CSE 264 – Web Systems Programming, Spring 2017
Class Project – Word Search Game - Due 4/27/17 - Day we will play in class.

- **Objective**
  The project will bring together a number of different topics we have covered in class and provide additional implementation experience.

- **Description**
  The completed project will implement a web application for playing wordsearch with multiple players over a network.

- **Instructions**

1. Setup

(a) Create an HTML5 application project in NetBeans and call it WordSearchGame-<lehigh-email-id>. Eg., my version would be called SetGame-jaf207.

(b) The basic structure of the application will be a single html page that uses Ajax calls and web sockets to communicate with the server.

(c) Include all your JavaScript code and CSS styles in separate files. Load the jQuery library from the CDN.

2. Play of the game

(a) In order to play the game, the user needs to enter a username into a text box on the screen and click the "Register" button which will send a request to the server to add the player to the game. The server will send back a string with the code that the player will have to submit with each additional request. This will identify the player to the server.

(b) The main playing board will be a large table of letters. Each letter should be in a square cell in the table. The table should have no borders or lines between the letter cells. The table will have to be both horizontally and vertically scrollable, since it will not fit on the page. Words will be embedded in the table in 8 different directions. All letters will be upper case.

(c) Above the letter grid should be a text field giving the subject of the wordsearch, indicating the category from which the words in the grid are drawn. For example, "Presidents of the US", "Mathematics Terms", etc. No list of words is provided.

(d) Each player gains points by identifying words by clicking on the letters in the word and then clicking the "Submit Word" button. As each letter is clicked the background and foreground colors should reverse. A user may toggle a letter by clicking on a selected letter again to unselect it, in which case the coloring of the letter should return to normal.

(e) When the user clicks "Submit Word", the client does an Ajax call to the server which sends the row/col coordinates of each letter in a list to the server. The server will determine if the letters constitute a valid word that has not already been selected and will send a response. If the word is accepted, the server will update the player's score with one point per letter.

(f) Periodically, as words on the grid are claimed, the server will use socket.io to send updates to all the clients with a list of any words that have been claimed. These words will be highlighted on the grid, by changing the background color of the cells. When two words cross, the color of the common cell is an addition of the colors of the other cells.

(g) Word submissions will be processed by the server in the order received. Only the first player to submit a word receives credit for it.

(h) There is a leader board which displays the list of players and their scores. The server will update this list and resend it to the clients using socket.io as the scores change. The client needs to redisplay the list every time a new one is sent.

(i) The games ends with all the words on the grid have been claimed. When the games ends, the winner will be highlighted in the leader board.

3. Implementation Details

(a) Ajax Calls

Function	URL	Parameters	Return

Login	.../wordsearch/login	{
		username: “.....”,
		}
			{
			success: true/false,
			id: “.....”,
			username: “.....”
			}

Get Puzzle	.../wordsearch/puzzle	{
		id: “.....”
		}
			{
			success: true/false,
			theme: “.....”,
			nrows: <integer>,
			ncols: <integer>,
			grid: “.....”
			}

Submit Word	.../wordsearch/submit	{
		id: “.....”,
		letters:
		[{ r:<integer>,
		c:<integer>}, ...]
		}
			{
			success: true/false
			}

(b) Socket Events

Event	Data
players	[{ name: “<username>”, score: <integer>, winner: true/false }, ...]

gridupdates	{
	words:[
	{
	text: “<the word that was claimed>”,
	letters: [{r:<integer>,c:<integer}, ...]
	}, ...
	]
	}

Refer to the set game client for an example of how to setup the socket.io event handlers.

To test your client, run from NetBeans or reload the html page.

To submit, clean the Netbeans project, zip it up and upload to Coursesite.