SOFT355 Referral - Chess Game

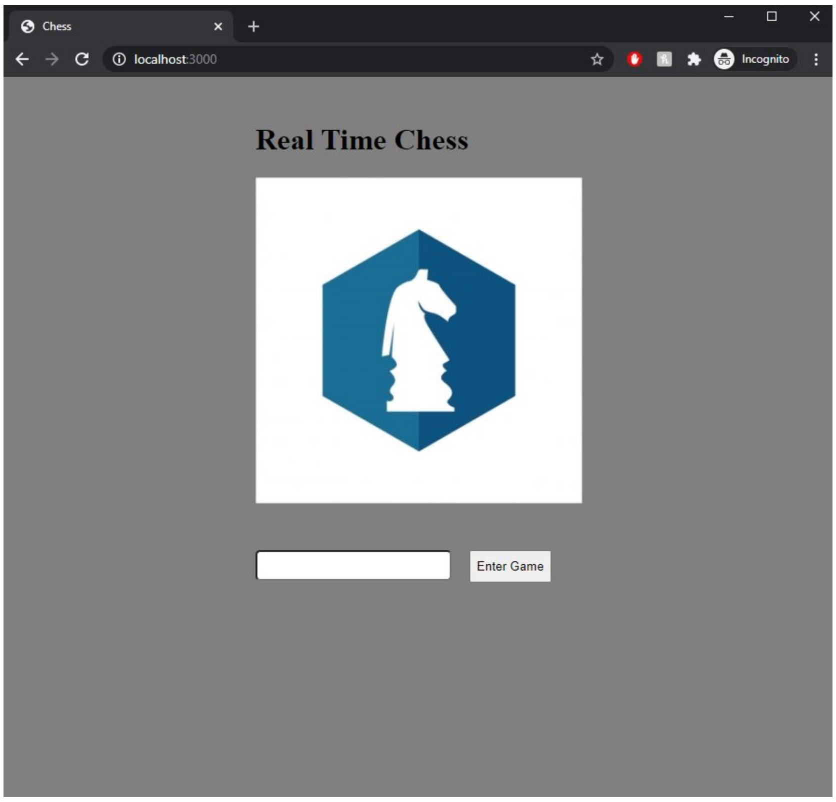
Ben Gearing - 10583504

Libraries used

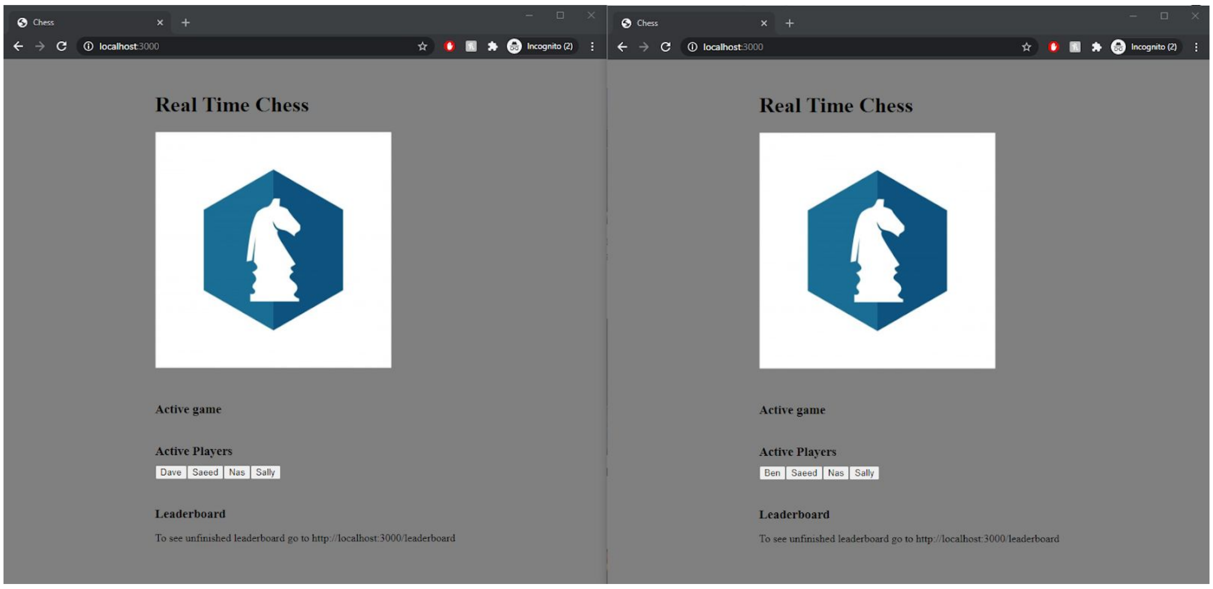
As a short introduction I will list the libraries that were used in the javascript since the number I used went up after research I did after the proposal:
chess.js - for chess logic, easy interaction with chessboard.js and gameboard saving using fen since coding it all out myself would have taken up a majority of the 2 weeks I had to do the referral.
chessboard.js - for creating a chessboard that easily interacts with chess.js and looks neat.
JQuery
win.js - for neater UI elements and design purposes

Functionality

When the user joins the website the connection is opened via websockets and the user is taken to the first login page.



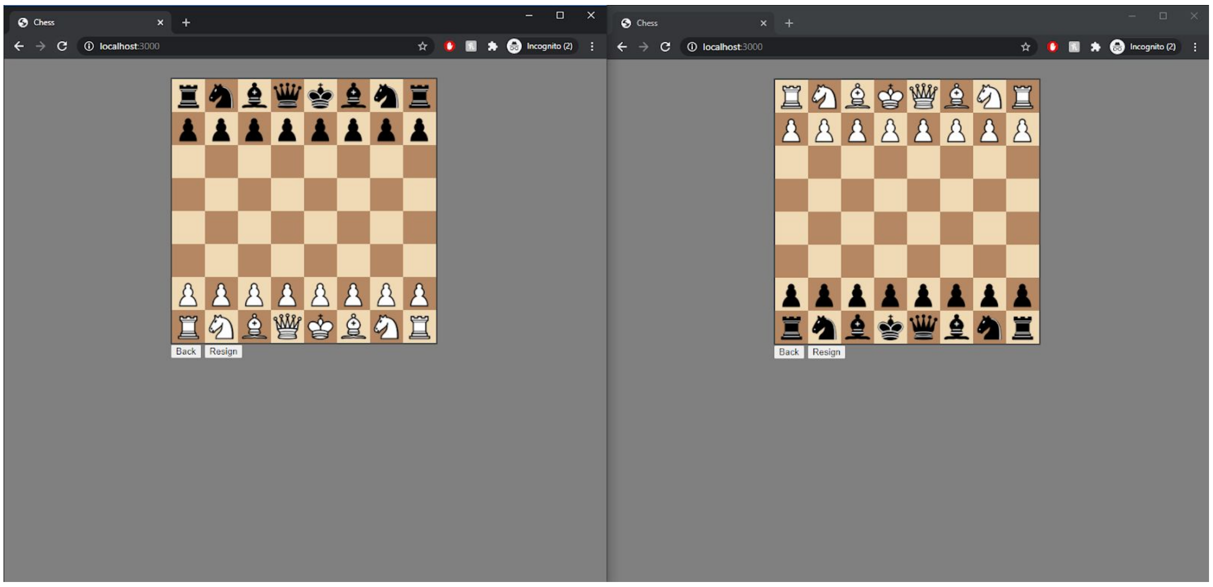
When a user first connects they are met with this page which contains a title, logo, text field and button. The user can enter a username to use for the game as long as it is above 0 characters and enter the lobby area. Additionally on the connection creates a user through the doLogin function where it takes the username entered into the field and creates a user object to be added to an array, this new object consists of a username and array of active games.



To see unfinished leaderboard go to http://localhost:3000/leaderboard

To see unfinished leaderboard go to http://localhost:3000/leaderboard

Once entered into the lobby the lobby page is shown with a list of active games for the user, active players and an unfinished leaderboard that I didn't have time to finish. From here the functionality the players experience is the ability to challenge other players to a game or return to their previously active games.
Once a challenge is issued the code uses the socket.on(‘invite’, ...etc) to create the game and remove the players from the lobby, then the players are taken to the gameboard where they can either play the game or resign if they didn’t want to accept the game in the first place.



When the game starts the program uses chess.js and chessboard.js to show each user the chessboard and init the game in its initial state. From here the players can each make moves in turns until one or the other wins. Thanks to the chess.js library when the game reaches a state it considers to be game over, which through the rules of chess are; check, checkmate, a draw via the 50 move rule or insufficient pieces are left, a stalemate where each move from each side gets stalemated, threefold repetition where the current board position happens more than three times using the fen state to check and if there is just insufficient material left for the game. Once the game is over the users can return to the lobby by clicking back or resign and challenge another player.

The communication for the moves within the chess game are also controlled by sockets sending and receiving information.

The main technologies at use in the project are; HTML, CSS, JavaScript, JQuery, Node.js, Express and Mocha and Chai are used for testing. An attempt to use MongoDB was made.

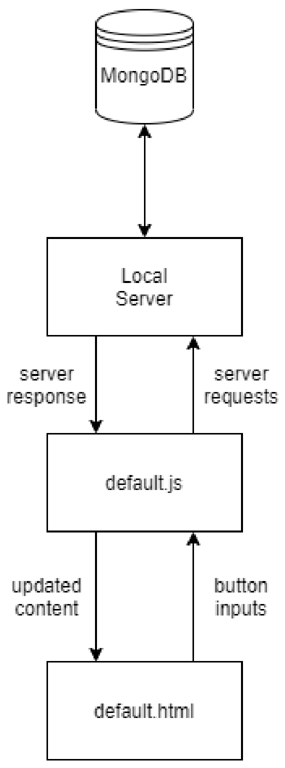
As far as functionality goes the game is extremely simple as this is one of my weaker skill sets and i didn't want to over complicate the project as i know i wouldn't have been able to produce a lot of extra features. This is also the reason I relied heavily on the chess.js library so I could focus on using the web sockets and actually understanding them.

Requirements

The application is ultimately aimed at anyone who wishes to play chess but the way the program turned out it is almost required to have a basic knowledge of chess before you use this application because i couldn't find a way to code in the instructions to show the user where each piece can move, they have to have that knowledge already. But chess rules are also easily accessed so anyone can play but it is more aimed at intermediate to high skill players.

The features included in the application are; specific user login so that other players can know who they can challenge and pick their opponents accordingly, a lobby page to show all active players, active games to allow some users to have multiple games going at once and the leaderboard which is unfinished at this stage and is unlikely to be finished by the deadline.

Design

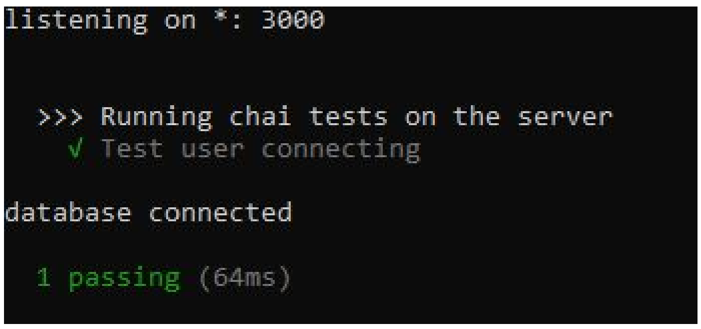


The client consists of the requisition information that the server provides in a visual format. In this project this consists of the lobby and its buttons and the gameboard and the position of the pieces. The client is not responsible for holding on to any information of importance, its only goal is to retrieve the information and display it for the user.
When a button is pressed on the web page a request is made to the server which then gives the appropriate response if needed. For example when a player challenges another player an invite is made which reaches the server and it takes the userID and opponentsID broadcast and emits them with 'leavelobby' which removes them from the lobby and after initiating a game places them both into that after assigning them a colour. After all this has been handled by the server the information is sent back out to both sockets.

When the game is actually being played the server is connected to multiple clients at once so when a move is made by a player in a game that client is sending information to the server and once the logic has been applied to the move and it has been deemed legal the information updating the game board is not only sent out to the original client but also their opponent.

The code is structured around each request from a socket and this is appropriate because as the project stands there is no need for constant updates and it is a turn based game so there is no need for users requests to all be handled at once.

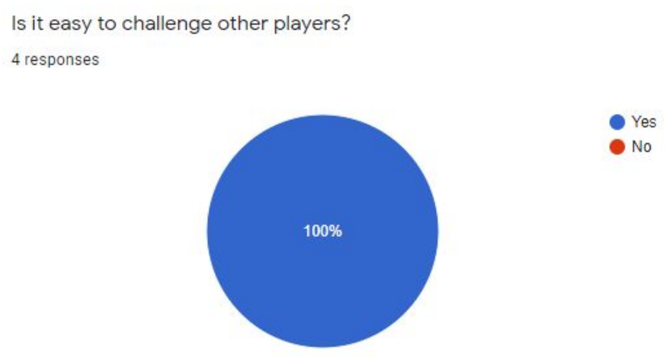
<u>Testing</u>
Because of the use of robust libraries such as chess.js and chessboard.js the only appropriate JUnit testing i could think of was making sure that the user connects to the server correctly. This was extended a bit to try and make sure the userID/username was a string but it kept throwing errors I couldn't fix. Current JUnit tests show 1 passing.



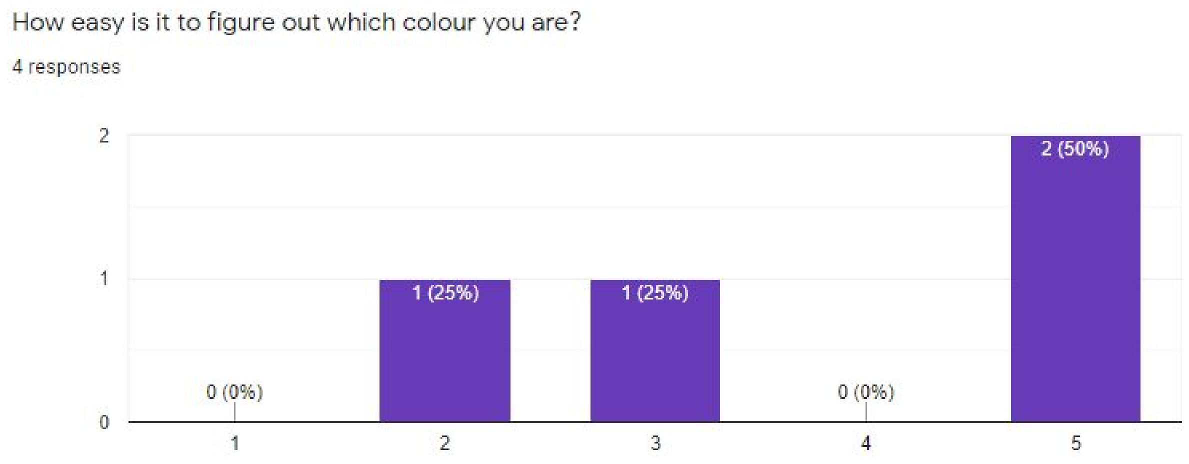
```
listening on *: 3000


  >>> Running chai tests on the server
    √ Test user connecting

database connected

  1 passing (64ms)
```

Since this is a game I decided to apply the type of usability testing that I would apply to any games project and ask users about the look and feel of the game and how easy or difficult they found it to use. Also due to the timing of this referral it was extremely difficult to find anyone who hadn’t seen this application before to test it which is why the test group consisted of 4 people rather than the usual ten to fifteen i would test the game on.

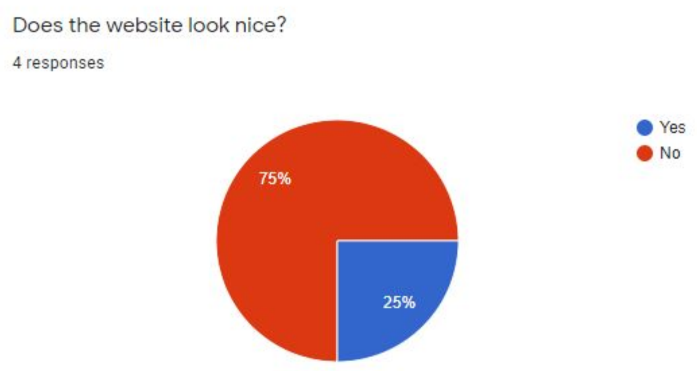
The first question I asked was about how easy players found it to challenge another user, the results were a unanimous yes on whether or not it is easy to challenge another player.



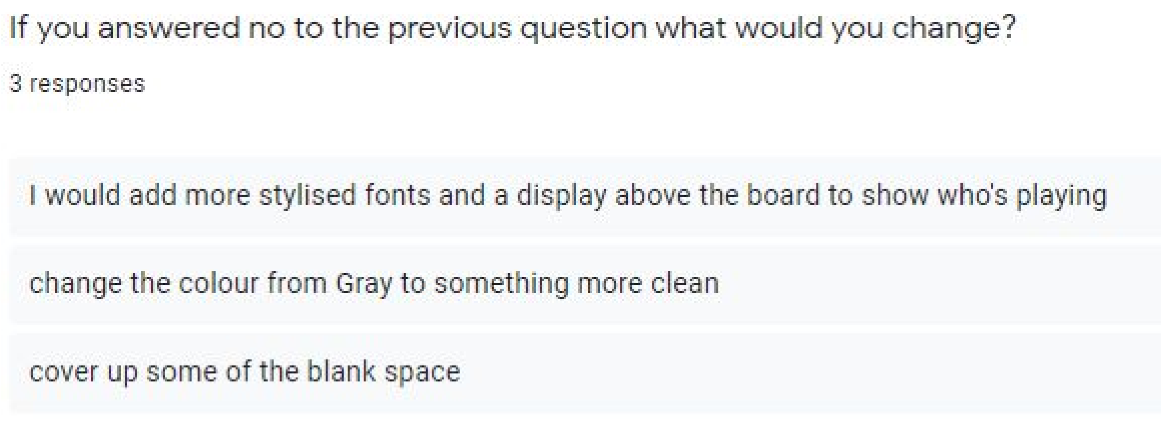
The second question I asked was how easy it was to figure out whether they were playing as the black or white pieces on the board. Some people found it difficult because of the lack of indication on the screen with the game but others found it easy based on what colour was at the bottom of the screen and they viewed the web page like a table where the bottom was the side closest to them.



The third question I asked was whether or not the website looks nice, unfortunately because of the large amounts of blank space and poor colour choice on my part a majority of users didn't like it.



The next question I asked was based around feedback on how to make the website look nicer, this was just a question incase i revisit this project in the future.


If you answered no to the previous question what would you change?

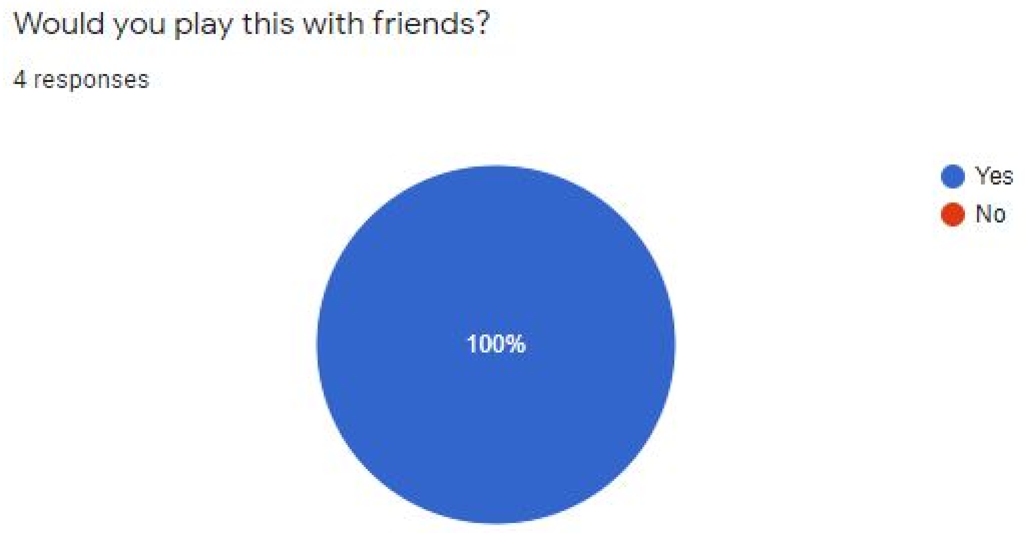
3 responses

I would add more stylised fonts and a display above the board to show who's playing

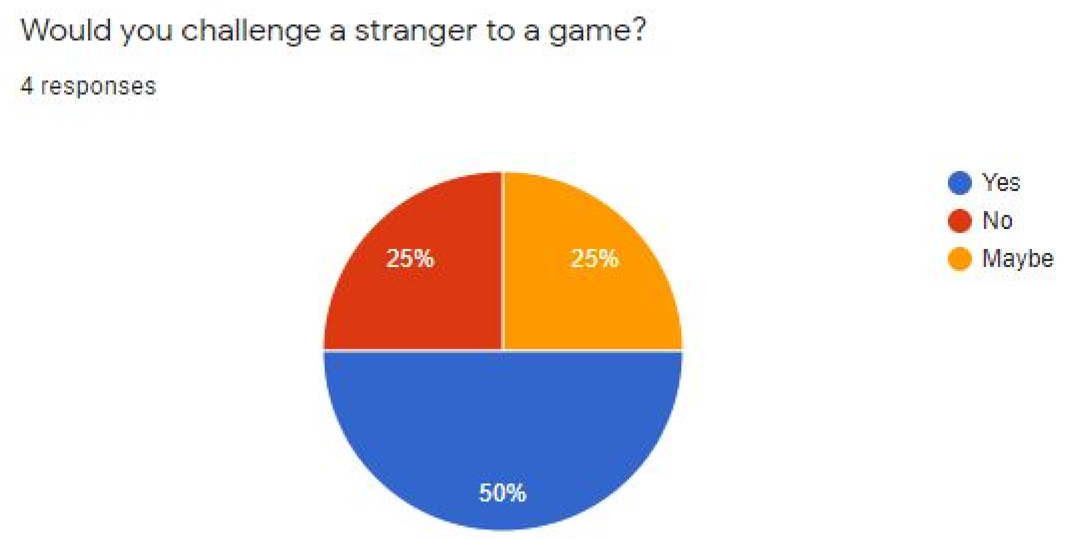
change the colour from Gray to something more clean

cover up some of the blank space

The fifth question I asked was whether or not they'd play this with their friends in case they don't have access to an actual chess board. The testers all said yes because they could communicate while they play.



The last question I asked was would they challenge a stranger and this got mixed results because some users just wanted to play with friends casually but some felt they'd have a more serious game against a random opponent.



I feel like the testing strategy was appropriate because the libraries used didn't require me testing them because they have been tried and tested before their release so all I had to test was the user experience and anything that had to do with the connecting to the server and websockets.

<u>DevOps pipeline</u>
https://github.com/RabidChinchilla/SOFT355-Referral---Chess-Game

The development environment I used involved Atom as an editor for coding, Google Chrome for leading the web pages, GitHub desktop for managing my repository and MongoDB for when I started attempting to use a database.

The CI pipeline for this project was not very well managed because this was a solo project and I was working exclusively from one computer unlike a team project. I have tried to keep my commits regular over the course of this project and development period. The commits have been based around each section being implemented so I can roll back the project if I were to break the project and were unable to pinpoint where things went wrong. Luckily when running the server through node.js most issues and errors were thrown there so i could see where exactly something went wrong.

Personal reflection

Overall I feel like the project completed what I set out to do in creating a multiplayer chess game but I am slightly disappointed in how confused I became over the course of this project. For a first attempt at distributed web development I'm not too disappointed but I relied heavily on forums and online resources to help me solve my issues that arose during the coding stage and a majority of my time was spent trying to teach myself the theory behind everything because I would regularly become lost when an issue arose or when approaching a technology I had never used before. The technologies worked extremely well for this project and the libraries I used calmed me down a lot during panicked moments when I felt like I couldn't code from scratch what I needed.

If i were to do this project again i would break down each file even further in an attempt to simplify the code so i can understand it even better. Also I would make a better attempt at trying to implement a database and leaderboard into the application. The skills I learnt from this project have definitely peaked my interest in web development and I would definitely like to do another passion project that revolves around these technologies.