CS 300 Spring 2021 Exam 2

- A textual description is REQUIRED with each part about how you solved it.
- Note that you must attempt this exam in the way taught in class and in skeleton code. In particular, you must not use 'document' except the one use already there, must not use the 'class' keyword, must not use the 'function' keyword, must not introduce global variables in client.js, and must not change client.html
- You may only email the instructor about the exam (junaid.siddiqui@lums.edu.pk). TAs will not be available.
- You can work on the exam till Sunday 4 April 11 pm. Submit on LMS.
- You must take the exam completely alone. You cannot consult with any other person regarding the exam or any material from the course for the duration of the exam. Except lecture summary emails sent by the instructor and resources linked at https://www.webfx.com/blog/web-design/free-javascript-books/, you must not consult any other resource including searching the internet, accessing shared documents, or accessing any prior code you had. Report any violations and remember that academic misconduct can result in disciplinary action.

High-level overview

Wordament is a game played on a 4x4 board where each tile is one letter of the English alphabet. —a word tournament—where you are competing with everyone else connected to the same server to be the best word searcher in every game. Every player is competing on the same board, in real time, to get the highest score.

At the beginning of each round, the player is presented with a board consisting of 16 tiles (4×4), each containing a single letter. Each letter is assigned a point value ranging from 1 – 10. Start searching the board for words in the puzzle. A word consists of any connected sequence of letters, in any direction, but you may not use a single tile more than once for a single word.

You are given startup code that shows a hard coded 3x3 grid (instead of 4x4 generated with loops) and only hooks the first button with mousedown, mouseup, mouseenter, and mouseleave event handlers. Run and understand how the code is working before moving on. Remember to run 'npm i ws' in the directory with these files.

You can only change client.js and server.js and only submit these two files. Make a copy after a part is correctly working. Submit three files of the last working part and then the same three files if you have attempted the next part but it is not working correctly.

Part 1: Game board and one letter selection

Make the grid 4x4 and generate it using a nested loop. When the user clicks Submit button above the grid, send the username entered in the input box to the server using web sockets. The server should send back sixteen letters that the client should show on the grid. The sixteen-letter string is to be hard-coded at the server in this part, but not at the client. If mousedown event is fired on any button, send the corresponding letter to the server, and the server should send it back a message to that one client only which will cause the client to show 'User X clicked the letter Y.' in the messages list.

Part 2: Word selection

For this part, the target is to capture all letters on which the user moved their mouse between mousedown and mouseup events. This can be done using the mousedown, mouseenter and mouseup events. You can keep a list of array indices in component state. Do not repeat code for each button. You do not need to handle the case if the mouse goes to a letter that is not adjacent in this part.

When mouseup is received, send all indices to server, which sends the "word" made back to the client which will then show "Word W scored 0 points." Background of selected letters should turn green, and all backgrounds should be reset on mouseup.

If the mouse goes back over the second last letter, the last letter should be unselected. For example, if EXAM is selected and the mouse goes back to the same A (there could be multiple A on the board), then the selection becomes EXA.

Part 3: Word validation

This must be done on the server. After receiving a set of indices, the server verifies that at least three indices are received (i.e., at least three letter words), no index is used twice, and that consecutive indices are adjacent on the board. If not, it tells the client which shows to the user “Word W was not accepted by the server.”

Add another check to the server that the word made is a correct word. If not, the client should be told which shows the user that “Word W is not a dictionary word.” Use the given “dictionary.txt” file that contains all valid words. You can use split(‘\n’) to split the string read from the file into an array of valid words.

Part 4: Scoring

The 26 letters are assigned the scores A 2, B 5, C 3, D 3, E 1, F 5, G 4, H 4, I 2, J 10, K 6, L 3, M 4, N 2, O 2, P 4, Q 8, R 2, S 2, T 2, U 4, V 6, W 6, X 9, Y 5, Z 8. Server maintains the total score of each client. Whenever a client makes a valid word, the server calculates its score, tells the client in its response message, and updates its own total score as well. For each valid word, the client will show the score it received for this word and the current total score.

Part 5: Multiplayer

So far, there was no broadcast. While we can broadcast every word every client makes, we will use a more efficient scheme. Every second (use the delay function given in server.js), only if anything has changed (use a flag for this), the server will send the username and score of top three users of this game. Make new divs to display this information near the top of the page.

Also, every 60 seconds, switch to another board and reset all scores. We have given 20 possible board arrangements in a file “boards.txt”. You can simply cycle through these 20 arrangements. When sending the board to the client, tell the client how many seconds are left, and the client should show a client-side timer using the delay function in client.js to display remaining time to the user.