Requirements met

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | How I implemented the requirement | Files that show proof | Lines that show proof |
| A clear use of HTML5 | I have used HTML5 code such as the “DOCTYPE” declaration for HTML5. However, I have used HTML5 using the correct practice. For example, instead of leaving a tag open such as < p> I would close it. I have also included an “html lang” so that the website knows that I am writing in English. I have also used the correct “Semantic Elements” such as: main, nav, article, section and header. I have also got the names of the ID/ element inside of the CSS instead of the element name and the tag like HTML4.1 did. | Proof of html5 being used can be found on all pages 3 examples:  Login1.html:  index.php:  Contact.php: | Line 16 – 22<br>  Line 45 - 67<br>  Line 46 - 66  Line 10<br>  Line 11<br>  Line 25 – 30<br>  Line 33 - 49  Line 11<br>  Line 12 – 103 <br>  Line 33 - 49 |
| Use of the Bootstrap framework providing a responsive layout | I used bootstrap grid layout on each page, allowing for any page to resize and the content on it to change location depending on the size of the view port. This will mean that the information on the pages will be easier to read.  I also used bootstrap to create tables that allow them to change as well, depending on the information that is inside of them.  If a user is on a mobile device, the site will change as well as the navigation bar. The content will reorganise itself to make it easier to read and the navigation will shrink into a drop down, hamburger button to navigate through. | index.php | Including Bootstrap:  Line 17  Line 20  Bootstrap Nav bar:  Line 33 – 49  Bootstrap grid layout:  Line 58 -70  Line 72 – 76  Line 78 - 80 |
| Use of JavaScript to manipulate the DOM based on an event | I have created a button that will change the content displayed inside of a table, depending on what type of contact information the user desires. On the page load it will hide one of the types of contact information, and once the button is clicked the DOM will change as the information showed will be toggled (so one will be hidden, and the other will show). | Contact.php  script.js | Table being manipulated:  Line 65 -92  Button to change info displayed:  Line 64  Automatic hide part of table on load:  Line 55  Javascript that auto hides part of the DOM  Line 36 – 43  Javascript function that will toggle the table:  Line 52 - 66 |
| JavaScript loading of dynamically changing information | I used a live updating twitter feed of Macklemore’s tweets and put that on the front page. This way every time Macklemore posts a new tweet it will upload and be shown on my site.  I also used a countdown clock to show the next time that an album will be dropping. This was done by getting the current date/time and subtracting the target date in order to determine how long it will be. This is refreshed every second in order to keep it up to date. | index.php  script.js | Live twitter feed location:  Line 79  Countdown clock display location  Line 60  Countdown clock JavaScript:  Line 1 - 31 |
| Use of jQuery in conjunction with the DOM | I used this to automatically hide the song names when the page loads. I then created a button that allows the user to view the song names once pressed and then hide them again after it is pressed again. I did this so the information wasn’t shown on the screen at first and then the user can toggle the information as many times as they like, to hide and show the song name. | index.php  script.js | Button location:  Line 62  Song locations  Line 63 – 67  Auto hide song names on page load:  Line 34 - 36  Javascript for button click to toggle display:  Line 46 - 50 |
| Use of a jQuery plugin to enhance your application | I used a plugin called “JQuery password strength checker” (<https://www.formget.com/password-strength-checker-in-jquery/>) I implemented this inside of my registration page, allowing the page to look at the password that the user has entered into the password field and inform them if the password is secure or not and how secure it is. This is done by looking at the password’s length and what characters are used which will then give it a score out of 4. The score is then converted into a string that describes the strength of the password. | Register.html  passwordscheck.js | Display results of password strength:  Line 55 – 58  All of this is the plugin |
| Use of AJAX (pure JavaScript i.e. without the use of a library) | I created a button in html that will call an AJAX function once clicked that will change a section of text depending on which button is clicked. The AJAX function will display the text from a .txt file. The information shown will depend on which button has been clicked. | About.php  script.js | Where the text gets displayed:  Line 66 – 68  Buttons for displaying info:  Line 56 - 62  JavaScript for showing the information  Line 68 -121 |
| Use of the jQuery AJAX function | On the ‘About’ page there is a section about Macklemore’s songs. When any of these buttons are clicked, a jQuery AJAX function will be ran depending on what button is clicked and the relating song lyrics will show. This is done by having a different button with different functions relating to each. When the button is clicked it will run the function that will change a paragraph to the text of a text file which will be related to the name of the button.  I have also got a “param” jQuery AJAX function to display the profile of Ryan Lewis and Macklemore on a button click. | About.php  script.js | Where the text gets displayed for both:  Line 66 – 68  Buttons for displaying song info:  Line 73 -76  Buttons to display the profiles:  Line 79 & 80  Jquery for showing the song info:  Line 123 – 152  Jquery for showing the profile info:  Line 154 - 174 |
| Use of cookies | When a user logs in and goes onto the home page they will be asked for their favourite colour and their age - these fields are optional. When they enter these values, they will be stored as cookies on the user’s computer. These are then used to customise the site depending on which cookie it is.  The age cookie will allow us to recommend a song to check out on the ‘About’ page and the favourite colour will change the background colour of their account page to their favourite colour. The users can delete the cookies whenever they want to. | index.php  About.php  script.js | Check to see if they have cookies and if not ask:  Line 54  To delete the cookies  Line 69  Recommending song location  Line 72  Fav colour background:  Line 37  Get recommended song  Line 52  Getting, setting, checking and deleting cookies:  Line 208 – 267  Changing colour depending on input:  Line 269 - 301  Getting recommended song and recommending one:  Line 303 -321 |
| User login functionality (PHP/MySQL) | I have created a registration page that will ask the user for a username, password and some information about themselves. Their username will then be checked to see if it is available. If it is, it will store all the data in the database and it will hash and salt the password before doing so. They will then be taken to the login page where they can use their login credentials to login to their account. Their inputs will be checked to prevent against SQL injection, the username will then be found with the relevant password and the password will then be compared against the encrypted password. If it matches, they will be directed to the index.php page. However, if a user already has an account on the site, they will not be able to get onto any of the other pages as it will redirect them to the login page. | Register.html  connect.php  Login1.html  login1.php | Form asking for info:  Line 43 – 77  Connecting and sending information from form to database:  Line 1 – 20  Line 48 – 58  Check to see if username is in use:  Line 24 – 47  Hashing password:  Line 17  Redirect to login page  Line 58  Not able to get onto other pages  On all pages other than register and login (example:  About.php  Line 1 – 8)  Form for entering in login information:  Line 47 – 65  Preventing sql injection:  Line 8 – 14  Check to see if users credentials match ones on database:  Line 17 – 45  Decrypting password from database and checking if there is a match:  Line 30  Storing variables in the session to maintain the login status for the users account:  Line 36 - 44 |
| Admin section of the website (PHP/MySQL) | If the user has got the Admin account credentials, they can login as Admin. The Admin can go to My Account > Update Account > Admin Access, where the admin can search to find a user’s account information and view it. They can then delete any user from the database, other than the Admin account. This is done by the Admin entering the username for the user they want to delete, and then clicking the delete button which will delete the whole row in the database that has the username equal to the one the Admin inputted. | AdminAccess.php  deleteUser.php | Form for finding user with submit button:  Line 57 – 62  Php for finding inputted username:  Line 63 – 104  Form that display the results of the found user:  Line 108 – 161  Php that gets user info:  Line 115 – 119  Line 127 - 131  Line 139-143  Line 151 - 155  Form for deleting user:  Line 166 – 170  Getting user inputting and selecting then deleting user:  Line 1 – 30  Preventing admin from deleting admin account:  Line 1 - 5 |
| Ability to select, add, edit and delete information from a database  (PHP/MySQL) | The user can go onto the ‘MyAccount’ page where they can view/ select their account information. This is done by retrieving the session user name and finding it on the database. It then outputs each column on the database into the correct field on the form page.  They can also go onto the ‘Update Account’ page where they are able to see their information and edit/ add new information.  The admin user is able to delete information from the database (as previously spoken about in the admin section).  Info can be added to the database when a user register. This is done by getting the input of the user and then making a connection to the database to then create a new row in the database with the new user’s credentials. | MyAccount.php  UpdateAccount.php  updateInfo.php  connect.php  deleteUser.php | selecting users account information:  Line 15 – 32  Displaying selected information:  Line 90 – 125  Php for selecting current account details:  Line 16 – 32  Showing gathered info:  Line 89 – 149  Input for new details:  Line 110  Line 123  Line 136  Prevent against SQL injection:  Line 10 – 23  Check to see if info needs updating and then edit current details to make them the new details:  Line 26 – 65  Redirect user depending on what they do:  Line 66 – 71  Connect and add information from form to database:  Line 1 – 20  Line 48 – 58  Getting user inputting and selecting then deleting user:  Line 1 – 30 |
| Appropriate consideration of relevant laws | When the user logins in and before any cookies are created, an alert popup will appear saying: “this website uses cookies to personalize content, you consent to our cookies if you continue to use this website, however to opt out go to the home page button and ‘Delete cookies’”.  -This complies with the GDPR cookies guidelines. This feature is called "*prior consent*" and will make sure that the site has consent to use cookies before storing them.  -The information about the cookies are "*accurate and specific*" as well as being "*presented in a clear plain language"* which are both requirements of the GDPR.  -There is only one category that this cookie will use – ‘personalize data’. This means an ‘okay’ button is enough, however, if there were more categories, there would have to be checkboxes for each category, so that the user can "*choose how the cookies are being used".*  -The user is always able to "*change their mind"* if they want to as there is a button on the home page that will delete any cookies stored. | script.php  index.php | Popup alert to meet GDPR guidelines:  Line 232 – 234  Running function to display message:  Line 54 |
|  |  |  |  |
| Security measures taken:  SQL queries should be written as prepared statements | Writing the SQLI queries as a prepared statement will prevent against SQLI injection. Therefore, I made all my sqli\_querys into statements. The pages I did this on were:  login1.php, MyAccount.php, updateInfo.php, UpdateAccount.php, AdminAccess.php, deleteUser.php,  connect.php | login1.php  MyAccount.php  updateInfo.php  UpdateAccount.php  AdminAccess.php  deleteUser.php  connect.php | Line 17 – 29  Line 16 – 32  Line 32 – 40  Line 44 – 52  Line 56 - 64  Line 16 – 32  Line 70 – 84  Line 12 – 29  Line 24 - 36 |
| Security measures taken:  Passwords should be salted and hashed | When a user enters a password in the registration page, that password is stored as a variable and then put through a function called “password\_hash”. This encrypts the password by hashing and salting it. It will change a regular word/password into a string with over 60 different letters, numbers and symbols which can only be translated/ decrypted by a computer with the decrypting key. | connect.php  login1.php | Salting and hashing password:  Line 16 – 17  Decrypting password:  Line 30 - 35 |
| Security measures taken:  Validation of user input | When a user is entering information into the register form, it will check each field if they have matched the requirements for the entry of that field (each input field has different requirements that must be met).  I have also made sure that when a user inputs data into the SQL database it is run through the functions stripcslashes and htmlspecialchars. This means that any attempt of SQL injection will be cancelled as any special characters such as < or ! are simply just characters of text and not potential injection code. | Register.html  script.js  login1.php | Onsubmit it will return true or false:  Line 44  Validate Function that will check all the fields:  Line 176 – 206  Prevent sql injection:  Line 8 - 14 |
| Security measures taken:  Any other relevant security features | When a user is inputting a password, it’s strength will be measured and given a score out of 4 which is marked as: short, weak, good and strong. This will mean that the user will make sure that their password is strong or at least good allowing for better passwords to be used. | Register.html  script.js  passwordcheck.js | Where the input for password is and where the strength rating is displayed:  Line 51 – 58  Validate function being ran:  Line 44  Not allowing user to continue if password is ‘to short’:  Line 190 – 193  Checks the input and gives it a score and a strength rating:  Line 10 – 43 |
| Security measures taken:  Any other relevant security features | I have also made sure that when a user registers their username, it will check to ensure it has not already been taken, otherwise this could cause problems. For example, if two users had the same usernames and one of them changed their name from “Jack” to “Paul” it could change the wrong account. | connect.php | Check to see if username avalable:  Line 24 – 47 |
| Security measures taken:  Any other relevant security features | If a user tries to browse the website without logging in, they will be redirected to the login page which will require them to login. This means that if something happens to the website, such as an SQL injection, it will be easier to find the user/ account that did it. | index.php  Contact.php | If user isn’t logged in they can’t access the page:  Line 1 – 9  If user isn’t logged in they can’t access the page:  Line 1 - 9 |