| | | | Weekly Sprint Sheet | | Team Members | Email |
|---|---|---|---|---|---|---|
| **Project: 5** | | | | | Arpita Kumari | akumar71@uic.edu |
| | | | | | Isaac Kim | ikim32@uic.edu |
| | | | | | Lydia Tse | ltse5@uic.edu |
| | | | | | Neil Thimmaiah | npemma2@uic.edu |
| | | | | | | |

| Action Item | Item ID | Team Member | Last Week | Week 1 | Week 2 | Issues |
|---|---|---|---|---|---|---|
| Setup server & client and allow multiple clients to connect to the server | | Lydia | N/A | Read through Socket.io and Node.JS documentation | 1. Implement HTTP server<br>2. Instantiate IO object whenever client connects to waiting room webpage<br>3. Create handlers for established connections and disconnections<br>4. Pass the correct information regarding client between different html pages | - Issue with keeping socket consistent throughout the different html pages<br>- Researching solution with Express and Socket.io "handshake" |
| Implement User Interface Designs for Webpages | | Isaac, Lydia, Arpita | N/A | 1. Read about Node.js, React.js<br>2. Learn how to implement UI elements<br>3. Prototype for the necessary user interfaces using Figma | ~~1. Create elements using React.js for each of the prototyped pages~~<br>1. Create the various html pages for each of the user interfaces<br>2. Make the Interfaces user friendly<br>3. Debug the code for GUI Elements. | - React has a very steep learning curve especially since we are all new to both Node and React<br>- We decided to use vanilla HTML/CSS/JavaScript to create the UI and handle events |
| Implement buttons to play again or to quit:<br>If Play Again button is chosen, player will be pushed to the end of the waiting room queue(the queue that is used to hold the new players)<br>If Quit button is chosen, disconnects player from the game and close the connection of that player to the socket | | Neil | N/A | 1.Create the buttons for each player to quit.<br>2. Try creating the button to play again if the first round works fine. | 1. Make sure implementation of all buttons work fine.<br>2. Check to make sure there is no error while closing the connection of a player using the quit button<br>3. Play again should reset everything for that specific player. | N/A |
| Develop the set of words to use in game | | Isaac & Lydia | N/A | 1. Review documentation for Oxford Dictionary API | ~~1. Develop filtering logic to filter words that are 5 - 7 letters in length and whose letters only produce one valid word~~<br>1. Use a data structure to store the 20 words that we came up with as a team and allow for 1 of the 20 to be selected at random<br>2. Allow for the chosen word to be scrambled and displayed to the user | - We no longer need to use the Oxford Dictionary API<br>- We created a list of words as a team |
| Use 4 variables to keep track of the score of each player | | Arpita | N/A | 1. Implement the 4 variables for each of the players. | 1. Make sure that the variables are not getting mixed up and each one corresponds to their respective players<br>2. using JSON to keep track of information regarding each player | Instead of having 4 variables into the game logic for each player, it is more efficient to have a player info(a JSON object), to be able to update each player's score depending on how fast they get the correct word. |
| Implement Waiting room logic | | Lydia | N/A | 1. Create a queue that holds all of the players in the order of descending wait time | 1. Test the queue implementation to ensure that waiting room allows players waiting the longest to enter the next available game | None |
| Use a time function on the server and calculate the time it took for winner to unscramble word | | Neil, Isaac | N/A | 1. Implement the time function for each of the 4 players 2. Exploring how to use Node.js, React<br>3. code a few small sample programs to see how these frameworks work.<br>4. pseudo code winner mechanics game logic | 1. Check to make sure the timer function is working<br>2. Required pausing should be working well too (when a player finished unscrambling the word, stop the time)<br>3. Debug and test this code thoroughly to check it is working | N/A |
| Use a boolean flag to tell the server that the client has done unscrambling the word and pause the time function for that player. Do the same for the other 3 players and check which player is the least and that player is the winner of the round (the fastest time to solve) | | Neil, Arpita | N/A | 1. Add a Boolean flag (4 flags)<br>2. Implement a way to keep track of the time of all the time limits of all the players | 1. Check to make sure that the boolean flag works correctly<br>2. Check to make sure that the least time is the one which being allocated as the winner.<br>3. Timer should pause every time a player unscrambles the word. | N/A |
| Use the timer module to have a max time limit for a game. | | Isaac | N/A | 1. Implement setTimeout() and pass it a function that will be activated to end the game. | 1. Create a Timer object from the built-in node.js Timers module<br>2. Make the code for the function that ends a game.<br>3. Debug to make sure a game ends for each possible game situation. | N/A |