**SOFT354 Client-side Web Scriping Report**

**Functionality**

My game is a web based multiplayer 2D Pong inspired game. It is played 1v1 with a public chat box that allows an unlimited amount of people to use as the game is being played.

**Technologies Used:**

● HTML5 & CSS

● HTML5 Canvas

● JavaScript

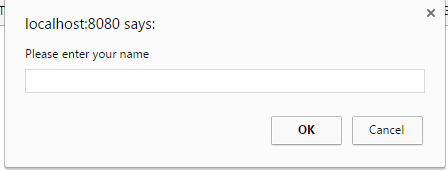
● Socket.IO (WebSocket)

● Node.js

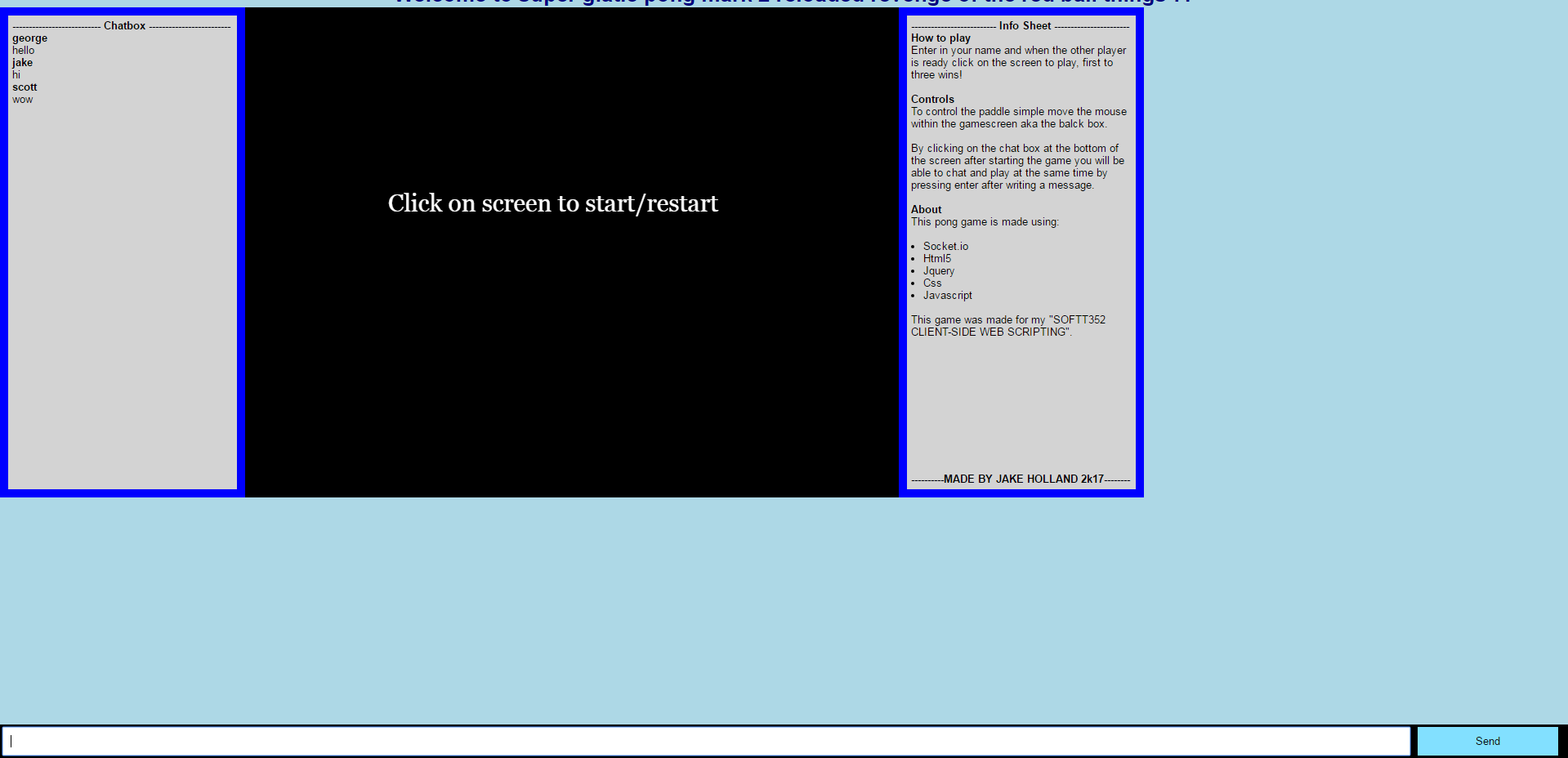
* JQuary

**Chat:**

To start the game the players will need to connect to their localhost port 8080. By doing this they will be greeted with this windows promp asking for them to input their name.

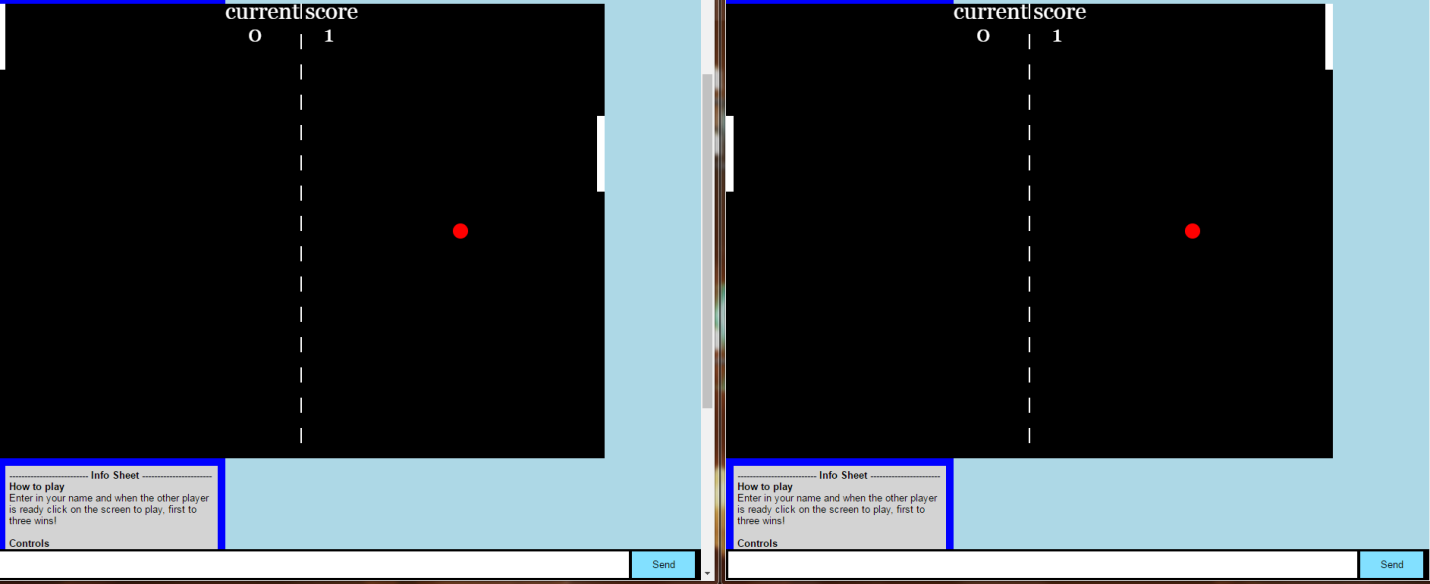


This name will be used to display who said what in the public chat which all players who join the server will use. The chat box is displayed to the left of the game canvas and displays everyone's name in bold with their message below. Since the game is designed to be played with the mouse, this allows users to type and send messages by clicking the button or pressing enter as they play.



**Game:**

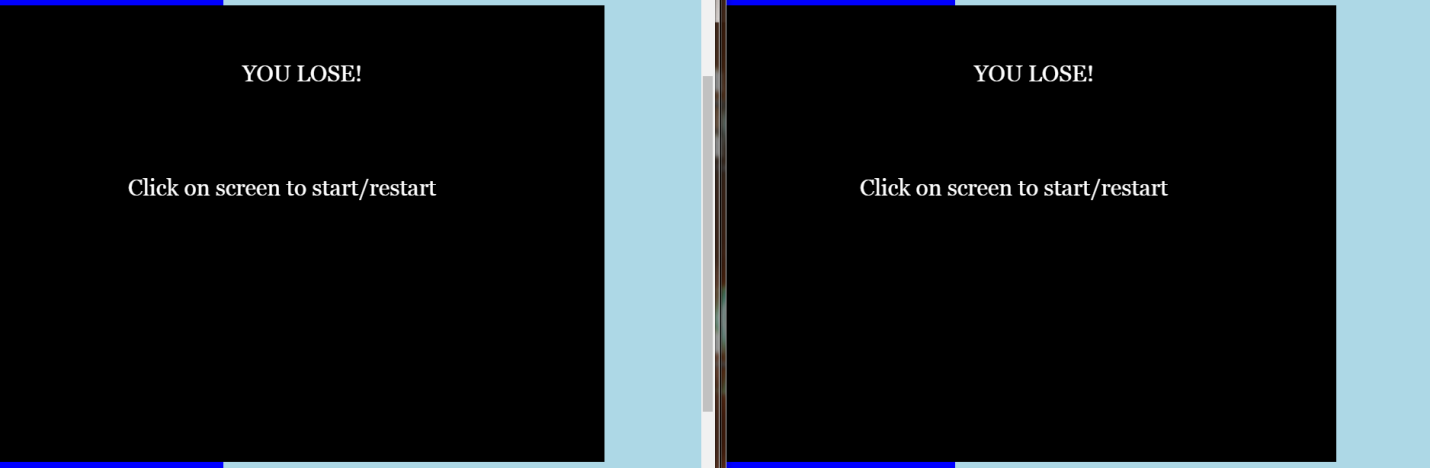
To start the game when both players are ready one player needs to click anywhere on the black game canvas. This will send a message to the other player that the game is starting and the game will begin. The game canvas will load this simple scene (seen below) with two paddles, a ball, a score chat and a boundary line. The balls direction will be randomly chosen at the start of the scene however whichever way it goes on one screen will be the opposite direction on the other. This is done since the game has been designed so that both players control the paddle on the left and their actions are displayed on the right paddle on the others players screen. If the ball was not mirrored the players would have to use the other player screen to see where the ball is. When the ball gets to their sides edge of the canvas it will check to see if the paddle is at the same location as the ball. If this is not the case the ball will restart in the middle, reverse its direction and give the other player a score that is displayed on both screens. However if the paddle is at the location of the ball it will reverse its direction, angle and speed depending on where it is on the paddle. This means that the closer to the side the ball hits the paddle the faster and at a sharper angle it will go on the way back to the other play.



Once someone reaches a score of three the game scene will change and display one of two messages either the victory or defeated scene based on who scored the third point.

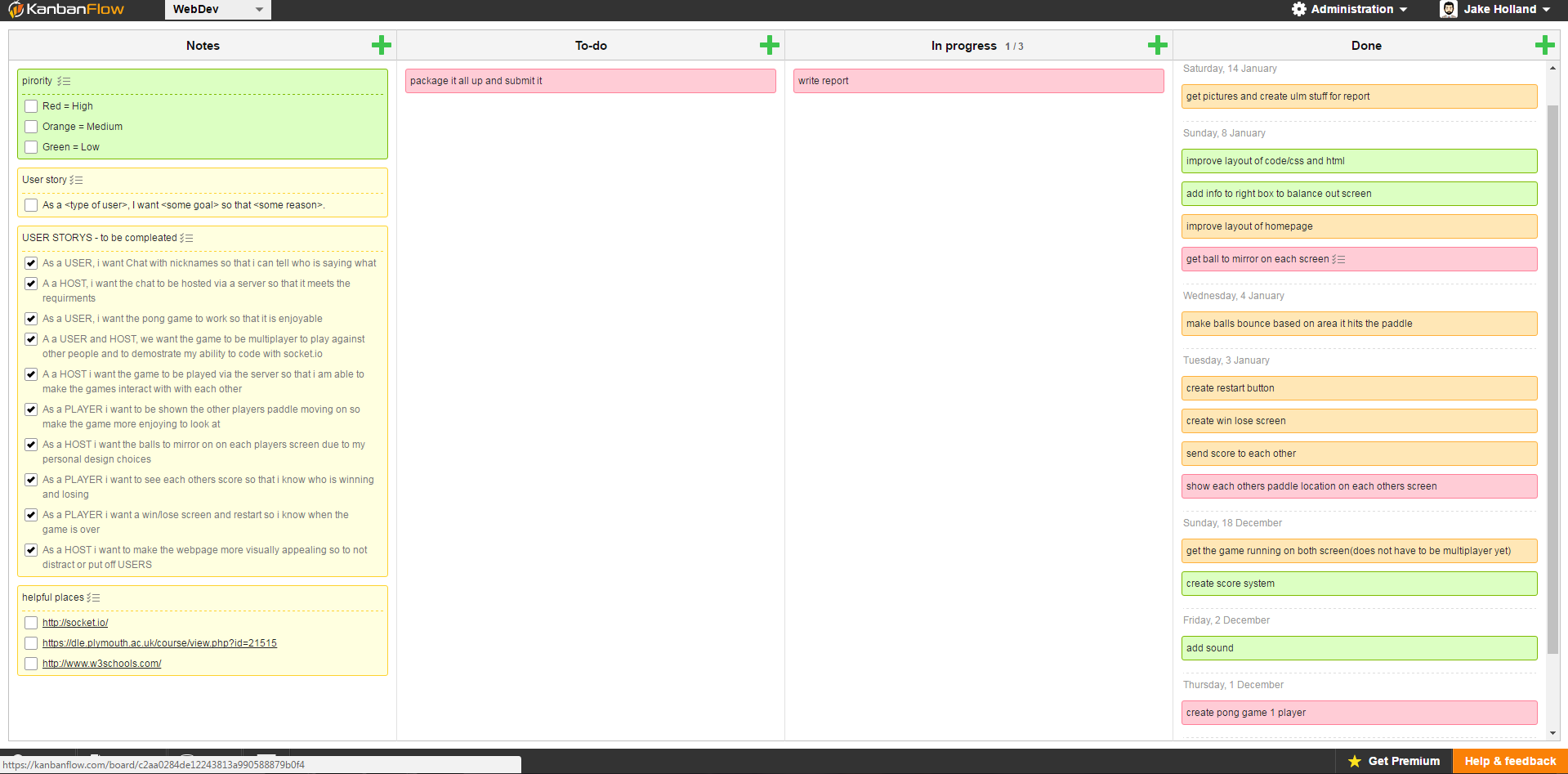
The game also has some sound effects with a ping sound being played whenever the ball hits a paddle, a victory sound when a user scores a goal/wins or a defeat sound when they miss the ball/lose.

Once the game is over users are able to restart by clicking on the game canvas. This will restart the score and ball.



**Requirements**

To help manage this project I decided to use agile development. To achieve this I used the website Kanbanflow where I page where I was able to keep track of my work. To better help myself manage things I created user stories and have priority to items in my to do list with a colour code.



**Notes:**

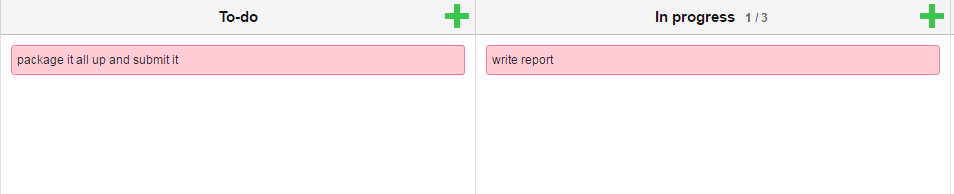
This section was used to remind myself of any important information. This is the area where I stored my user stories, helpful links, and keys for my project



**To-do & In Progress:**

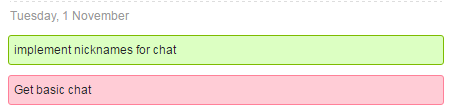
The to-do section is where I stored all the tasks I had to do. They have been colour coded red, orange and green for high, medium and low priority features.

The In-progress tab was used to store any features that were in current development. This was done to remind me what thing were half finished and helped to keep me focused and to stop myself from starting many tasks and completing none.



**Done:**

This section was used to record all the features I had completed and what date they were done. The advantage of this is that it helped showed how productive I had been for better or worse.



**Design**

This game was inspired by arcade classic pong. It has been designed for people who enjoyed the game and would be interested in playing an online version with friends.

A unique aspect of the game is that the paddle is controlled using the mouse on the game screen instead of the more traditional arrow or wasd controls. This design trait was intended so to allow users to be able to type in the messaging box at the bottom of the screen as the game is played so users can talk as they play. To make the game more interesting on a design aspect each player will see them self on the left hand side of the screen. The ball will then be mirrored and the others players paddle position will be translated to the right hand side paddle. This effect will give the illusion the other player is controlling the right hand paddle.

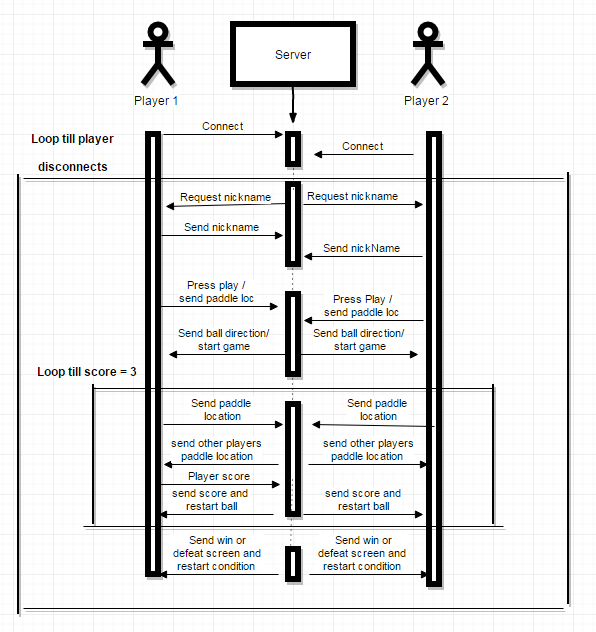
**ServerSide:**

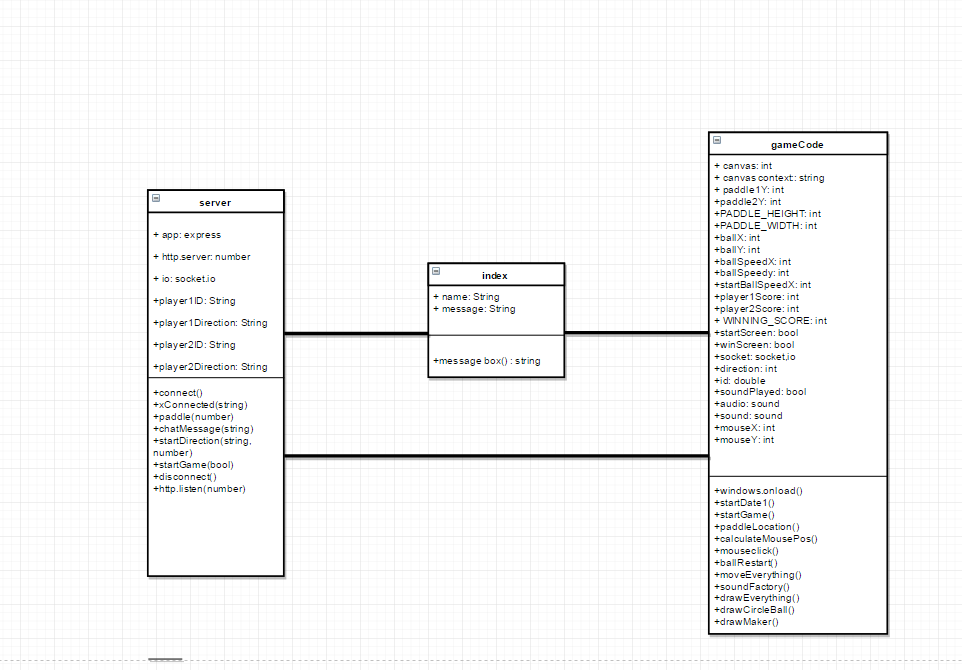
|  |  |
| --- | --- |
| **Events** | **Description** |
| Chat message | Receives a package from a user containing their nickname and message. This is then emited so that all users will receive it. It also displays the message on the command line |
| xConnect | Receives a message with the chosen nickname of a newly connected user and displays it on the command line |
| Start direction | Receives the random start direction and unique player id. It then stores it till it receives the other player's unique id and start direction. It then ensures that the start directions are not the same and packages all this information together and emits it's to both players |
| startGame | Receives a message saying a user as clicked the screen to start the game and emits this out to the other player |
| disconnect | Tells the server via command line that a user has disconnected |

ClientSide:

|  |  |
| --- | --- |
| Events | Description |
| drawMaker | A function is used to pass in variables to make object e.g. paddles. |
| drawCircleBall | A slightly different method than drawMaker used to create the move complex ball shape. |
| drawEverything | This method contains all the code for drawing objects onto the canvas such as text, size, font, colour, shapes, score and where to put it on the canvas |
| soundFactory | A function that contains a switch method for all the sounds for the game |
| moveEverything | Contains the methods used to move objects in the game and how they interact while moving e.g. hitting the side or paddle. |
| ballRestart | A function called when the ball needs to be sent back to the middle and to find out if the winning score has been meet. |
| mouseclick | An event called when the mouse clicks on the game canvas. It is also used to restart the whole game |
| calculateMousePos | Used to find the location of the mouse on the game canvas |
| paddle Location | Used to find the location of the other players paddle and to transfer that location to the right hand side paddle |
| startGame | A function called at the start of every game and when the game restarts and is used to restart everything |
| startData1 | Used to calculate the balls direction and to ensure it does not match the others players ball direction |
| window.onload | Contains main loop of the game and calls any functions that need to happen before anything else in the code |

Below are the UML diagrams for the project

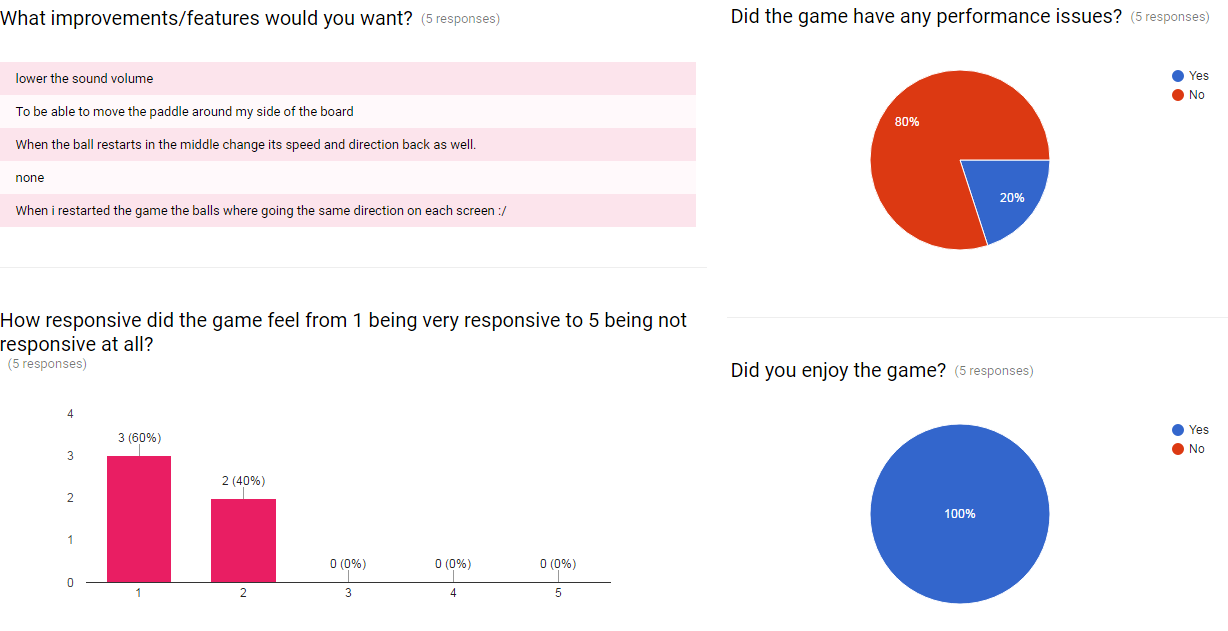


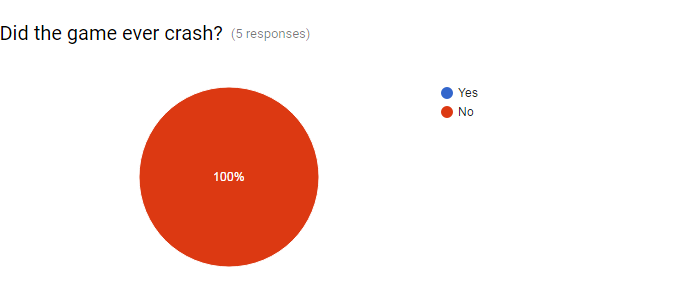


**Performance**

The performance of the game both quantitatively and qualitative have been positive.

User testing was used to gage how people apart from me found the game. To do this I had them play the games a couple of times and then fill in a Google form survey. The result of this can be seen below.

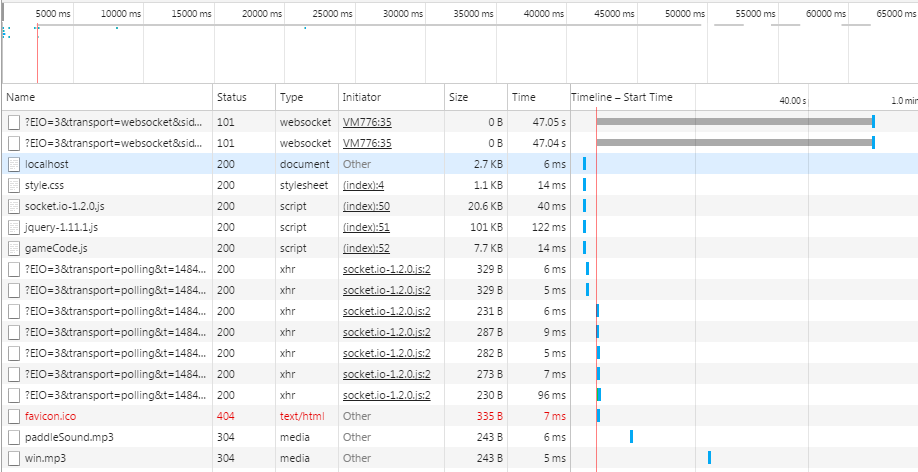




The result of the survey treads towards the positive. This can be seen by the fact that all users enjoyed the game even those who stated that the game needs to be improved in areas. What more the game was tested on a number of difference deceives with varying degrees of processing power. In all of these cases not once did the code ever crash or uses seem to experience any latency problems. The case that a performance issue did arise highlighted an interesting design fault that if a player refreshes the page after a game their ball will not always mirror that of the other players if the game is then restarted. In future projects I will have to ensure that I do more user testing since it helps find issues that even the creator who knows the code in and out could not find. In terms of fixing this issue for the mean time I will have to display a message advising users to not refresh the page if they want to replay with a player.

In terms of the chat box feature it has never crashed or stopped working at any point of the development once it was set up. This however may be due to my limited ability to test it. In the future I will have to find a way to brute force it and see how the serve is able to handle under more stress.

The files that are needed to be loaded for the game are very small and the information it exchanges between users and clients are very simple. Due to this Google chromes networking tab displays very healthy readings. This is due to the game being very memory efficient and does not waste space constantly reloading or sending things that are not needed. This can all be proven in the survey results that displays that no user has any frame rate or latency issues due to it being capped at 30.



**Personal Reflection**

Overall I am happy with my end product. At the start of this module I knew next to nothing about JavaScript and HTML so to end up with a game that people enjoyed is really fulfilling, however there are a few things I would change. One being that I would try and separate my game code into more classes such as one for the paddles, balls and score. This would have made my code cleaner and as the project got bigger it become more and more difficult to find the methods and functions I wanted. Another thing would be that I would have liked to have used a more test driven development such as using Qunits. This for example would have made it easier to find if any new code was interfering with old.

A big lesson I did learn from this model is that if I do not understand something I should not try and ignore it and find a work around. An example of this is that due to my lack of understand of how the server side worked I spent a large amount of time trying to make sure the balls would mirror on the client side without any assistance from the server. Once I finally started to look into using the server side I found an easy solution to the problem relatively quickly.