CS 300: Advanced Programming Spring 2022
Assignment 3
Sequence Board Game

In this assignment you are required to implement a single multiplayer game of Sequence for four players (2 in each team). Player 1 and 3 will be team 1 and player 2 and 4 will be team 2 by default. You are not required to support multiple parallel games or to support a second game after the first one finishes or to support disconnecting players. You can read more about the game here. However, for simplicity we will not be following all the rules of this game.

Part 1 [20 marks]:
Remove the hard-coded sequence board from client.html and generate it dynamically using a React Component called Sequence. Remember that class attribute is written className in React. You do not need to understand or change sequence.css.

The file server.js has a hardcoded constant variable (2d array) named board Change the initial board to [[]] in the client.js. Then introduce websockets and have the server send the initial board to the client. The message format will be JSON with a key named 'type' where the only message we will be handling in this part is where type = 'newboard' and a second key named 'board' which has the sequence board. Replace the board in your state (on client side) with the one coming from the server. Now display the board dynamically.

After completing the steps above, edit the message that was being sent by adding one more key with the message. This key will be named 'positionBoard', which will contain the positionBoard which is a 10 by 10 board of "-" initially. On the client side, create a state variable named 'positionBoard' and initialise it as [[]]. Replace the positionBoard in your state (on the client side) with the one coming from the server.

Part 2 [20 marks]:
Now, you need to send the client 6 cards by randomly generating sets of 6 cards from the deck. The file server.js has a hard coded variable (1d array) named deck and a function named divideDeckIntoPieces. The function divides the deck randomly into pieces of 6 (8 pieces of 6 cards and a piece of 4 cards). You can use the first piece of 6 cards for now since we are not differentiating between clients for now.

On the client side, create a state variable named 'deck' and initialise it as []. Then send the cards to the client by either creating a new message or editing the previous message from part 1. Replace the deck in your state (on the client side) with the one coming from the server. Then display the cards dynamically.

Use the className "text_box" to add a text box as well along with a circle of colour blue or green to denote the colour for that certain player. You can do this by adding the following lines:

 Add message here
<div className = “color green” >

Please note that the className for the circle can be “color blue” as well. It will be assigned by the server once your game starts to support multiple players.

Part 3 [20 marks]:
You will now introduce taking a turn. Write a function that is called every time a card is clicked on the sequence board (not the 6 cards given to the client - just the cards on the board). The function checks if the card clicked is present in the cards that have been provided to the client. If it is present then move is valid and that certain card is removed from the cards array (remember that the cards array is being displayed as well, so it must update instantly!). Then the client sends a message to the server which consists of the index (row index and column index) of the card that was clicked.

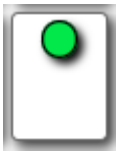
Upon receiving the message from the client, the server updates the positionBoard by replacing the “-” with a “green” or a “blue” where g and b stand for green and blue *(Please note that 2 players will be assigned with green colour and 2 with blue. But your game is for 1 player at the moment, so you can come back and edit this later when your game starts to support multiple players)*.

After updating the positionBoard, the server will send it to the client and the client will replace the positionBoard with its newer version.

Now you will use the positionBoard to add conditions in your html so that a circle of colour green or blue is shown on the card according to the positionBoard.

For example, if the positionBoard has a “-” on index [0][1] then it will be shown as:

If the positionBoard has a “green” on index [0][1] then the card above will be shown as:



Similarly, if the positionBoard had a “blue” then the circle would be of colour blue.

The html code for a blank card with a green circle is:

The html code for a blank card with a blue circle is:

However, if the move was not valid (if the client doesn't have the card they clicked) then an appropriate error message is shown to the client. Please note that the client must make a valid move and only then the game will move forward!

Part 4 [20 marks]:
Now modify the game into an actual 4 player game. Follow the below steps to do so:

Wait for four players to connect. Assign each player a colour using a new websocket message that is sent when the player connects. This colour should be visible to the user. Player 1 and 3 will have the same colour, while player 2 and 4 will have a similar colour. However, the colours can only be blue or green.

Once the fourth player arrives, send them the board, positionBoard and cards (6 cards each) as done in part 1 and part 2 of this assignment. The function divideDeckIntoPieces made 8 sets of 6 cards, so for 4 players we send them the first 4 sets only.

Now modify your program such that all four players can play turn by turn. The turn of the player should be displayed in the text box. All your clients should be able to see whose turn it is. You can assign them a unique name, such as player1, player2 etc. After each player's turn, the server sends the updated positionBoard to all of the clients.

If it's player1's turn but any of the other players click on the cards then show a message to them that it is not their turn. Similarly, during their turn if a player clicks on a card (on the board) that they don't have in the set of cards they received, then an appropriate message is shown in the text box to them.

Part 5 [20 marks]:
In this part you will create a winning condition. However, before that some of the rules and functionality needs to be implemented.

Firstly, we will be implementing the functionality of jacks in your code. Previously, when a card was clicked on the board then it was checked if the client had that specific card with them. If they didn't have it, then an appropriate message was shown; otherwise, a coin was placed on top of that card. However, if the client has a jack then the client will not be able to make a move according to your code (at the moment - we will make it work tho!). This is because jacks don't exist on the board.

Change your code in such a way that if a client has a jack then they can place a coin anywhere on the board, provided there is no coin on that position already.

Now, we create the winning condition. At any point, the first team to create a sequence of 5 chips either up or down, across or diagonally wins. This is the same as having 5 chips of the same colour up, down, across or diagonally. When a team wins, you have to inform all the players of the winning team in the text/message box.

After all the players run out of cards for the first time, which will be after all of them take 6 moves each, the server then sends the unused cards to each player. Please recall that in part 3 we sent the first 4 sets of 6 cards, while there were 4 sets left. Now the server will send the other 4 sets to the players. This way every client will always have a unique card with them and the dead card condition will never hold. If there is no winner after 12 moves each, then the game is considered as a draw. You have to inform all the players about the draw in the text/message box.

Submission Instructions and guidelines:
Submit client.js, server.js, sequence.css and client.html. The deadline for this assignment is **11:55 PM on 17th April**.
Please email/DM Ahmed or Sameer for any queries related to this assignment. Happy Coding! 😀