Submission Worksheet

Submission Data

Course: IT114-450-M2025
Assignment: IT114 - Milestone 3 - Go Fish
Student: Colin R. (ctr26)
Status: Submitted | **Worksheet Progress:** 92%
Potential Grade: 9.28/10.00 (92.80%)
Received Grade: 0.00/10.00 (0.00%)
Started: 8/7/2025 9:34:36 PM
Updated: 8/9/2025 2:04:30 AM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-3-go-fish/grading/ctr26
View Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-3-go-fish/view/ctr26

Instructions

1. Refer to Milestone3 of Go Fish
    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

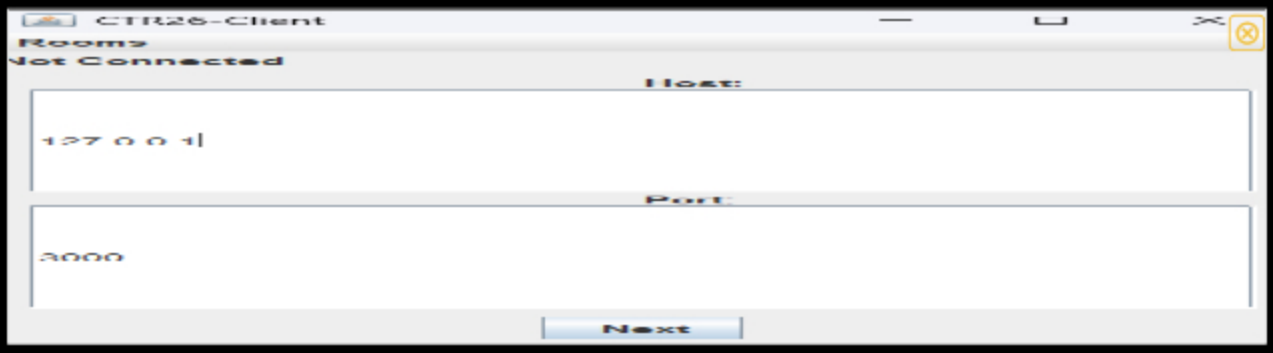
Progress: 100%

Part 1:

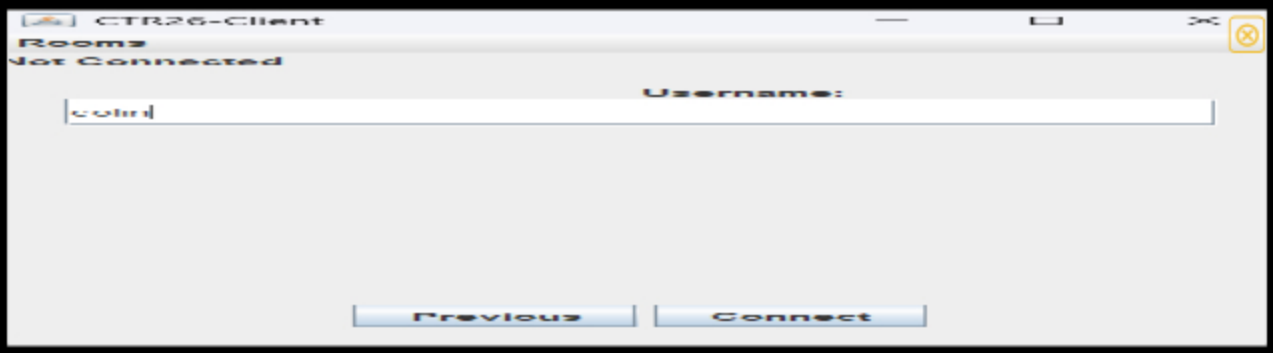
Progress: 100%

Details:

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



Connection panel



User details panel

Saved: 8/7/2025 9:40:43 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:

For the connection panel, the button ui element has a function that is triggered whenever it is clicked. That function pulls the data that is inputted into the text filed and passes them on to the ClientUI script. Then, it triggers for the next view to be displayed, which is the user details view. This view does the same thing with the text box and button, but after clicking it also triggers the connect function in the ClientUI script. This function take all the data from those two views and uses it to connect to the server.

Saved: 8/7/2025 9:40:43 PM

Task #2 (0.50 pts.) - Ready Panel

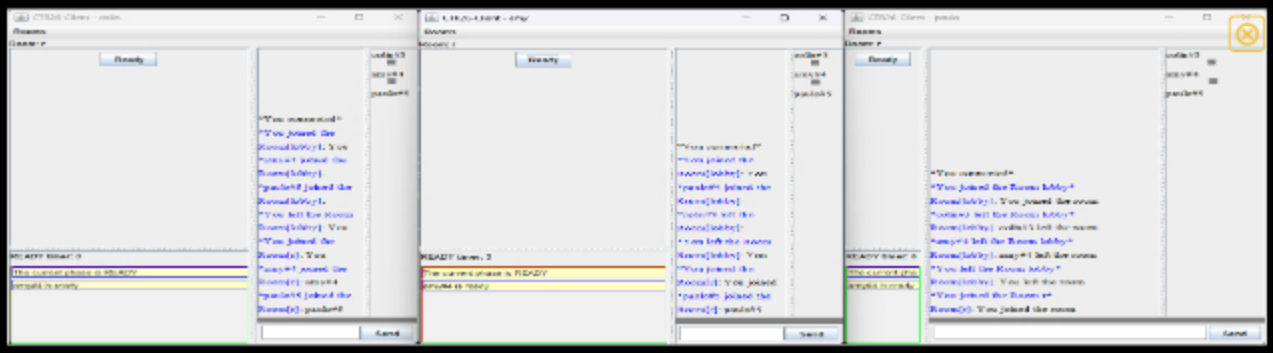
Progress: 100%

Part 1:

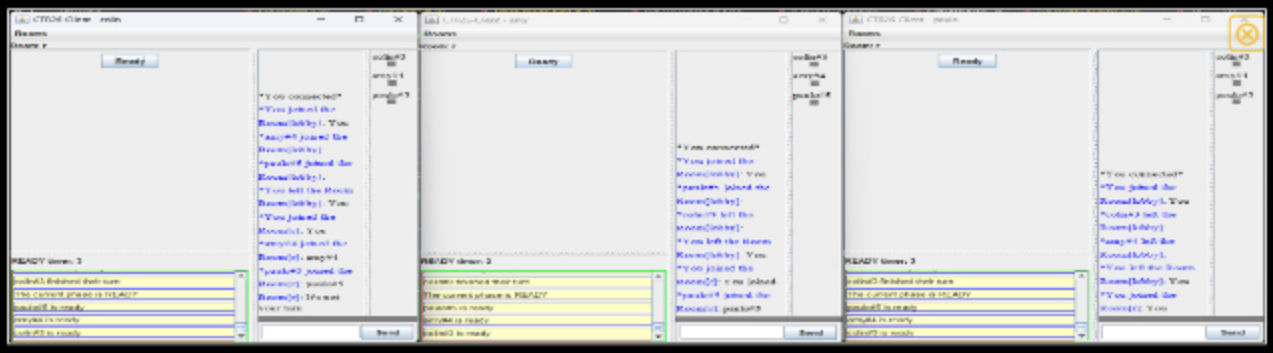
Progress: 100%

Details:

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



2 users ready, 1 not



All users ready

Saved: 8/7/2025 9:46:32 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:

When in the ready phase, the turn status boolean is used to track which players are ready. Upon clicking the ready button, the sendReady function is triggered in client.java. This function sends a ready payload to the server, which then reads it and triggers the ready timer.

Saved: 8/7/2025 9:46:32 PM

Section #2: (2 pts.) Project Ui

Progress: 66%

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

Progress: 100%

Details:

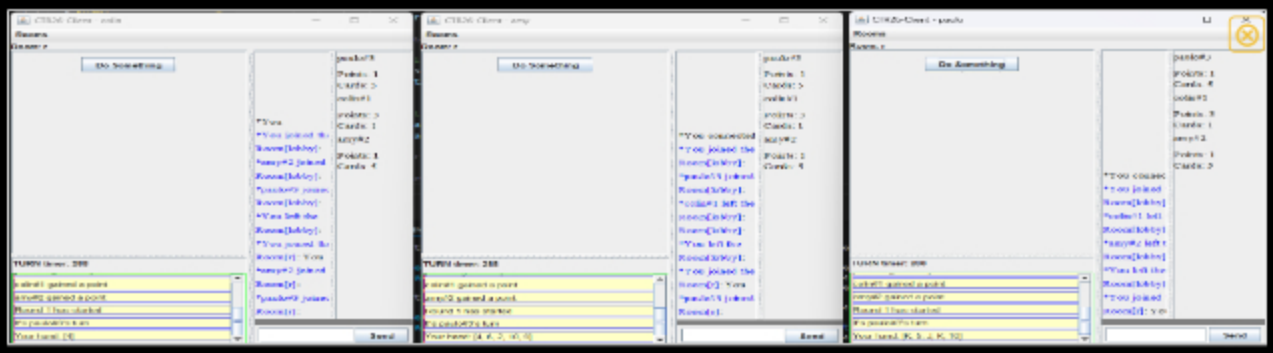
- Show the username and id of each user
- Show the current points of each user
- Users should appear in turn order across each client
- Show the number of cards in a hand for each user

Part 1:

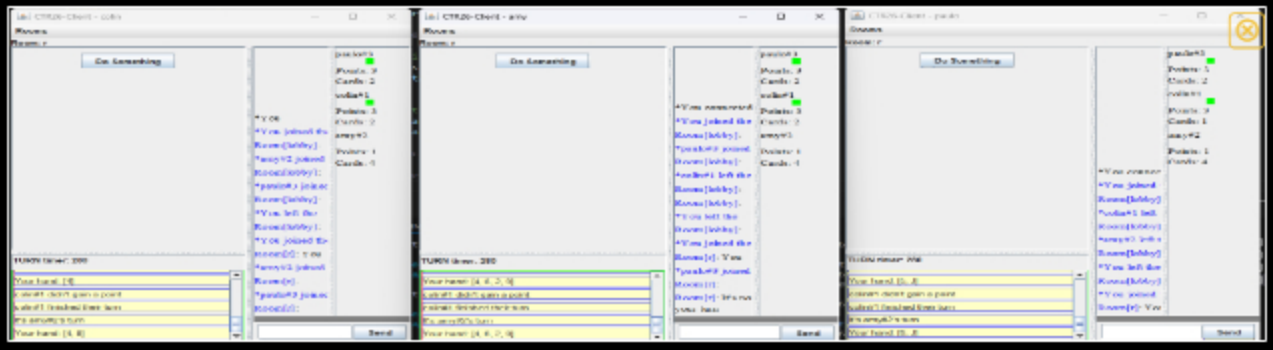
Progress: 100%

Details:

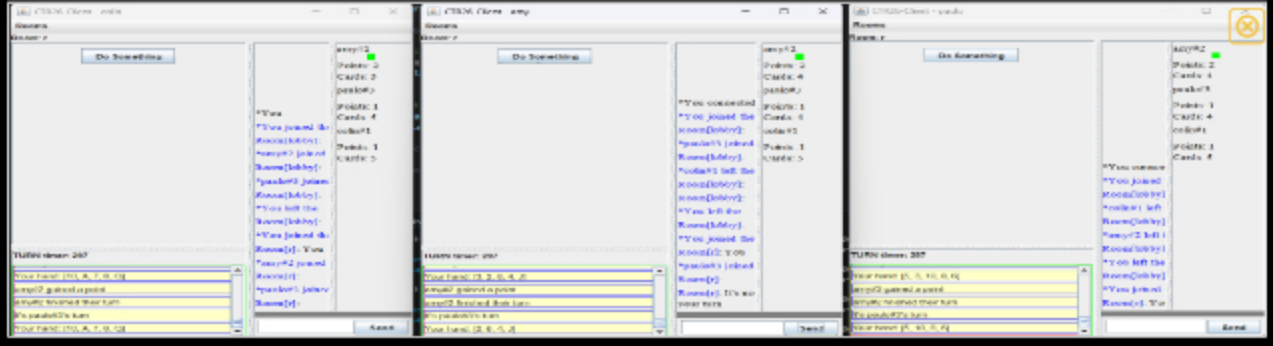
- Show various examples of points (3+ clients visible)
  - Include code snippets showing the code flow for this from server-side to UI
- Show that the sorting is maintained across clients
  - Include code snippets showing the code that handles this
- Show various examples of card count for each user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



Game example 1/3



Game example 2/3



Game example 3/3

```
for (ServerThread targetUser : turnOrder)
{
    if (targetUser.getClientId() == targetId)
    {
        if (targetUser.getHand().contains(targetCard))
        {
            targetUser.removeCard(targetCard);
            currentUser.addCard(targetCard);
        }
        else
        {
            currentUser.addCard(deck.draw());
        }
        updatePoints(currentUser);
        sendHand(targetUser);
        sendHand(currentUser);
    } <- #512-525 if (targetUser.getClientId() == target
} <- #510-526 for (ServerThread targetUser : turnOrder)
```

Points: GameRoom.java turn action

```
328     private void updatePoints(ServerThread player)        You, yesterday • Play
329     {
330         int points = player.checkForPair();
331         sendGameEvent(String.format(format:"%s %s", player.getDisplayName(),
332         points > 0 ? "gained a point" : "didn't gain a point"));
333         if (points > 0) {
334             player.changePoints(points);
335             sendPlayerPoints(player);
336         }
337     } <- #329-337 private void updatePoints(ServerThread player)
```

Points: GameRoom.java server message

```
267     private void sendPlayerPoints(ServerThread sp) {
268         clientsInRoom.values().removeIf(spInRoom -> {
269             boolean failedToSend = !spInRoom.sendPlayerPoints(sp.getClientId(), sp.getPoints());
270             if (failedToSend) {
271                 removeClient(spInRoom);
272             }
273             return failedToSend;
274         }); <- #268-274 clientsInRoom.values().removeIf
275     } <- #267-275 private void sendPlayerPoints(ServerThread sp)
```

Points: GameRoom.java send to server thread

```
72      public boolean sendPlayerPoints(long clientId, int points) {
73          PointsPayload rp = new PointsPayload();
74          rp.setPoints(points);
75          rp.setClientId(clientId);
76          return sendToClient(rp);
77      } <- #72-77 public boolean sendPlayerPoints(long clientId, int pc
```

Points: ServerThread.java create payload

```
    private void processPoints(Payload payload) {
        if (!(payload instanceof PointsPayload)) {
            error(message:"Invalid payload subclass for processCardAdd");
            return;
        }
        PointsPayload pp = (PointsPayload) payload;
        long targetId = pp.getClientId();
        int points = pp.getPoints();
        if (targetId    Constants.DEFAULT_CLIENT_ID) {
            // reset all
            knownClients.values().forEach(cp -> cp.setPoints(-1));

            passToUICallback(type:IPointsEvent.class, e -> e.onPointsUpdate(Constants.DEFAULT_CLIENT_ID, -1));
        } else if (knownClients.containsKey(targetId)) {
            knownClients.get(targetId).setPoints(points);

            passToUICallback(type:IPointsEvent.class, e  > e.onPointsUpdate(targetId, points));

        } <- #651 else if (knownClients.containsKey(targetId))
    } <- #638-657 private void processPoints(Payload payload)
```

Points: Client.java process payload

```
    @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);
                }
            }); <- #174-180 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);
                }

            }); <- #182-189 SwingUtilities.invokeLater
        } <- #181-190 else if (userItemsMap.containsKey(clientId))
    } <- #172 191 public void onPointsUpdate(long clientId, int points)
```

Points: UserListView.java update ui

```
    /**
     * Sets 'turnOrder' to a shuffled list of players who are ready.
     */
    private void setTurnOrder() {
        turnOrder.clear();
        turnOrder   clientsInRoom.values().stream().filter(ServerThread::isReady).collect(Collectors.toList());
        Collections.shuffle(turnOrder);
        List<Long> clientIds = turnOrder.stream().map(ServerThread::getClientId).collect(Collectors.toList());
        sendTurnOrder(clientIds);

    } <- #350-357 private void setTurnOrder()

    private void sendTurnOrder(List<Long> clients)
    {
        clientsInRoom.values().forEach( c -> {
            c.sendTurnOrder(clients);
        });
    } <- #360-364 private void sendTurnOrder(List<Long> clients)
```

List Order: GameRoom.java randomize list and send to server thread

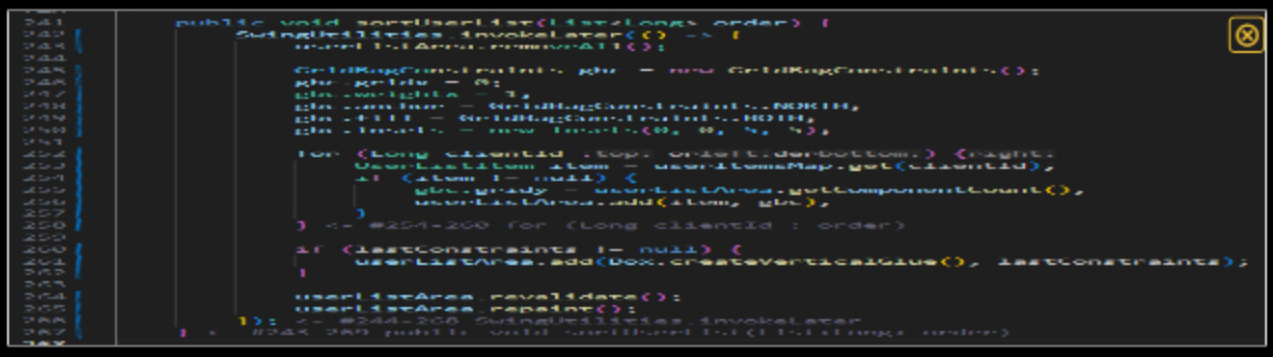
```
    public boolean sendTurnOrder(List<Long> clients)
    {
        ClientListPayload clp = new ClientListPayload(clients);
        clp.setClientId(getClientId());
        clp.setPayloadType(PayloadType.CLIENT_LIST);
        return sendToClient(clp);
    }
```

List Order: ServerThread.java create payload

```
    private void processOrderList(Payload payload)
    {
        ClientListPayload clp = (ClientListPayload) payload;
```

```
633             List<Long> clients = clp.getClients();
634             passToUICallback(type:IRoomEvents.class, e -> e.sortUserList(clients));
635
636         } <- #631-636 private void processOrderList(Payload payload)
```

List Order: Client.java process payload



List Order: UserListView.java update ui

```
for (ServerThread targetUser : turnOrder)
{
    if (targetUser.getClientId() == targetId)
    {
        if (targetUser.getHand().contains(targetCard))
        {
            targetUser.removeCard(targetCard);
            currentUser.addCard(targetCard);
        }
        else
        {
            currentUser.addCard(deck.draw());
        }
        updatePoints(currentUser);
        sendHand(targetUser);
        sendHand(currentUser);
    } <- #512-525 if (targetUser.getClientId() == target
} <- #510-526 for (ServerThread targetUser : turnOrder)
```

Card Count: GameRoom.java turn action

```
322     private void sendHand(ServerThread player)
323     {
324         syncPlayerCards(player);
325         player.sendCurrentHand(player.getClientId(), player.getHand());
326     }
```

Card Count: GameRoom.java send to server thread

```
254     private void syncPlayerCards(ServerThread incomingClient) {
255         clientsInRoom.values().forEach(serverUser -> {
256             if (serverUser.getClientId() != incomingClient.getClientId()) {
257                 boolean failedToSync = !incomingClient.sendCurrentHand(serverUser.getClientId(), serverUser.getHand());
258                 if (failedToSync) {
259                     LoggerUtil.INSTANCE.warning(
260                             String.format(format:"Removing disconnected %s from list", serverUser.getDisplayName()));
261                     disconnect(serverUser);
262                 } <- #258-262 if (failedToSync)
263             } <- #256-263 if (serverUser.getClientId() != incomingClient.getClientId())
264         }); <- #255-264 clientsInRoom.values().forEach
265     } <- #254-265 private void syncPlayerCards(ServerThread incomingClient)
```

Card Count: GameRoom.java sync card counts for all players

```java
protected boolean sendCurrentHand(long clientId, List<CardType> cards)
{
    CardsPayload cp = new CardsPayload(cards);
    cp.setPayloadType(PayloadType.CARDS);
    cp.setClientId(clientId);
    return sendToClient(cp);
} <- #237-242 protected boolean sendCurrentHand(long clientId, List<Car
```

Card Count: ServerThread.java create payload

```java
// Start process*() methods
private void processCardsSync(Payload payload)
{
    CardsPayload cp = (CardsPayload) payload;
    myUser.syncCards(cp.getCards());
    long targetId = cp.getClientId();
    int cards = cp.getCards().size();
    if (targetId == Constants.DEFAULT_CLIENT_ID) {
        passToUICallback(type:ICardsEvent.class, e -> e.onCardsUpdate(Constants.DEFAULT_CLIENT_ID, -1));
    } else if (knownClients.containsKey(targetId)) {
        LoggerUtil.INSTANCE.info(message:"passToUICallBack");
        knownClients.get(targetId).setCardCount(cards);
        passToUICallback(type:ICardsEvent.class, e -> e.onCardsUpdate(targetId, cards));
    } <  #623 627 else if (knownClients.containsKey(targetId))
} <  #616 628 private void processCardsSync(Payload payload)
```

Card Count: Client.java process payload

```java
@Override
public void onCardsUpdate(long clientId, int cards) {
    if (clientId == Constants.DEFAULT_CLIENT_ID) {
        SwingUtilities.invokeLater(() -> {
            try {
                userItemsMap.values().forEach(u -> u.setCards(-1));// reset all
            } catch (Exception e) {
                LoggerUtil.INSTANCE.severe(message:"Error resetting user items", e);
            }
        }); <- #194-202 SwingUtilities.invokeLater
    } else if (userItemsMap.containsKey(clientId)) {
        SwingUtilities.invokeLater(() -> {
            try {
                LoggerUtil.INSTANCE.info(message:"");
                userItemsMap.get(clientId).setCards(cards);
            } catch (Exception e) {
                LoggerUtil.INSTANCE.severe(message:"Error setting user item", e);
            }

        }); <- #204-212 SwingUtilities.invokeLater
    } else {
        LoggerUtil.INSTANCE.info("Not working!");
    }
}message: <- #193-216 public void onCardsUpdate(long clientId, int cards)
```

Card Count: UserListView.java update ui

Saved: 8/8/2025 1:43:51 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 card count

Your Response:

Points: The code works by handling the logic from the server side. When the server finishes the logic for updating points, it triggers the server thread to send the payload to the client. When the client processes this payload that contains the points, it sends it to the ui script in order to update it properly. List Sorting: After the turn order is shuffled, the server triggers the server thread for each user to create and send a client list payload to each client. The client processes this data

and sends the list to the ui script. The ui script strips each client element from the screen and readds them in the order of the given list. Card Count: This function in the exact same way as the points, but instead of sending a points payload that contains an integer, it send the cards of the player in a list. The count is then taken from this list and given to the ui in order to update the screen. This is synced between all players by triggering the function among all clients.

Saved: 8/8/2025 1:43:51 AM

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

Progress: 100%

Details:

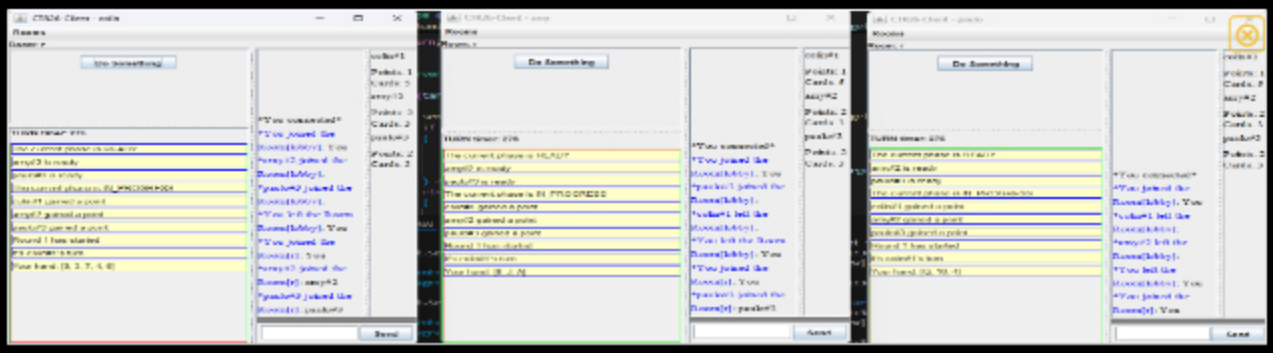
- Show a message of whose turn it is when this value changes
- Show a message of who the person targeted, what they attempted to take, and what was received (including if it was a “Go Fish”, but you can use your own words)
- Show a message of any points acquired for pairs/sets

Part 1:

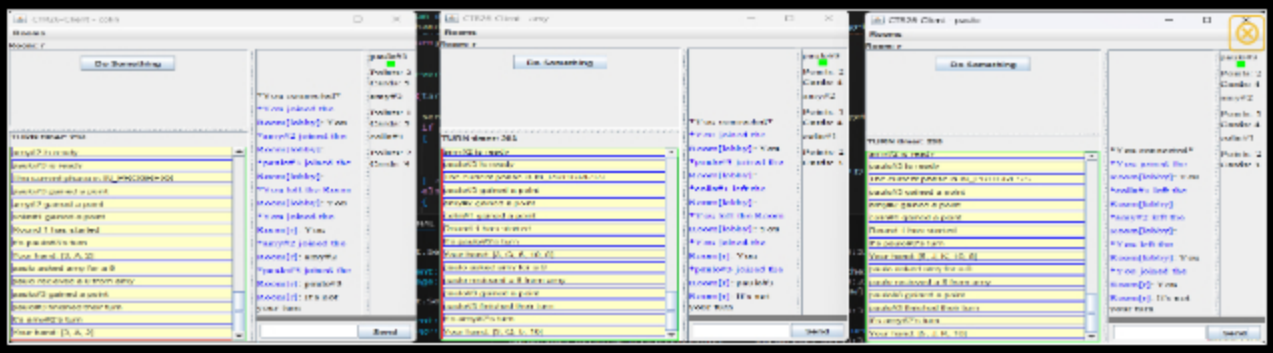
Progress: 100%

Details:

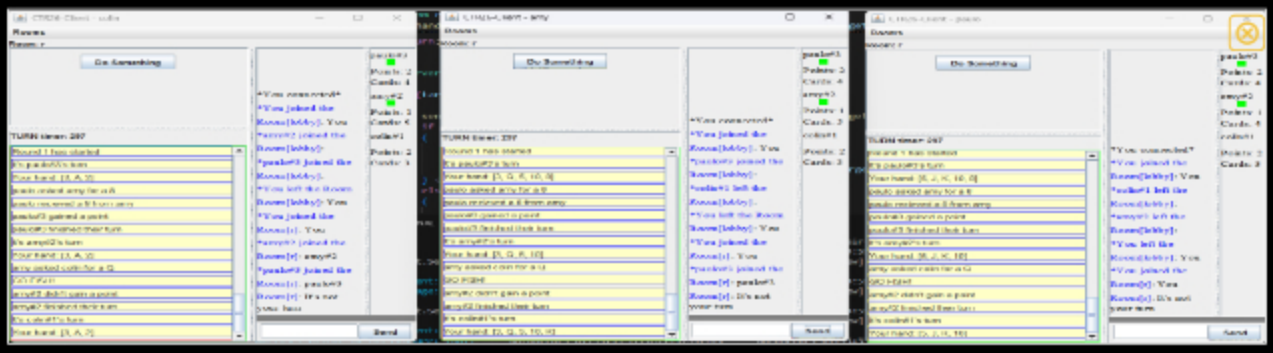
- Show various examples of each of the messages/visuals
- Show code snippets related to these messages from server-side to UI
- Show the countdown timer for the round



Game example 1/3



Game example 2/3



Game example 3/3

```
        for (ServerThread targetUser : turnOrder)
        {
            if (targetUser.getClientId() == targetId)
            {
                sendGameEvent(currentUser.getClientName() + " asked " + targetUser.getClientName() + " for a " + targetCard.toString());
                if (targetUser.getHand().contains(targetCard))
                {
                    targetUser.removeCard(targetCard);
                    currentUser.addCard(targetCard);
                    sendGameEvent(currentUser.getClientName() + " recieved a " + targetCard.toString() + " from " + targetUser.getClientName());
                } <- 515-519 if (targetUser.getHand().contains(targetCard))
                else
                {
                    currentUser.addCard(deck.draw());
                    sendGameEvent(str:"GO FISH!");
                }
                updatePoints(currentUser);
                sendHand(targetUser);
                sendHand(currentUser);
            } <- 512-528 if (targetUser.getClientId() == targetId)
        } <- 510-529 for (ServerThread targetUser : turnOrder)
```

GameRoom.java target messages

```
    private void updatePoints(ServerThread player)
    {
        int points = player.checkForPair();
        sendGameEvent(String.format(format:"%s %s", player.getDisplayName()
        points > 0 ? "gained a point" : "didn't gain a point"));
        if (points > 0) {
            player.changePoints(points);
            sendPlayerPoints(player);
        }
    } <- #329-337 private void updatePoints(ServerThread player)
```

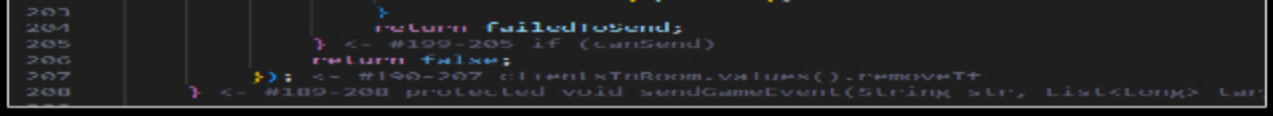
GameRoom.java points message

```
    /** {@inheritDoc} */
    @Override
    protected void onTurnStart() {
        LoggerUtil.INSTANCE.info(message:"onTurnStart() start");
        resetTurnTimer();
        try {
            ServerThread currentPlayer = getNextPlayer();
            // relay(null, String.format("It's %s's turn", currentPlayer.getDisplayName()));
            sendGameEvent(String.format(format:"It's %s's turn", currentPlayer.getDisplayName()));
            showHands();
        } catch (MissingCurrentPlayerException | PlayerNotFoundException e) {

            e.printStackTrace();
        }
        startTurnTimer();
        LoggerUtil.INSTANCE.info(message:"onTurnStart() end");
    } <- #172-186 protected void onTurnStart()
```

GameRoom.java turn message

```
    protected void sendGameEvent(String str, List<Long> targets) {
        clientsInRoom.values().removeIf(spinRoom -> {
            boolean canSend = false;
            if (targets != null) {
                if (targets.contains(spinRoom.getClientId())) {
                    canSend = true;
                }
            } else {
                canSend = true;
            }
            if (canSend) {
                boolean failedToSend = !spinRoom.sendGameEvent(str);
                if (failedToSend) {
                    removeClient(spinRoom);
```



BaseGameRoom.java send game event

```java
79      public boolean sendGameEvent(String str) {
80          return sendMessage(Constants.GAME_EVENT_CHANNEL, str);
81      }
```

ServerThread.java send message

```java
251     protected boolean sendMessage(long clientId, String message) {
252         Payload payload = new Payload();
253         payload.setPayloadType(PayloadType.MESSAGE);
254         payload.setMessage(message);
255         payload.setClientId(clientId);
256         return sendToClient(payload);
257     } <- #251-257 protected boolean sendMessage(long clientId, Strin
```

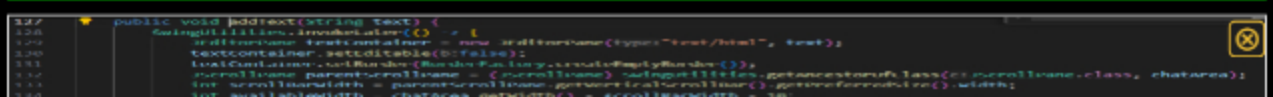
ServerThread.java message payload

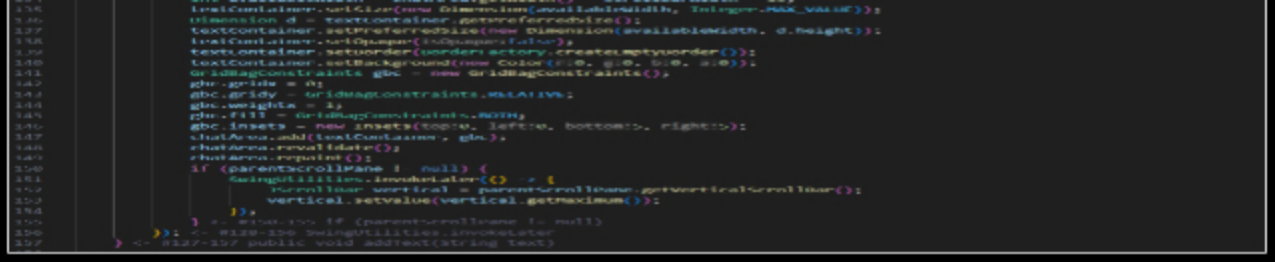
```java
866     private void processMessage(Payload payload) {
867         LoggerUtil.INSTANCE.info(TextFX.colorize(payload.getMessage(), Color.BLUE));
868
869         passToUICallback(type:IMessageEvents.class, e -> e.onMessageReceive(payload.getClientId(),
870                 payload.getMessage()));
871     } <- #866-871 private void processMessage(Payload payload)
```

Client.java process message

```java
180     @Override
181     public void onMessageReceive(long clientId, String message) {
182         if (clientId < Constants.DEFAULT_CLIENT_ID) {
183             return;
184         }
185         String displayName = Client.INSTANCE.getDisplayNameFromId(clientId);
186         // added color to differentiate between room and user messages
187         String name = clientId == Constants.DEFAULT_CLIENT_ID ? "<font color=blue>%s</font>"
188                 : "<font color=purple>%s</font>";
189         name = String.format(name, displayName);
190         addText(String.format(format:"%s: %s", name, message));
191     } <- #181-191 public void onMessageReceive(long clientId, String message)
```

ChatView.java receive message





ChatView.java display message

Saved: 8/8/2025 2:06:38 AM

Part 2:

Progress: 100%

Details:

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

Your Response:

The code just uses the same information that is used for the logic in order to create these messages. Specifically, the client that is being handeled can easily have their name added to the message using the get client name command. These messages are then packaged into a payload by the server thread and sent off to the client. When the client recieves the message payload, it triggers the the chat view ui script to add the message onto the screen.

Saved: 8/8/2025 2:06:38 AM

Task #3 (0.67 pts.) - Game Area

Progress: 0%

Details:

- Grid or similar of buttons for each player that becomes active only for the current Player (i.e., if it's not the specific player's turn, the interactions should be disabled or hidden)
  - Clicking a button will target that player and select them as part of the turn
- Grid or similar of card information about the Player's hand
  - Clicking a card will select the suite as part of the turn
- Once both a target Player and desired suit are selected, a prompt should ask to confirm their choices (yes/no)
  - Confirming yes should send the request
  - Confirming no will close the dialog and let them change one of the options

Part 1:

Progress: 0%

Details:

- Show a few examples of targetting/selecting player/card combinations
- Clearly show the UI across 3+ clients (only the current player should have their interactions enabled/visible)

Missing Caption

Saved: 8/5/2025 1:28:18 AM

Part 2:

Progress: 0%

Details:

- Briefly explain the code flow for handling the UI visibility
- Briefly explain the code for selecting a target and a card and sending the it to the server-side

Your Response:

Missing Response

Saved: 8/5/2025 1:28:18 AM

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

Progress: 100%

Task #1 (2 pts.) - Extra Decks

Progress: 100%

Details:

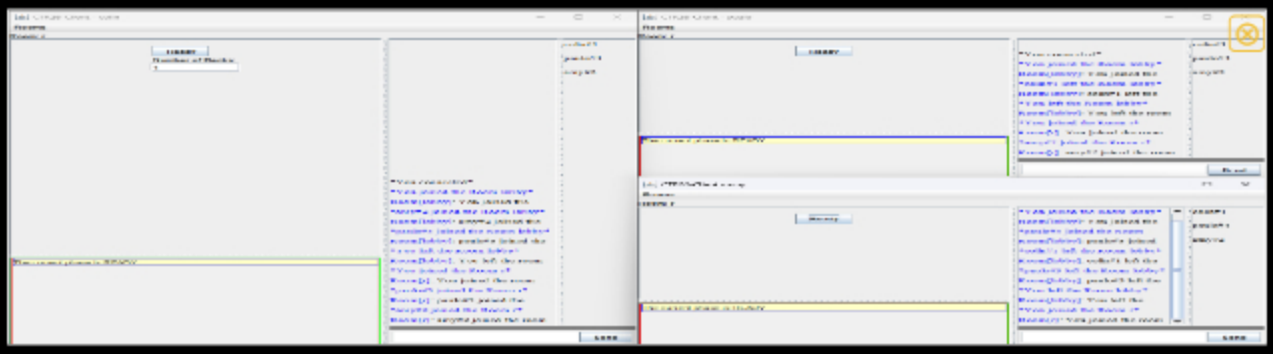
- Extra decks can be set/toggled by session creator

Part 1:

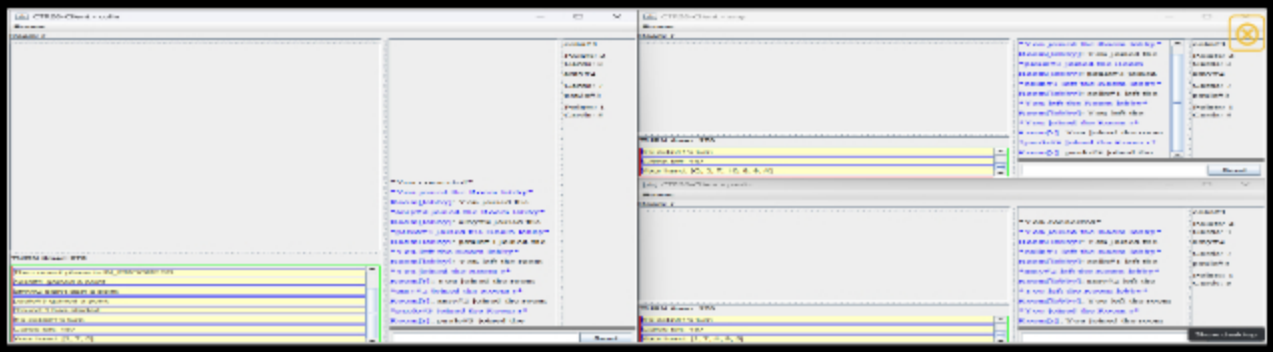
Progress: 100%

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 code related to having the server-side utilize multiple decks



ReadyView



GameView with card count

```
271     private void syncHost(ServerThread sp)
272     {
273         clientsInRoom.values().forEach(serverUser -> {
274             boolean failedToSync = !sp.sendHost(serverUser.getClientId(), creator.getClientId());
275             if (failedToSync) {
276                 LoggerUtil.INSTANCE.warning(
277                         String.format(format:"Removing disconnected %s from list", serverUser.getDisplayName()));
278                 disconnect(serverUser);
279             } <- #275-279 if (failedToSync)
280         }); <- #273-280 clientsInRoom.values().forEach
281     } <- #272-281 private void syncHost(ServerThread sp)
282
```

GameRoom.java send hostId

```
641     private void processHost(Payload payload)
642     {
643         hostId = payload.getClientId();
644         if (myUser.getClientId() == hostId)
645         {
646             passToUICallback(type:IConnectionEvents.class, e -> e.roomCreator());
647         }
648     } <- #642-648 private void processHost(Payload payload)
```

Client.java call ui update

```
47      @Override
48      public void roomCreator()
```

```
49          {
50              content.add(decktext);
51              content.add(numDecks);
52              isHost = true;
53          } <- #49-53 public void roomC
```

ReadyView.java add ui element

```
readyButton.addActionListener(_ -> {
    try {
        Client.INSTANCE.sendReady(isHost ? numDecks.getText().trim() : "-1");
    } catch (IOException e1) {
        e1.printStackTrace();
    }
}); <- #35-41 readyButton.addActionListener
content.add(readyButton);       You, 40 minutes ago • Added multiple decks as a
```

ReadyView.java send ready data

BaseGameRoom.java process ready data

```
14      public Deck(int numDecks)
15      {
16          loadCards(numDecks);
17      }
18
19      public void loadCards(int numDecks)
20      {
21          for (CardType card : CardType.values())
22          {
23              for (int i = 0; i < 4 * numDecks; i++)
24              {
25                  cards.add(card);
26              }
27          } <- #21-26 for (CardType card : CardType.values())
28          shuffle();
29      } <- #19-28 public void loadCards(int numDecks)
30
31      public void shuffle()
32      {
33          Collections.shuffle(cards);          You, 4 days ago • Ad
34      }
```

Deck.java populating deck

Saved: 8/8/2025 4:57:27 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code for the host's option to toggle this feature
- Briefly explain the code for handling multiple decks

Your Response:

In order to have the ui only display to the host, I needed to add a new value to the room object, which is the serverthread that created the room. This serverthread is then passed on to each user when they join the room. This allows for each client to know who the host is. When the ready screen is sent to the players, each client will check if they are the host and call the ui function accordingly. For the multiple decks, the code utilizes the already existing ready data that is passed to server. I just added a new message to the ready payload that contains the user input from the ui text box. The server recieves this payload and sets its number of decks to that value. Then, when the server calls for the deck to be created, the parameter value scales how many of each card are added into the deck.

Saved: 8/8/2025 4:57:27 PM

Task #2 (2 pts.) - Add Wildcards

Progress: 100%

Details:

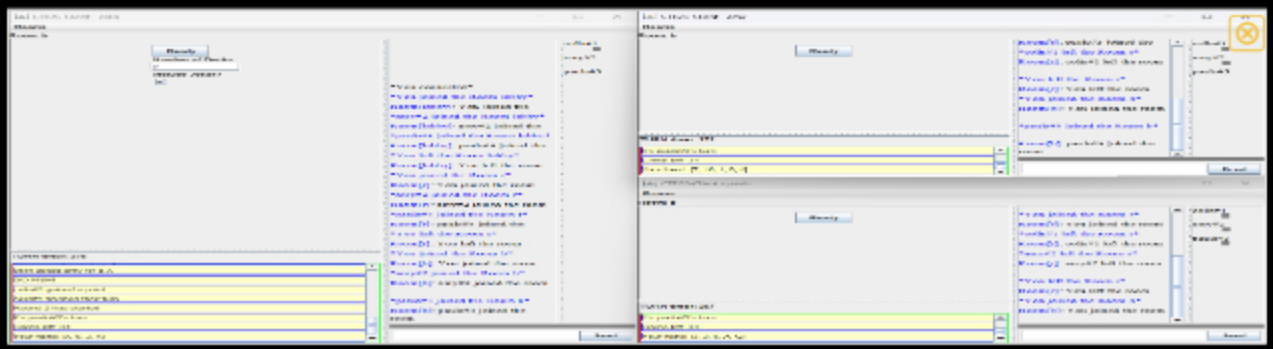
- Wildcards can be set/toggled by session creator

Part 1:

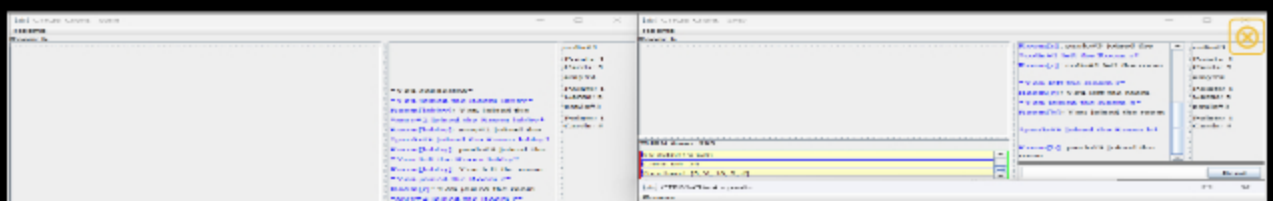
Progress: 100%

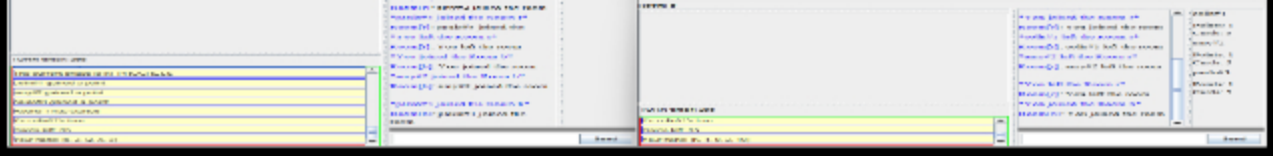
Details:

- 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 code related to having the server-side utilize wildcards and any visual representation of such like messages

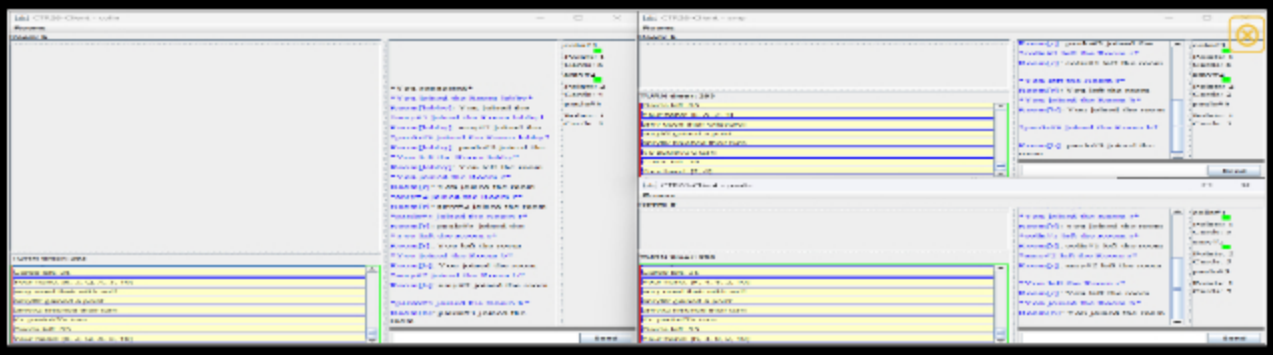


GameView 1/3





GameView 2/3



GameView 3/3

```
662     private void processHost(Payload payload)
663     {
664         hostId = payload.getClientId();
665         if (myUser.getClientId() == hostId)
666         {
667             passToUICallback(type:IConnectionEvents.class, e -> e.roomCreator());
668         }
669     } <- #663-669 private void processHost(Payload payload)
```

Client.java process host payload and trigger ui event

```
50          @Override
51          public void roomCreator()
52          {
53              content.add(decktext);
54              content.add(numDecks);
55              content.add(toggletext);
56              content.add(jokerToggle);
57              isHost = true;
58          } <- #52-58 public void roomCreat
```

ReadyView.java add ui elements for host only

```
578     protected void handleWildcard(ServerThread currentUser, CardType card)
579     {
580         try {
581             checkPlayerInRoom(currentUser);
582             checkCurrentPhase(currentUser, Phase.IN_PROGRESS);
583             checkCurrentPlayer(currentUser.getClientId());
584             checkIsReady(currentUser);
585             if (currentUser.didTakeTurn()) {
586                 currentUser.sendMessage(Constants.DEFAULT_CLIENT_ID, message:"You have already taken your turn this round");
587                 return;
588             }
589
590             sendGameEvent(currentUser.getClientName() + " used their wildcard!");
591             currentUser.removeCard(CardType._X);
592             currentUser.removeCard(card);
593             sendHand(currentUser);
594             updatePoints(currentUser, wildcard:true);
```

GameRoom.java processes wildcard use and triggers game updates

Saved: 8/8/2025 8:08:28 PM

Saved: 8/8/2025 8:08:28 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the code for the host's option to toggle this feature
- Briefly explain the code for handling wildcards

Your Response:

All clients recieve a hostId from the game room when it is created. If the host id matches the client's id, it calls for the roomCreator ui event. This event is implemented in the readyview script and adds the ui elements for this feature. Only the host will have these features and all other clients will have their data discarded upon pressing ready.

Saved: 8/8/2025 8:08:28 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)

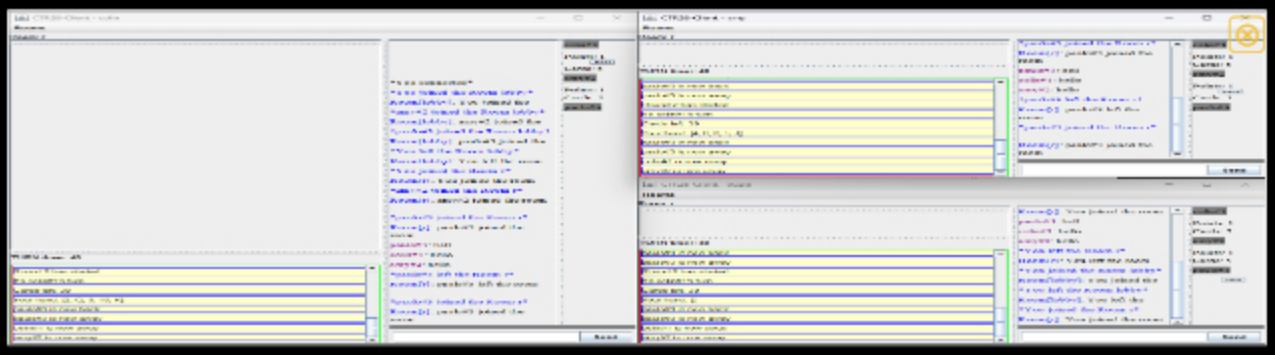
Part 1:

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

GameView 1/2



GameView 2/2

```
awayButton = new JButton(text:"Away");
awayButton.setPreferredSize(new Dimension(width:30, height:10));
awayButton.setFont(awayButton.getFont().deriveFont(size:8f)); // Bit font size
awayButton.setMargin(new Insets(top:1, left:2, bottom:1, right:2));
awayButton.addActionListener(_ -> {
    // don't set away directly, use the server reply
    try {
        Client.INSTANCE.sendAwayAction();// using it as a toggle
    } catch (IOException e) {
        // example of user feedback via a UI component
        JOptionPane.showMessageDialog(this, message:"Failed to send away action.", title:"Error", JOptionPane.ERROR_MESSAGE);
        LoggerUtil.INSTANCE.severe(message:"Error sending away action", e);
    } <- #87-91 catch (IOException e)

}); <  #83 93 awayButton.addActionListener
awayButton.setVisible(aFlag:false); // initially hidden
rowPanel.add(awayButton);
```

UserListItem.java button trigger

```
654     public void sendAwayAction() throws IOException
655     {
656         Payload payload = new Payload();
657         payload.setPayloadType(PayloadType.AWAY);
658         sendToServer(payload);
659     }
```

Client.java payload creation

```
352             case AWAY:
353                 try {
354                     // cast to GameRoom as the subclass will handle all Game logic
355                     ((GameRoom) currentRoom).handleAwayAction(this);
356                 } catch (Exception e) {
357                     sendMessage(Constants.DEFAULT_CLIENT_ID, message:"You must be in a GameRoom to set away status");
358                 }
```

ServerThread.java receive payload

```java
protected void handleAwayAction(ServerThread currentUser) {
    try {
        checkPlayerInRoom(currentUser);
        // anyone can be away whenever so there are less required "checks" here
        // toggle away status
        currentUser.setAway(!currentUser.isAway());
        sendAwayStatus(currentUser);
    } catch (PlayerNotFoundException e) {
        currentUser.sendGameEvent(str:"You must be in a GameRoom to do the away action");
        LoggerUtil.INSTANCE.severe(message:"handleAwayAction exception", e);
    } catch (Exception e) {
        LoggerUtil.INSTANCE.severe(message:"handleAwayAction exception", e);
    }
} <- #526-539 protected void handleAwayAction(ServerThread currentUser)
```

GameRoom.java set away status

```java
private void sendAwayStatus(ServerThread sp) {
    clientsInRoom.values().removeIf(spInRoom -> {
        boolean failedToSend = !spInRoom.sendAwayStatus(sp.getClientId(), sp.isAway(), isQuiet:false);
        if (failedToSend) {
            removeClient(spInRoom);
        }
        return failedToSend;
    }); <- #338-344 clientsInRoom.values().removeIf
} <- #337-345 private void sendAwayStatus(ServerThread sp)
```

GameRoom.java send to client

```java
// Start Send*() Methods
public boolean sendAwayStatus(long clientId, boolean isAway, boolean isQuiet) {
    ReadyPayload rp = new ReadyPayload(deckCount:"");
    rp.setClientId(clientId);
    rp.setReady(isAway);
    rp.setPayloadType(isQuiet ? PayloadType.SYNC_AWAY : PayloadType.AWAY);

    return sendToClient(rp);
} <- #75-82 public boolean sendAwayStatus(long clientId, boolean isAway, ...
```

ServerThread.java send back to client

```java
private void processAwayStatus(Payload payload) {
    // Note: Using ReadyPayload since it's my only payload with just a boolean
    // it'd be best to rename it to something generic or to subclass it for
    // readability (even if the subclass adds nothing)
    if (!(payload instanceof ReadyPayload)) {
        error(message:"invalid payload subclass for processAwayStatus");
        return;
    }
    ReadyPayload rp = (ReadyPayload) payload;
    if (rp.getClientId() == Constants.DEFAULT_CLIENT_ID) {
        // reset all
        knownClients.values().forEach(c -> c.setAway(isAway:false));
        getRoomCallback(type:TurnEvent.class, r -> r.onAwayStatusUpdate(Constants.DEFAULT_CLIENT_ID, isAway:false));
    } else if (knownClients.containsKey(rp.getClientId())) {
        boolean isAway = rp.isReady(); // Ready is just the boolean getter, assume it means isAway here
        knownClients.get(rp.getClientId()).setAway(isAway);
        getRoomCallback(
                type:TurnEvent.class,
                r -> r.onAwayStatusUpdate(rp.getClientId(), isAway));
        // updated Game Events view if not a quiet sync
        if (payload.getPayloadType() != PayloadType.SYNC_AWAY) {
            clientSideGameEvent(
                    String.format(locale:"%s is now %s", getDisplayNameFromId(rp.getClientId()),
                            isAway ? "away" : "back"));
        } <- #672-676 if (payload.getPayloadType() != PayloadType.SYNC_AWAY)
    } else {
        LoggerUtil.INSTANCE.warning(String.format(format:"unknown client id %s for away status update", rp.getClientId()));
    }
} <- #663-680 private void processAwayStatus(Payload payload)
```

Client.java sync status and send to ui

```java
@Override
public void onAwayStatusUpdate(long clientId, boolean isAway) {
    if (clientId == Constants.DEFAULT_CLIENT_ID) {
        SwingUtilities.invokeLater(() -> {
            userItemsMap.values().forEach(u -> u.setAway(isAway:false)); // reset all
        });
```

```
256         } else if (userItemsMap.containsKey(clientId)) {
257             SwingUtilities.invokeLater(() -> {
258                 userItemsMap.get(clientId).setAway(isAway);
259             });
260         } <- #256-260 else if (userItemsMap.containsKey(clientId))
261     } <- #251-261 public void onAwayStatusUpdate(long clientId, boolean isAway)
```

UserListView.java trigger status change

```
215     public void setAway(boolean isAway) {
216         this.isAway = isAway;
217         applyStatusStyles();
218         // use the server reply as confirmation
219         awayButton.setText(isAway ? "Back" : "Away");
220         repaint();
221     } <- #215-221 public void setAway(boolean isAway)
```

UserListItem.java ui update

```
681         // updated Game Events View if not a quiet sync
682         if (payload.getPayloadType() != PayloadType.SYNC_AWAY) {
683             clientSideGameEvent(
684                     String.format(format:"%s is now %s", getDisplayNameFromId(cp.getClientId()),
685                             isAway ? "away" : "back"));
```

Client.java away message

```
177     @Override
178     protected void onTurnStart() {
179         LoggerUtil.INSTANCE.info(message:"onTurnStart() start");
180         resetTurnTimer();
181         try {
182             ServerThread currentPlayer = getNextPlayer();
183             // handle away status
184             if (currentPlayer.isAway()) {
185                 sendGameEvent(String.format(format:"%s is currently away and is getting skipped",
186                         currentPlayer.getDisplayName()));
187
188                 onTurnEnd(); // skip the turn
189                 return;
190             } else if (currentPlayer.isAway())
191             // relay(null, String.format("It's %s's turn", currentPlayer.getDisplayName()));
192             sendGameEvent(String.format(format:"It's %s's turn", currentPlayer.getDisplayName()));
193             sendGameEvent("Cards left: " + deck.cardsLeft());
194             showHands();
195         } catch (MissingCurrentPlayerException | PlayerNotFoundException e) {
196
197             e.printStackTrace();      You, 2 weeks ago • Adding phase files ...
198         }
199         startTurnTimer();
200         LoggerUtil.INSTANCE.info(message:"onTurnStart() end");
```

GameRoom.java skip away players

Saved: 8/9/2025 12:50:25 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:

The code starts off with the ui button that when clicked, sends a payload off to the server saying that the away status is changing. The server then receives this payload and updates the clients away status on the server side. After doing that, it sends a payload back to the client to let it know that it can update its ui. Once the client receives that payload, it calls a ui function that changes the userlistitem of the client to reflect their away status. The code skips over away clients by checking their away status on turn start. if the client is away, it sends a game event message and ends their turn.

Saved: 8/9/2025 12:50:25 AM

Task #2 (1 pt.) - Spectators

Progress: 100%

Details:

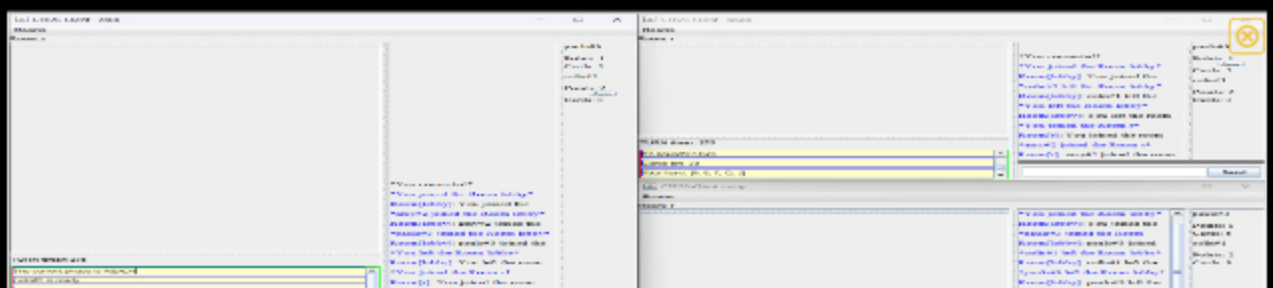
- Spectators are users who didn't mark themselves ready
  - Optionally you can include a toggle on the Ready Check page
- The can see all chat but are ignored from turn/round actions and can't send messages
- Spectators will have a visual representation in the user list to distinguish them from other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

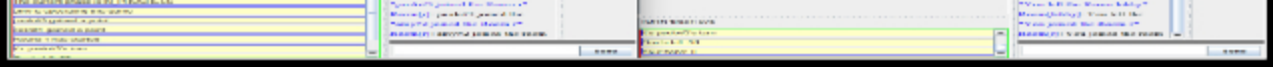
Part 1:

Progress: 100%

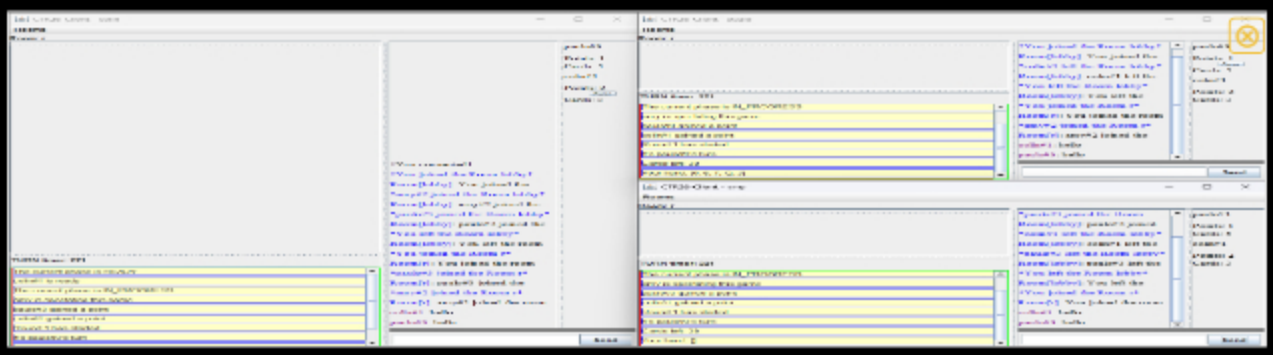
Details:

- 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)





Joined as spectator message



All 3 sent messages, but only non spectators are displayed

```
417     */
418     private void setTurnOrder() {
419         turnOrder.clear();
420         turnOrder = clientsInRoom.values().stream().filter(ServerThread::isReady).collect(Collectors.toList());
421         Collections.shuffle(turnOrder);
422         List<Long> clientIds = turnOrder.stream().map(ServerThread::getClientId).collect(Collectors.toList());
423         sendTurnOrder(clientIds);
424         clientsInRoom.values().stream().filter(client -> !client.isReady()).collect(Collectors.toList()).forEach( e -> {
425             sendGameEvent(e.getClientName() + " is spectating this game");      You, now • Uncommitted changes
426         });
427
428     } <- #418-428 private void setTurnOrder()
```

GameRoom.java exclude non ready players from turn order and send spectator message

Room.java ignore messages from not ready players

Saved: 8/9/2025 1:56:23 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:

When the turn order is created, it is sent to the clients ui in order to update the userlistview. If a client is not set to ready, they will not be included in the turn order and won't be displayed by the userlistview. The game room only goes through clients that are in the turn order. Clients will only be added to the turn order if they are ready. In the relay command, it checks to see if the ready status of the sender is true. If it is not, the relat function is closed and the message will not be sent. NA

Saved: 8/9/2025 1:56:23 AM

Section #5: (1 pt.) Misc

Progress: 95%

Task #1 (0.33 pts.) - Github Details

Progress: 87%

Part 1:

Progress: 75%

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

Missing Caption

Saved: 8/9/2025 1:58:14 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/ColinRafferty7/ctr26-IT11[illegible]

URL
https://github.com/ColinRafferty7

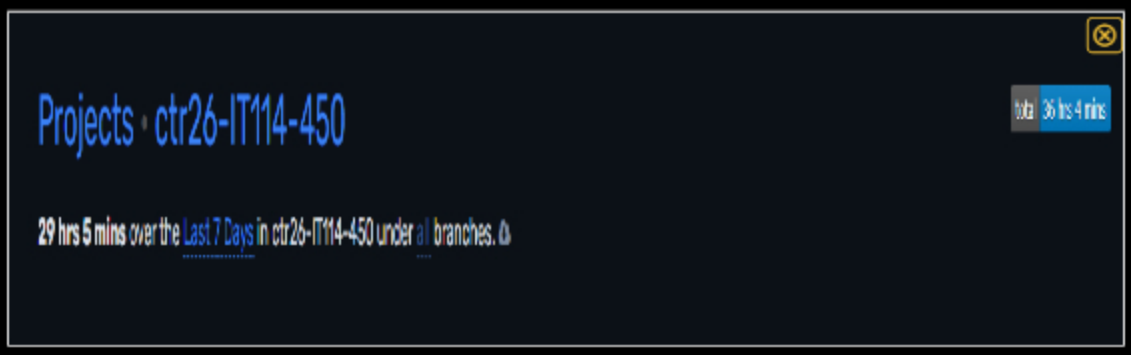
Saved: 8/9/2025 1:58:14 AM

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 top

WakaTime bottom

Saved: 8/9/2025 1:59:34 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)

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

Your Response:

For this final milestone, I learned how to communicate back and forth between client and server in order to maintain ui updates. It was a lot more tedious than I had assumed because changes do not just happen universally. To change an item on the screen, that information needs to be sent out to every client.

Saved: 8/9/2025 2:01:10 AM

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 this assignment was plugging in the baseline code and going through revisions page. It was fun to go through each line or code and function and revert the ones that were specific to my prjoect.

Saved: 8/9/2025 2:02:19 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:

The hardest part of this assignment were all the little bugs that would add hours onto my development time. In such a bare bones client-server setup, the smallest mistake of a payload can cause the entire program to halt. The most annoying part is going through all of the different files and keeping track of the long function changes. As the project got bigger, it would take longer to open each file as well.

Saved: 8/9/2025 2:04:30 AM