Submission Worksheet

Submission Data

Course: IT114-007-F2025
Assignment: IT114 - Milestone 3 - RPS
Student: Lukas P. (lap5)
Status: Submitted | **Worksheet Progress:** 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Started: 12/9/2025 3:05:03 AM
Updated: 12/9/2025 4:03:38 AM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-007-F2025/it114-milestone-3-rps/grading/lap5
View Link: https://learn.ethereallab.app/assignment/v3/IT114-007-F2025/it114-milestone-3-rps/view/lap5

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%

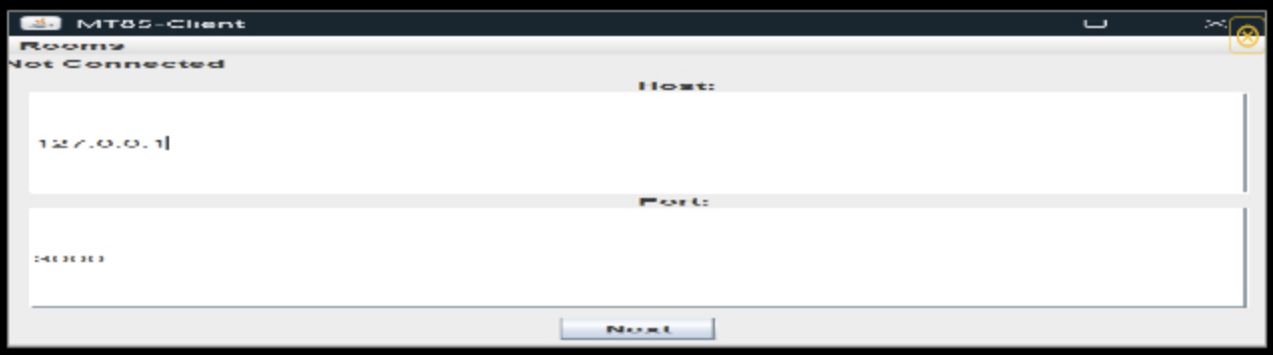
Part 1:

Progress: 100%

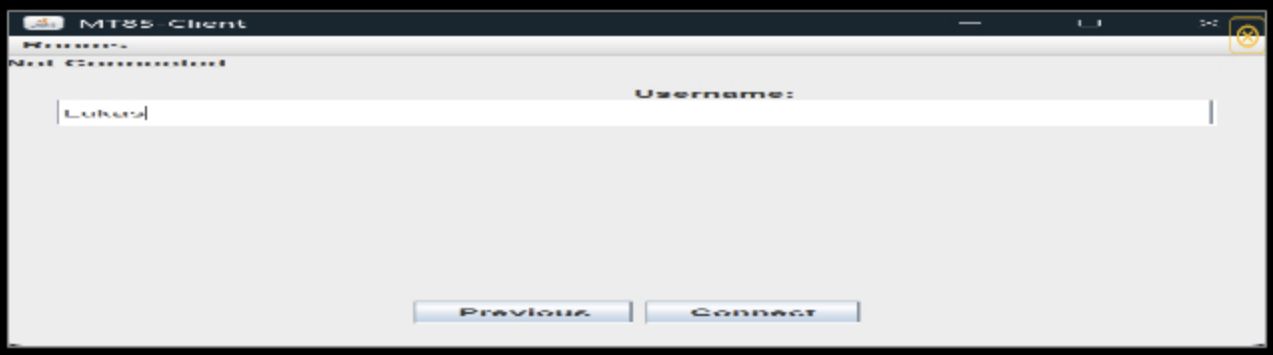
Details:

- Show the connection panel with valid data

- Show the user details panel with valid data



panel



panel

Saved: 12/9/2025 3:07:17 AM

Part 2:

Progress: 100%

Details:

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

Your Response:

The UI captures input and `Client.connect()` initializes a `Socket`. On success, it spawns a `CompletableFuture` to `listenToServer()` and immediately sends a `CLIENT_CONNECT` payload with the username to handshake.

Saved: 12/9/2025 3:07:17 AM

Task #2 (0.50 pts.) - Ready Panel

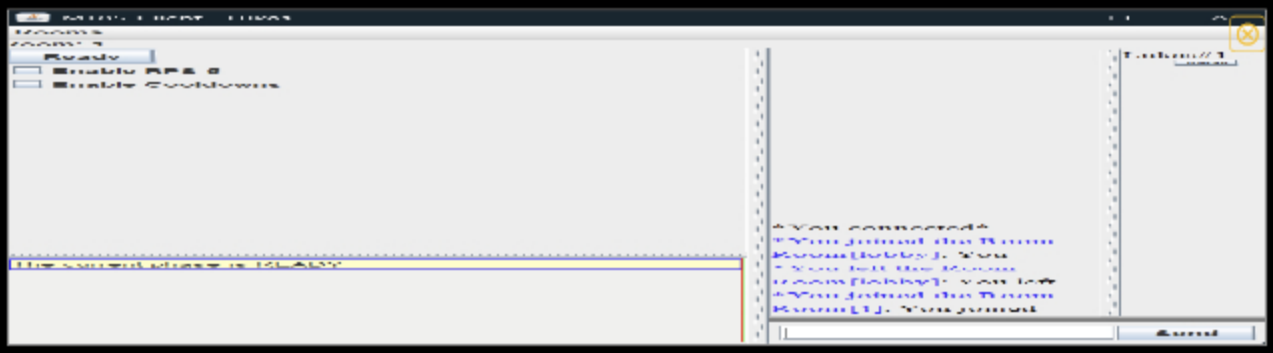
Progress: 100%

Part 1:

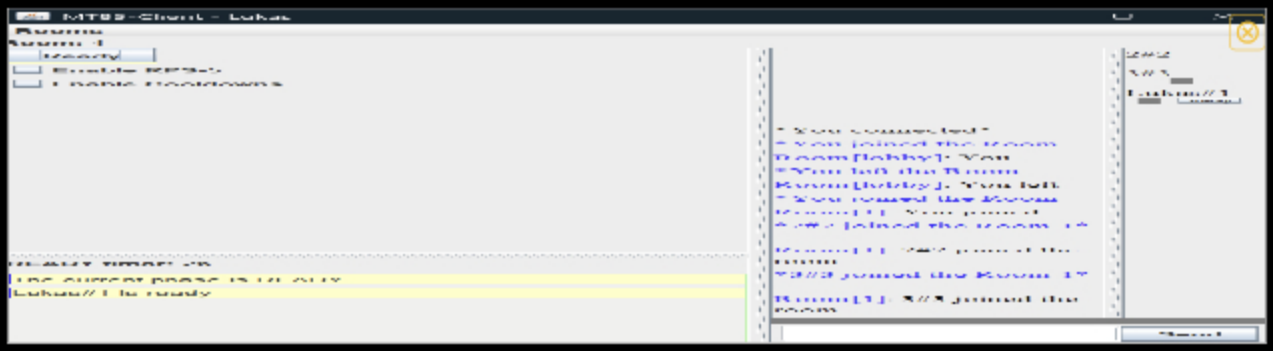
Progress: 100%

Details:

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



button



ready

Saved: 12/9/2025 3:09:17 AM

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:

Clicking the button calls `Client.sendReady()`, creating a `ReadyPayload` with `isReady=true` and sending it via `sendToServer()`, which serializes the object to the output stream.

`Client.processReadyStatus` receives the payload, updates the local `User` object, and triggers the `onReceiveReady` callback. `UserListView` intercepts this and updates the specific `UserListItem` visual state.

Saved: 12/9/2025 3:09:17 AM

Section #2: (2 pts.) Project Ui

Progress: 100%

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

Progress: 100%

Details:

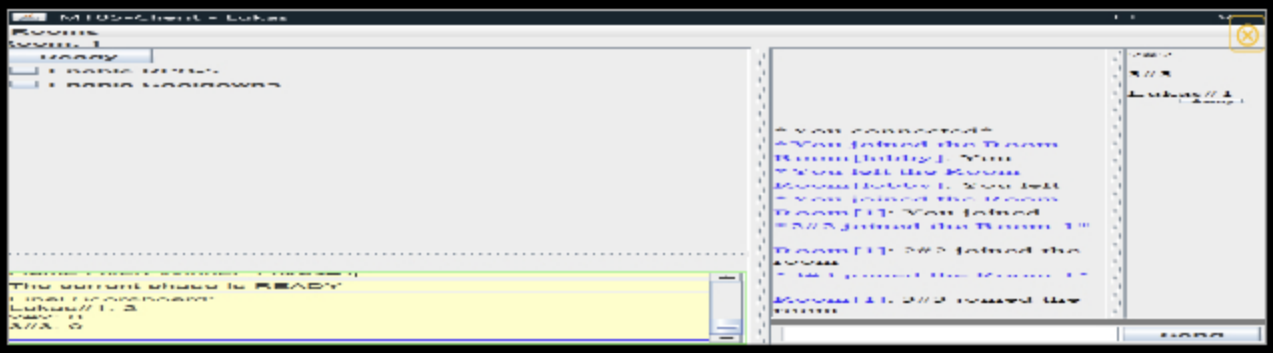
- Show the username and id of each user
- Show the current points of each user
- Users should appear in score order, sub-sort by name when ties occur
- Pending-to-pick users should be marked accordingly
- Eliminated users should be marked accordingly

Part 1:

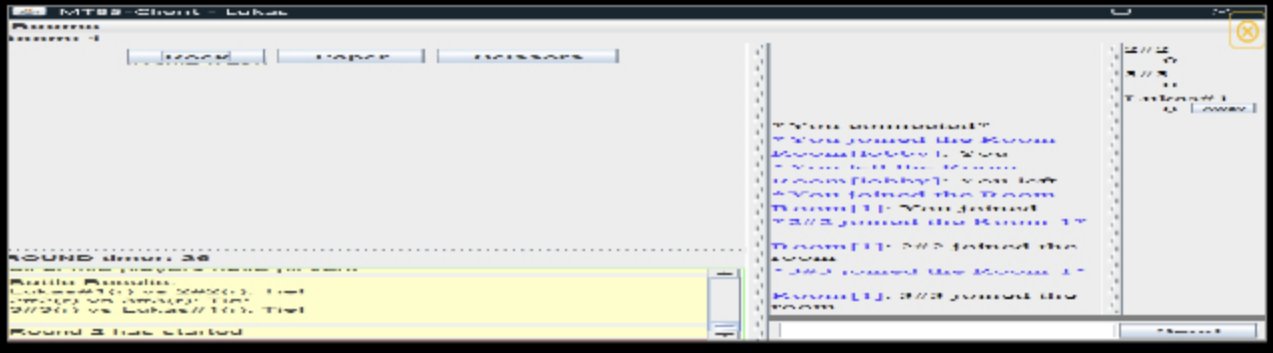
Progress: 100%

Details:

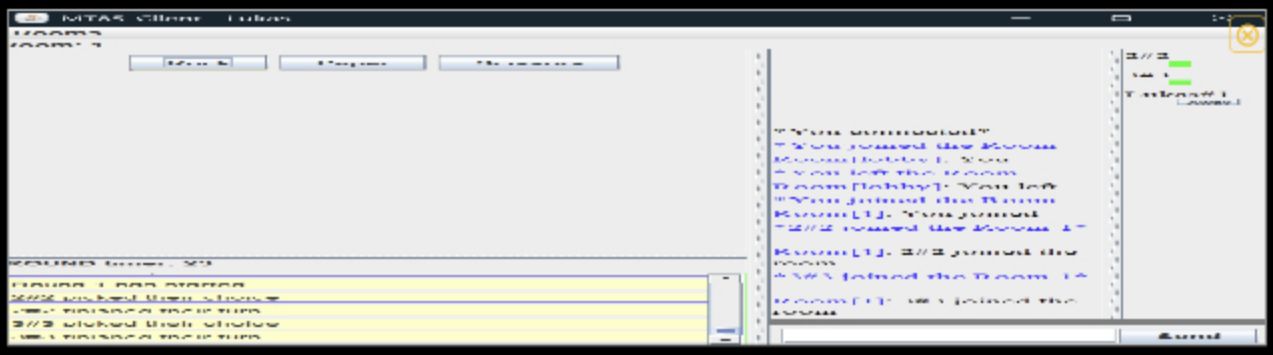
- Show various examples of points (3+ clients visible)
    - Include code snippets showing the code flow for this from server-side to UI
- Show that the sorting is maintained across clients
    - Include code snippets showing the code that handles this
- Show various examples of the pending-to-pick indicators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



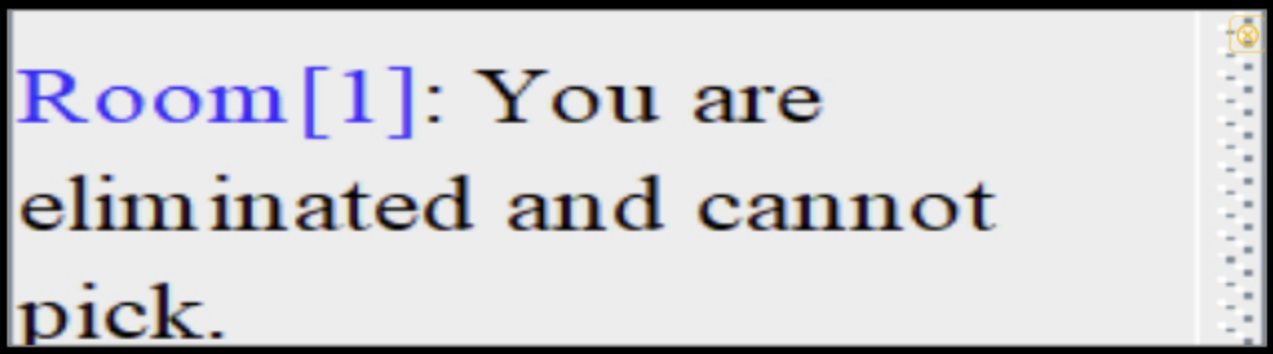
username and id



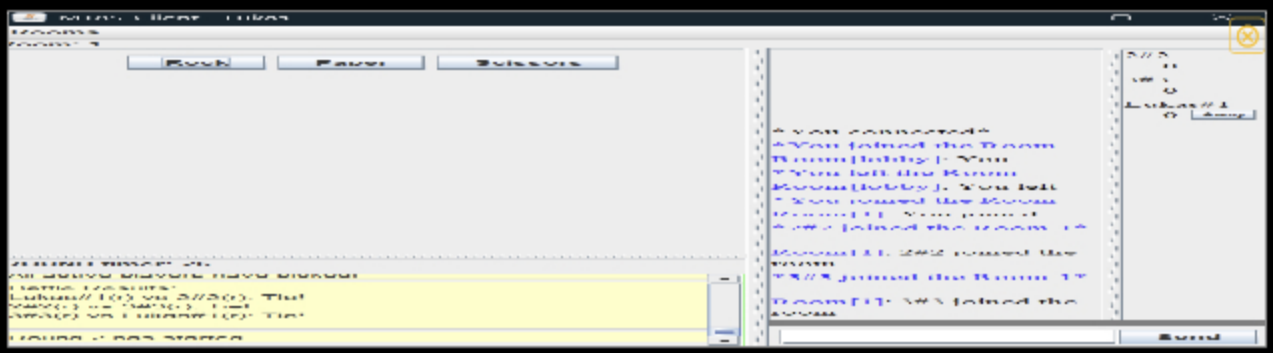
points



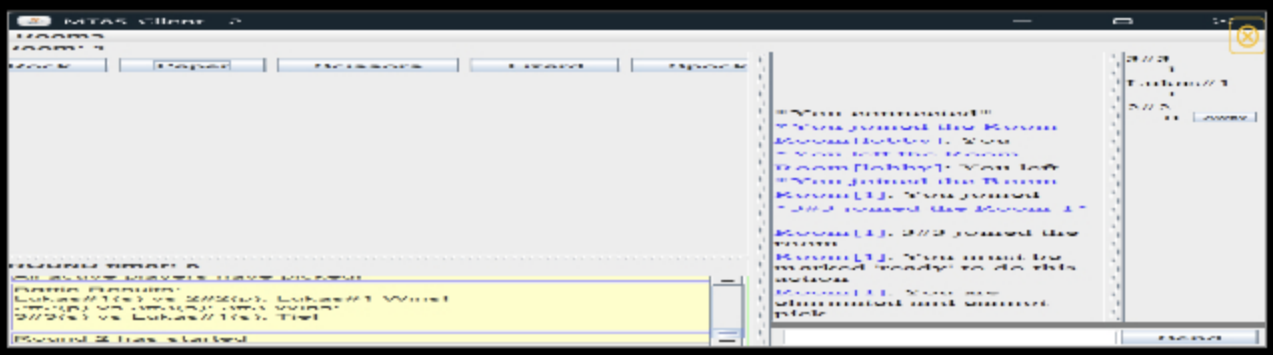
pending to pick is indicated by nothing. Like thats what indicates them as needing to pick. The lack of anything means pick smth



marked as eliminated



point example



point example

```
private void processPoints(Payload payload) {
    PointsPayload pp = (PointsPayload) payload;
    if (knownClients.containsKey(pp.getClientId())) {
        knownClients.get(pp.getClientId()).setPoints(pp.getPoints());
        passToUICallback(IPointsEvent.class, e -> e.onPointsUpdate(pp.getClientId
```

```
        (), pp.getPoints()));
    }
}
```

code

```
private void sortUserList() {
    List<UserListItem> sortedItems = new ArrayList<>(userItemsMap.values());
    sortedItems.sort((a, b) -> {
        int scoreCompare = Integer.compare(b.getPoints(), a.getPoints());
        return scoreCompare != 0 ? scoreCompare : a.getClientName().
        compareToIgnoreCase(b.getClientName());
    });
    // ... re-add items in order
}
```

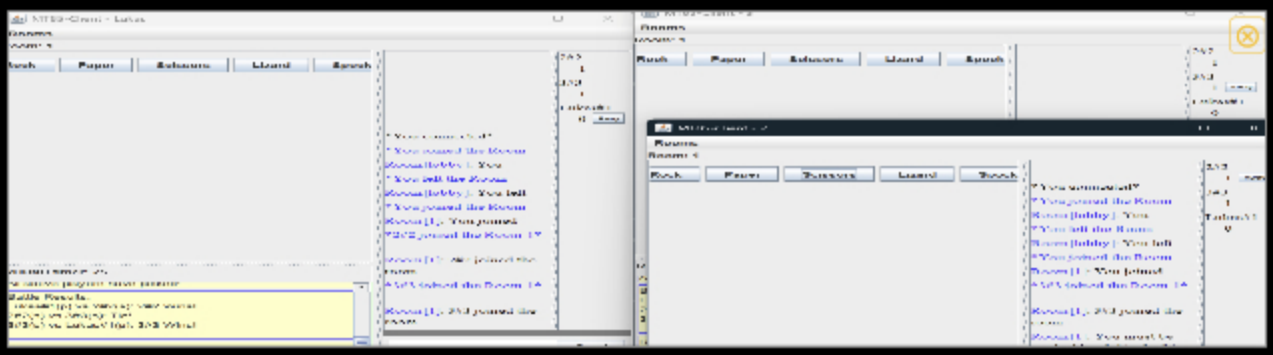
code

```
if (isAway) {
    html.append("<span style='background-color: gray;'>");
}
```

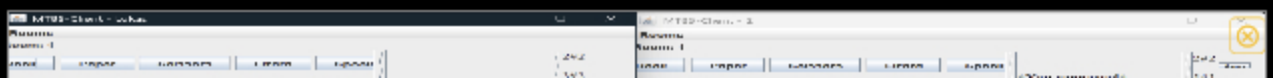
code

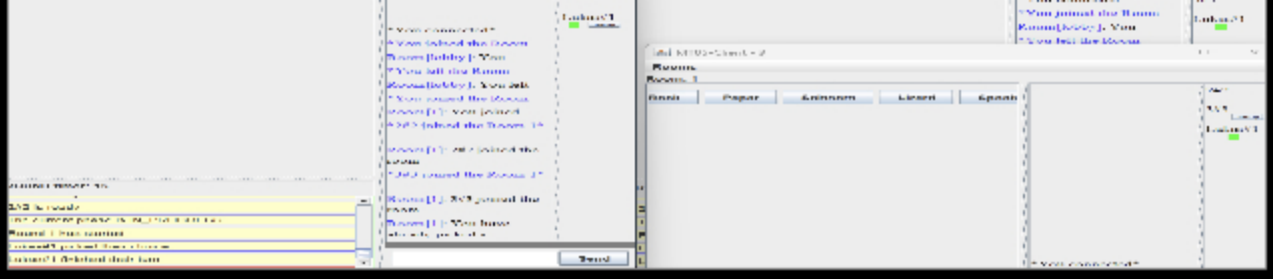
```
public void setTurn(boolean didTakeTurn) {
    turnIndicator.setBackground(didTakeTurn ? Color.GREEN : new Color(0, 0, 0, 0));
    // ...
}
```

code

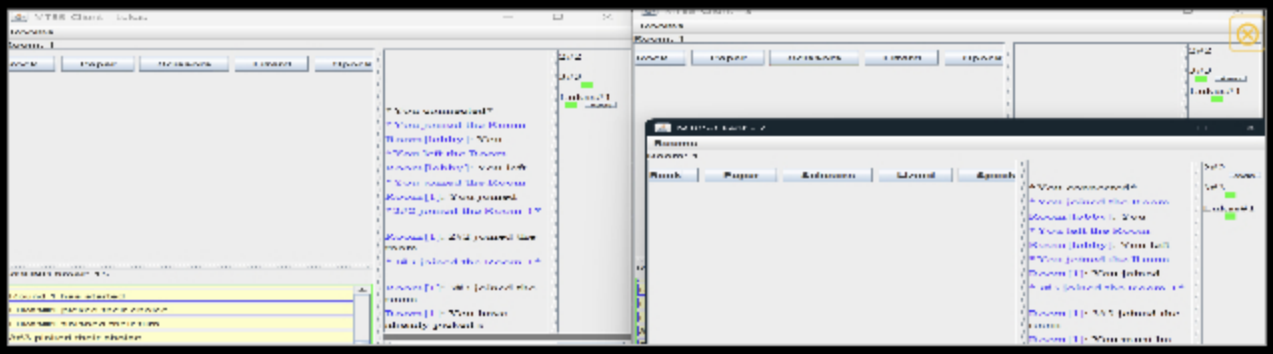


sorting and points across 3 client





pending to pick



pending to pick

Saved: 12/9/2025 3:28:38 AM

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:

`GameRoom` calculates points and broadcasts `PointsPayload`. `Client` receives it, updates the local `User` model, and notifies `UserListView` to update the specific item's points label.

`UserListView` listens for `onPointsUpdate`. When triggered, it creates a list of current items, uses a custom `Comparator` (Points DESC, Name ASC), and re-adds them to the panel.

When `GameRoom` receives a valid turn, it broadcasts a generic "Player Picked" event. `Client` parses this and calls `onTookTurn`, causing `UserListItem` to set the indicator panel to Green.

Elimination is handled server-side in `onRoundEnd`. While no specific payload sends "Eliminated", the Game Event text "X was eliminated" is displayed, and they cease to take turns.

Saved: 12/9/2025 3:28:38 AM

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

Progress: 100%

Details:

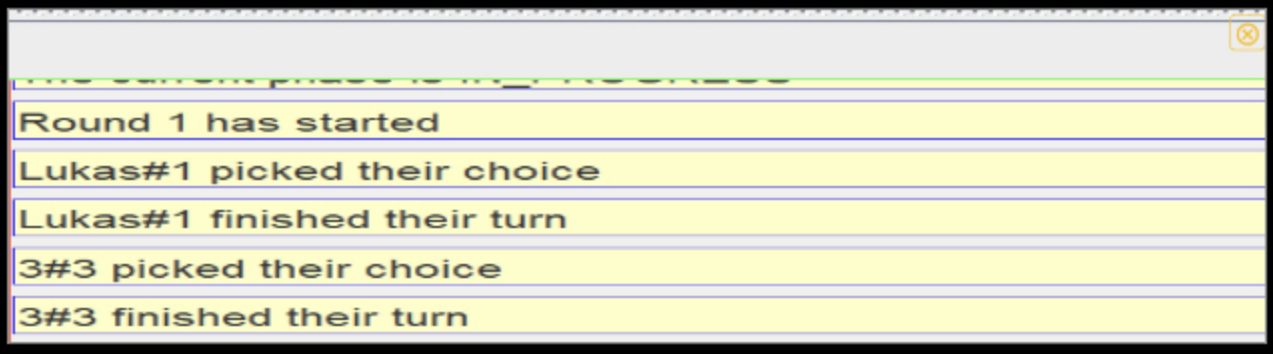
- Show the status of users picking choices
- Show the battle resolution messages from Milestone 2
  - Include messages about elimination
- Show the countdown timer for the round

Part 1:

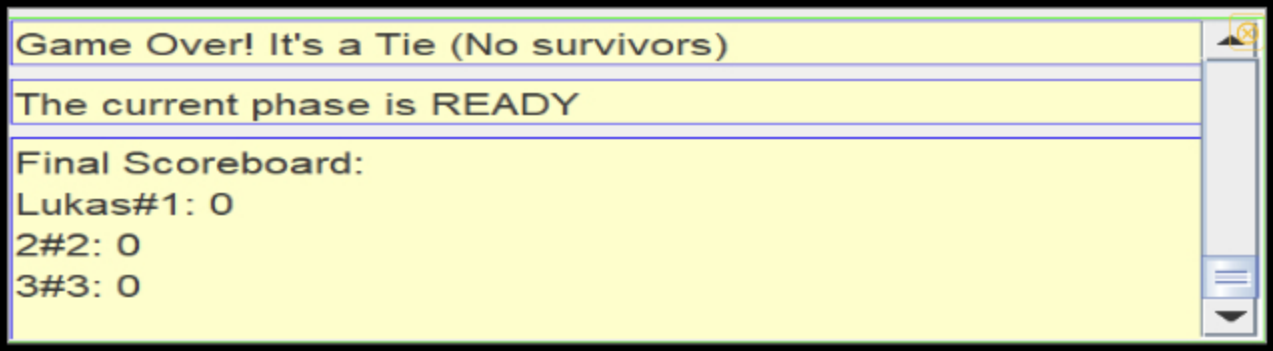
Progress: 100%

Details:

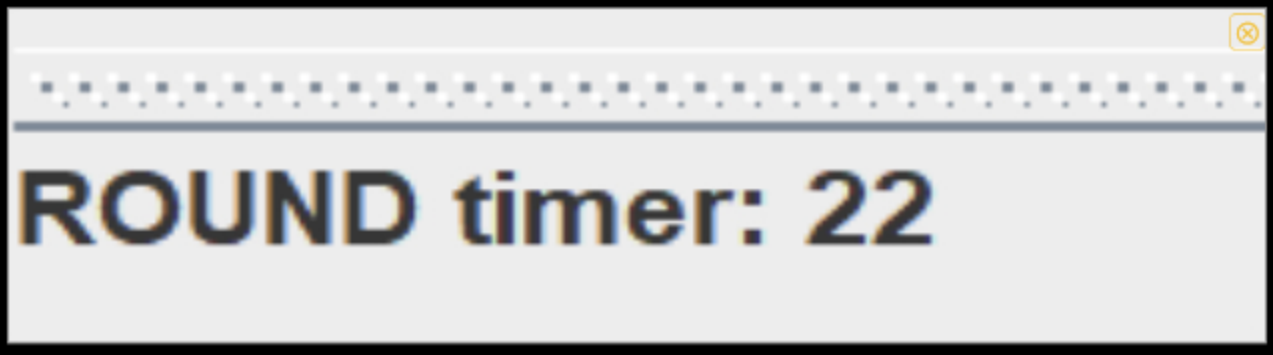
- Show various examples of each of the messages/visuals
- Show code snippets related to these messages from server-side to UI



picking



resolution



countdown

Battle Results:
Lukas#1(s) vs 3#3(s): Tie!
3#3(s) vs Lukas#1(s): Tie!

Round 2 has started

Game Over! It's a Tie (No survivors)

message examples

Battle Results:
Lukas#1(s) vs 3#3(s): Tie!
3#3(s) vs Lukas#1(s): Tie!

message examples

```
sendGameEvent(String.format("%s picked their choice", currentUser.getDisplayName
()));
sendGameEvent(battleLog.toString());
```

code

Saved: 12/9/2025 3:31:57 AM

Part 2:

Progress: 100%

Details:

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

Your Response:

`GameRoom` constructs the string and calls `sendGameEvent`, which packages it into a `MESSAGE` payload. `Client` receives it, and the `ChatGameView` appends it to the history text area.

Saved: 12/9/2025 3:31:57 AM

Task #3 (0.67 pts.) - Game Area

Progress: 100%

Details:

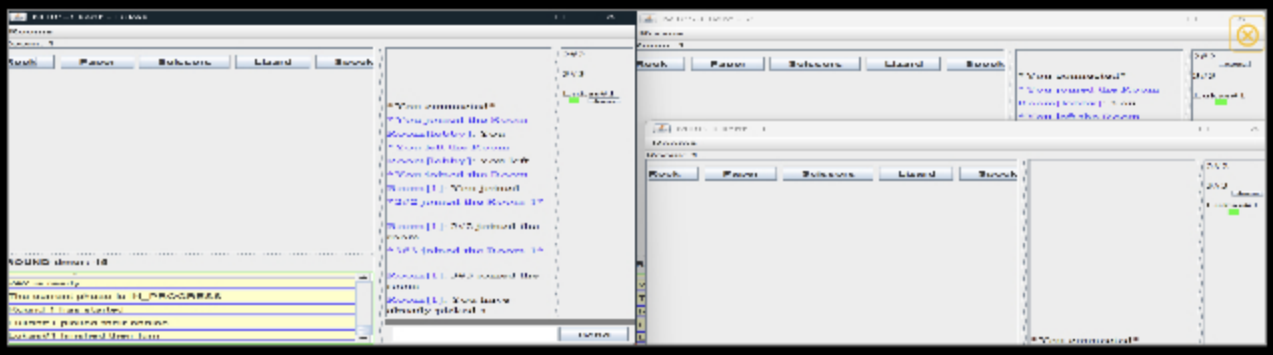
- UI should have components to allow the user to select their choice

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

```
currentUser.setLookTurn(true);

sendTurnStatus(currentUser, true);
```

code

```
public void sendDoTurn(String text) {
    ReadyPayload rp = new ReadyPayload();
    rp.setPayloadType(PayloadType.TURN);
    rp.setMessage(text);
    sendToServer(rp);
}
```

code

Saved: 12/9/2025 3:35:02 AM

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 UI button listener calls `Client.sendDoTurn("r")` (or p/s). This sends a `TURN` payload containing the choice string to the server, where `GameRoom.handleTurnAction` processes it.

The server validates the choice and sends back a `SYNC_TURN` payload. The `Client` fires `onTookTurn`, identifying the sender ID and updating their specific `UserListItem` indicator to Green.

Saved: 12/9/2025 3:35:02 AM

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

Progress: 100%

Task #1 (2 pts.) - Extra Choices

Progress: 100%

Details:

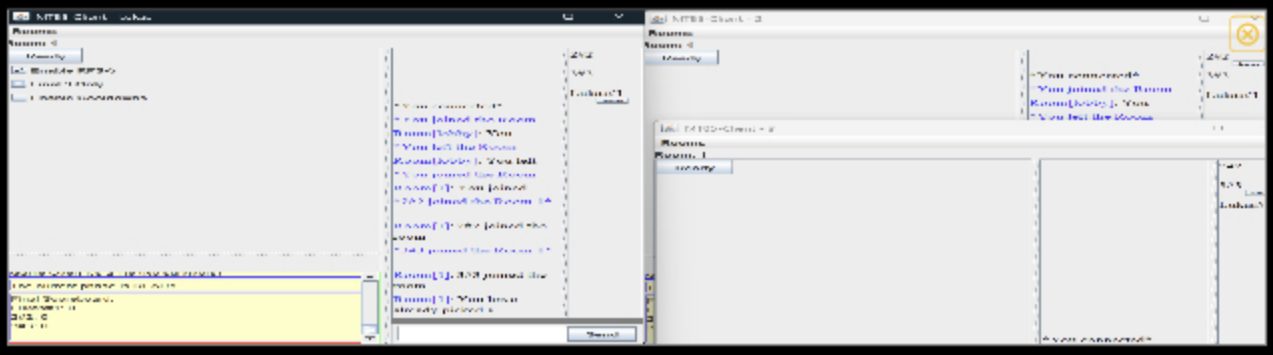
- Setting should be toggleable during Ready Check by session creator
  - (Option 1) Extra choices are available during the full session
  - (Option 2) Only activate extra options at different stages (i.e., last 3 players remaining)
- There should be at least 2 extra options for rps-5

Part 1:

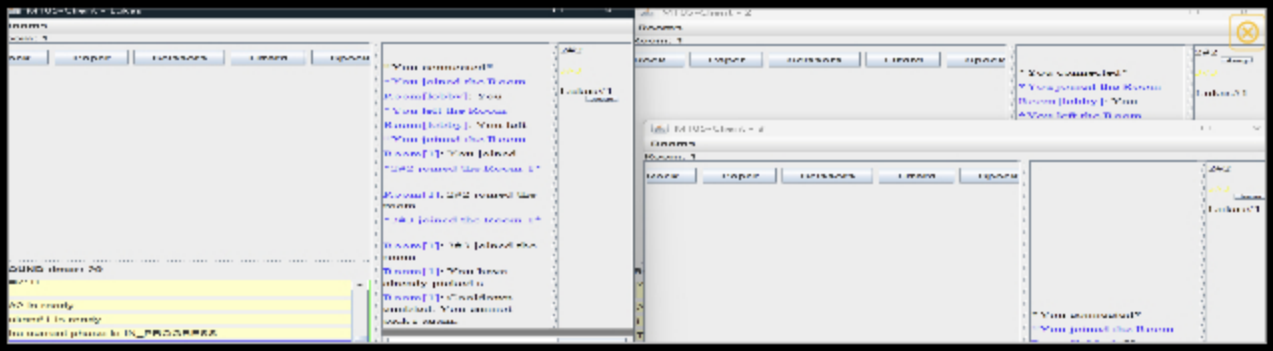
Progress: 100%

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



play screen with extra options

```
if (isRPS5Allowed && enableRPS5Final3) {
    long activeCount = clientsInRoom.values().stream().filter(...).count();
    if (activeCount > 3) isRPS5Allowed = false;
}
```

code

```
private void updateHostControls() {
    chkRPS5.setVisible(Client.INSTANCE.isHost());
    chkRPS5Final3.setVisible(Client.INSTANCE.isHost() && chkRPS5.isSelected());
}
```

code

Saved: 12/9/2025 3:41:33 AM

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:

The checkboxes send a `/settings` command. `GameRoom.handleSettings` verifies the sender is the Host before updating the room's boolean flags (`enableRPS5`, etc.) and broadcasting the change.

In `handleTurnAction`, before recording a choice, the server checks `enableRPS5`. If enabled and "Final 3" is active, it counts active players to determine if extended choices are legal.

I implemented **Option 2** (Final 3 Players), where RPS-5 options are restricted until the active player count drops to 3 or fewer.

Saved: 12/9/2025 3:41:33 AM

Task #2 (2 pts.) - Choice cooldown

Progress: 100%

Details:

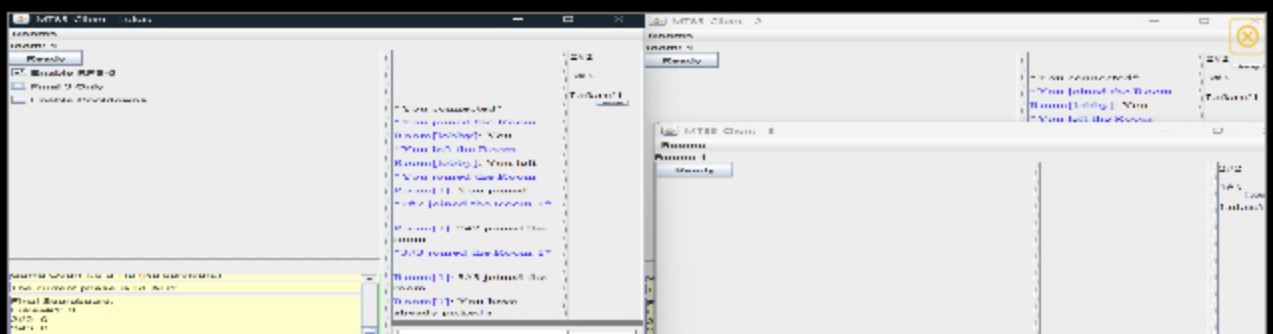
- Setting should be toggleable during Ready Check by session creator
- The choice on cooldown must be disable on the UI for the User

Part 1:

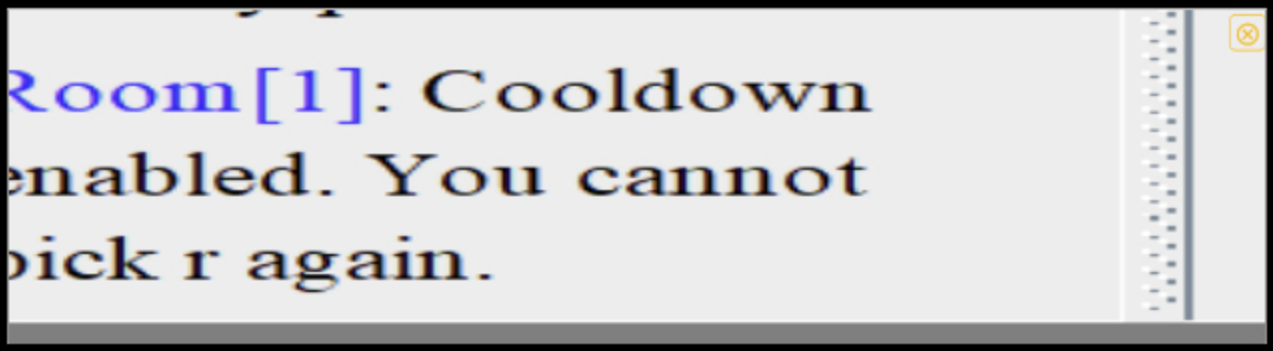
Progress: 100%

Details:

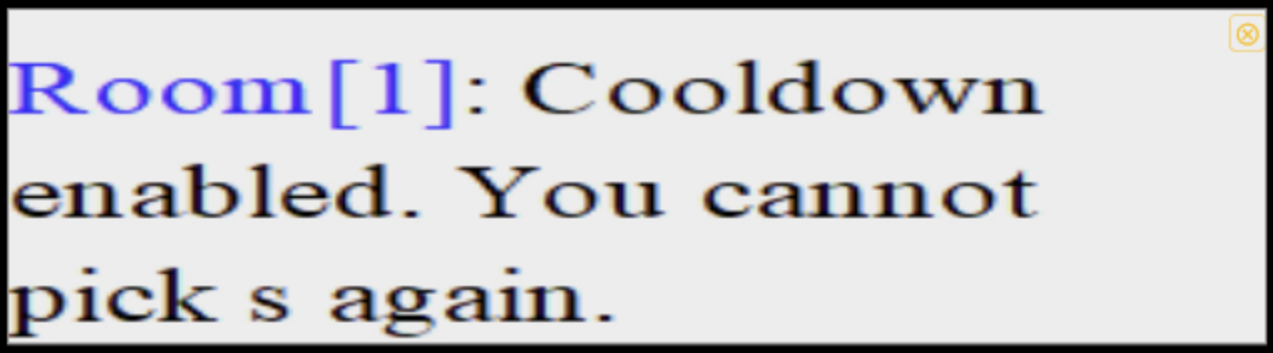
- Show the Ready Check screen with the option for the host (3+ clients must be visible)
  - Show the related code that makes this interactable only for the host
- Show a few examples of the play screen with the choice on cooldown
  - Show the related code for the UI and handling of the cooldown and server-side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



ready check



cooldown



cooldown

```
if (enableCooldown && c.equals(currentUser.getLastChoice())) {
    currentUser.sendMessage(Constants.DEFAULT_CLIENT_ID, "Cooldown active: You
    cannot pick the same option twice!");
    return;
}
```

code

```
chkCooldown.setVisible(Client.INSTANCE.isHost());
```

code

Saved: 12/9/2025 3:43:48 AM

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:

Similar to RPS-5, the Host toggles a checkbox which sends `/settings cooldown on`. The server sets `enableCooldown = true` in the `GameRoom` instance.

The server tracks `lastChoice` on the `ServerThread`. During `handleTurnAction`, it compares the incoming choice with `lastChoice`. If they match, the turn is rejected with an error message.

Saved: 12/9/2025 3:43:48 AM

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

Progress: 100%

Task #1 (1 pt.) - Away Status

Progress: 100%

Details:

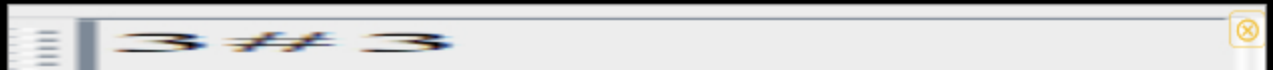
- Clients can mark themselves away and be skipped in turn flow but still part of the game
- The status should be visible to all participants
- A message should be relayed to the Game Events Panel (i.e., Bob is away or Bob is no longer away)
- The user list should have a visual representation (i.e., grayed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

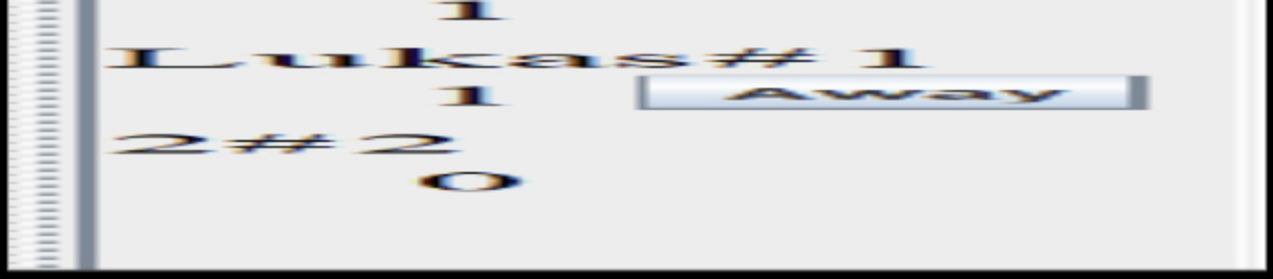
Part 1:

Progress: 100%

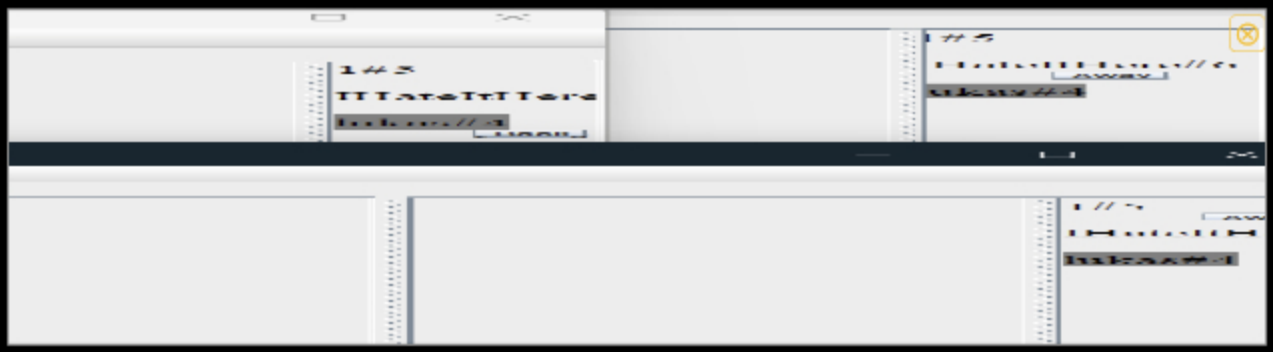
Details:

- 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





away



away 3 client

```
public void sendAwayAction() {
    Payload payload = new Payload();
    payload.setPayloadType(PayloadType.AWAY);
    sendToServer(payload);
}
```

code

```
sender.setAway(isAway);
clientsInRoom.values().forEach(client -> {
    client.sendClientInfo(..., sender.isAway());
});
```

code

```
clientsInRoom.values().stream()
    .filter(client -> ... && !client.isAway())
    .forEach(client -> client.setEliminated(true));
```

code

```
sendGameEvent(sender.getDisplayName() + (isAway ? " is now Away" : " is no longer Away"));
```

code

Saved: 12/9/2025 3:49:57 AM

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:

Clicking "Away" sends an `AWAY` payload. The server toggles the `isAway` flag on the `ServerThread`, broadcasts a `SYNC_CLIENT` payload to update visuals, and sends a Game Event text message.

In `onRoundEnd`, the stream filter specifically checks `!client.isAway()`. This ensures that away players are excluded from "Did Not Pick" elimination checks and battle pair generation.

Saved: 12/9/2025 3:49:57 AM

Task #2 (1 pt.) - Spectators

Progress: 100%

Details:

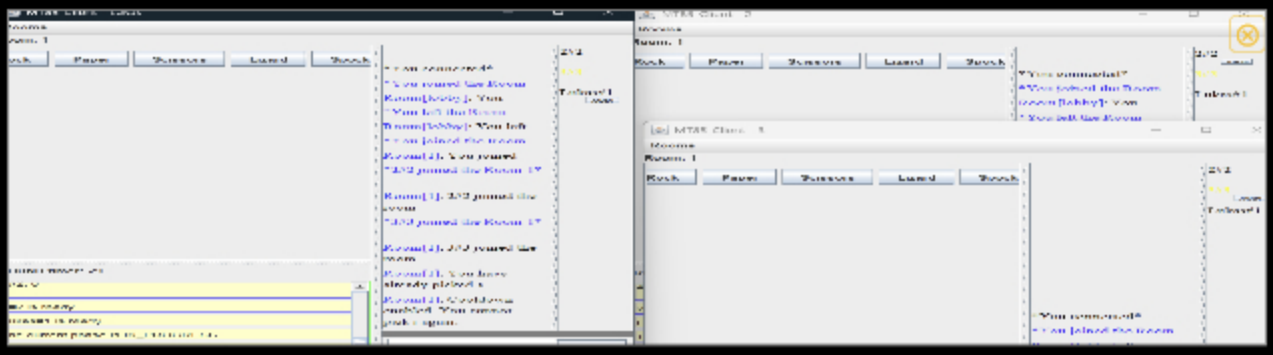
- Spectators are users who didn't mark themselves ready
  - Optionally you can include a toggle on the Ready Check page
- The can see all chat but are ignored from turn/round actions and can't send messages
- Spectators will have a visual representation in the user list to distinguish them from other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

Part 1:

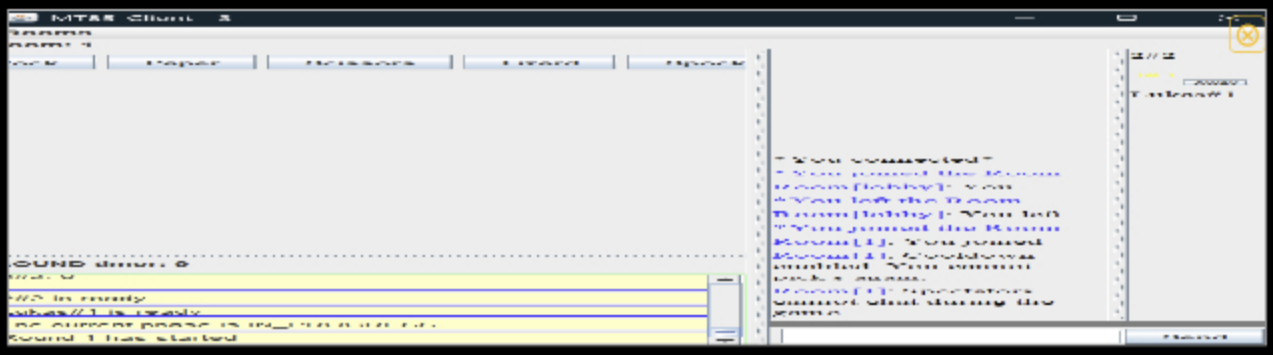
Progress: 100%

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)



visaul spectator across 3 client



chat message about spectator and spec view

```
// They are only excluded from logic processing.

clientsInRoom.values().forEach(...) // Includes spectators
```

code

```
if (currentPhase == Phase.IN_PROGRESS && sender.isSpectator()) {
    sender.sendMessage(Constants.DEFAULT_CLIENT_ID, "Spectators cannot chat during
```

```
    the game.");
    return;
}
```

code

```
filter(client -> ... && !client.isSpectator())
```

code

```
String formattedMessage = String.format("%s %s%s", client.getDisplayName(),
didJoin ? "joined" : "left", suffix);
broadcast(formattedMessage);
```

code

Saved: 12/9/2025 3:54:33 AM

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:

If a user joins during `IN_PROGRESS`, they are flagged as a spectator via the join message. Since they are implicitly `!isReady`, the server treats them as non-participants.

Similar to Away status, `onRoundEnd` and `processBattles` filter the client stream with `!client.isSpectator()`, preventing them from being processed for wins/losses/elimination.

`GameRoom` overrides `handleMessage`. It intercepts chat payloads and checks `isSpectator()`. If true, it blocks the relay and replies with a private error message to the sender.

The system is built on broadcasting to the `clientsInRoom` map. Since spectators are in this map, they automatically receive all `GAME_EVENT` and `TIMER` payloads without requiring extra "opt-in" code.

Saved: 12/9/2025 3:54:33 AM

Section #5: (1 pt.) Misc

Progress: 100%

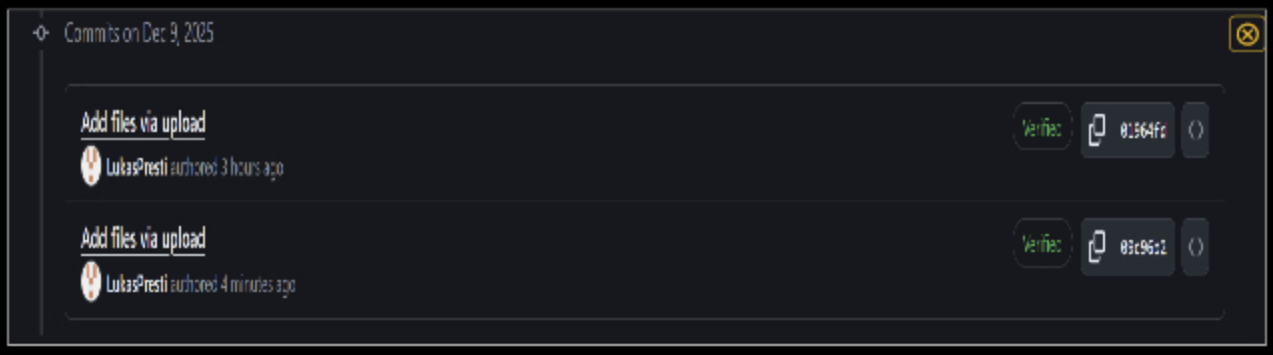
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

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



i think this is right. i still have no idea

Saved: 12/9/2025 4:01:19 AM

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/LukasPresti/lap5-it1114-007/compare/main%40%7B1day%7D...main

URL
https://github.com/LukasPresti/la

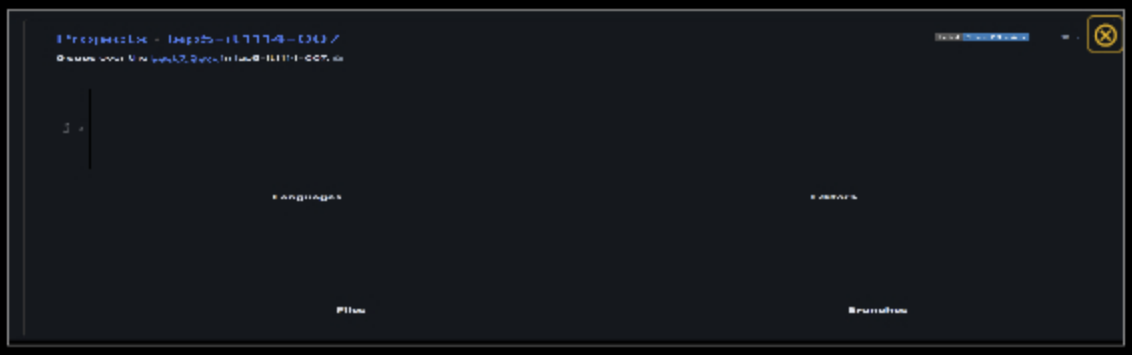
Saved: 12/9/2025 4:01:19 AM

Task #2 (0.33 pts.) - WakaTime - Activity

Progress: 100%

Details:

- Visit the WakaTime.com Dashboard
- Click `Projects` and find your repository
- Capture the overall time at the top that includes the repository name
- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary



wakatime STILL broken. I tried everything to fix it

Saved: 12/9/2025 4:02:16 AM

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:

Not to wait until the last minute bc this sucks.

Saved: 12/9/2025 4:02:37 AM

Task #2 (0.33 pts.) - What was the easiest part of the assignment?

assignment?

Progress: 100%

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

clicking the submit button in about 30seconds from writing this.

Saved: 12/9/2025 4:02:55 AM

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:

literally everything else. Just taking the screenshots of everything and adding them to this toook about 2 hours.

Saved: 12/9/2025 4:03:38 AM