**Web Application Development Project Submission 2022**

Name: Enda Lynch

Student ID: G00398795

Date: 18th May 2022

Home page/index page/start page (eg., page user should open first): Login.html

**Project Requirements Implementation**

**NB – Please run python local host on folder and open login.html via the host, otherwise plot will not be generated for merch.html when user navigates to that page**

The HTML Files and files they use.

|  |  |  |
| --- | --- | --- |
| **HTML** | **CSS** | **JavaScript** |
| ***login.html*** | **logincss.css** | **login.js** |
| ***home\_and\_signup.html*** | **stylesForSecAtt.css** and **modal.css** | **app.js** |
| ***bar.html*** | **bar.css** | **d3.js** |
| ***merch.html*** | **merch.css** | ***merch.html*** |

|  |  |
| --- | --- |
| **ITEM 1** | **Reference** |
| *Allow the customer to enter their login details:* | Open Login Page  ***Login.html*** lines 14 and 15 create the text boxes  ***Login.js*** has the function that validates the login data |
| *Login details validated (via a login screen) before receiving a summary of the order:* | If login Successful pop up tells you and brings you to home page.  If login unsuccessful pop up tells you and stops you progressing any further |
| *Username:* | gmit |
| *Password:* | gmit |
| *Brief description of how this was implemented:* | Used the course material and youtube videos to help with the structure and then added my own twist to the style.  ***Login.html*** has the html code and structure – creates the input fields and buttons.  ***Logincss.css*** is the stylesheet used to define the attributes and style to the structure of the page.  ***Login.js*** has the javascript DOM query to extract the values and the function that is called to validate that the login details that are input via login.html are correct. |

|  |  |
| --- | --- |
| **ITEM 2** | **Reference** |
| *Perform form validation through JavaScript or HTML to ensure that text fields are not empty, and a valid email address is entered* | On Homepage click on Sign Up button mid-way down the page.  Click with all fields empty to show text validation errors. |
| *Brief description of how this was implemented:* | In ***home\_and\_signup.html*** there is a div class ”modal”. This is activated by clicking the sign up button. It generates a pop-up registration. 4 fields, Name, email, password, password-confirm.  ***modal.css*** styles the popup and positions the images, text-fields and submit button.  ***app.js*** then has an event listener that waits for the click on the “submit” input tag in the form tag and uses the checkRequired, checkLength and passwordMatch functions to validate the inputs and return errors if they fail validation. Name has to be more than 3 letters – email has to have @ and passwords have to match – errors will tell user if any info is incorrect via showError function. Fields highlight green if submit is valid. Data does not go anywhere as this is front end development only. |

|  |  |
| --- | --- |
| **ITEM 3** | **Reference** |
| Access and change HTML on the web page through the DOM; | This is done multiple times in the project – most notably in the data validation on the registration pop-up on ***home\_and\_signup.html*** mentioned in Item 2 above.  Also used to gather the values input on the login page |
| *Brief description of how this was implemented:* | For register section in ***home\_and\_signup.html*** it accesses the values that are input via the div class form-validation in **home\_and\_signup.html** and then uses the DOM to gather the values and checks those values against the parameters set up in the functions created on ***app.js*** to ensure they are valid inputs.  Also used is querySelector() – this returns the first Element within the document that matches the specified selector, or group of selectors.  This is used to input the error message to the p tag in the div class="form-validation" in ***home\_and\_signup.html*** |

|  |  |
| --- | --- |
| **ITEM 4** | **Reference** |
| Access and change styling through the DOM; | This is also done multiple times in the project – in **bar.html** the DOM accesses data using ***d3.js.*** |
| *Brief description of how this was implemented:* | This is done to draw and change the color and size of the bar chart generated by the array in ***d3.js.***  It uses getElementById("chartHeight") to allow the user to pick between 3 sizes of charts.  It also uses getElementById("colorPicker") to allow the user to access the svg attributes and pick the color of the bars via the ***bar.html***. |

|  |  |
| --- | --- |
| **ITEM 5** | **Reference** |
| Demonstrate the use of events; | Onclick seems to be the most common event and it is used in ***login.html*** and ***home\_and\_signup.html*.** |
| *Brief description of how this was implemented:* | This was implemented on the sponsors section of the ***home\_and\_signup.html.*** It is used to navigate to the sponsers pages (nike.com and nfl.com) via the buttons under the pics of Serena Williams and Tom Brady.  Onclick is also used in the ***login.html*** page to call the validate function from ***login.js*** that then verifies the username and password that was input via html for a successful logon. |

|  |  |
| --- | --- |
| **ITEM 6** | **Reference** |
| Contain two D3 data visualisations (e.g., Bar Chart) of your choosing  a. One from a CSV file  b. One from an array | ***Bar.html*** in conjunction with ***d3.js*** generates a hardcoded array based bar chart from the values provided in the array.  ***Merch.html*** this uses the file merch.csv by reading in the data and generating the label – this must be done via the python localhost server for the graph to be generated. |
| *Brief description of how this was implemented:* | Created a hardcoded array in ***d3.js*** and then created a svg const from the div id d3-container – then put it all into a function that could be called and the chart generated once the user had picked a size and color. |

|  |  |
| --- | --- |
| **ITEM 7** | **Reference** |
| Both visualisations should allow the user to specify display settings, including an option to change colour, display size and animations | ***Bar.html*** allows the user to pick the size (3 options) and color (using a color bar) before drawing the plot – if the user wants to change size or color or both the previous graph will be cleared.  ***Merch.html*** does not allow for user manipulation – but it does show off some transition features. |
| *Brief description of how this was implemented:* | For ***bar.html*** I created a hardcoded array in ***d3.js*** and then created a svg const from the select tag id chartHeight – this container is then accessed and allow the user to pick from a dropdown different sizes (Small = 400px Medium = 600px Large = 1000px) and will generate the plot the appropriate size by using the DOM to gather the users choices.  For Merch.html – as mentioned above it does not allow for user interaction as was specified, however if the data in the csv file is updated the graph will automatically update – also the bars are color-coded depending on values and the way it is generated shows off some nice .transition() features, including updating the labels. |

|  |  |
| --- | --- |
| **ITEM 8** | **Reference** |
| Have a minimum of 3 linked pages; | ***login.html, home\_and\_signup.html, bar.html, merch.html*** |
| *Brief description of how this was implemented:* | These links were made using the href attributes on the relavent tags or the window.open() feature using javascript.  ***login.html*** 🡪 uses logincss.css for stylesheet 🡪 uses login.js for javascript (Should be first page opened and if successful line 12 in login.js will bring you to):  ***home\_and\_signup.html***🡪  uses stylesheets stylesForSecAtt.css for main page and modal.css for register pop up window 🡪 uses app.js for javascript from here you can click the results button(see li tag with class “navbar\_\_btn” for href) on the navigation bar that will open ***bar.html*** 🡪 uses bar.css stylesheet 🡪 uses d3.js to hold the hardcoded array data and generate the chart, it uses inputs from the user on the html page to generate the plot with specifications 🡪 from here you can navigate back to the home page or click the Merch Sales link in the navigation bar that will bring you to 🡪  ***merch.html*** 🡪 uses merch.css style sheet🡪javascript here is done on the merch.html page |

***Additional information:*** At the beginning of this project I had some very lofty ideas of what my website may be and look like – however I was soon humbled as to the extent of work and complexity that can go into a single webpage. I found that I am pretty good at following instructions and delivering if I know what I want to do, but I do not really have the creative flair as of yet to have the visualization, that comes with experience, of a front end developer. However between the lectures, lecture notes, YouTube videos and tutorial websites such as W3Schools I was able to put together these webpages.

The initial idea I had was to create a website to where the user was able to login, vote on their favorite sports star, see where their favorite star ranks among other stars, have a “shop” where the user could order and buy their favorite stars gear and see where their star ranks via merchandising sales. Not a very useful website, however as you can probably tell I am a big sports fan so I wanted to try to keep the project relatable to my own interests. This initial visualization did not really go as planned as I found I was getting lost in the complexity of the html file and trying to create a suitable dynamic interface with a polling feature that updated plots and graphs with input was, for now, beyond my skill level and I was doing too much research on topics that were not in the scope of this project. Because of this I scaled back the complexity and decided to simplify my site to a simple ***login page*** where the users username and password is validated. This would open the ***homepage*** where there is a navigation bar at the top of the page that would link to the graphs/plots that will be discussed later. On the main body of the homepage I have an image displayed with a button that will allow you to “register” your email on the web site – This then check to ensure there is text in each field and will force you to enter a valid email – and make sure your passwords are the correct length. One issue I ran into here that I could not fix was that I wanted to count how many valid entries there were, and if there was 4 then an alert would say registration successful. This works unless you have a valid name and email, and if you have a valid password and no text in the *confirm password* field. However if you put any text in the *confirm password* field the alert is triggered even though an error is displayed – I left this in as I wanted to show what I was trying to do – see app.js function checkRequired(inputArr) for code. If the registration is successful the pop-up window will disappear after the alert is acknowledged.

If you click on the “Results” button on the top right of the page that will navigate you to another page and allow you to see where the top athletes votes have been counted(bar.html). On this page you can change the size and color of the bar chart. The navigation bar is at the top of the page and from here you can navigate back home, or if you have the localhost running from the folder where the files are stored you can navigate to the Merch Sales that displays what athletes from the list have sold the most merchandise since 2020. Unfortunately I ran out of time here and did not get the opportunity to add the buttons to change the size and color, however when you log on you can see that I used some nice features where the plot drops down from the top of the page and the labels transition to show the money brought in via merchandise sales for each athlete.

One of the reasons I did not meet the specifications required was I spent far too much time on the homepage creating the sponsors section and the footer section underneath. The sponsors section links to big sports brands nike.com and nfl.com and if you have a look you will see that transitions and events are used. Also in the footers section I spent more time than I would like to admit trying to create the table effect for the *About Us, Contact Us* section, and more time again trying to generate the social media icons under the fake copyright. But I do think the page looks far better with them – even though they only lead to the social media homepages.

In all it was a very enjoyable and fulfilling project and I will look forward to playing with the features and trying to correct some of the functionality that I could not master.