Web-Tech Report

tr17544, fw17231

June 2020

1 Introduction

Our site provides a way for people to track and analyse their history of workouts in the gym, and view their progress. They can edit their account details and create workouts which they can view on the homepage. Admins have the ability to create/add/delete exercises which users can add to workouts.
To run the server, first run ***npm install*** to install the node packages, then run ***node server.js*** to start the server.
We have provided an Admin account which you can use to login and see all the features of the site:
- username: Admin
- password: Admin
Registering an account will only create a 'Basic' account, so will not have access to admin features.

2 Headings

2.1 HTML

For this section we will be claiming an: A.
We have written 6 different pages for the site. We have also made use of EJS partials for the Menu and Header bar. This meant that we only had the edit the html for these sections of the page in one file, and all pages would have the changes. Further, we have used xhtml delivery for each page. We have both developed a high confidence in designing html pages, and were able to quickly put them together.

2.2 CSS

For this section we will be claiming an: A.

We believe we have a high level of confidence working with css to style and edit our pages. We have multiple style sheets, and have written all the css ourselves, albeit following some tutorials to form a baseline. We have removed all style tags from the html files, so all styling is done in the stylesheets.

2.3 JS

For this section we will be claiming an: A+.

We have gained a lot of experience writing client-side scripts, and for each page, a separate script file can be found. We both have a good understanding of JQuery and Ajax, and are both able to write client side scripts using the two libraries very comfortably. We also made use of another JS library called croppie.js, which we used to allow users to crop images down for custom profile pictures. Croppie allows us to grab the base64 encoding of the cropped image, which we store in the database. It took some time to implement this as there wasn't much documentation I could find for the initial set up of croppie. We have used client-side JS and css to create some animations and effects for the site. For instance, the background of the login and register page, the sticky header, modals fading in/out, and the side menu. We have added some functionality that prevents numbers being inserted into text only fields, when the keypress happens rather than on submit. Additionally, the JS library Chart.js has been used to implement the graphs seen on the homepage easily. An AJAX request is used to collect the data from one of the database tables, and it is then displayed onto the graphs on page load.

2.4 PNG

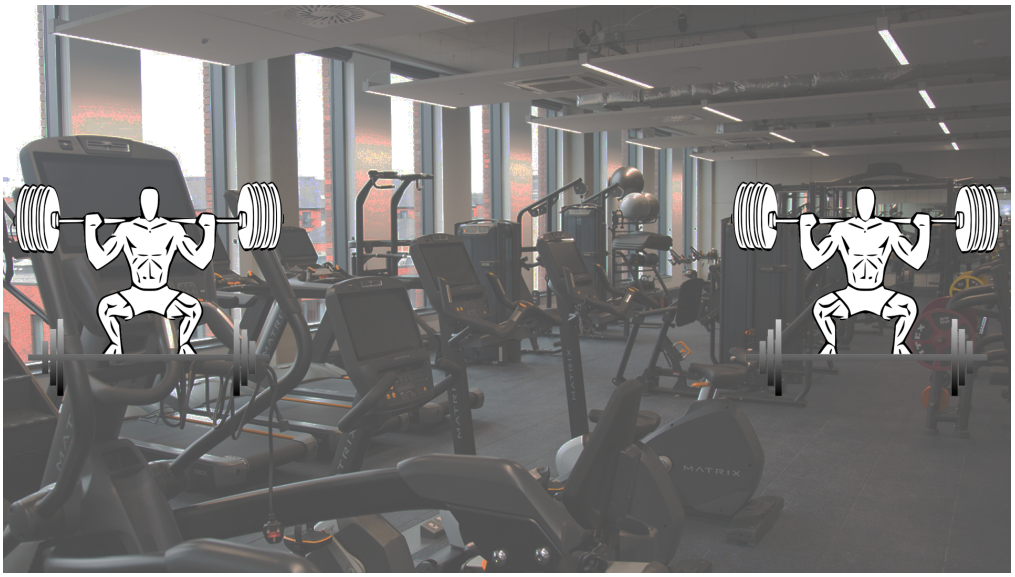
For this section we will be claiming an: A

We have created 4 background images for the login and register pages, which fade from one to another if the user does not login. These were created in Photoshop as we are more accustom to this than GIMP. Techniques such as filters, layering, colour change, transparency were all used to create the aesthetic we wanted. All images collected were cropped and scaled to

1920x1080 due to this being the most common monitor resolution in use, as well as the most common aspect ratio, meaning that it should scale up and down with little distortion.

For background 1, we altered the brightness and contrast to darken the image and make some areas more vibrant. Then a new layer was added, filled white, and given a low opacity to create a clouded look. Lastly, another layer was added so that the site logo could be positioned on the left and right hand sides of the image.

Original



Final Image

For background 2, the colour mode was set to black and white and a field blur filter was applied as to not draw much attention to the background image.

Original

Final Image

For background 3, we liked the image, however the athletes top was different to that of the site colour theme. Therefore, to tie in this background image better with out site, a layer mask was used on a duplicate of the image to single out the tank top. From here, the hue and saturation was altered in order to change it from a red top to a blue one. After this, the image was made more vibrant to allow the altered top to stand out more, and the colour curves were adjusted too to remove the washed over look the original image had.

Layer Masked Top

Original

Final Image

Background 4 again was converted to black and white and had an average-blur filter applied to it for similar reasons to background 2.

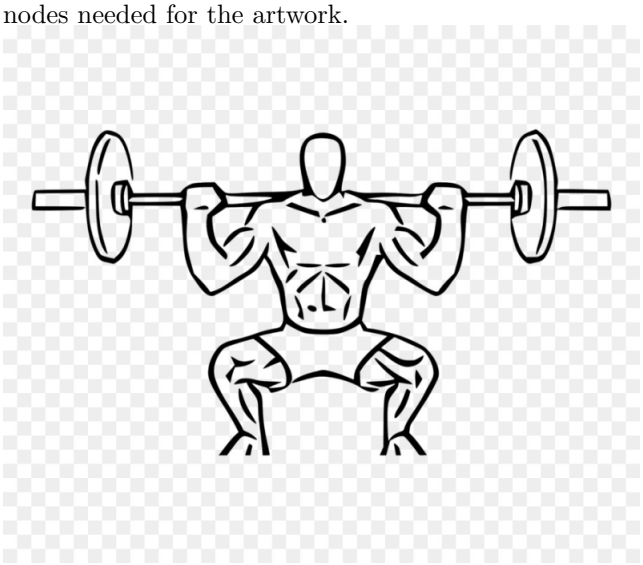
Original


Final Image

2.5 SVG

For this section we will be claiming an: A.

To create our vector art we used Inkscape as it gives easy creation of complex vector art. Our main logo (seen on the header once the user is logged in) was heavily inspired from a logo found online, which was traced over, and then added to and slightly altered to fit the final design we wanted. As we wanted the art to be filled, path-edit was used in order to ensure this was possible, and to also fix any unwanted overlapping between paths (on the barbell weights the athlete is carrying for example). The barbell that the athlete is standing on was created freehand using the bezier-curve/straight line tool to draw half of the barbell, then duplication and flipping it in order to achieve the desired symmetrical look. A gradient was applied to the barbell do enhance the look of the icon. Each of the main logo features was grouped together to allow for easy size changes and transformations of larger sections. This proved useful for the body especially due to the large amount of routes needed for all of the muscular detail. Once the logo had been completed, the simplify tool was used to reduce the number of nodes needed for the artwork.


Site Logo Inspiration

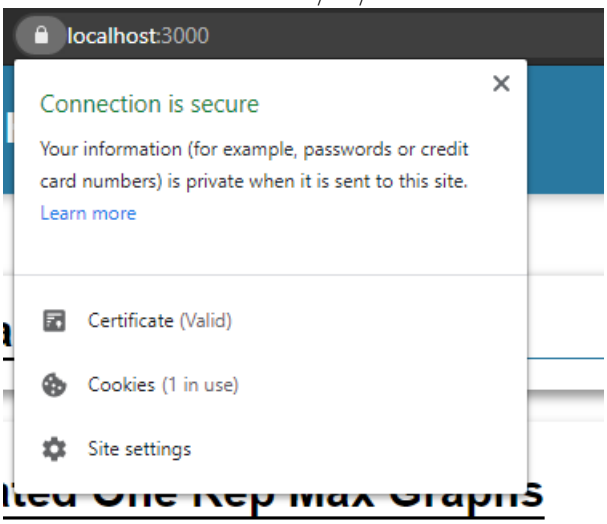
Vector art was also used for the menu icons. These were all drawn freehand using the straight line tool.

2.6 Server

For this section we will be claiming an: A+.

We have created a new server using express, rather than adapting from the one provided. The server is split up into several files in order to make it more manageable for development. It is split up into routers, with one router handling each page. The main server.js file handles the login and registration for the site, as well as bringing in all the other routers, and setting all the required parameters for libraries and settings.

We have provided two ports that access the server. Port 8080 access the http version of the server. Port 3000 will access the https version of the server. We have generated a self-signed certificate to demonstrate that https would work. In order for your browser to trust this certificate (and therefore trust the https connection) it must be installed. On Windows, installing it to your 'Trusted Route Certification Authorities' worked. I have also included a screenshot, to show that the https connection was secure on my own machine. Firefox was the only browser that still did not trust the certificate. The certificate can be found in site/ssl/localhost.cert



Our site makes use of passport authentication to handle logins. The setup for this can be seen in the file passport-config.js. Using middleware functions, we can easily check if a user is authenticated to access a page on the site. A similar process was employed for the admin access. A middleware function will check that the user role is 'admin' before allowing any access. Admins are the only people that can create, edit, and delete exercises in the database. Basic accounts will not be able to do this.

We have included some basic URL validation. If a url is not specified by the server then the user will be redirected to the homepage (or the login page if the user is not authenticated).

We have not stored user passwords as plain-text, but instead we implemented a hashing with random salt algorithm. The server recomputes a password guess on the server side, and then authenticates the users if it is a correct password guess.

2.7 Database

For this section we will be claiming an: A.

We wrote a script (database-config.js) that would build/rebuild the database whenever we made changes to it's structure. We organised are database accesses to be part of the different routers, so there is not one single database access module, but instead it is a module per page. We have used prepared statements whenever we access the database. We thought about the design of our database, having 4 separate tables, users, exercises, workouts, and user record data. We have ensured there are foreign keys relating the tables together were necessary. We have gained a solid understanding of callbacks, and have thus designed the database access to work so that handling the database results happens only when the database has finished the request. We have made sure that we use ps.finalise() whenever we have made a database access, to ensure the database does not get locked. We can update, insert and delete data from all tables, with some access only provided to the admin account, e.g. only admin can edit the exercises table.

2.8 Dynamic Pages

For this section we will be claiming an A+.

We used EJS which is a template engine. We rendered pages with data that would be insert into appropriate tags on the page, e.g. profile picture and page title in header. We could also control if HTML elements were rendered or not, depending on other data. This is how we controlled if the admin link appears in the menu or not. The workout planner is entirely dynamic, with the user being able to add and remove workouts (and underlying sets) at will. The page is empty on arrival when accessed through the menu, with only the workout name being pre-set to the current date as to not cause a database error when saving. The list of workouts the user can choose from is populated by requesting the data from the server via a database request and then inserting this data into the appropriate list on page. Within the workout planner, the workouts can be dragged and dropped onto each other so that the user can re-organise the order of their workout. When the workout is saved, the title will appear in the right hand section of the homepage, where the user can either delete the workout or view and edit it within the workout planner page.

The homepage also has some dynamic elements to it, with the graph data being requested from the server on page load for the specific user, as well as the workouts that they have saved. The admin section too is populated with data requested from the server. The majority of the site's functionality is dynamic, with basic templates being filled with data from the server. A lot of effort has been put in to ensure that the site is seamless and that the user experience is fully dynamic. We have a deep understanding of the frameworks and are both comfortable in implementing any feature.

3 Development Process

During development we used the node module Nodemon, which restarts the server every time it detects a change to the javascript, client side or server side. We made sure that we used browser development tools to test any client side functionality. Git was also used in order to have version control, and also allowed us to work remotely from each other.

4 Summary

In summary, we think the grades for each section should be:
- HTML: A
- CSS: A
- JS: A+
- PNG: A
- SVG: A
- Server: A+
- Database: A
- Dynamic Pages: A+