Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-milestone-4-rps-2024-m24/grade/cae6

Course: IT114-003-F2024
Assigment: [IT114] Milestone 4 RPS 2024 M24
Student: Chizorom E. (cae6)

Submissions:

Submission Selection

1 Submission [submitted] 12/12/2024 3:54:53 AM

Instructions

COLLAPSE

- Implement the Milestone 4 features from the project's proposal document: https://docs.google.com/document/d/11SRMo7JkLAMM-PuuiGwl_Z-QXP3pyQ7xN3lRxwmcwCc/view
- Make sure you add your ucid/date as code comments where code changes are done
- All code changes should reach the Milestone4 branch
- Create a pull request from Milestone4 to main and keep it open until you get the output PDF from this assignment.
- Gather the evidence of feature completion based on the below tasks.
- Once finished, get the output PDF and copy/move it to your repository folder on your local machine.
- Run the necessary git add, commit, and push steps to move it to GitHub
- Complete the pull request that was opened earlier
- Upload the same output PDF to Canvas

Branch name: Milestone4

Group

100%

Group: Away
Tasks: 1
Points: 2.5

COLLAPSE

Task

100%

Group: Away
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game
Weight: ~100%
Points: ~2.50

COLLAPSE

Columns: 1

Sub-Task

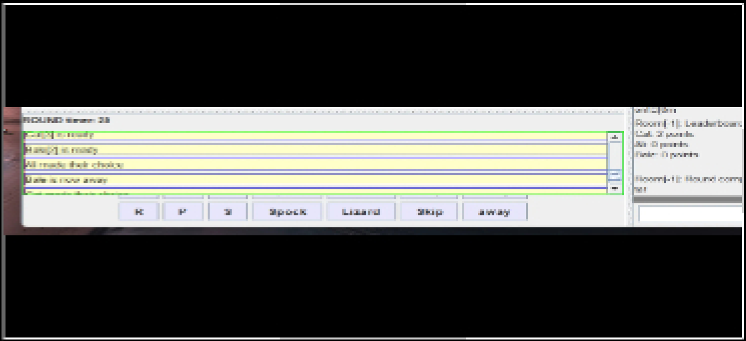
100%

Group: Away
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game
Sub Task #1: Screenshots of editors must have the frame title visible with your ucid and the client name. Code screenshots must have ucid/data comments.

Task Screenshots

Gallery Style: 2 Columns

4 2 1



ROUND timer: 28
Cat[2] is ready
Kate[2] is ready
All made their choice
Kate is now away
Cat made their choice

Room[-1]: Leaderboard
Cat: 2 points
Ali: 0 points
Kate: 0 points

Room[-1]: Round comp

Mark away

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

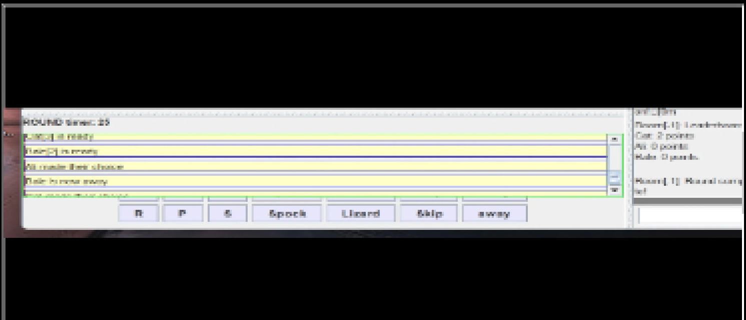
100%

Group: Away
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game
Sub Task #2: Show a few examples of the Game Events Panel showing away/not away status in a clear message

Task Screenshots

Gallery Style: 2 Columns

4 2 1



ROUND timer: 25
Cat[2] is ready
Kate[2] is ready
All made their choice
Kate is now away

Room[-1]: Leaderboard
Cat: 2 points
Ali: 0 points
Kate: 0 points

Room[-1]: Round comp

Showing away status i

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Away

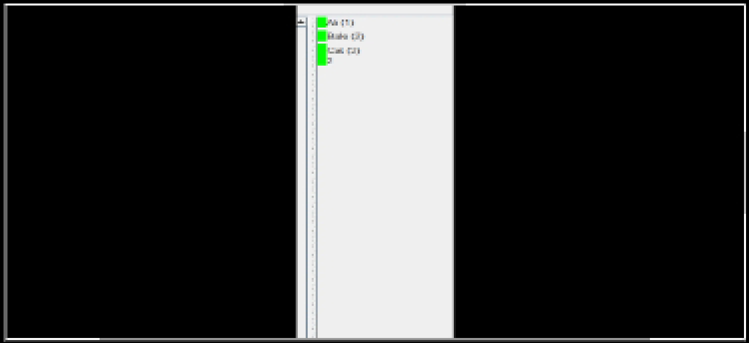
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game

Sub Task #3: Show a few examples of the User List panel showing a Client away (demonstrate that this changes when the status is toggled)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



UserList

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Away

Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game

Sub Task #4: Show the code flow that handles the away toggle (from Client interaction to updating server-side state)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Was not able to implement

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

I was not able to full implement the away feature, nor get it to work in full. What I ended up trying to do in due time was letting users mark themselves away as a choice, however, that implementation was not fully implemented as well.

Sub-Task

100%

Group: Away

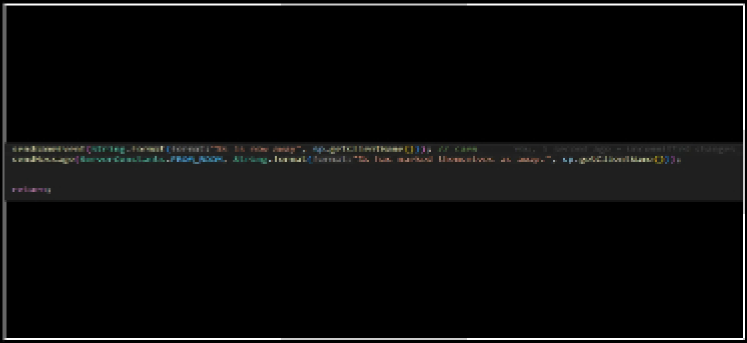
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game

Sub Task #5: Show the code that handles the Game Events Panel message (from the server-side state changing to sending the payload)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
sendGameEvent(String.format(format:"%s is now away", sp.getClientName())); // Care [illegible]
sendMessage(ServerConstants.FROM_ROOM, String.format(format:"%s has marked themselves as away.", sp.getClientName()));


return;
```

Game event panel message

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

In order to show that a client had been marked away, I sent a game event back to the client side in the game room and it will send the player from the room's name and the message saying that they marked themselves away.

Sub-Task

100%

Group: Away

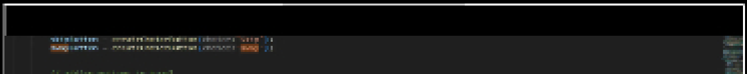
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game

Sub Task #6: Show the code that handles updating the User List Panel (from Client receiving -> the UI change)

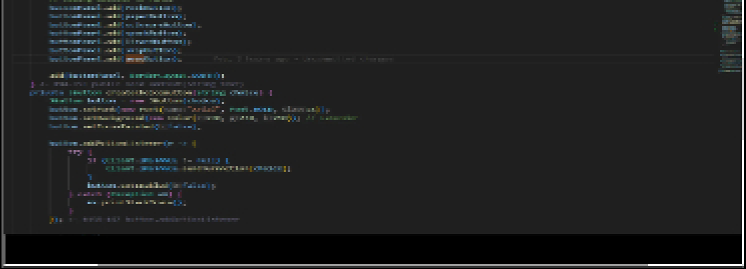
Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
skipButton = createCenterButton([illegible] "Skip");
awayButton = createCenterButton([illegible] "Away");
```



```
        buttonPanel.add(rockButton);
        buttonPanel.add(paperButton);
        buttonPanel.add([illegible]);
        buttonPanel.add([illegible]);
        buttonPanel.add([illegible]);
        buttonPanel.add([illegible]);
        buttonPanel.add(awayButton);

        add(buttonPanel, BorderLayout.[illegible]);
    } [illegible]
    private JButton [illegible](String choice) {
        JButton button = new JButton(choice);
        button.setFont(new Font(name:"Arial", Font.BOLD, [illegible]));
        button.setBackground(new Color([illegible])); // [illegible]
        button.setFocusPainted(b:false);

        button.addActionListener(e -> {
            try {
                if (Client.INSTANCE != null) {
                    Client.INSTANCE.[illegible](choice);
                }
                button.setEnabled(b:false);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }); // [illegible] button.addActionListener
```

Was not able to add in time

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

I was not able to add the code updating the user List panel. I think I may have mixed up some of the instructions, so what I attempted to do was add the away option to the play panel where the users choices are, and have it handled here.

Sub-Task

100%

Group: Away

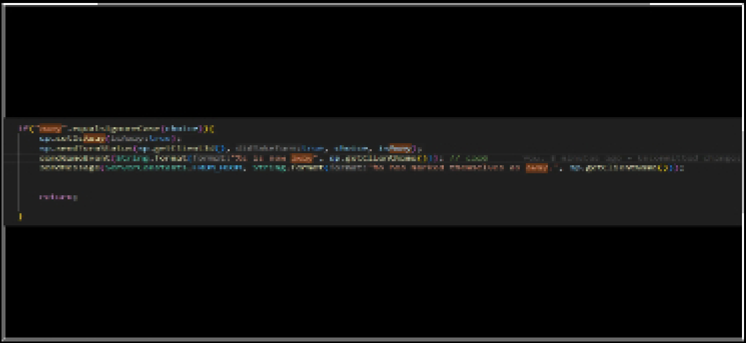
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game

Sub Task #7: Show the project logic that skips/ignores away players (away players can't take a turn) and include a UI screenshot of an applicable message if someone tries the action while away

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
if("away".equalsIgnoreCase(choice)){
    sp.setIsAway(isAway:true);
    sp.sendTurnStatus(sp.getClientId(), didTakeTurn:true, choice, isAway);
    [illegible](String.format(format:"%s is now away", sp.getClientName())); // [illegible]
    sendMessage([illegible], String.format(format:"%s has marked themselves as away.", sp.getClientName()));

    return;
}
```

Project Logic

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

If the choice is away, then the client or the player is set as away. The turn status will be sent sending, that they took their turn, their name, their ID, and the fact that they are away.Then a game event is sent out stating that the player has marked themselves away, along with a regular message.

End of Task 1

End of Group: Away
Task Status: 1/1

Group

100%

Group: Spectator
Tasks: 1
Points: 2.5

⌃ COLLAPSE ⌃

Task

100%

Group: Spectator
Task #1: Spectator: Client can join as spectator
Weight: ~100%
Points: ~2.50

⌃ COLLAPSE ⌃

ⓘ Details:

Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.
Spectator control/access logic must be handled in the GameRoom.
They can see all chat but are ignored from turns and can’t send messages

Columns: 1

Sub-Task

100%

Group: Spectator
Task #1: Spectator: Client can join as spectator
Sub Task #1: Show the spectator UI and demonstrate how they're blocked/ignored from game actions (turns and messages)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Was not able to add a spectator

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Spectator

Task #1: Spectator: Client can join as spectator

Sub Task #2: Show the User List Panel representation of Spectators (must differ from active Players)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Was not able to add spectator

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Spectator

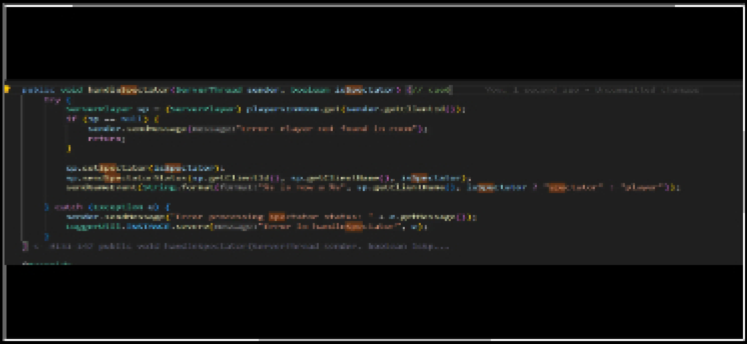
Task #1: Spectator: Client can join as spectator

Sub Task #3: Show the code logic that blocks/ignores spectators from game actions (turns and messages)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
public void handleSpectator(ServerThread sender, boolean isSpectator) {// cas        You, 1 second ago • Uncommitted changes
    try {
        ServerPlayer sp = (ServerPlayer) playersInRoom.get(sender.getClientId());
        if (sp == null) {
            sender.sendMessage(message:"Error: player not found in room");
            return;
        }

        sp.setSpectator(isSpectator);
        sp.sendSpectatorStatus(sp.getClientId(), sp.getClientName(), isSpectator);
        sendGameEvent(String.format(format:"%s is now a %s", sp.getClientName(), isSpectator ? "spectator" : "player"));

    } catch (Exception e) {
        sender.sendMessage("Error processing spectator status: " + e.getMessage());
        LoggerUtil.INSTANCE.severe(message:"Error in handleSpectator", e);
    }
} // public void handleSpectator(ServerThread sender, boolean isSp...
```

code logic

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

This code was supposed to handle changing a player's status in a game room to either a spectator or an active player. First it checks if the player exists and if not, it will sends an error message. If the player is found, it updates their status based on whether isSpectator is true (spectator) or false meaning they are a player. It sends a message to the player about theit role and if any errors occur, it logs the issue and sends an error message to the player.

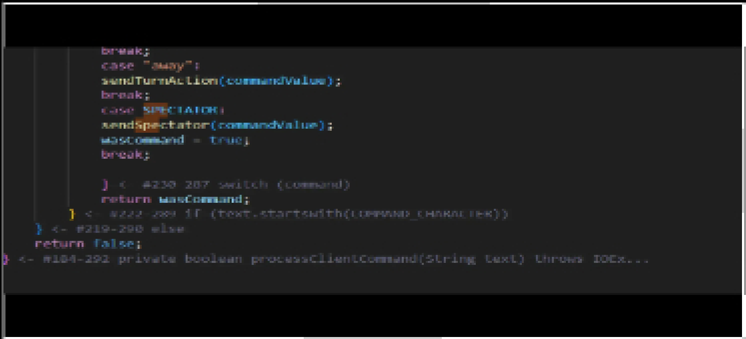
Sub-Task
100%

Group: Spectator
Task #1: Spectator: Client can join as spectator
Sub Task #4: Show the code flow of the server-side sending the spectator status to Clients and having the User List Panel updated accordingly

Task Screenshots

Gallery Style: 2 Columns

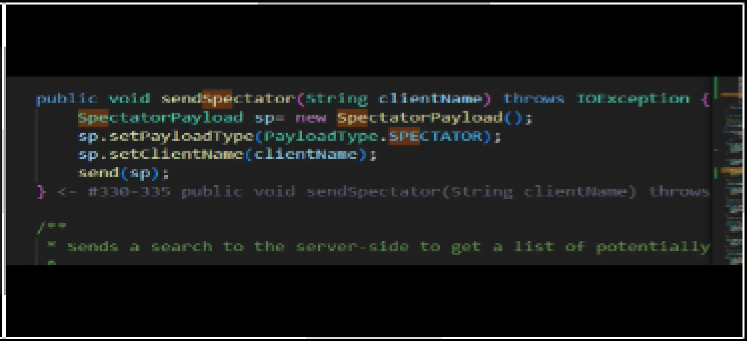
4 2 1



```
                break;
                case "away":
                sendTurnAction(commandValue);
                break;
                case SPECTATOR:
                sendSpectator(commandValue);
                wasCommand = true;
                break;

                } <- #230-287 switch (command)
                return wasCommand;
            } <- #222-289 if (text.startsWith(COMMAND_CHARACTER))
        } <- #219-290 else
        return false;
} <- #184-292 private boolean processClientCommand(String text) throws IOEx...
```

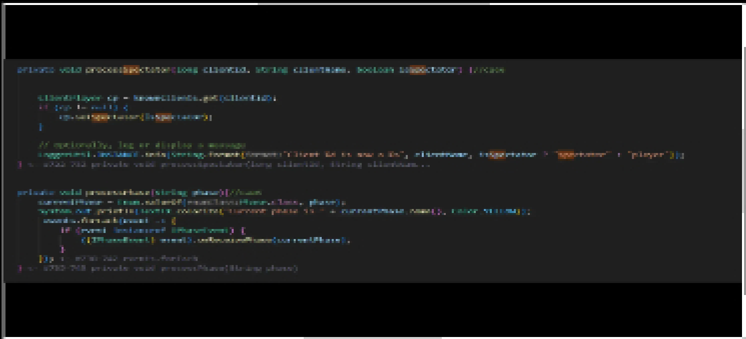
Spectator status 1: client



```
    public void sendSpectator(String clientName) throws IOException {
        SpectatorPayload sp= new SpectatorPayload();
        sp.setPayloadType(PayloadType.SPECTATOR);
        sp.setClientName(clientName);
        send(sp);
    } <- #330-335 public void sendSpectator(String clientName) throws

    /**
     * sends a search to the server-side to get a list of potentially
```

Spectator status 2: client



```
private void processSpectator(long clientId, String clientName, boolean isSpectator) {//case


    ClientPlayer cp = knownClients.get(clientId);
    if (cp != null) {
        cp.setSpectator(isSpectator);
    }

    // optionally, log or display a message
    LoggerUtil.INSTANCE.info(String.format(format:"Client %s is now a %s", clientName, isSpectator ? "spectator" : "player"));
} <- #722-732 private void processSpectator(long clientId, String clientName...


private void processPhase(String phase){//[illegible]
    currentPhase = Enum.valueOf(Phase.class, phase);
    System.out.println(TextFX.colorize("Current phase is " + currentPhase.name(), Color.YELLOW));
     events.forEach(event -> {
        if (event instanceof IPhaseEvent) {
            ((IPhaseEvent) event).onReceivePhase(currentPhase);
        }
    }); <- [illegible] events.forEach
} <- [illegible] private void processPhase(String phase)
```

3: client

was not able to add to user list panel correctly

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

Although the spectator does not work, this is the code I did to attempt it. First I added Spectator as a command, giving clients the option to join in as a spectator.Then I added a send spectator in which it sends the spectator data to the server. I also added a process spectator which processes the payload type spectator finds the client, checks if the client exists, and if the client does they are set as a spectator. Then a message that specifies whether a client is a spectator or player is logged.

End of Task 1

End of Group: Spectator
Task Status: 1/1

Group

100%

Group: Project Specific
Tasks: 2
Points: 4

^ COLLAPSE ^

Task

100%

Group: Project Specific
Task #1: Implement extra options beyond Rock Paper and Scissors
Weight: ~50%
Points: ~2.00

^ COLLAPSE ^

ⓘDetails:
Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

- Optionally only activate this at different stages of the game (i.e., last 3 people)

Columns: 1

Sub-Task

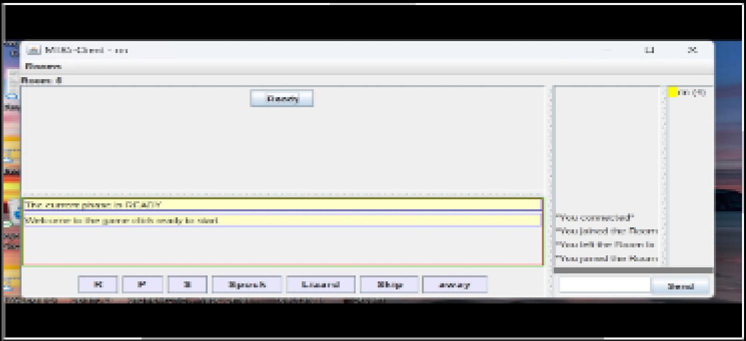
100%

Group: Project Specific
Task #1: Implement extra options beyond Rock Paper and Scissors
Sub Task #1: Show the pre-game screen (can be ready panel) that toggles the extra option mode

Task Screenshots

Gallery Style: 2 Columns

4 2 1



The current phase is READY
Welcome to the game click ready to start

You connected
*You joined the Room
*You left the Room la
*You joined the Room

Pre-game screen

Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Note the extra options you went with and how the work
Response:

The extra options I added were spock and Lizard. So Rock beats crushes scissors and lizard, Paper covers rock and disproves spock, scissors cuts paper and lizard, spock smashes scissors and vaporizes rock, and lizard eats paper disproves Spock, scissors cuts paper and lizard, spock smashes scissors and vaporizes rock, and lizard eats paper and poisons Spock. So now instead of a range of 3, there is a range of 5 players, with indices 0 through 4. So player one will win if the difference is 1 or 3 because the equation is if ((a - b + 5) % 5 == 1 || (a - b + 5) % 5 == 3) and player 2 wins if the difference is 2 or 4. If there is no difference, then that means there was a tie.

Sub-Task

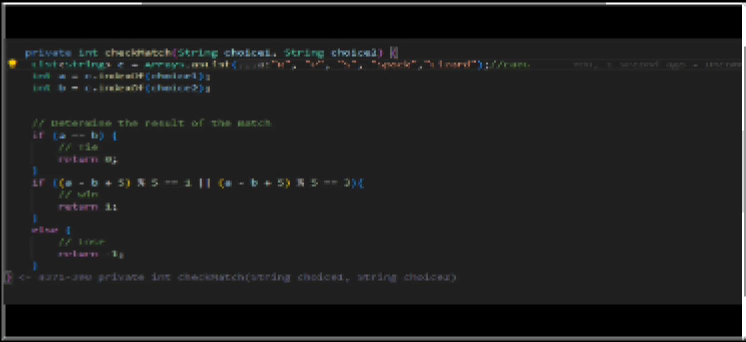
100%

Group: Project Specific
Task #1: Implement extra options beyond Rock Paper and Scissors
Sub Task #2: Show the code related to handling these extra options and where it's used/set (on session start or towards the last few rounds, note this as well)

Task Screenshots

Gallery Style: 2 Columns

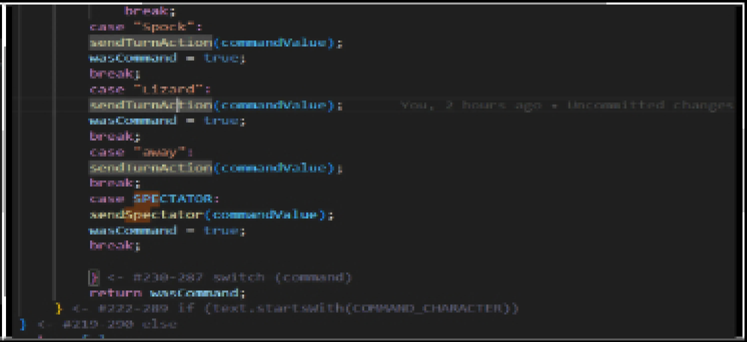
4 2 1



```
private int checkMatch(String choice1, String choice2) {
    List<String> c = Arrays.asList(...a:"R", "P", "S", "Spock","Lizard");//rars        You, 1 second ago • Uncommi
    int a = c.indexOf(choice1);
    int b = c.indexOf(choice2);


    // Determine the result of the match
    if (a == b) {
        // tie
        return 0;
    }
    if ((a - b + 5) % 5 == 1 || (a - b + 5) % 5 == 3){
        // win
        return 1;
    }
    else {
        // lose
        return -1;
    }
} <- #271-288 private int checkMatch(String choice1, String choice2)
```

handling new options



```
                break;
            case "Spock":
            sendTurnAction(commandValue);
            wasCommand = true;
            break;
            case "Lizard":
            sendTurnAction(commandValue);        You, 2 hours ago • Uncommitted changes
            wasCommand = true;
            break;
            case "away":
            sendTurnAction(commandValue);
            break;
            case SPECTATOR:
            sendSpectator(commandValue);
            wasCommand = true;
            break;

            } <- #230-287 switch (command)
            return wasCommand;
        } <- #222-289 if (text.startsWith(COMMAND_CHARACTER))
    } <- #219-290 else
```

handling new options

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

Instead of a range of 3, there is a range of 5 players, with indices 0 through 4. So player one will win if the difference is 1 or 3 because the equation is if ((a - b + 5) % 5 == 1 || (a - b + 5) % 5 == 3) and player 2 wins if the difference is 2 or 4. If there is no difference, then that means there was a tie. I've also created more command options from the players to choose form and more buttons as well, with the option.

End of Task 1

Task

100%

Group: Project Specific
Task #2: Implement a cooldown on an option (i.e., same option can't be picked twice in a row by the same player)
Weight: ~50%
Points: ~2.00

COLLAPSE

Details:

Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/date comments

Code screenshots must have ucid/date comments.

Columns: 1

Sub-Task

100%

Group: Project Specific

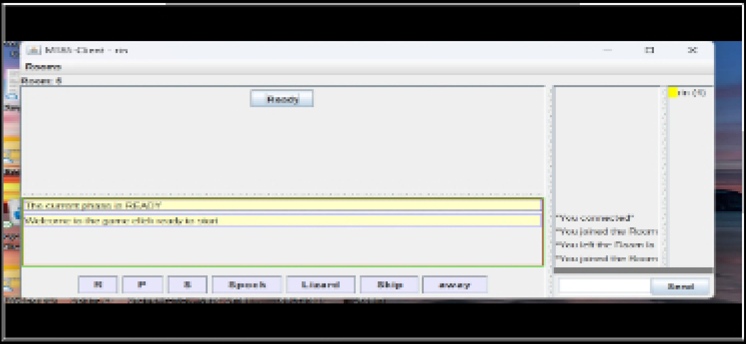
Task #2: Implement a cooldown on an option (i.e., same option can't be picked twice in a row by the same player)

Sub Task #1: Show the pre-game screen (can be ready panel) that toggles this feature

Task Screenshots

Gallery Style: 2 Columns

4 2 1



The current phase is READY

Welcome to the game click ready to start

You connected
*You joined the Room
*You left the Room la
*You joined the Room

Pre game screen for cooldown

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Project Specific

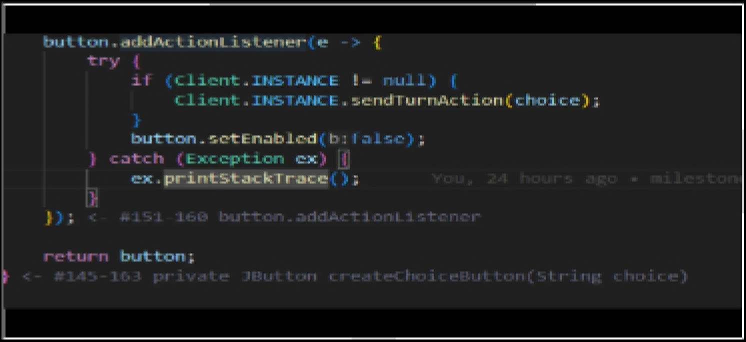
Task #2: Implement a cooldown on an option (i.e., same option can't be picked twice in a row by the same player)

Sub Task #2: Show the code related to handling this on the server-side and client-side

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
    button.addActionListener(e -> {
        try {
            if (Client.INSTANCE != null) {
                Client.INSTANCE.sendTurnAction(choice);
            }
            button.setEnabled(b:false);
        } catch (Exception ex) {
            ex.printStackTrace();       You, 24 hours ago • mileston
        }
    }); <- #151-160 button.addActionListener

    return button;
} <- #145-163 private JButton createChoiceButton(String choice)
```

cool down, client -side

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works and explain how you handle the "prevent twice in a row" logic

Response:

I was not able to implement a standard or an actual cooldown that includes a server-side, however I was able to disable the button after the user has already chosen an option so they can't pick the same choice again.

Sub-Task

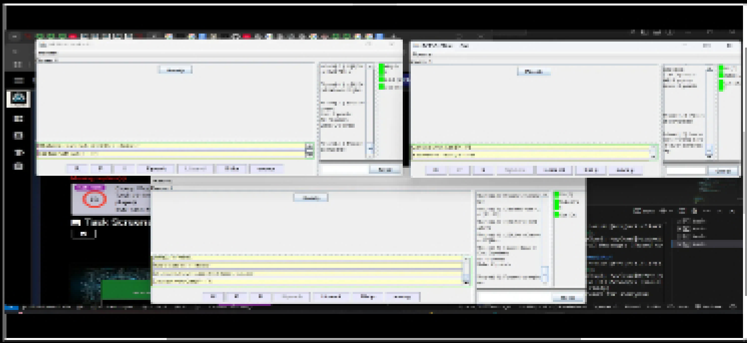
100%

Group: Project Specific
Task #2: Implement a cooldown on an option (i.e., same option can't be picked twice in a row by the same player)
Sub Task #3: Demonstrate from the UI this feature in effect including any potential related Game Event messages

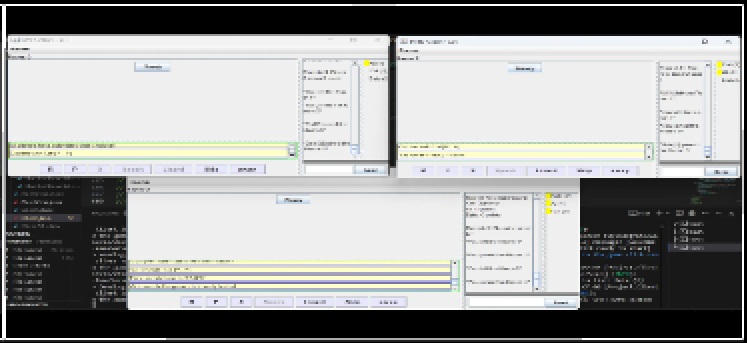
Task Screenshots

Gallery Style: 2 Columns

4 2 1



Feature in affect



feature in affect in a different room

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

When clients click on an option, the button fades out and the client isn't able to click on the option again. This stays in affect until another round is begun.

End of Task 2

End of Group: Project Specific
Task Status: 2/2

Group

100%

Group: Misc
Tasks: 3
Points: 1

COLLAPSE

Task

Group: Misc

100%

Task #1: Add the pull request link for the branch
Weight: ~33%
Points: ~0.33

^ COLLAPSE ^

Details:
Note: the link should end with /pull/#

Task URLs

URL #1
https://github.com/Cae6/cae6-IT114-003/pull/12

URL
https://github.com/Cae6/cae6-IT114-003/pull/12

End of Task 1

Task

100%

Group: Misc
Task #2: Talk about any issues or learnings during this assignment
Weight: ~33%
Points: ~0.33

^ COLLAPSE ^

Task Response Prompt

Response:

I had a few issues when it came to milestone 4, and the main ones surrounded being able to mark away properly and adding a cooldown feature properly. I have parts of those features but they were not implemented well. I also had a few new issues occurring such as the buttons glitching a bit and the size adjustment to fit the new choices. I also was not able to implement the spectator well in due time. I did attempt the code, however the button for it was not appearing.

End of Task 2

Task

100%

Group: Misc
Task #3: WakaTime Screenshot
Weight: ~33%
Points: ~0.33

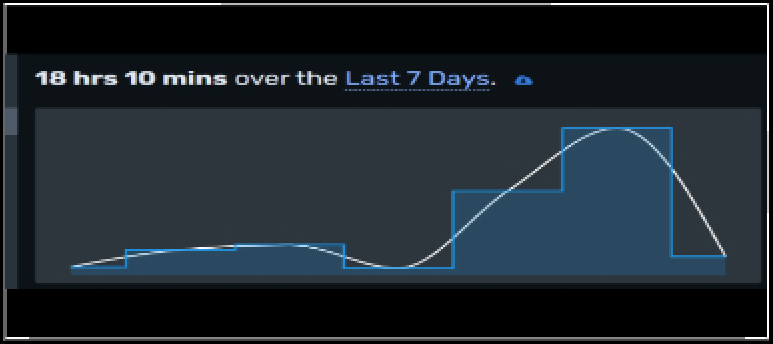
^ COLLAPSE ^

Details:
Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

Task Screenshots

Gallery Style: 2 Columns

4 2 1



18 hrs 10 mins over the Last 7 Days.

Waka Time ScreenShot

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment