Submission Worksheet

Submission Data

Course: IT114-450-M2025
Assignment: IT114 - Milestone 3 - RPS
Student: Lamia M. (lm87)
Status: In Progress | **Worksheet Progress:** 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Started: 8/11/2025 2:08:18 AM
Updated: 8/11/2025 11:10:13 PM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-3-rps/grading/lm87
View Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-3-rps/view/lm87

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

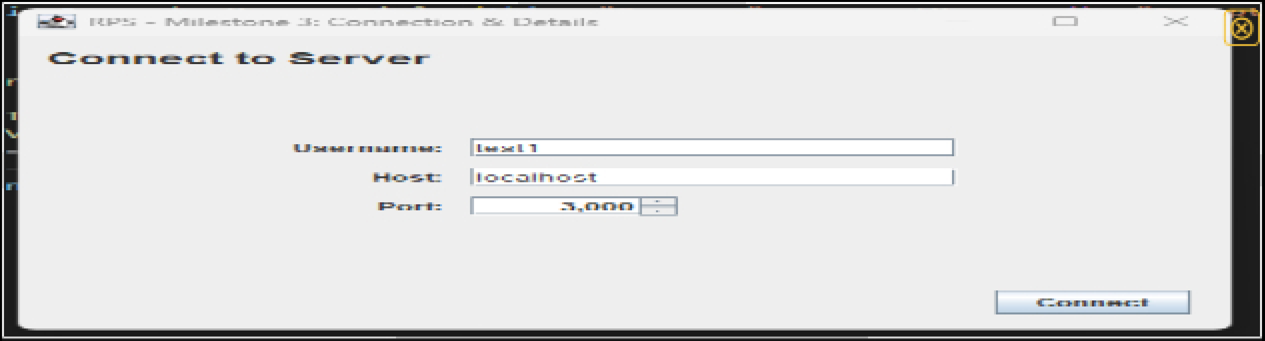
Progress: 100%

Part 1:

Progress: 100%

Details:

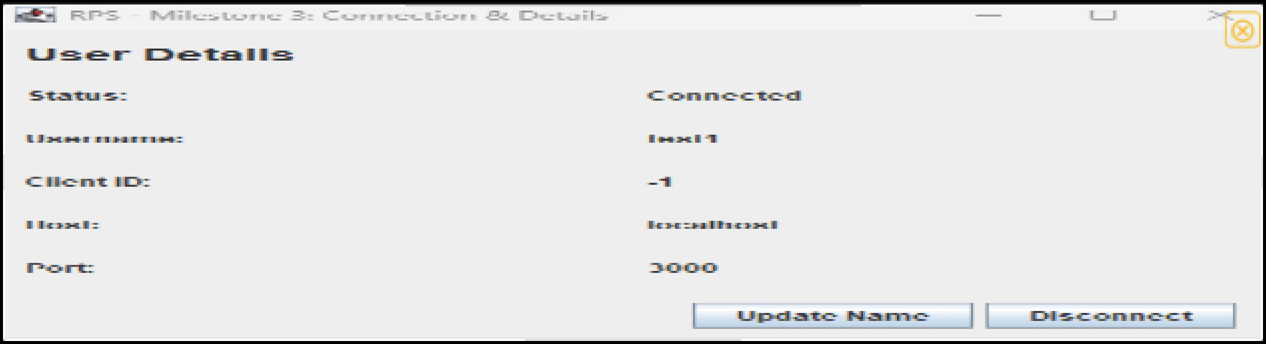
- Show the connection panel with valid data
- Show the user details panel with valid data


RPS – Milestone 3: Connection & Details

Connect to Server

Username:	test1
Host:	localhost
Port:	3,000

Connection Panel Window


RPS – Milestone 3: Connection & Details

User Details

Status:	Connected
Username:	test1
Client ID:	-1
Host:	localhost
Port:	3000

User Details Panel

Saved: 8/11/2025 2:17:06 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:

When the user enters their username, host, and port in the Connection Panel and clicks connect, the panel reads those values and calls methods on the Client singleton to store the username locally and open a connection to the server. The uiConnect method remembers the host and port, starts the socket connection, and launches the listener thread that handles messages from the server. Immediately after connecting, the client sends a handshake payload to the server containing the username, and the server stores this information in the user's session, assigns them to the default “lobby” room, and announces their arrival to other clients in that room. Once connected, the UI switches to the User Details Panel, which calls refreshFromClient() to read the stored username, client ID, host, and port from the Client object and display them. If the user clicks disconnect, the details panel calls shutdown() on the client, which stops the listener thread, closes the network streams, and marks the client as no longer running, after which the UI switches back to the Connection Panel.

Saved: 8/11/2025 2:17:06 AM

Task #2 (0.50 pts.) - Ready Panel

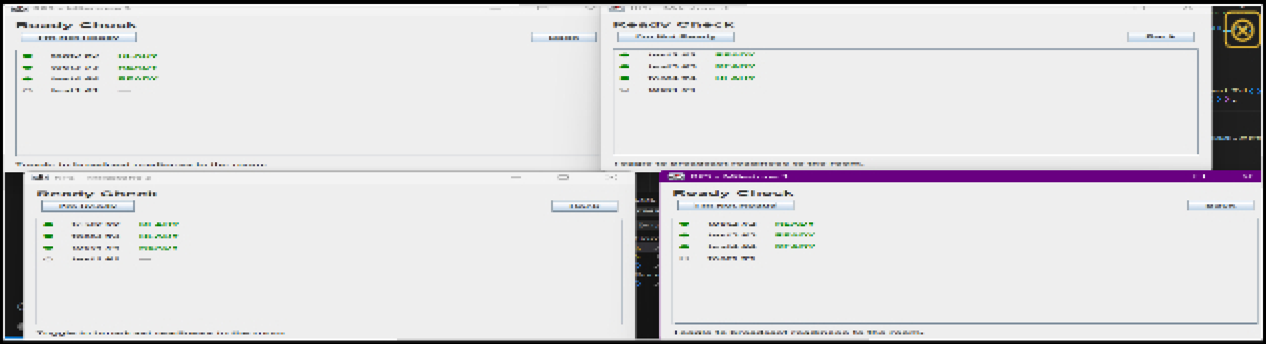
Progress: 100%

Part 1:

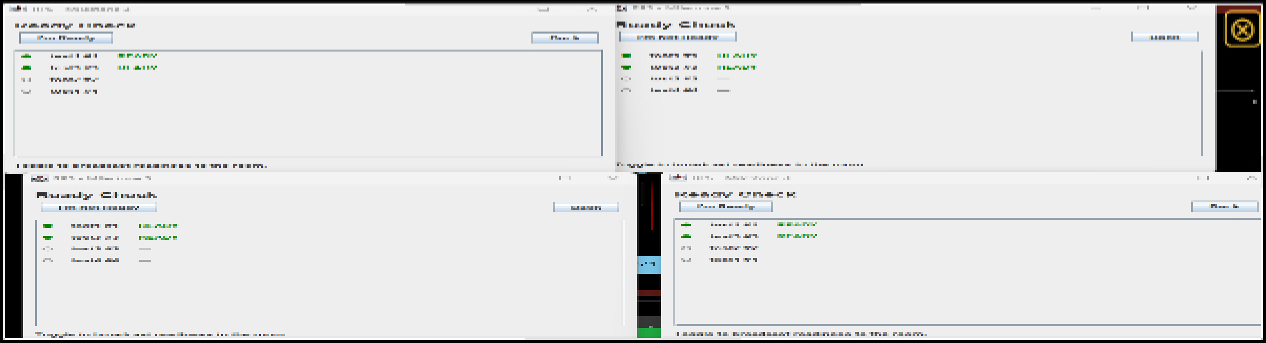
Progress: 100%

Details:

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



variation 1 of 4 clients being ready



variation 2 of 4 clients being ready

Saved: 8/11/2025 3:13:35 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:

When a player clicks the I'm Ready button in the Ready Panel, the panel toggles its local myReady flag and calls Client.uiToggleReady(ready). This method stores the player's readiness in the readyMap and sends a normal chat message to the server in the format [READY] <0|1>. The server broadcasts this message to all connected clients in the same room, so every client receives the READY status change.

When a client receives any chat message, processMessage() runs. It checks if the message contains [READY], extracts the client ID and readiness value, and updates the local readyMap. The Ready Panel's Swing timer (refreshing every 500 ms) calls refreshList(), which reads readyMap and knownClients to rebuild the on-screen list, so the READY or not-ready indicator is updated for all players in real time.

Saved: 8/11/2025 3:13:35 AM

Section #2: (2 pts.) Project Ui

Progress: 100%

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

Progress: 100%

Details:

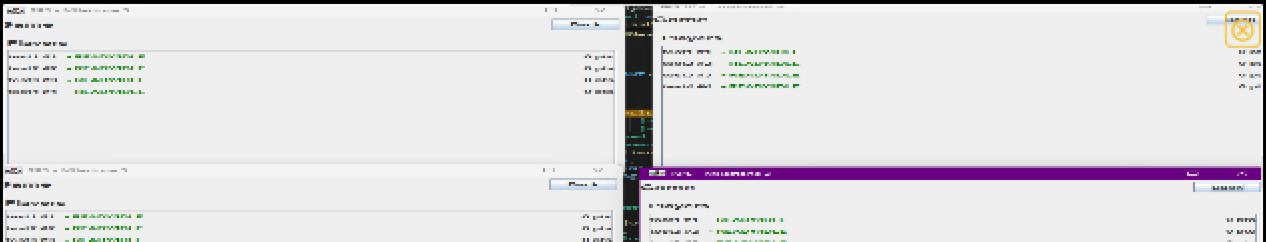
- Show the username and id of each user
- Show the current points of each user
- Users should appear in score order, sub-sort by name when ties occur
- Pending-to-pick users should be marked accordingly
- Eliminated users should be marked accordingly

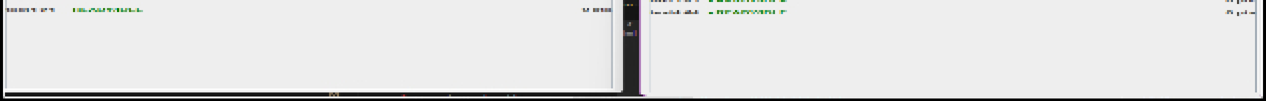
Part 1:

Progress: 100%

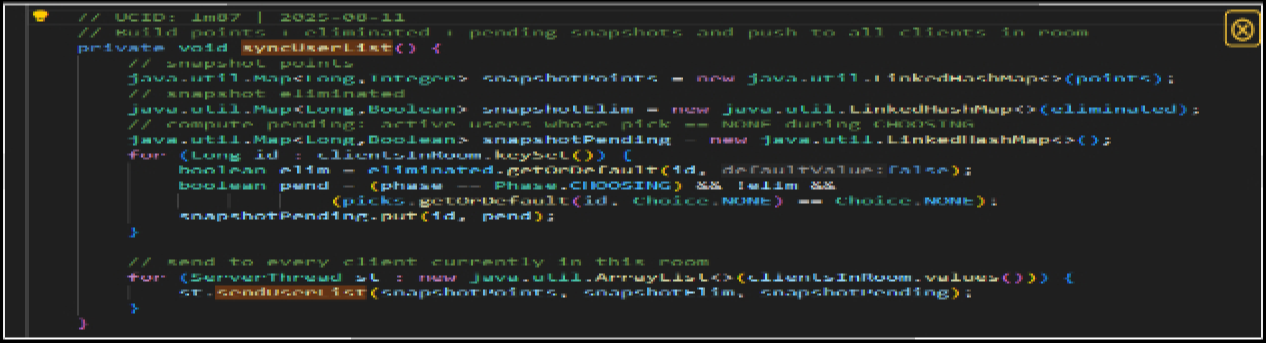
Details:

- 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



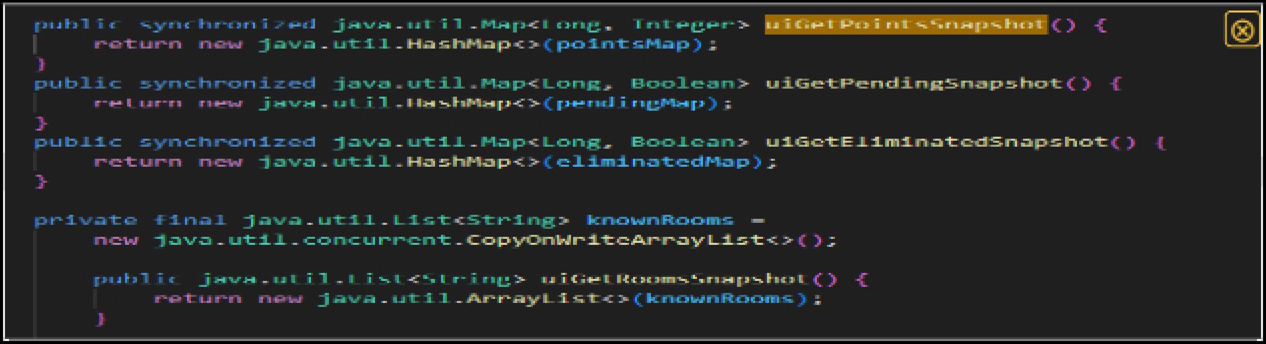


screenshots of clients



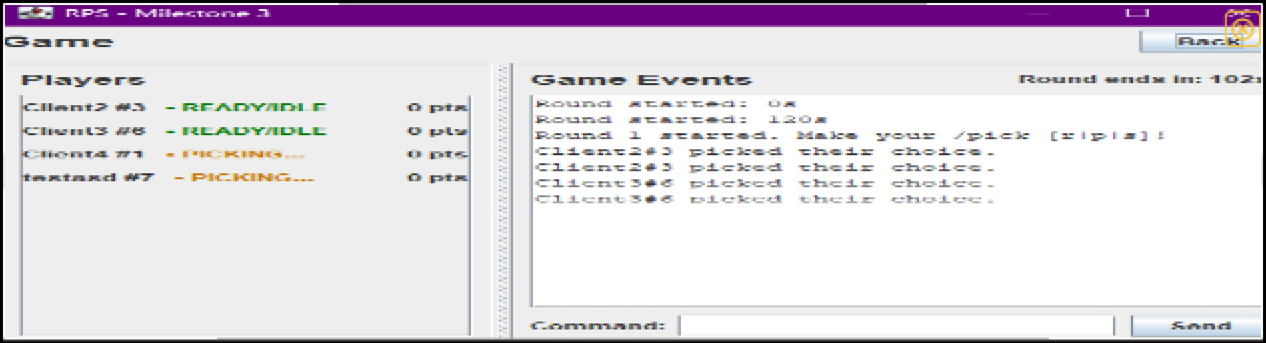
```java
// UCID: lm87 | 2025-08-11
// Build points + eliminated + pending snapshots and push to all clients in room
private void syncUserList() {
    // snapshot points
    java.util.Map<Long,Integer> snapshotPoints = new java.util.LinkedHashMap<>(points);
    // snapshot eliminated
    java.util.Map<Long,Boolean> snapshotElim = new java.util.LinkedHashMap<>(eliminated);
    // compute pending: active users whose pick == NONE during CHOOSING
    java.util.Map<Long,Boolean> snapshotPending = new java.util.LinkedHashMap<>();
    for (Long id : clientsInRoom.keySet()) {
        boolean elim = eliminated.getOrDefault(id, defaultValue:false);
        boolean pend = (phase == Phase.CHOOSING) && !elim &&
                (picks.getOrDefault(id, Choice.NONE) == Choice.NONE);
        snapshotPending.put(id, pend);
    }

    // send to every client currently in this room
    for (ServerThread st : new java.util.ArrayList<>(clientsInRoom.values())) {
        st.sendUserList(snapshotPoints, snapshotElim, snapshotPending);
    }
}
```

code snippet of sync User List to update the UI



```java
public synchronized java.util.Map<Long, Integer> uiGetPointsSnapshot() {
    return new java.util.HashMap<>(pointsMap);
}
public synchronized java.util.Map<Long, Boolean> uiGetPendingSnapshot() {
    return new java.util.HashMap<>(pendingMap);
}
public synchronized java.util.Map<Long, Boolean> uiGetEliminatedSnapshot() {
    return new java.util.HashMap<>(eliminatedMap);
}

private final java.util.List<String> knownRooms =
    new java.util.concurrent.CopyOnWriteArrayList<>();

    public java.util.List<String> uiGetRoomsSnapshot() {
        return new java.util.ArrayList<>(knownRooms);
    }
```

code snippet of ui snapshots

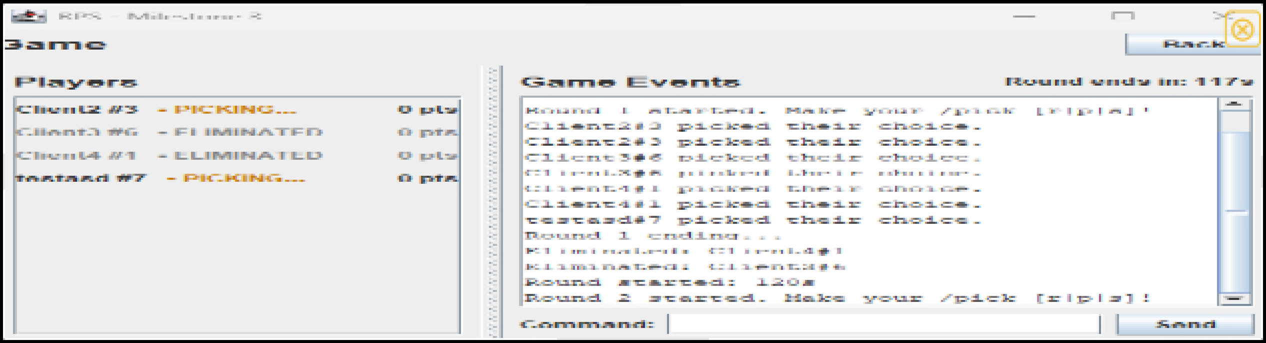


```
Client2 #3   - READY/IDLE     0 pts
Client3 #6   - READY/IDLE     0 pts
Client4 #1   - PICKING...     0 pts
testasd #7   - PICKING...     0 pts
```

Round ends in: 102s

```
Round started: 0s
Round started: 120s
Round 1 started. Make your /pick [r|p|s]!
Client2#3 picked their choice.
Client2#3 picked their choice.
Client3#6 picked their choice.
Client3#6 picked their choice.
```

screenshot of events situation 1



```
Client2 #3   - PICKING...     0 pts
Client3 #6   - ELIMINATED     0 pts
Client4 #1   - ELIMINATED     0 pts
testasd #7   - PICKING...     0 pts
```

Round ends in: 117s

```
Round 1 started. Make your /pick [r|p|s]!
Client2#3 picked their choice.
Client2#3 picked their choice.
Client3#6 picked their choice.
Client3#6 picked their choice.
Client4#1 picked their choice.
Client4#1 picked their choice.
testasd#7 picked their choice.
Round 1 ending...
Eliminated: Client4#1
Eliminated: Client3#6
Round started: 120s
Round 2 started. Make your /pick [r|p|s]!
```

screenshot of events situation 2

Saved: 8/11/2025 5:28:16 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:

Points updates (server → UI): When points change (e.g., a winner is awarded in GameRoom), the server calls syncUserList(). That method snapshots points, builds a UserListPayload (with points, plus the current eliminated/pending flags), and sends it to every client via ServerThread.sendUserList(...). On the client, processPayload(USER_LIST) copies the incoming map into pointsMap. The UserListPanel polls uiGetPointsSnapshot() on a short Swing timer and redraws rows, so scores on-screen update immediately for all players.

User list sorting (in the UI): During each refresh, UserListPanel merges knownClients with pointsMap and sorts a local list: first by points descending, then by player name ascending (case-insensitive) to break ties, then by id as a final tiebreaker. It then rebuilds the list of rows, so every client renders the same stable order as long as they receive the same USER_LIST snapshot.

Pending-to-pick indicators (server → UI): At onRoundStart(), the server sets phase=CHOOSING and initializes each active player's picks[id]=NONE. In syncUserList(), the server computes pending[id] = (phase==CHOOSING && !eliminated[id] && picks[id]==NONE) and includes that boolean map in the UserListPayload. The client stores it in pendingMap when handling USER_LIST, and UserListPanel shows "PICKING…" for those ids. When a player picks (handlePick()), the server updates picks, calls syncUserList() again, and the client flips that row to "READY/IDLE".

Elimination indicators (server → UI): During onRoundEnd(), the server marks non-pickers and battle losers with eliminated[id]=true, then calls syncUserList(). That payload carries the updated eliminated map; the client writes it into eliminatedMap. On refresh, UserListPanel renders those rows grayed with "ELIMINATED" (and they're no longer considered pending). On session reset, the server clears state and calls syncUserList() so the UI removes elimination marks.

Saved: 8/11/2025 5:28:16 AM

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

Progress: 100%

Details:

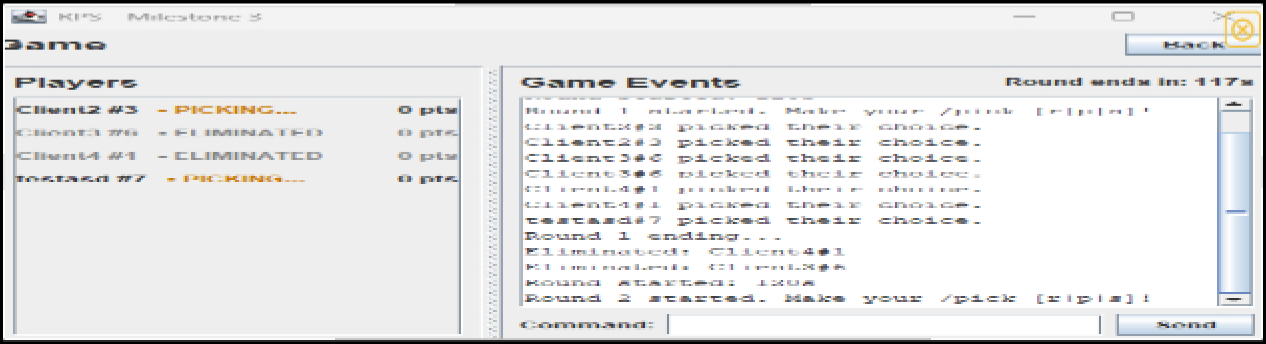
- Show the status of users picking choices
- Show the battle resolution messages from Milestone 2
  - Include messages about elimination
- Show the countdown timer for the round

Part 1:

Progress: 100%

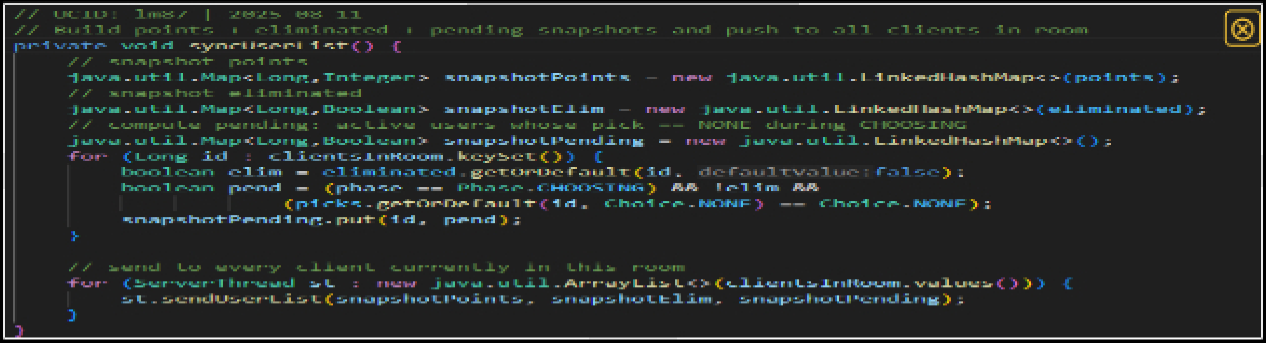
Details:

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



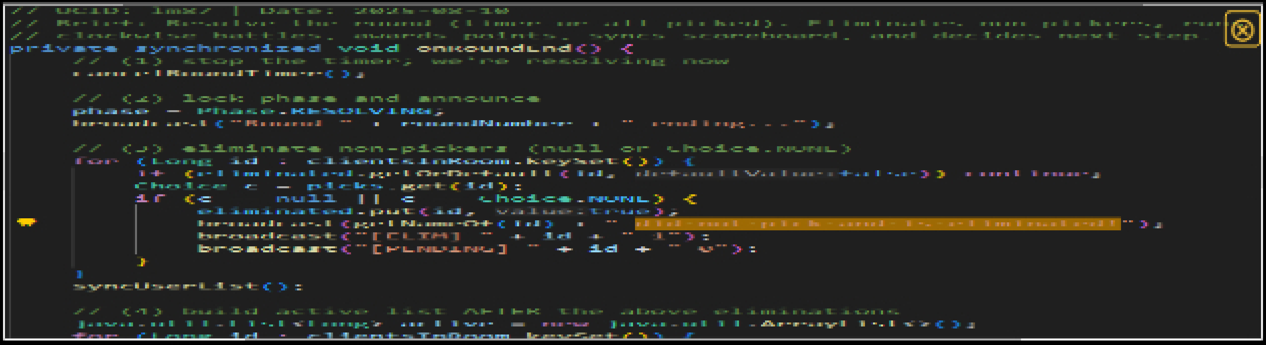
```
RPS - Milestone 3
Game                                                     Back
Players                          Game Events                    Round ends in: 117s
Client2 #3  - PICKING...  0 pts  Round 1 started. Make your /pick [r|p|s]!
Client3 #6  - ELIMINATED  0 pts  ClientX#3 picked their choice.
Client4 #1  - ELIMINATED  0 pts  Client2#3 picked their choice.
testasd #7  - PICKING...  0 pts  Client3#6 picked their choice.
                                 Client3#6 picked their choice.
                                 Client4#1 picked their choice.
                                 Client4#1 picked their choice.
                                 testasd#7 picked their choice.
                                 Round 1 ending...
                                 Eliminated: Client4#1
                                 Eliminated: ClientX#6
                                 Round started: 120s
                                 Round 2 started. Make your /pick [r|p|s]!
                                 Command:                                  Send
```

countdown timer + battle logs and various events being shown in the "game events" window



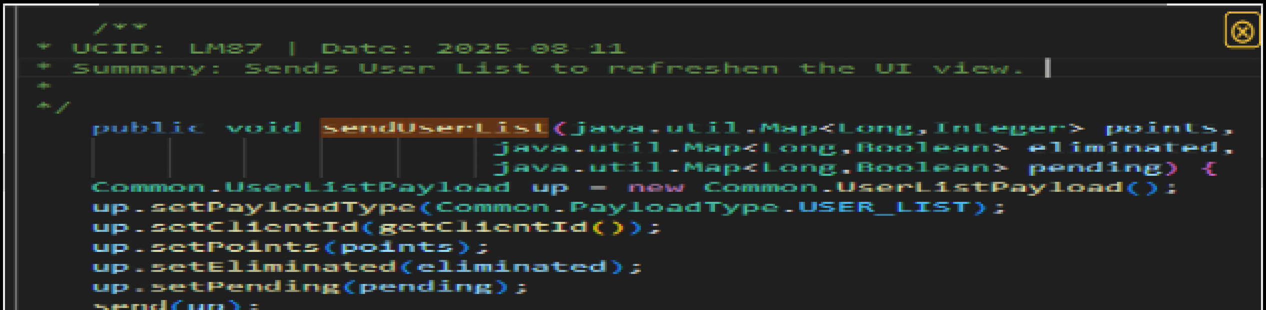
```java
// UCID: lm87 | 2025 08 11
// Build points + eliminated + pending snapshots and push to all clients in room
private void syncUserList() {
    // snapshot points
    java.util.Map<Long,Integer> snapshotPoints = new java.util.LinkedHashMap<>(points);
    // snapshot eliminated
    java.util.Map<Long,Boolean> snapshotElim = new java.util.LinkedHashMap<>(eliminated);
    // compute pending: active users whose pick == NONE during CHOOSING
    java.util.Map<Long,Boolean> snapshotPending = new java.util.LinkedHashMap<>();
    for (Long id : clientsInRoom.keySet()) {
        boolean elim = eliminated.getOrDefault(id, defaultValue:false);
        boolean pend = (phase == Phase.CHOOSING) && !elim &&
                    (picks.getOrDefault(id, Choice.NONE) == Choice.NONE);
        snapshotPending.put(id, pend);
    }

    // send to every client currently in this room
    for (ServerThread st : new java.util.ArrayList<>(clientsInRoom.values())) {
        st.sendUserList(snapshotPoints, snapshotElim, snapshotPending);
    }
}
```

snippet of syncUserList

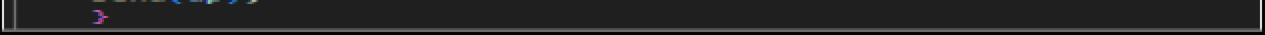


```java
// UCID: lm87 | Date: 2025-08-10
// Brief: Resolve the round (timer or all picked). Eliminate non pickers, run
// clockwise battles, awards points, syncs scoreboard, and decides next step.
private synchronized void onRoundEnd() {
    // (1) stop the timer; we're resolving now
    cancelRoundTimer();

    // (2) lock phase and announce
    phase = Phase.RESOLVING;
    broadcast("Round " + roundNumber + " ending...");

    // (3) eliminate non-pickers (null or Choice.NONE)
    for (Long id : clientsInRoom.keySet()) {
        if (eliminated.getOrDefault(id, defaultValue:false)) continue;
        Choice c = picks.get(id);
        if (c == null || c == Choice.NONE) {
            eliminated.put(id, value:true);
            broadcast(getNameOf(id) + " did not pick and is eliminated!");
            broadcast("[ELIM] " + id + " 1");
            broadcast("[PENDING] " + id + " 0");
        }
    }
    syncUserList();

    // (4) build active list AFTER the above eliminations
    java.util.List<Long> active = new java.util.ArrayList<>();
    for (Long id : clientsInRoom.keySet()) {
```

code snippet of UI updates on onRoundEnd()

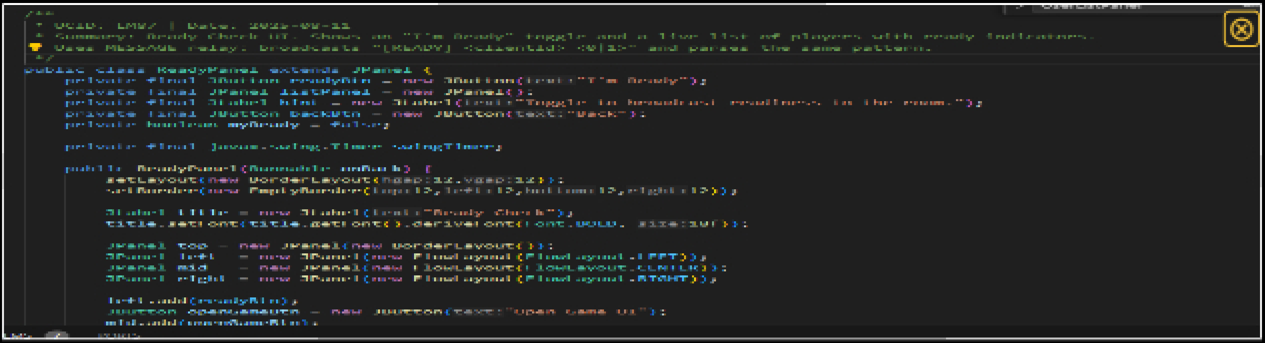


```java
    /**
 * UCID: LM87 | Date: 2025-08-11
 * Summary: Sends User List to refreshen the UI view.
 *
 */
    public void sendUserList(java.util.Map<Long,Integer> points,
                             java.util.Map<Long,Boolean> eliminated,
                             java.util.Map<Long,Boolean> pending) {
    Common.UserListPayload up = new Common.UserListPayload();
    up.setPayloadType(Common.PayloadType.USER_LIST);
    up.setClientId(getClientId());
    up.setPoints(points);
    up.setEliminated(eliminated);
    up.setPending(pending);
    send(up);
```



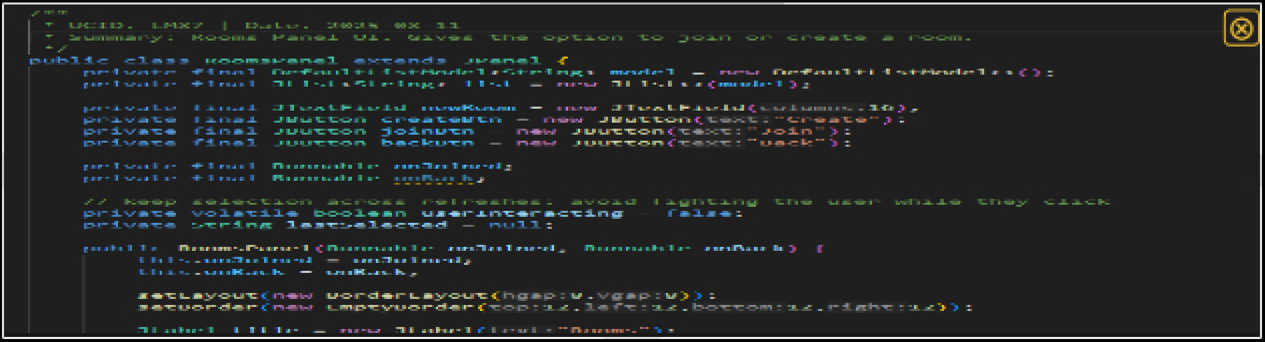
```
    }
```

code snippet of sendUserList()



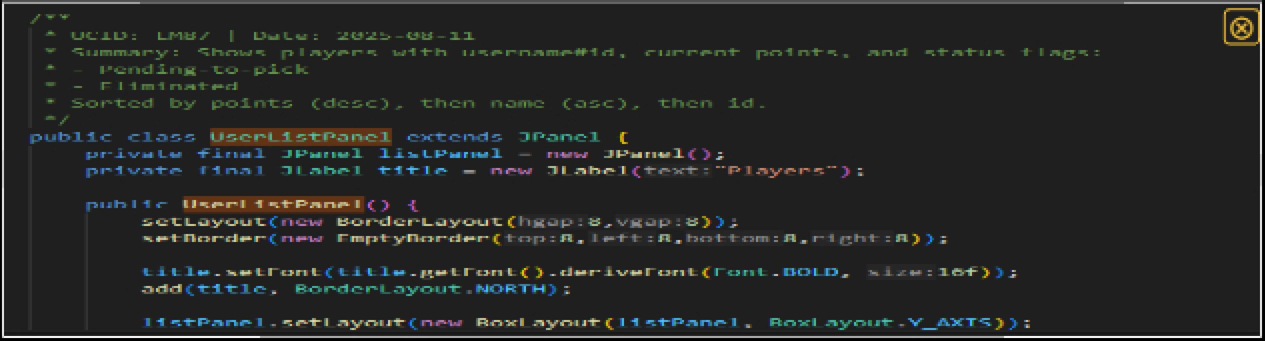
```
/**
 * UCID: LM87 | Date: 2025-08-11
 * Summary: Ready Check UI. Shows an "I'm Ready" toggle and a live list of players with ready indicators.
 * Uses MESSAGE relay: broadcasts "[READY] <clientId> <0|1>" and parses the same pattern.
 */
public class ReadyPanel extends JPanel {
    private final JButton readyBtn = new JButton(text:"I'm Ready");
    private final JPanel listPanel = new JPanel();
    private final JLabel hint = new JLabel(text:"Toggle to broadcast readiness to the room.");
    private final JButton backBtn = new JButton(text:"Back");
    private boolean myReady = false;

    private final javax.swing.Timer swingTimer;

    public ReadyPanel(Runnable onBack) {
        setLayout(new BorderLayout(hgap:12,vgap:12));
        setBorder(new EmptyBorder(top:12,left:12,bottom:12,right:12));

        JLabel title = new JLabel(text:"Ready Check");
        title.setFont(title.getFont().deriveFont(Font.BOLD, size:18f));

        JPanel top = new JPanel(new BorderLayout());
        JPanel left  = new JPanel(new FlowLayout(FlowLayout.LEFT));
        JPanel mid   = new JPanel(new FlowLayout(FlowLayout.CENTER));
        JPanel right = new JPanel(new FlowLayout(FlowLayout.RIGHT));

        left.add(readyBtn);
        JButton openGameBtn = new JButton(text:"Open Game UI");
        mid.add(openGameBtn);
```

Code Snippet for "Ready Pane"



```
/**
 * UCID: LM87 | Date: 2025-08-11
 * Summary: Rooms Panel UI. Gives the option to join or create a room.
 */
public class RoomsPanel extends JPanel {
    private final DefaultListModel<String> model = new DefaultListModel<>();
    private final JList<String> list = new JList<>(model);

    private final JTextField newRoom = new JTextField(columns:16);
    private final JButton createBtn = new JButton(text:"Create");
    private final JButton joinBtn = new JButton(text:"Join");
    private final JButton backBtn = new JButton(text:"Back");

    private final Runnable onJoined;
    private final Runnable onBack;

    // Keep selection across refreshes; avoid fighting the user while they click
    private volatile boolean userInteracting = false;
    private String lastSelected = null;

    public RoomsPanel(Runnable onJoined, Runnable onBack) {
        this.onJoined = onJoined;
        this.onBack = onBack;

        setLayout(new BorderLayout(hgap:8,vgap:8));
        setBorder(new EmptyBorder(top:12,left:12,bottom:12,right:12));

        JLabel title = new JLabel(text:"Rooms");
```

code snippet for RoomsPanel



```
/**
 * UCID: LM87 | Date: 2025-08-11
 * Summary: Shows players with username#id, current points, and status flags:
 *  - Pending-to-pick
 *  - Eliminated
 * Sorted by points (desc), then name (asc), then id.
 */
public class UserListPanel extends JPanel {
    private final JPanel listPanel = new JPanel();
    private final JLabel title = new JLabel(text:"Players");

    public UserListPanel() {
        setLayout(new BorderLayout(hgap:8,vgap:8));
        setBorder(new EmptyBorder(top:8,left:8,bottom:8,right:8));

        title.setFont(title.getFont().deriveFont(Font.BOLD, size:16f));
        add(title, BorderLayout.NORTH);

        listPanel.setLayout(new BoxLayout(listPanel, BoxLayout.Y_AXIS));
```

code snippet for users list panel

Saved: 8/11/2025 5:42:01 AM

Part 2:

Progress: 100%

Details:

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

Your Response:

When a round starts, the server's GameRoom.onRoundStart() sets phase=CHOOSING, initializes each active player's pick to NONE, and broadcasts two things: a chat-style line (e.g., "Round 3 started…") and a structured marker "[ROUND_START] " so clients can start a local countdown. It then calls syncUserList(), which snapshots points, eliminated, and “pending-to-pick” (derived from phase==CHOOSING and pick==NONE) and sends them to everyone via a USER_LIST payload (ServerThread.sendUserList(...)). As players choose, handlePick(...) records the choice, broadcasts “X picked their choice.”, and calls syncUserList() again so “PICKING...” flips to “READY/IDLE.” When resolving, onRoundEnd() broadcasts battle/elimination lines (“Eliminated: ...” or “did not pick and is eliminated!”), updates points for any winner, and calls syncPoints() (scoreboard) and syncUserList() (pending/eliminated) to keep clients in sync.

On the client, processMessage(...) funnels interesting server messages (round start/end, “picked their choice”, elimination, “Game over!”, etc.) into an in-memory event log and, when it sees "[ROUND_START]", it computes the round’s end time for a live countdown. In parallel, processPayload(USER_LIST) replaces local maps for points, pending, and eliminated with the server snapshots. The UI simply reads those models on a short Swing timer: GameEventsPanel renders the event log and the ticking countdown; UserListPanel merges knownClients with points/pending/eliminated, applies the fixed sort (points desc, name asc), and labels each row “PICKING...”, “READY/IDLE”, or “ELIMINATED.” Because updates originate on the server and are pushed in payloads, any action (CLI or UI) shows up immediately and consistently on every client.

Saved: 8/11/2025 5:42:01 AM

Task #3 (0.67 pts.) - Game Area

Progress: 100%

--- Task Collapsed ---

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

Progress: 100%

--- Section Collapsed ---

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

Progress: 100%

Task #1 (1 pt.) - Away Status

Progress: 100%

--- Task Collapsed ---

Task #2 (1 pt.) - Spectators

Progress: 100%

--- Task Collapsed ---

Section #5: (1 pt.) Misc

Progress: 100%

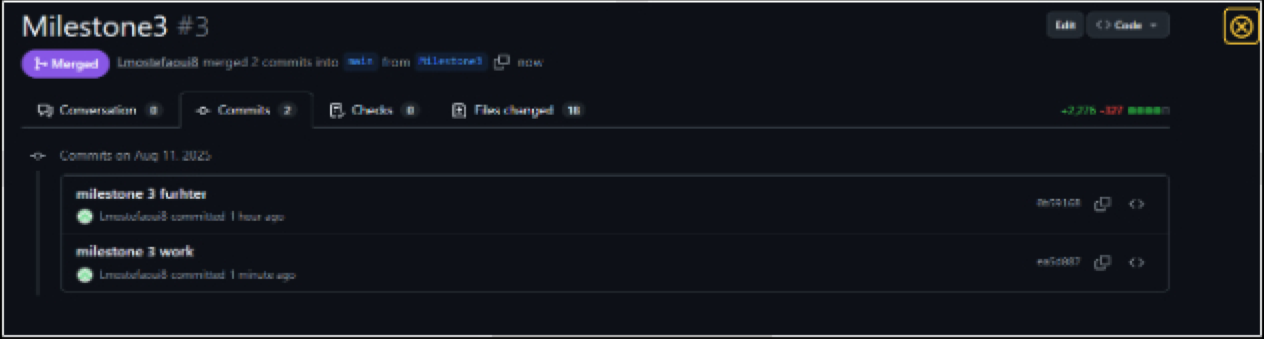
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

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



Milestone3 #3

Merged Lmostefaoui8 merged 2 commits into `main` from `Milestone3` now

Conversation 0 | Commits 2 | Checks 0 | Files changed 18 | +2,276 -327

Commits on Aug 11, 2025

milestone 3 furhter
Lmostefaoui8 committed 1 hour ago

milestone 3 work
Lmostefaoui8 committed 1 minute ago

SS of commits history

Saved: 8/11/2025 11:07:28 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/Lmostefaoui8/lm87-it114/pull/3

URL
https://github.com/Lmostefaoui8/

Saved: 8/11/2025 11:07:28 PM

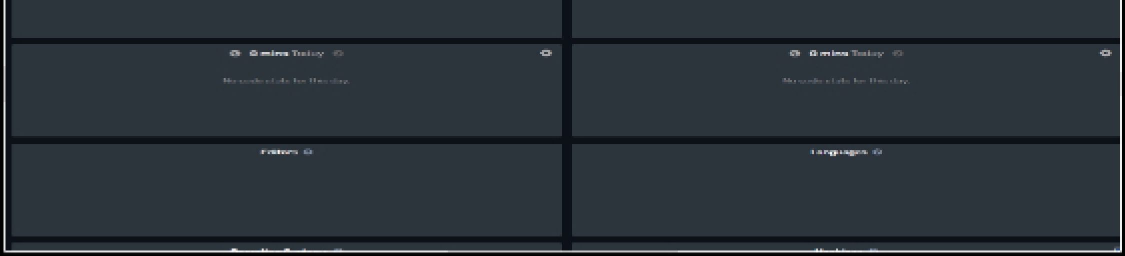
Task #2 (0.33 pts.) - WakaTime - Activity

Progress: 100%

Details:

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

0 mins over the Last 7 Days.



i didn't use waka time

Saved: 8/11/2025 11:07:42 PM

Task #3 (0.33 pts.) - Reflection

Progress: 100%

--- Task Collapsed ---