CMPE2550 – Assignment 02 – Othello

In this assignment, you will explore use of a PHP session, continuing to work with PHP and the different ways in which it may be included within your page, and brush up on some client-side skills such as AJAX calls and dynamic page updating.

Othello is a fairly common game. The players take turns placing stones, attempting to capture the most squares by "bookmarking" their opponent's stones. This means that when a player places a stone, all of the opponent's stones in between two of the placer's stones are flipped to the placer's colour. Think of this as a much simplified version of Go.

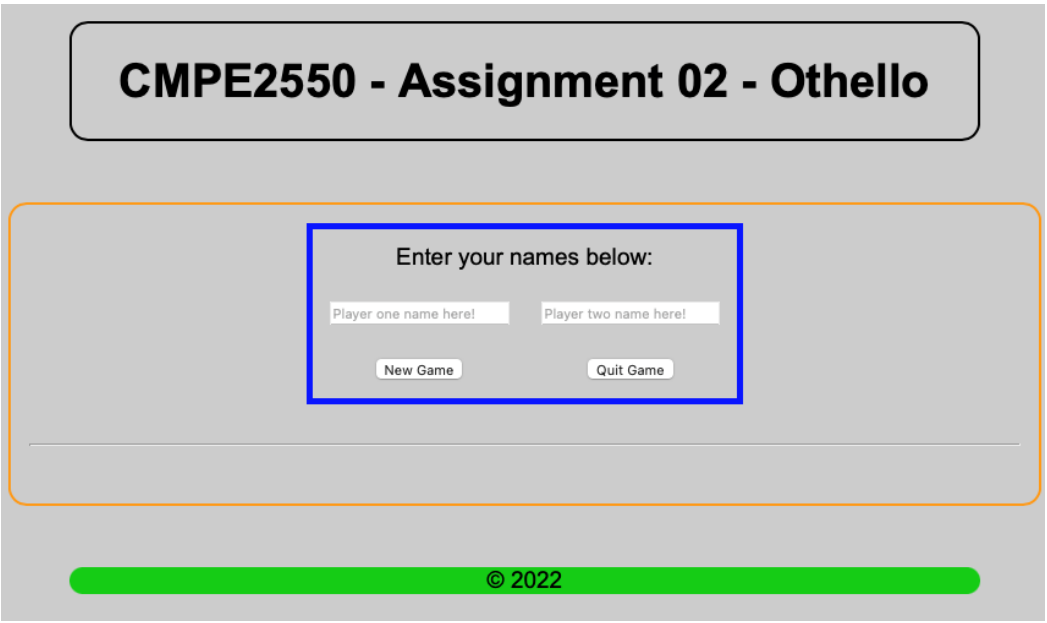
Create a new Assignment02 folder. You will require at least 4 files (exact names not required):

index.php	This file will be your landing page and will contain the code for creating and destroying sessions.
index.js	This page will process the user's actions from the index.php page.
Index.css	This page will be used to contain both regular static classes, and classes that may be added and removed as required by the application logic.
gameplay.php	This will be a service file for containing the "brains" of the application.

You may include other pages as you feel are necessary for organization and mobility purposes.

An example layout and ***very*** simple style has been shown, but it is not mandatory to replicate either. Feel free to be artistic as you please, but it is advised to complete the functionality first.

When a user first navigates to the page, a simple layout is seen such as that shown below:.

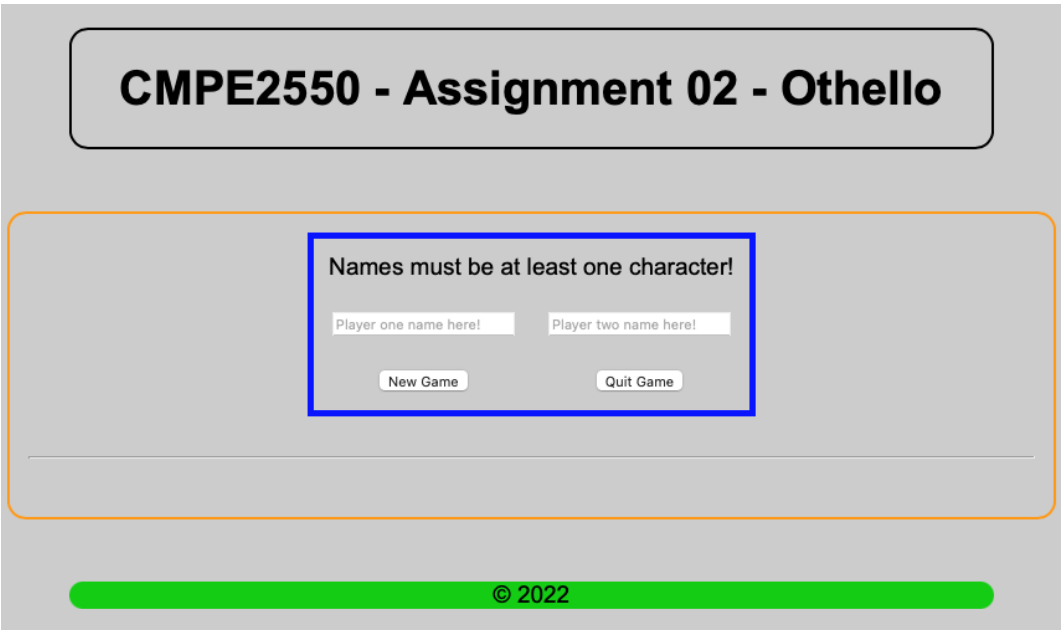


The controls in the sample image are:

1) One Label for messaging the user
2) Two TextBoxes for the players' names
3) Two Buttons for starting and resetting the game

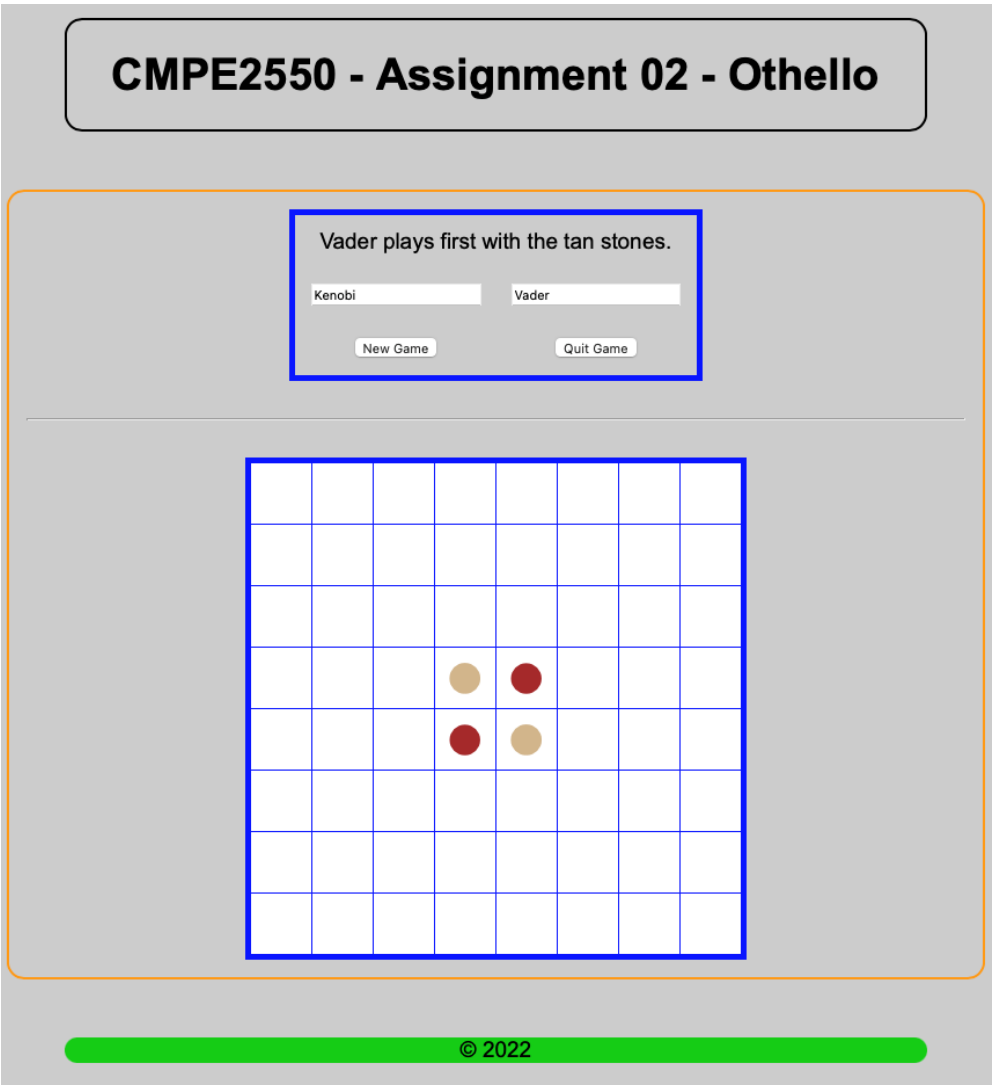
This example is a minimum bar to strive for. As you complete the assignment, you may choose to add more to the basic requirements.

When the user presses the [New Game] Button, the submitted data will be cleaned and examined to ensure both names were provided. An appropriate error message should be displayed if either or both of the names is missing.



Once both names have been supplied, the initial gameboard shall be shown, along with the random player that has been chosen to place the first stone, and their colour of stone. Values should be stored into the session so that the server may keep track of the game details. See the first image on the following page for a sample game layout. Note that as they are constructed, you will require a naming scheme for the individual cell ids. An AJAX call will be sent to the server when the page is loaded indicating that the server should initialize a new game. It is suggested you send an "action" to the server with all AJAX calls.

The shown gameboard at the top of the following page has been constructed using readonly TextBoxes, placed immediately beside each other (no table structure), and some single bordering (some borders are set to 0px to make sure duplicate borders do not show up. This is a good base of controls to which we may add events. Hint: I constructed this in Javascript as a string that was then poked into a waiting div on the HTML page.



If the [Quit Game] Button is pressed, the session variables will be unset and the session destroyed. This should return you to the game state seen in the first image in this document.

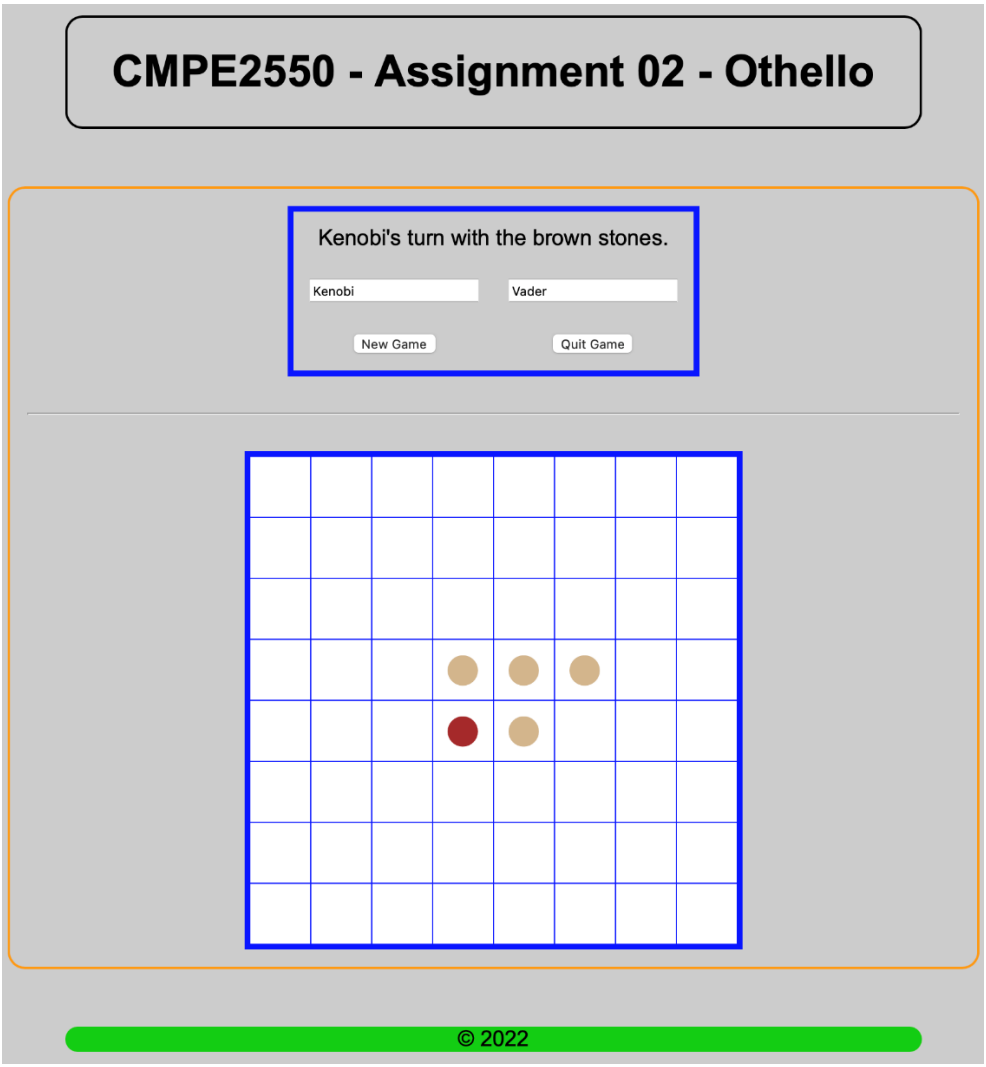
Note that there are four stones placed to begin the game. You must follow this pattern. If you wish to replicate the filled in circle character, use ●, similar to the © you may have used in the past for the copyright symbol.

The game play itself is rather simple. We will provide a click method for each of the included Textboxes. When clicked, the handler will extract the row and column coordinates of the cell and send them to the server for processing via AJAX call.

The server will recursively examine the 8 straight line directions from the placed stone. If a stone the same colour is encountered, then all stones of the opponent's will be captured and the colour displayed changed to match the current player's stones. No changes occur if the

edge of the board or a blank space is encountered before another of the current player's stones. Hint: get one direction working properly, then rinse and repeat for the other 7 directions. The only thing that should change is the step away from the placed stone.

As a simple example, the following image shows one possible result from the first stone placement.



On the server side, each interaction should include interrogating what action has been requested by the client. Game initialization and processing a stone placement should be the new actions you need.

For game initialization, it is suggested that you create a 2D array of integers that will be initialized to 0 to indicate empty cells. Then set player one stone to 1 and player two stones to 2. This 2D array may then be saved into the session as a json_encoded string, and also returned to the client for display processing. You should also be storing the player names and whose is

currently supposed to be placing the next stone. That way a move can be properly validated and processed upon request.

For a stone placement, json_decode the game data string from the session into a local variable (which will now be the 2D array of integers, change the value at the indicated location, check whether any other stones need to be changed over and if so make the changes, then re-encode the 2D array for saving back into the session and for delivery back to the client.

For all actions, the game data received back on the client will need to be examined and the proper coloured stones placed in the appropriate cells determined by the value of the integer for that space in the 2D array. Other data may also be passed back to support other features you wish to build in to your application. It is advised to get the basic functionality sorted out first.

Note: This will be a living document as it is the first time it is being delivered. If typos are encountered, or an explanation is not clear or seems incomplete, please approach me about it. If the fix is complex enough, I will add to or change this document and alert the classes that it has been changed.