Tug of Words

Team 05: Ammar Husain, Anoop Jain, Charlie Crouse,
Jiwon “Daniel” Kim, Jenna Ellis, and Prashanth Koushik

Design Inspection
Code Inspection
Unit Testing Log

Design Inspection:

Product	Tug Of Words, an IO type racing game		
Date	3/4/2018		
Author	Jenna Ellis		
Inspectors	Charlie Crouse and Jenna Ellis		
Recorder	Jenna Ellis and Anoop Jain		
Defect #	Description	Severity	How it was corrected
1	During in-game operation, we initially had the “current word” that the player was attempting to type stored as an attribute in that user’s object in the ‘users/’ section of the Firebase. This was unintuitive, and should be fixed to make more sense.	1	We added a section in the lobby portion of Firebase to include a field that stores the current word that the user is typing.
2	Previously, the socket operations for sendWord and verifyWord functionality was thought to operate entirely through the app.js file in the server repository, which contained all of the listeners for socket operations initiated by the clients. However, this design was inefficient, and could be optimized to prevent unnecessary debugging and errors with socket operations.	2	Socket operations will now be handled by the methods that actually generate new words and verify words submitted by each client, rather than through another function in the app.js file. Specifically, this required the sendWord and verifyWord methods to include a socket.emit() call that would either send a new word to each client or a boolean of whether the previous word submitted was correctly typed or not.

3	We initially designed our application around the concept of having pre-game holding rooms. We also used the concept of lobbies to simulate the portion of the application in which users pick their teams or are sorted into teams.	1	During the implementation process, we realized that the use of separate rooms and lobbies was redundant, and decided to combine both module's functionality into a single lobby module (on both the client and server code bases).

Code Inspection:

Code inspection revealed the following defects:

Product	Tug Of Words, an IO type racing game		
Date	03/04/2018		
Author	Development Team 5		
Inspectors	Development Team 5		
Recorders	Development Team 5		
Defect #	Description	Severity	How
1	While modifying the functionality of the sendWord and verifyWord methods, the name of the sendWord method was changed from getWord to sendWord in order to better reflect the method's added functionality in the method name. This caused the test for getWord to break, because the method no longer existed.	2	The solution was to recreate the getWord method exactly as it was before. The test would then perform its intended functionality, which was to check that the API that we are using to generate random words was performed as expected.
2	When reviewing a large number of code changes on the client code base, it was noticed that an action was exported twice, which caused a reducer to crash	3	The second instance of the action being exported was removed, and the code returned to a functional state.
3	When reviewing code written on the client side for lobbies, numerous variables were defined and unused, and were also incorrectly referenced undefined variables. This	1	Deleted unused variables from source code and verified console errors were gone.

	threw console errors when running any lobby code.		
4	While reviewing the lobby socket function, we noticed a firebase issue with updating the user's team	2	We added the correct firebase reference so the user is nested inside the team 1 attribute.

Unit Testing Log:

At the moment, we do have automated unit testing for the front end portion of our project. In order to automate our unit tests, we have used a library called enzyme that will allow us to initialize specific containers and components of our application and traverse through the output to make sure that there the render was successful. As any other testing framework, we are able to define suites for a full container, and different cases for properties and states of the component.

For the server unit testing framework, we decided to use mocha, a serial unit test framework which results in really nice reporting.

Our system relies on two sets of modules, one set for the client component, and one for the server component.

Client Modules:

- Main-Menu Module
- Lobby Module

Server Modules:

- App Module
- Firebase (Database connection) Module
- Game Module
- Lobby Module
- User Module

Client Module Testing Summary

Main-Menu Module:

The main menu module is a front end container that is used to direct the user to create a username; upon creating a username, the user is directed to a page that allows them to specify what sort of game to create (public or private). This page also allows users to "reset" their username, which allows them to return to the username input screen.

Product	Tug Of Words, an IO type racing game		
Module	Main-Menu Module		
Date	03/04/2018		
Author	Development Team 05		
Defect #	Description	Severity	How it was corrected
1	No changes were made		

Lobby Module:

The lobby module is a front end container that is used to hold users in between creating their name and starting a game. The lobby module focuses on a single page that a user is directed to after they choose a public or private game option. When a user is entering a private game, the lobby module allows the user to choose what team they are on, and then click start to start the game.

Product	Tug Of Words, an IO type racing game		
Module	Lobby Module		
Date	03/04/2018		
Author	Development Team 05		
Defect #	Description	Severity	How it was corrected
1	When joining a team in a private lobby, unit tests generated by the react testing framework implied that the default (null) case was always being encountered	2	This was corrected by fixing an incorrect parameter passing in the redux for lobbies, in which an activity variable was passed in instead of the proper user input

			variable.
2	When joining teams, other users cannot see live updates of other users joining teams unless they refresh the page (which is wrong in terms of our socket.io system)	1	Change emit to io.sockets.emit in order to update all users sharing the same socket connection

Server Module Testing Summary

<u>App Module:</u> This contains the actual server implementation. The code initializes the express/socket.io instance and has the logic for the socket listener functions.

Product	Tug Of Words, an IO type racing game		
Date	03/04/2018		
Author	Development Team 5		
Defect #	Description	Severity	How
1	In order for all users that have access to a private game to effectively share a lobby, http requests are not sufficient. Users cannot interact with each other.	2	Made joining a lobby a socket event instead of a http request
2	In order for users to pick and switch teams dynamically, http requests are not sufficient.	2	Added socket listeners and broadcasters to be used in lobby and in-game events (joining teams and starting the game specifically)

<u>Firebase Module:</u> The firebase module is a module that allows our application to connect to our Firebase real-time database.

Product	Tug Of Words, an IO type racing game		
Date	03/04/2018		
Author	Development Team 5		
Defect #	Description	Severity	How
1	No changes made		

<u>Game Module:</u> The game module supports all server operations for the in-game aspect of Tug Of Words. This includes incrementing and decrementing point values for users as they type words correctly and incorrectly as well as returning random words to clients as they request them. This also includes functionality that will verify whether a word submitted by the client was typed correctly.

Product	Tug Of Words, an IO type racing game		
Date	03/04/2018		
Author	Development Team		
Defect #	Description	Severity	How
1	Our test case for random word generation considered a threshold for 0.5% to avoid repetition of words to different team members or each team in general. However, we found that as the number of words generated increases, a 0.5% threshold was not enough because the probability of generating the same word again is higher with a higher n-value.	1	In order to resolve this issue, we have increased our threshold to 1% so that the words are still random, but allows for more words to be repeated given generation of more random words. In this case, we test a thousand words and allow for repetition of upto ten words. In most cases, this method only repeats about 2-4 words.
2	One of our design decisions during this sprint was that we changed the location of a user's current word and current score from the 'users/' section of the Firebase to the 'rooms/' section of the Firebase. This caused all of the tests for addPoints and removePoints to break, because they were still pointing to the old section of the Firebase.	1	In order to resolve the issue, we had to update the tests to reflect the new organization of the 'rooms/' section of the Firebase. In addition, we had to modify the steps to create a new room in order to reflect other new design changes.

<u>Lobby Module:</u> The lobby module is where the app is in a pre-game state. This includes several definitions of functions that user can access in order to join or be placed on teams.

Product	Tug Of Words, an IO type racing game		
Date	03/04/2018		
Author	Development Team 5		
Defect #	Description	Severity	How
1	All functions in the lobby module were returning undefined values, and we needed to add support for asynchronous actions	3	ES6 async await syntax was used to fix the undefined values
2	Users had no ability to leave or change teams once they joined one	1	Add implementation for functions to support this user action specifically using sockets.

<u>User Module:</u> The user module is where the players of Tug Of Words inherit their different attributes. This module contains the attributes of a user such as user id (uid) and username. This module is important because using this information the players can interact with other players by joining lobbies and collecting points. This user module will allow us to expand upon what the users can do.

Product	Tug Of Words, an IO type racing game		
Date	03/04/2018		
Author	Development Team 5		
Defect #	Description	Severity	How
1	No changes made		