CMPE2550 – Assignment 02 – Tic Tac Toe

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.

Tic-Tac-Toe is a fairly common game. The players take turns placing marks ("X"s and "O"s), attempting to capture three squares in a straight line.

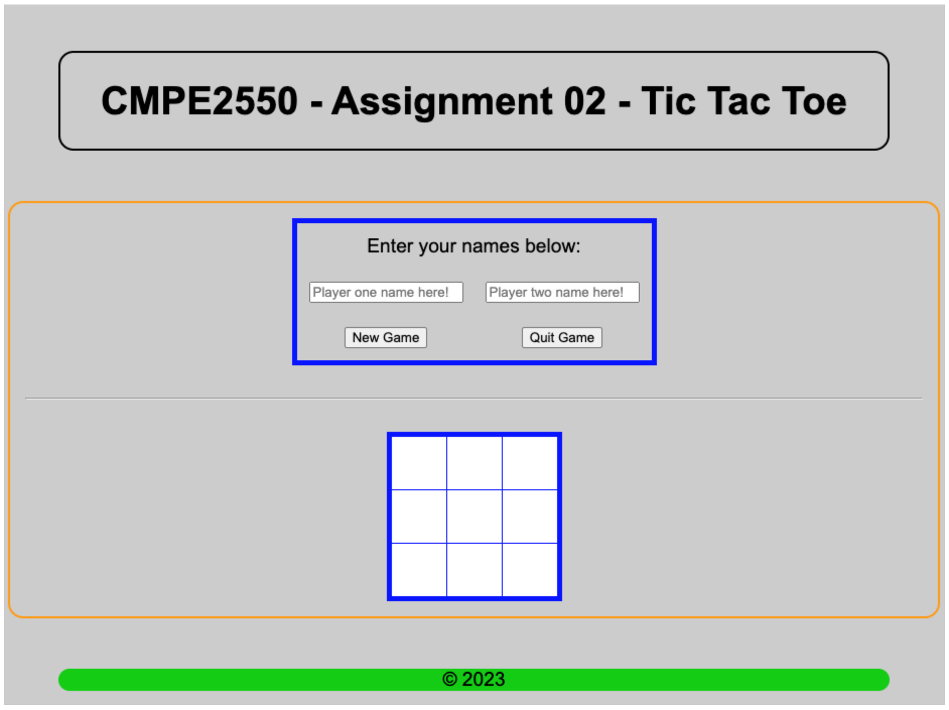
Create a new Assignment02 folder. You will require at least 4 files (exact names not required):

index.php	This file will be your landing page and will contain the code for creating and destroying sessions, and "sessioning" player information.
TicTacToe.js	This page will process the player's actions from the index.php page.
TicTacToe.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 (mostly for indicating types of user messages).
gameFlow.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 ***<u>very</u>*** simple style has been shown, but it is not mandatory to replicate exactly. Feel free to be artistic as you please, but it is advised to focus on the functionality first.

When the players first navigate to the page, a simple layout is seen such as that shown below:

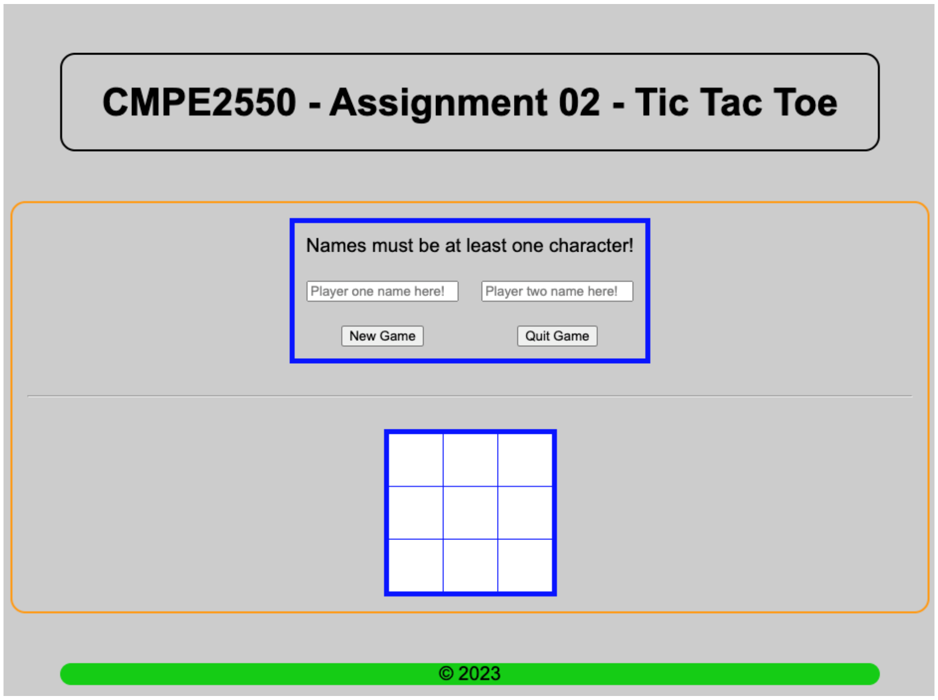


The controls in the sample image are (1-3 built into a form that posts back to index.php):

1) One Label for messaging the user
2) Two TextBoxes for the players' names
3) Two Buttons for new games (same players) and resetting the game (including players)
4) The blank gameboard (you could choose to show or hide this on New and Quit clicks)

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.


CMPE2550 - Assignment 02 - Tic Tac Toe

Names must be at least one character!

Once both names have been supplied, the random player that has been chosen to place the first mark (always "X") shall be indicated in the status message label. Values should be stored into the session so that the server may keep track of the game details.

Note that as they are constructed, you will require a naming scheme for the individual cell IDs in the gameboard (row_column perhaps?).

An AJAX call will be sent to the server when the page is loaded indicating that the server should initialize a new game and "session" any initialized data items. It is suggested you send an "action" to the server with all AJAX calls (perhaps "NewGame" here?).

The shown gameboard has been constructed using readonly TextBoxes, 50px x 50px in size (set in CSS), 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: The inputs were constructed in Javascript as a string that was then poked into a waiting div on the index.php page, and then the borders were added using the DOM, but there are other options.

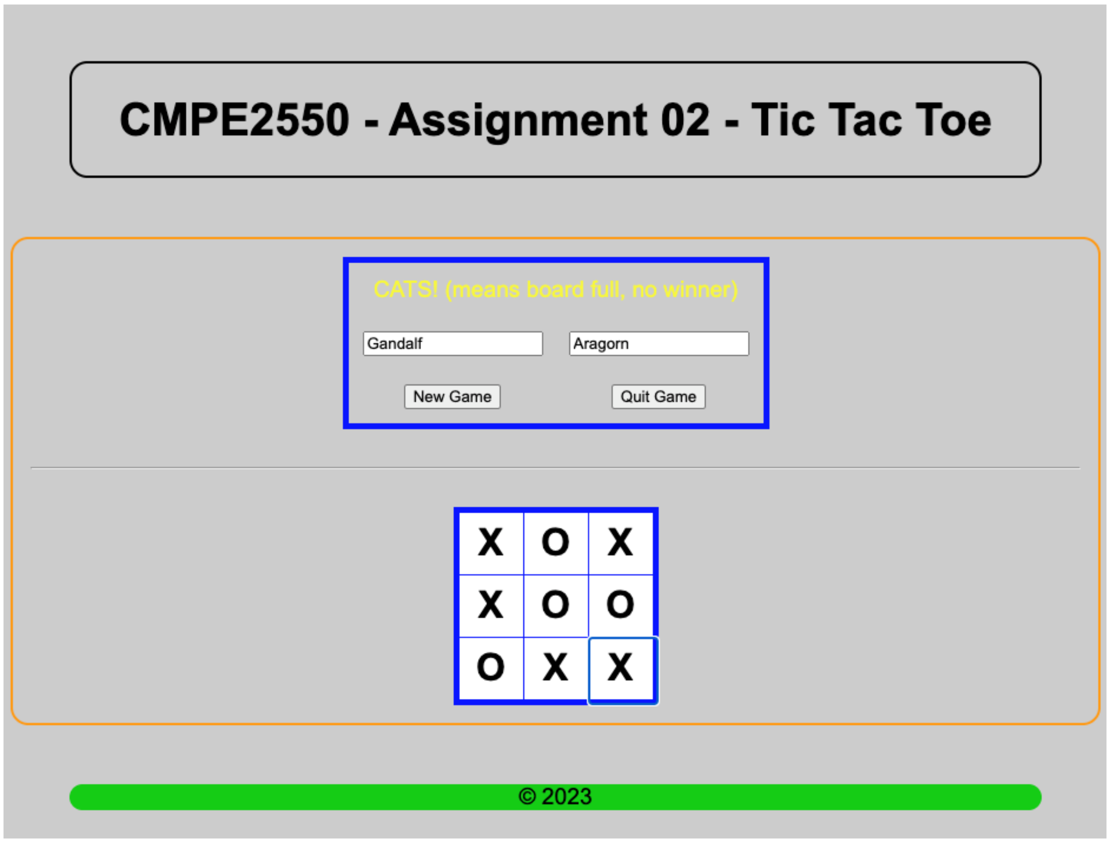
If the [Quit Game] Button is pressed, the server session variables will be unset and the session destroyed. This should return you to the game state seen in the first image in this document.

The game play itself is rather simple. We will provide a click method for each of the included Textboxes (there is more than one way to accomplish the task, but “this” will be useful). When clicked, the handler will extract the row and column coordinates of the cell from the clicked Textbox ID and send them to the server for processing via an AJAX call.

The server maintains ALL of the gameplay information. The ONLY data that should be passed to the server as a result of clicking a cell should be the row and column that was clicked. This means that the server “knows” the current state of the gameboard, which player’s turn it is, what that player’s mark is (“X” or “O”), whether the action or move is valid, etc. Also remember that your intentions may be skirted by malicious users, so in all cases of data sent from client to server, the data shall be cleaned and validated (think data types, ranges, etc).

After the data has been validated and a player mark placed on the gameboard, you must check for a win condition. Again, this is the successful placing of the same mark in a line of three in a row, column, or diagonal. If all squares have been occupied and there is no line of three for either player, then the game ends in CATS! (no winner).

The following page shows a couple of possible game endings (note how the player names have persisted in the textboxes all along):

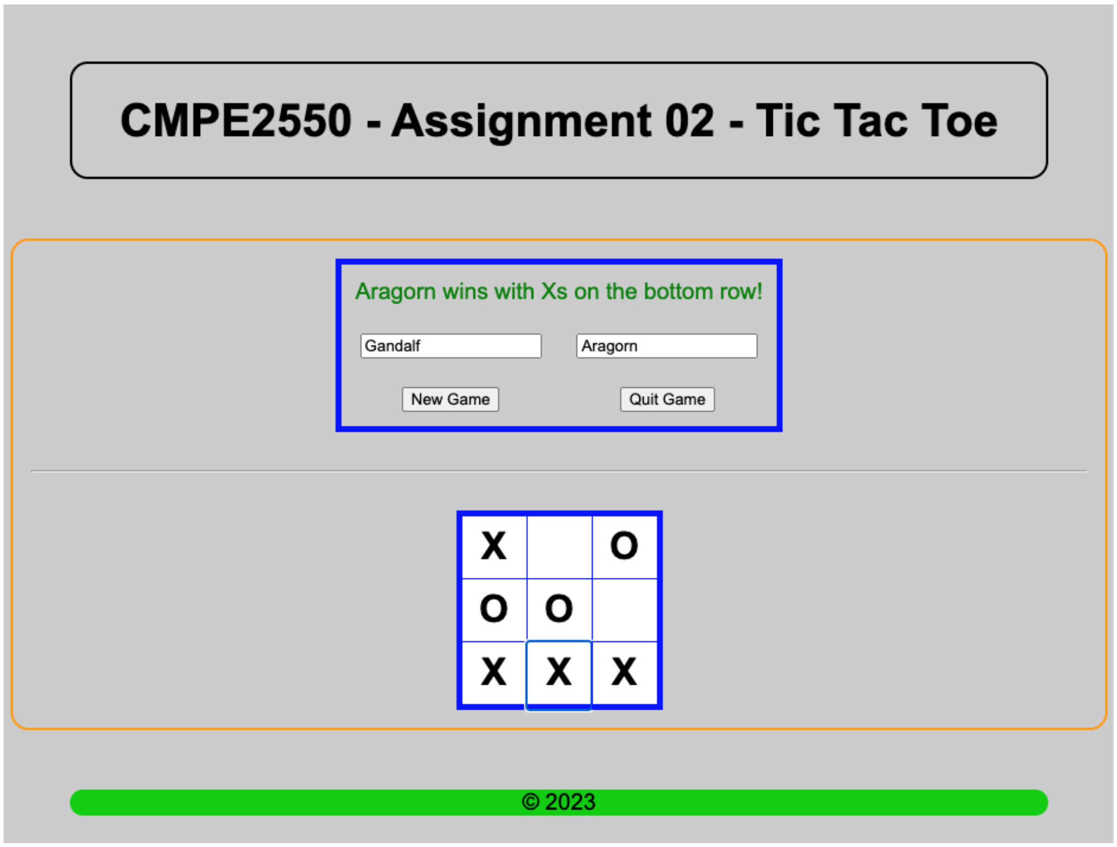

CMPE2550 - Assignment 02 - Tic Tac Toe

CATS! (means board full, no winner)

Gandalf | Aragorn

X	O	X
X	O	O
O	X	X

© 2023


CMPE2550 - Assignment 02 - Tic Tac Toe

Aragorn wins with Xs on the bottom row!

Gandalf | Aragorn

X		O
O	O	
X	X	X

© 2023

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

For game initialization, it is suggested that you create a 2D array that will be initialized to 0 to indicate empty cells. You will already have saved the user data for which order the players will place their marks and which mark they will play within your pre-processing block of the index page. You may use other variables saved to the session to help you process player moves and control the overall game play.

Hint: You may save any type of data to the session, but you may only error_log strings to view in the error_log file in your web space, so json_encode and json_decode will be your friend for converting arrays and objects to and from json string format.

For all actions, the game data received back on the client will need to be examined and the proper marks placed in the appropriate cells determined by the values returned from the server.

As a best practice, you should set up a "return data" array and populate it while completing a required action. Program flow should be laid out very minimally at the top of your service PHP file in an if-else structure, and the implementation for each action should be build into functions that are called. A single echo following the if-else ladder should be used to send the "return data" back to the client where it may be processed in Javascript. Remember that not just data but user messaging may be returned.

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 your instructor about it so that the feedback may be passed to the document creator. If the fix is complex enough, this document will be added to or changed and the class alerted. Do keep in mind that you are expected to do some of the thinking and specific design, so this document has not been designed as a step-by-step recipe for completing the assignment, but more intended as a discussion of the requirements.