Web Technologies Report

Julian Loscombe
Username: jl14910

Raul Mihoc
Username: rm14834

1 Introduction

For this coursework our pair made a website for the game of draughts(checkers). It offers both a single player version, in which two people play on the same computer by taking turns and a multiplayer version in which a websocket allows two people to play eachother over the network.

To run the server you can us the npm tasks provided. If you only want to play locally without websockets then you just need to run: npm run start.

To run the full game with multiplayer enabled then you should run the following commands: 1)npm run start; 2)npm run start:matchmaker 3)npm run start:router.

2 HTML (claim A)

Our submission contains 3 html pages written without use of any framework. These are served using XHTML delivery when possible. Additionally, we have an automated way of running the vnu.jar validation script on all of the HTML pages we have developped.

3 CSS (claim A)

All the styling for our website, apart from the actual board for the game, has been generated using CSS. Some of the issues we have investigated include: linking the Quicksand font from Google Fonts, use of various selectors such as tags, id's, classes, children, actions such as hover and tag and attribute value. Furthermore, we have also implemented a simple pulsing animation for the logo of our site using the @keyframes rule of CSS3.

4 Client-side JS (claim A)

The majority of the work for our submission has been concentrated on this part. We have used plain ES5 in our client side code which consists of a model of the game of checkers, a protocol of message passing using a websocket in order to allow for remote multiplayer games as well as a front-end for the actual game.

The game board has been drawn using canvas 2D and has been animated using window.requestAnimationFrame().

5 PNG (claim A)

We have generated PNG screenshots of our index.html page and modified these using GIMP in order to create a short tutorial for the use of our website.

Topics that have been investigated include layers, transparency, converting images and changing resolutions (since the original image was a print screen). In fact, all of the 3 images (localgame.png, logout.png and remotegame.png) have been obtained from a single .xcf file which contains 4 layers. The transparencies of these have been manipulated in order to obtain the desired highliting effect of the button.

6 SVG

7 Server (claim A)

We have used the initial server provided and adapted/enhanced it to suit our needs. Specific issues we have tackled include managing websockets to allow multiplayer games, https and SSL certificates.

8 Database (claim A)

Our website uses a database in order to store an "Elo" score for each player which gets updated after every multiplayer game they play. Each player starts at 1000 and his score grows or decreases based on performance in the last game. A leaderboard consisting of the best players is shown at the bottom of the index page.

An important point to make is that the name and ids of each player are provided using the Google SignIn API which has been integrated in our index page. This simplifies the issues with duplicated usernames or ids as we can now use the data from the google account unambiguously.

We have organised all database accesses into a separate server side module which we have developped in the test-driven development fashion.

9 Dynamic pages

10 Depth