie.



Players pending to make a pick are marked with an orange square

Code for sending the pending status from the server side to the client

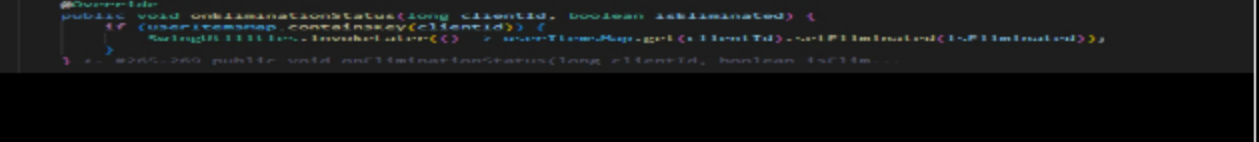
Client side case for pending pick and sending it to UI

UI dealing with the pending updates and picks actions

Players are mark as eliminated with a red square

Code for sending the eliminated status from the server side to the client

Client side case for players eliminated and sending it to UI



UI dealing with the elimination status and updateding it for users

Saved: 7/23/2025 1:53:22 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:

For the code flow of points being updated, the server will use sendPoints to be called when points change for a specific player, and it will send the information to the client side. The client will then process the point change with processPoints. Finally, the ui will use onPointsUpdate to update the points in the list view for all the clients, so every user can see the point change when a player gains a point.

For the code flow of sorting players by points or by names if there is a tie, I made sortUserList which reatreves the items from UserListItem and sorts the list in descending order based on points and sorts alphabetically if the points are equal. Then it repaints the list and adds it showing the correct sorted list.

For the code flow of pending picks being marked, on the server side, the server sends the status of a pending pick, saying this player's pick is pending, and sends it over to the client in a Pending payload. Then, the client has a case for that pending pick payload, which handles if a pick is pending, and if it is, it sends it over to the UI, which the UI then handles the pending pick action by marking them as pending pick with an orange square.

For the code flow of marking a player that is eliminated, on the server side, the server sends the status whether a player is eliminated, and it sends it over to the client in an Eliminated payload. Then the client has a case for the eliminated payload, which does kinda the same as the pending pick one, but is for a player that is eliminated, and it will then send the updates to the UI. The UI does the same as the pending pick and handles the elimination update by marking eliminated players with a red square, and it also makes sure that it doesn't mark them with a pending pick square because eliminated players can't make a pick.

Saved: 7/23/2025 1:53:22 PM

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

Progress: 100%

Details:

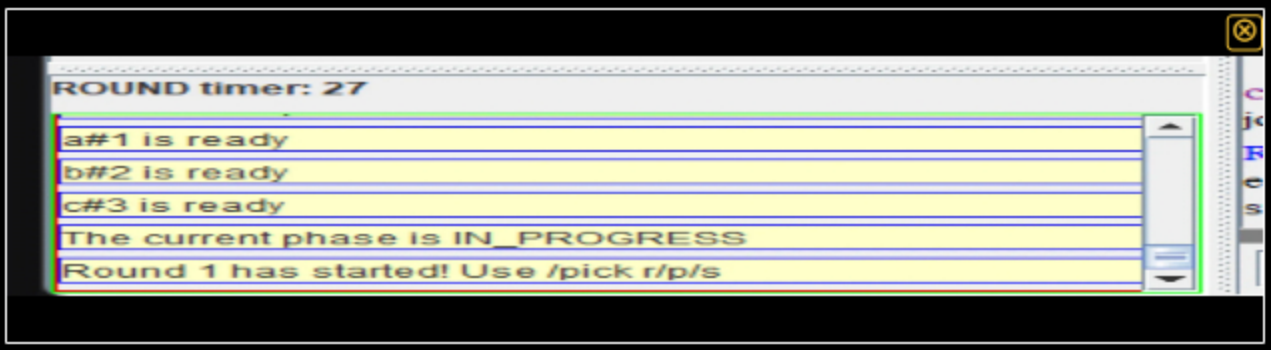
- Show the status of users picking choices
- Show the battle resolution messages from Milestone 2
  - Include messages about elimination
- Show the countdown timer for the round

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

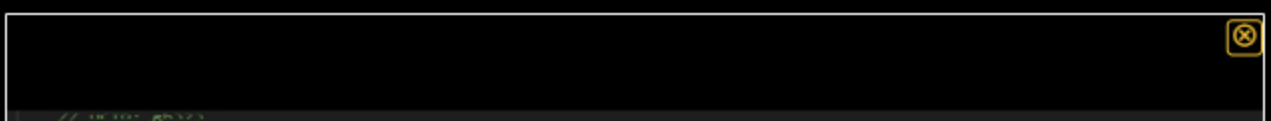


Showing round timer and ready messages

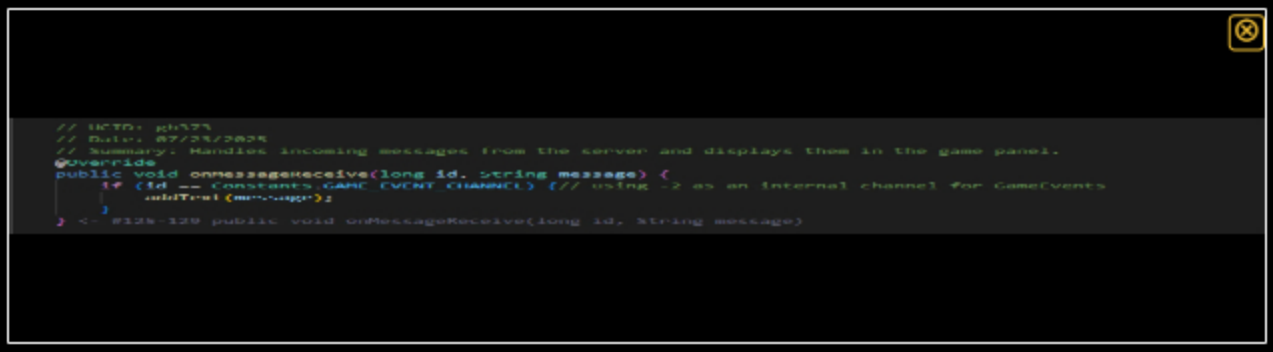
Showimg some more messages in the game panel

Example of messages going to the game panel on the server side.

Server sending game events to the client side



Client processing messages from the server and sending it to the UI



UI handles the games panal messages here and displays them to all clients

Saved: 7/23/2025 2:33:32 PM

Part 2:

Progress: 100%

Details:

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

Your Response:

The code flow on how messages are sent into the game panel is that at the start, the server side will send game events that include all the messages that will go into it, and send it to the client in a payload. That payload is used by the client, and also the client uses processGameEvent to handle all game events and game panel messages, and sends it to the UI, where the UI handles the received messages and displays them with addText into the game panel at the bottom of the UI.

Saved: 7/23/2025 2:33:32 PM

Task #3 (0.67 pts.) - Game Area

Progress: 100%

Details:

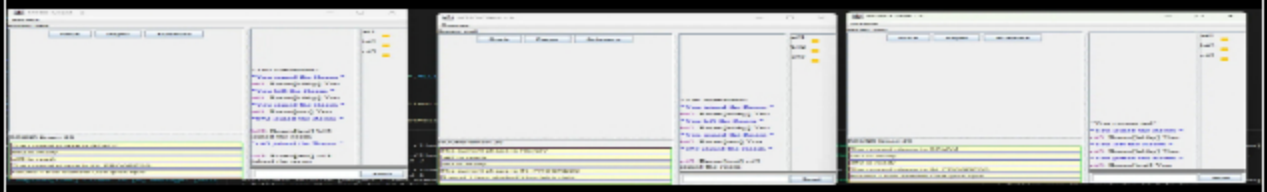
- UI should have components to allow the user to select their choice

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



Buttons for choosing RPS when in game

2 players picked so the buttons disappeared

Showing at the end that when a player clicks a button it makes that choice

Code for sending the choices and also the visual of the buttons

Code for sending the choices and also the visual of the buttons

Saved: 7/23/2025 4:50:35 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:

The code flow for selecting a choice and having it reach the server-side when a user clicks one of the buttons it uses sendPick with the choice that corresponds with the button and sends that pick to the server for the user's turn. For the code flow for receiving the selection for the current player to update the UI, the client receives the server's updates about the state change from the payload. Then, when the picks start, the phase changes, and it shows the 3 buttons to pick. Once a user has picked a button, the buttons will disappear, meaning they have selected a choice that is sent ot the server.

Saved: 7/23/2025 4:50:35 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:

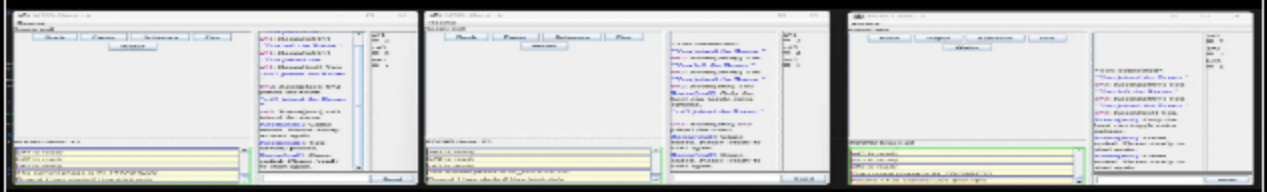
Progress: 100%

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

- Show the related code for the UI and handling of these extra options (including battle logic)

shwoing old version still works



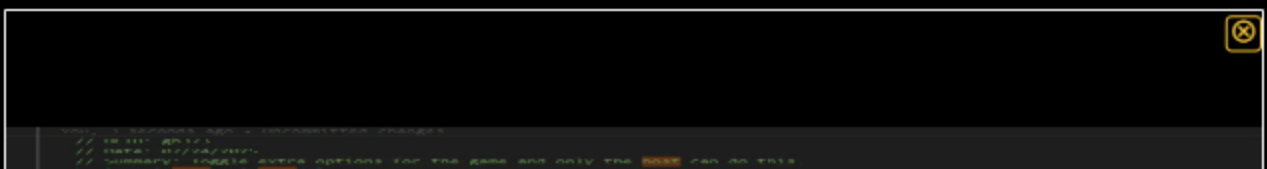
in game showing extra options

showing button can only be activated by a who is the host

```
    You, 1 second ago • Uncommitted changes
    // UCID: gba73
    // Date: 07/24/2025
    // Summary: Setting the host status for the client.
    boolean isHost = clientsInRoom.size() == 1;

    Payload hostPayload = new Payload();
    hostPayload.setPayloadType(PayloadType.HOST_STATUS);
    hostPayload.setMessage(Boolean.toString(isHost));
    sp.sendToClient(hostPayload);
} <- #45-62 protected void onClientAdded(ServerThread sp)
```

code for setting the host when the client is first to join



code for saying only the host can toggle the extra option

code for buttons being visible during the extra option being enabled

code for the cases of extra options toggled and when it's enabled

battle logic of how to win and old version too

Code for the case for clitn to update them on the status of if they are the host or not

Saved: 7/24/2025 8:39:32 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:

I went with (Option 1). Extra choices are available during the full session, which is available every time there is a ready check. The host's options are that they are the only ones allowed to toggle the extra options. The host is the first one to join the room, and is the host, retaining that status. hee UI will send a message to any non-host who tries to toggle the extra option, saying only the host can do this. The client uses a case for Host status to update if they are the host. The code related to the options during the battle logic is the same as how it is for the old RPS, where there is logic on who wins against what. I added so that one can beat two different picks, for example, the Rock beats Scissors and Fire, so this makes sure that there is more excitement in the ways where there are possibilities you can win or lose. The way it works with code-wise is the same, and these buttons and picks will only work if the extra option is enabled.

Saved: 7/24/2025 8:39:32 PM

Task #2 (2 pts.) - Choice cooldown

Progress: 100%

Details:

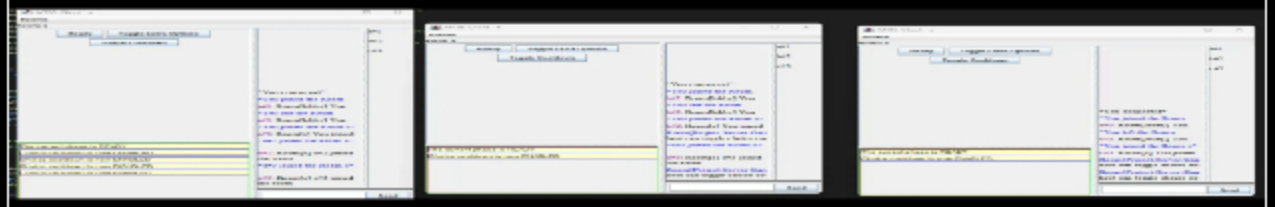
- Setting should be toggleable during Ready Check by session creator
- The choice on cooldown must be disable on the UI for the User

Part 1:

Progress: 100%

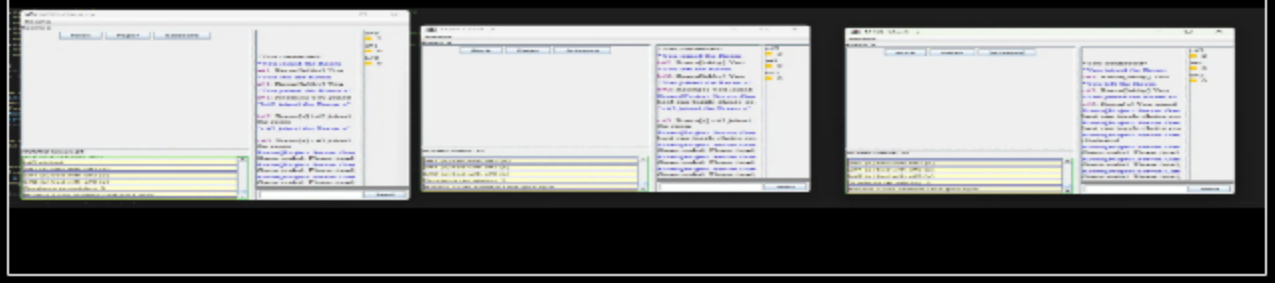
Details:

- 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

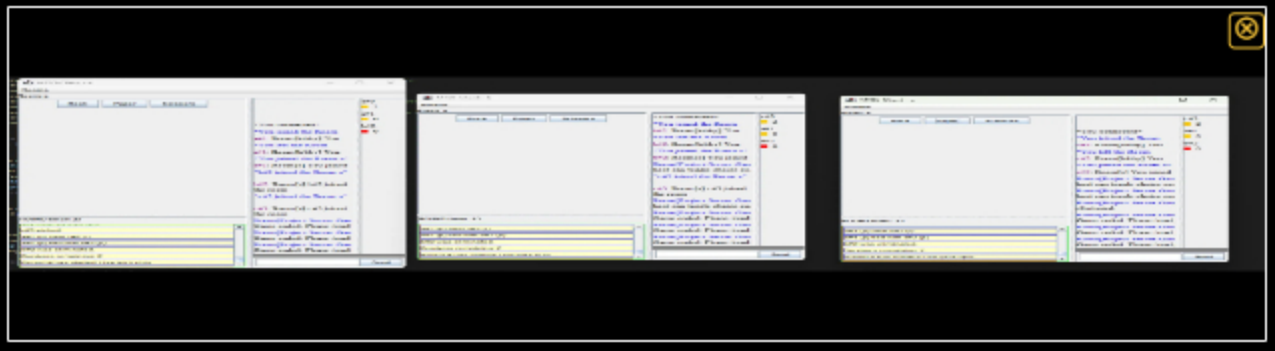


output showing that only the host can activate cooldown toggle

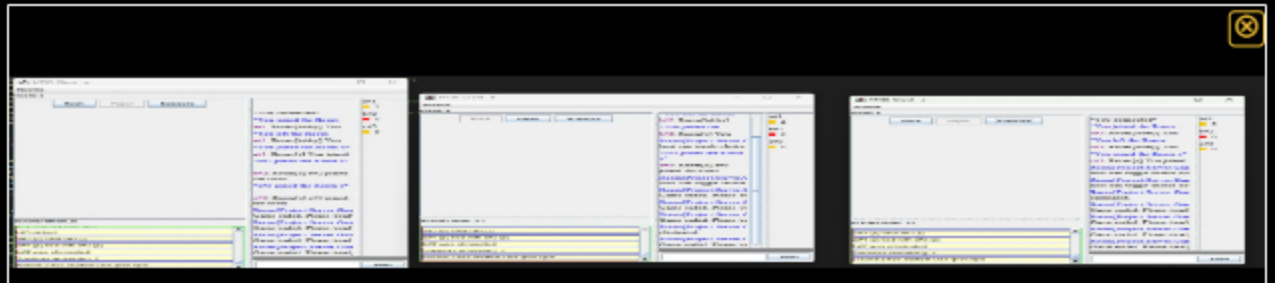
code for only making sure that the host can toggle the cooldown and server side enforcing it



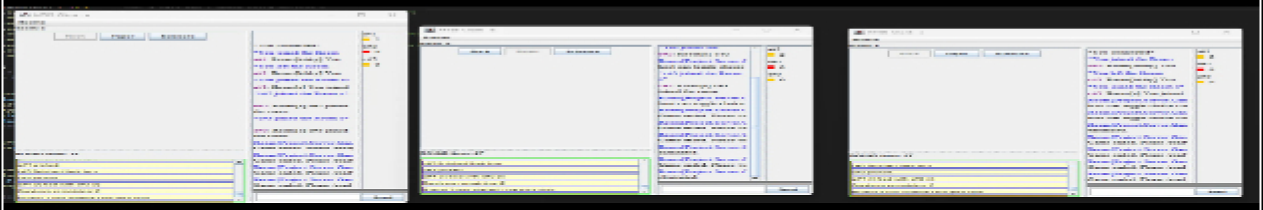
DIsable Example 1



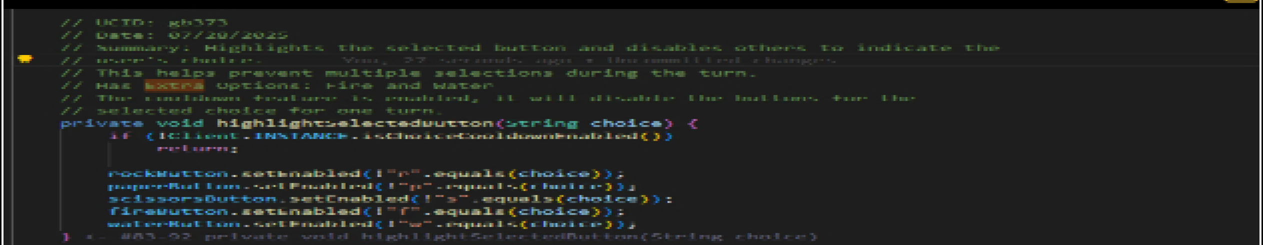
Disabled Example 2



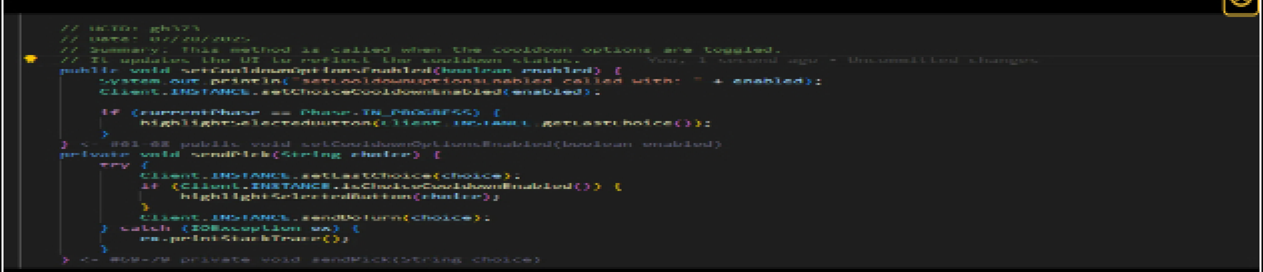
Enabled Example 1



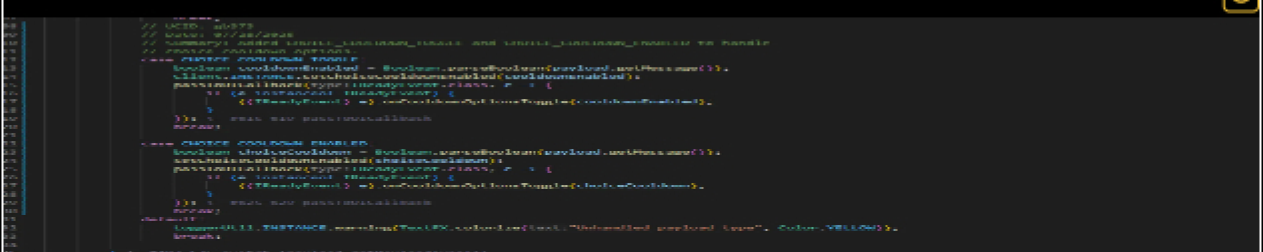
Enabled Example 2



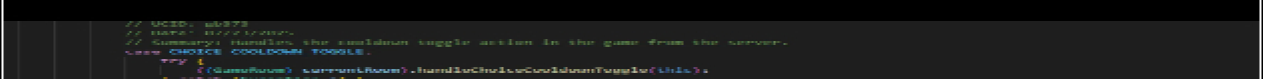
Code for UI making sure that when enabled, player cant choose the same pick twice.



UI code for it setting enabled for the toggle of cooldown and what it does



Client dealing with the two cooldown cases



Code from server handiling the cool down toggle action in the game

Saved: 7/28/2025 1:53:49 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:

The code for host toggle is as follows: when the host is determined, which is the first to join the server, they are marked as the host and can only toggle certain buttons available. On the server side, the game room handles the cooldown toggle to see if the are the host using it is the host, and if they are, they are allowed to activate the cooldown toggle. On the Client side, them being able to set the enable on the cooldown toggle can only be done by the host. The code related to enforcing the cooldown is simple because what it does is the server will use the only host for the toggle enable, but it mainly happens in the ui where the UI uses a highlightSelected button and disables the button that was last clicked, and this is looked at by your send pick. And this resets after the next round, which means that the logic is the player will not be able to choose the same pick twice in a row because the last pick is disabled by the UI.

Saved: 7/28/2025 1:53:49 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%

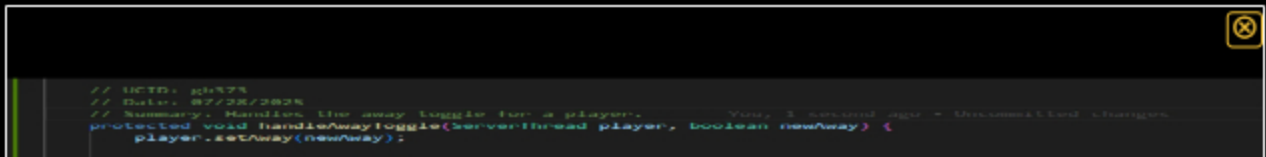
Details:

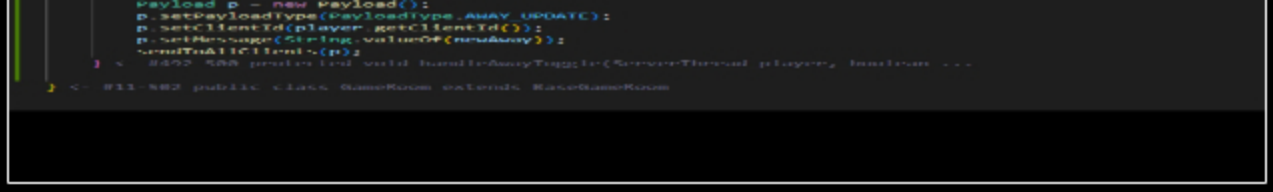
- Show the UI button to toggle away
- Show the related code flow from UI to server-side back to UI for showing the status
- Show the related code flow for sending the message to Game Events Panel
- Show various examples across 3+ clients of away status (including Game Events Panel messages)
- Show the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

mark away button with a being away as an example

Client code handiling various parts of the away toggle and status

code for showing gray over a user who is away

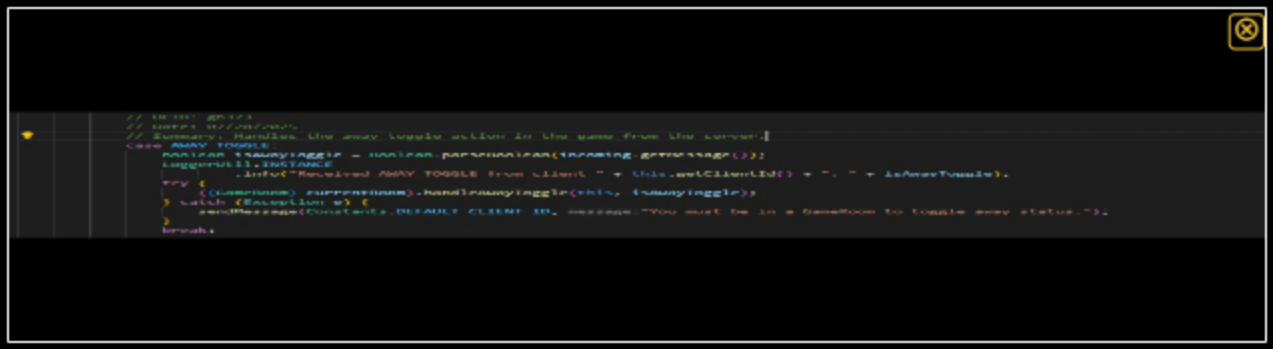




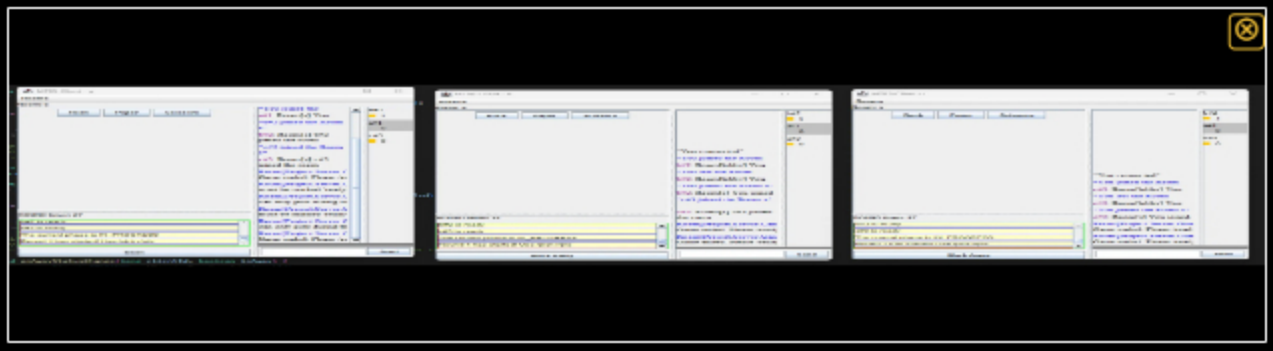
gameroom handiling the away toggle

Code for Sending Messages to Game Event Panel

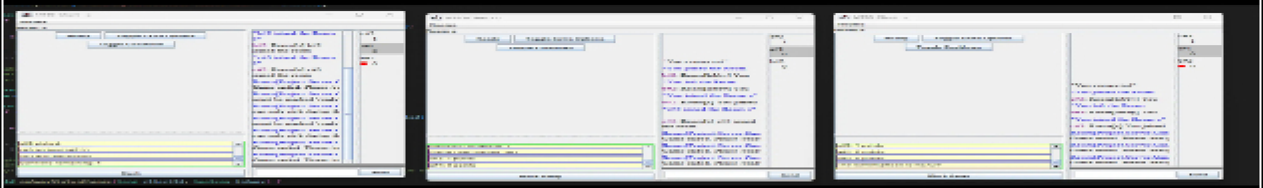
Code for button for toggiling away status



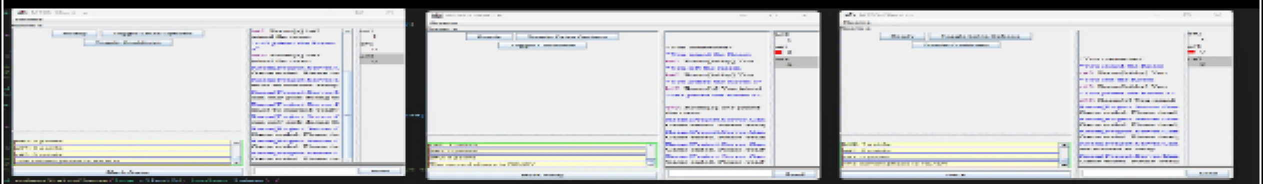
Code for dealing with the away toggle case in the server.



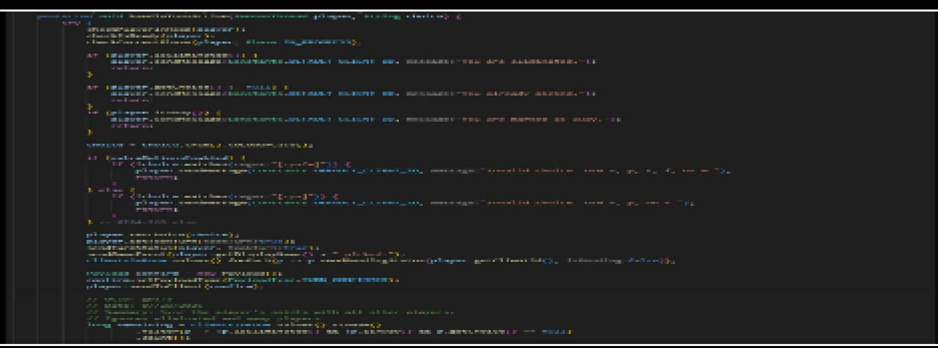
Output for showing a being away and the game starting without him and him not being inclided in the pick process



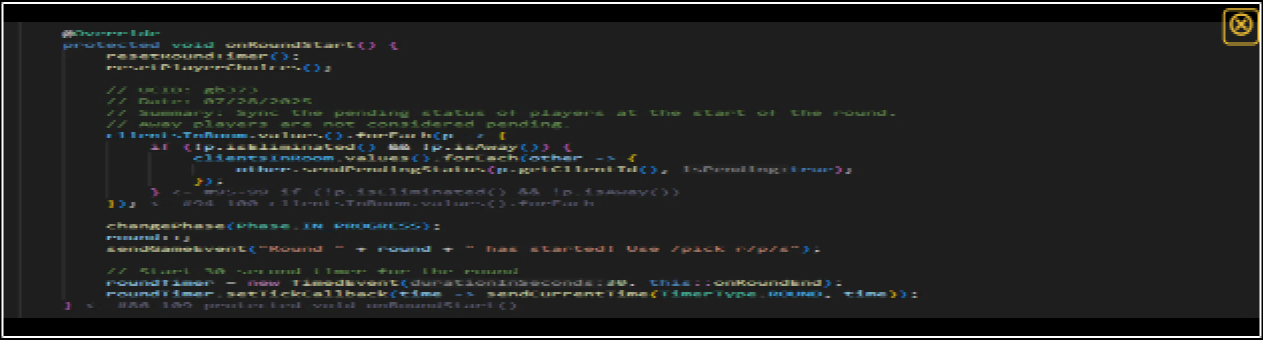
Output for a who is away not being counted for the round end and also not being eliminated.



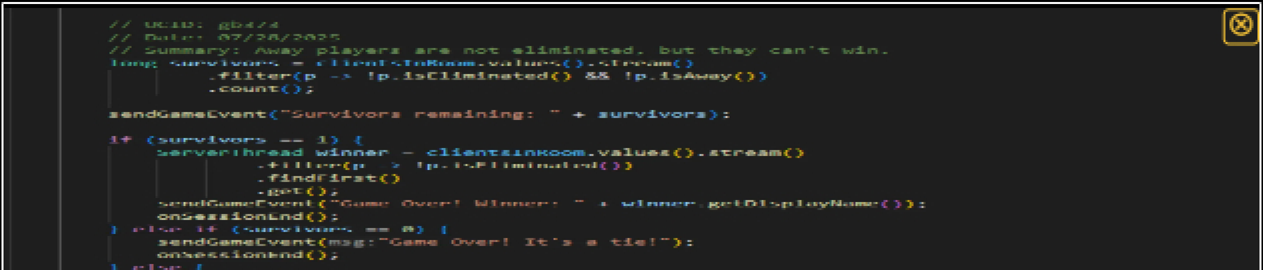
output for c being away and them trying to make a pick but they can't because they are away.



turn action ignoring away players with points



Away players are not considered pending on a round start meaning there skipped from making a pick



```
        onRoundStart();
    }
} <-- #11V-1VV protected void onRoundEnd()
```

On round end, an away player are not eliminated but can't win.

Output of c clicking being away and now a few time and the messages in the game event panel

Saved: 7/28/2025 6:48:46 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:

For the code flow of the away action, it starts in the UI where the player clicks the away button, which makes them toggles the away status. It then goes to the client part, where the away status is now true, and a payload is sent about that status. Finally, the server recognizes that this player is away and skips their turn, and doesn't affect them with eliminations or wins. They are excluded from elimination but can't win a round. Away players are also ignored for points because they are away. Finally, they are ignored for a pending pick, meaning their pick is skipped because they are away.

Saved: 7/28/2025 6:48:46 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

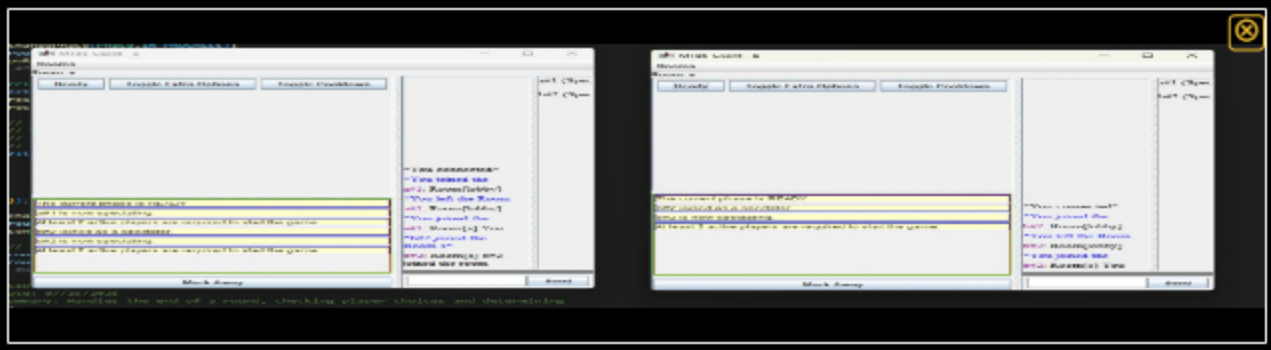
an in-progress session)

Part 1:

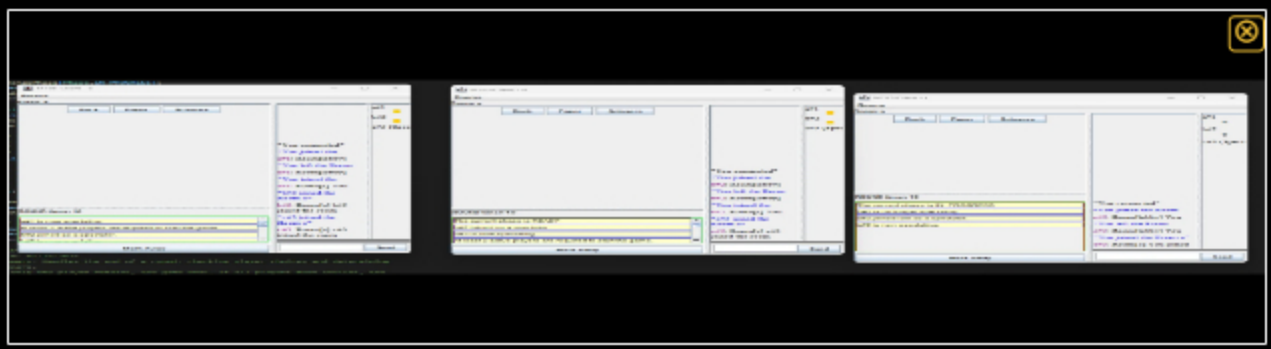
Progress: 100%

Details:

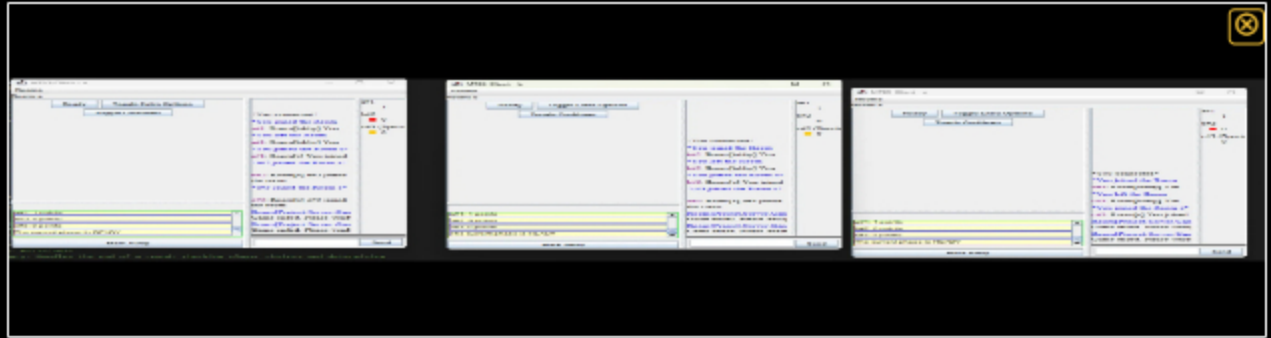
- Show the UI indicator of a spectator (visual and message)
- Show the related code flow from UI to server-side back to UI for showing the status
- Show the related code flow for sending the message to Game Events Panel
- Show various examples across 3+ clients of spectator status (including Game Events 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)



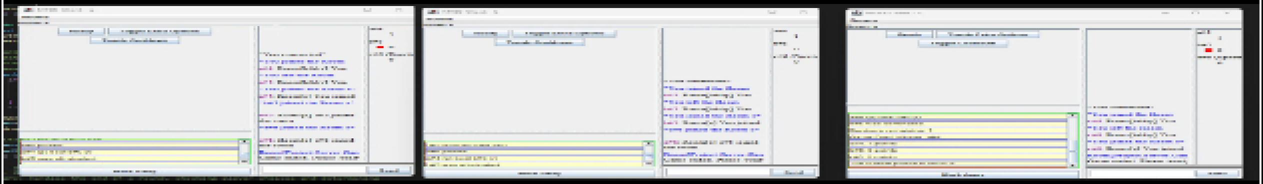
UI visual a message for spectators



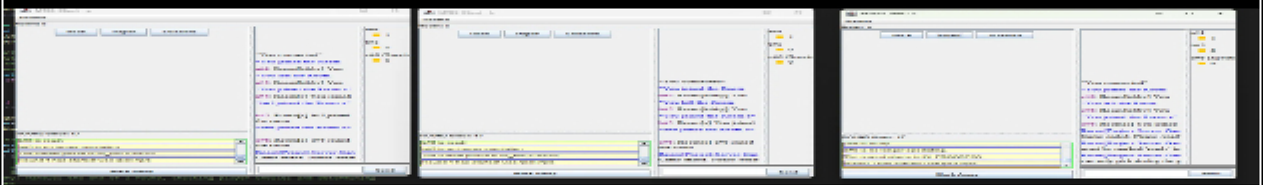
C joining late so he's a spectator.



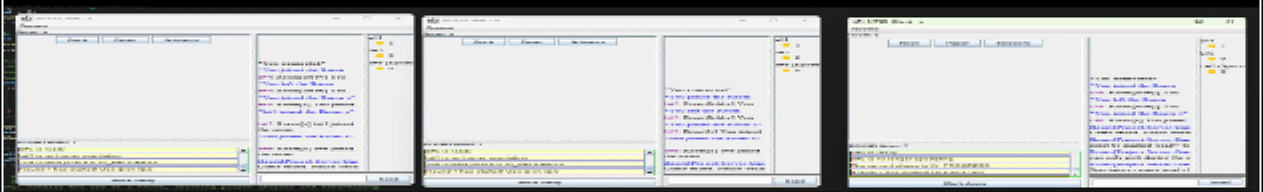
output example 2



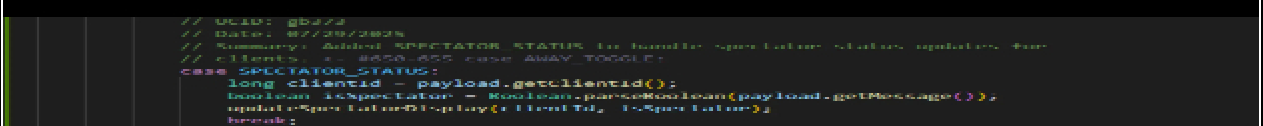
c not being in the pick or round logic



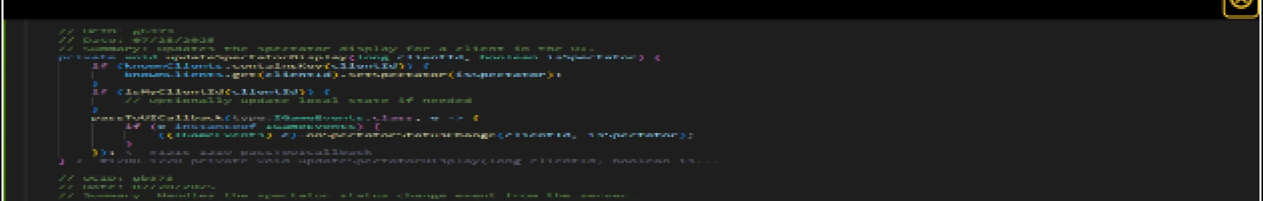
Messages for when c trys to make a pick saying no



Message to c saying they cant send messages



Code for case in client to deal with the status of spectators.



Clients dealing with spectators and updating UI and handiling server information

```
// UCID: gb373
// Date: 07/07/2025
// Summary: Sends the spectator status to the client, indicating whether the
// player is a spectator.
public void sendSpectatorStatus(long clientId, boolean isSpectator) {
    Payload payload = new Payload();
    payload.setPayloadType(PayloadType.SPECTATOR_STATUS);
    payload.setClientId(clientId);
    payload.setMessage(Boolean.toString(isSpectator));
    sendToClient(payload);
} <-- 8424-429 public void sendSpectatorStatus(long clientId, boolean isSpec...
```

Server sending spectator status to client

Gameroom handiling different ignoire logic for spectators and when they are ready they are not a spectator

```
// UCID: gb373
// Date: 07/07/2025
// Summary: Handles the spectator status change event and sending a message to the panel about the status change.
@Override
public void onSpectatorStatusChange(long clientId, boolean isSpectator) {
    String name = client.getClient(clientId).getDisplayNameOrDefault(clientId);
    addItem(name + (isSpectator ? " is now spectating." : " is no longer spectating."));
}
} <-- 830-833 public class GameEventView extends JPanel implements IMessa...
```

Messages sent in the game event panel about spectators

```
// UCID: gb373
// Date: 07/07/2025
// Summary: Handles the message payload from the client.
// Spectators cannot send messages.
case MESSAGE:
    if (this.isSpectator()) {
        sendMessage(CmdLimits.DEFAULT_CLIENT_ID, message:"Spectators cannot send chat messages.");
    } else {
        currentRoom.handleMessage(this, incoming.getMessage());
    }
    break;
```

Server making sure spectators cant send messages

Saved: 7/28/2025 8:06:02 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:

The code flow is to start the gameroom, handle the ready check, and see if a player is not ready, they are a spectator, and also if they are ready, they are not a spectator. It also looks out for whether a player joins later, and they are a spectator. Then it code goes to the client, where they get the status of whether they are a spectator or not, and then finally, the UI will broadcast the message saying who is a spectator. The server side ignores the logic the same way as away status, but adds the factor of they don't check for ready, meaning if you are a spectator, the other players can be ready and start a round. For preventing messages, there is a spectator check in the message case, saying that if it's true, they are a spectator, they can't send a message. Extra details that I have are the certain messages they get and only get, restricting them from picking and choosing messages. Spectators are also marked in the UI.

Saved: 7/28/2025 8:06:02 PM

Section #5: (1 pt.) Misc

Progress: 100%

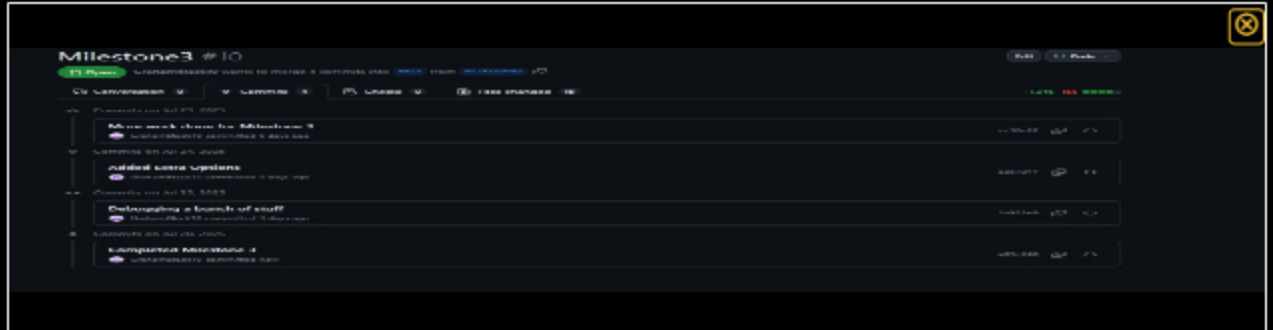
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

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



Commits (some are on for Milestone 2 because of rework in that branch)

Saved: 7/28/2025 8:08:35 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/GrahamBlack10/gb373-IT114-450/pull/10

URL
https://github.com/GrahamBlack1

Saved: 7/28/2025 8:08:35 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

individual time

overall time

Saved: 7/28/2025 8:09:23 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 the longest milestone, so I learn a lot about using ui and adding additional features to my RPS game. I learn a lot more about the flow between the UI and to client, to the server, and how to implement things so that the UI would show different aspects like pending pick and elimination. I learn about teaser ways to get that connection and found simpler ways to understand all the code between them. I also learn a lot more about game logic because of all the new features we have to add, like extra options, and if a player was away or spectating, what would the logic ignore for those certain players. I learned a lot more about the different panels of our UI and how I can edit them to show different things and new things.

Saved: 7/28/2025 5:08:41 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 tasks like elimination, pending picks, away status, and cooldown. I found these pretty simple because I found the code flow to be very easy to understand on how I would implement these features into my game. For things like the elimination mark and pending mark, I looked at how

the ready mark looked and implemented the other features with the logic that was needed for them to appear. The away status was simple because now I knew the code flow very well, so it was simple to add the flow from the UI to the server to show that a player was away and ignore game logic, so they were skipped. Finally, I had cooldowns implemented when I was doing the extra options, so that was helpful, and all I needed to do was have a toggle for it that only the host could enable.

Saved: 7/28/2025 5:12:33 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 I found was the extra option. I had difficulty with the logic at the start with what I wanted the 2 new options to do, and I found that there was some difficulty with figuring out a fair game that would make rps-3 turn into rps-5. After this, implementing it was more difficult. I found issues with making sure the UI knew what buttons to click, and the toggle had some difficulties. I found that adding a host check was also really difficult because I had to figure out how to make sure that there was a host, and the server and client knew it. Also, there were some issues with figuring out that the host had to be the first one to join and make it that way. I did eventually figure this out, and how I did this task was to start with the extra options, which took me some time, but I was able to get that working with the correct logic. When adding these options, it also started to mess with my other code and made things not work, so that was very annoying. But after that, I figured out how to have only the host toggle the extra options, and it eventually was all working with no issues.

Saved: 7/28/2025 5:19:31 PM