University of the West Indies – Cave Hill Campus, Barbados
COMP 2245 – Web Development Concepts, Tools and Practice
Semester I: September 2024
Class Assignment
Due Date & Time: September 29, 2024, 11:59pm

IMPORTANT SUBMISSION INSTRUCTIONS

- **Submit your assignment early and submit any improvements you have made often.** No excuses for the server being down or your computer crashing or the Martians attacked... will be accepted.
- **Your assignment must be submitted to the eLearning system using the *Submit Assignment 1* link in one zip file called assignment1_xx.zip** where *xx* stands for the version number of your assignment. Use the version number to help **you** keep track of **your** submissions.
- If you are submitting multiple versions of your assignment, only the one with the **most recent date** will be marked regardless of version number.
- **DO NOT EMAIL YOUR ASSIGNMENT** to either the instructor or the teaching assistant. **It will not be considered as submitted** if you do, you will receive zero for the assignment if it is not in the E-Learning system by the deadline.

Plagiarism

- Simply cutting and pasting code found on the web or generating code using generative AI, attributing it to the author or system and then submitting it as a completed assignment will not benefit you. It must be your idea. **Do yourself a favour... do your own work!**
- Students are reminded that plagiarism is a serious offence.

Late Submissions

- You are allowed late submissions for up to four days after the due date. Late submissions will incur penalties as shown below. Penalties are cumulative, so for example, an assignment that is two days late will receive an 6% penalty.

Days Late	% Penalty
1	2
2	4
3	6
4	8

Assignment Details

You are to use HTML5, CSS, JavaScript to create a Web application for a gym management system. This system will allow administrators to enrol new gym members and sign on personal trainers. It will allow personal trainers to see the training session schedule and book personal training sessions with members. Members will be able to see their accounts and any personal training sessions they have. Everyone will be able to view the activities scheduled in the gym using the ubiquitous Gym Schedule link that is part of the hamburger menu that is found on the top left corner of most of the pages. The functional requirements that must be met for this assignment are described below.

You are to only use DOM Level 0 ***event handling***. **You will be penalized** if you use DOM Level 2 event listeners.

1. Figure 1 shows the starting console page of the management system. The HTML file for this page is named **index.html**. It is the first page that everyone who is not logged in, or has been logged out, sees of the application. **(3 marks)**

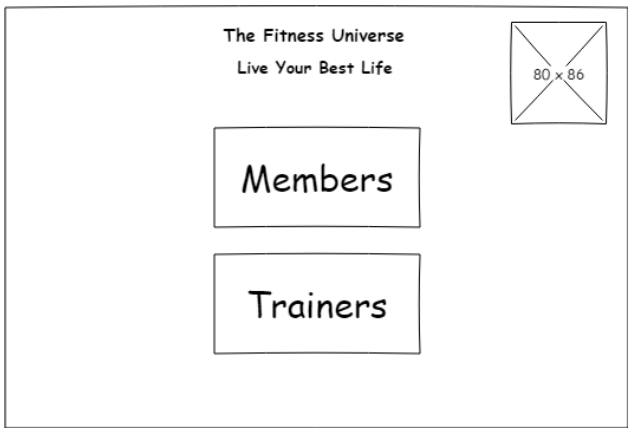

Figure 1: The main page for the application

   a. The image in the top right-hand corner is the company's logo. Select or create an image that is 80x86 pixels in size to make the logo. If you have selected an image from the web, ensure that it is at least under the Creative Commons licence. **(3 marks)**
   b. The two large buttons are clickable. The Members button goes to Members log in page and the Trainers button goes to the Trainers login page. **(4 marks)**

[Q1 Total: 10 marks]

2. The Members Login page is shown in Figure 2. The HTML file for this page must be named **member_login.html**. **(2 marks)**
   a. The hamburger menu in the top left corner of the page, when clicked, displays a drop-down menu that has the non-clickable menu item **View Schedule** as its only menu item. **(2 marks)**
   b. The page requires two pieces of information: the gym member's id number and their password. The Forgot Password link remains on the page when clicked. The data format for the membership id and password must be validated by separate JavaScript functions, which are invoked when the Log In button is clicked. These

functions can use JavaScript string manipulation and other functions, ***but no regular expressions***.

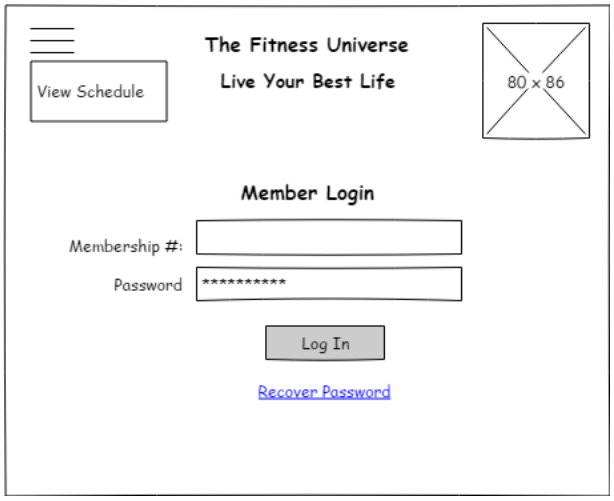

Figure 2: The Member Login page

i. **`verifyMemberNumber(`*`mid`*`)`**. Takes the string entered in the ***Membership #*** textbox as parameter *mid* and returns a Boolean value that indicates whether the string is in the correct format. A valid membership number starts with the number 9 followed by five numeric characters to form a six-digit number. **(5 marks)**

ii. **`verifyPassword(`*`pwd`*`)`**. Takes the string entered in the password textbox as parameter ***pwd*** and returns a Boolean value that indicates whether the string is in the correct format. A valid password must: (i) be between 8 - 16 characters long; (ii) mainly consist of alphanumeric characters but must have **at least one** of the following special characters: **&**, **$**, **#**, **@** and; (iii) have **at least one** upper case letter and at least one number. **(5 marks)**

c. Create a JavaScript function called **`loginMember(`*`data`*`)`** that takes the form's user input in **JSON** format through the ***data*** parameter. It invokes the **`verifyMemberNumber`** and **`verifyPassword`** functions to check the user input and if either of these functions return false, then loginMember generates one or more error messages and stops any further processing.

If no errors are generated, the verified membership number and password are checked against entries in the ***members*** localStorage dataset. If the user-entered membership number/password pair matches an entry in localStorage with the same membership and password, then the full entry is retrieved and used to update the ***members*** sessionStorage dataset with the first name, last name and membership number. If no valid entry is found, then the function should return an error message as a string, otherwise, it returns an empty error string. **(8 marks)**

d. Error messages generated either from an improperly formatted username or password, or if the user is not a valid user, must be displayed in red text between the Member Login label and the Membership # textbox. **No dialog boxes are to be used to display JavaScript error messages**. **(3 marks)**

[Q2 Total: 25 marks]

3. Figure 3 shows the Personal Trainer Login page. The HTML file for this page must be named **trainer_login.html**. Include a hidden field called *employee* with a value of "*trainer*." **(3 marks)**

   a. The hamburger menu in the top left corner of the page, when clicked, shows a dropdown menu with two non-clickable menu items: Change Location and Gym Schedule. **(2 marks)**

   b. The page requires two pieces of information: the trainer's email address and password. The format for the email address and password must be validated by two

Figure 3: The Trainer login page

separate JavaScript functions, which are invoked when the Log In button is clicked. These functions can use JavaScript string manipulation and other functions, ***but no regular expressions***.

   i. **`verifyEmail(`*`email`*`)`**. Takes ***a valid email*** entered in the email textbox and ensures that the domain name for the email address is one of three values: `fitnessuniverse.com`, `fitnessuniverse.bb` or `fitnessu.life`. If the domains are valid, it returns true and false otherwise. The input to this function must be a correctly formatted email address. **(3 marks)**

   ii. **`verifyPassword(`*`pwd`*`)`**. This is the same function that is used in the members login page. **(2 marks)**

   c. Create a JavaScript function called **`loginEmployee(`*`data`*`)`** that takes the form's user input in **JSON** format through the ***data*** parameter. It invokes the **`verifyEmail`** and **`verifyPassword`** functions to check the user input and if either of these functions return false, then loginEmployee generates error messages and stops any further processing.

If no errors are generated the verified email address and password are checked. If the value of the employee hidden field is "trainer" then a matching entry in the ***trainers*** localStorage dataset is searched for. If the email/password pair has a corresponding email and password ***trainers*** entry in localStorage, then the full entry is retrieved and used to update the ***trainers*** sessionStorage dataset with the first name, last name and email address. If no valid entry is found, then the function should return an error message as a string, otherwise, it returns an empty error string. **(6 marks)**

d. Error messages generated either from an improperly formatted email or password – or if the user is not a valid employee – must be displayed in red text between the Personal Trainer Login label and the Email textbox. **No dialog boxes are to be used to display JavaScript error messages**. **(3 marks)**

e. The Forgot Password link, goes to the **forgot_trainer.html** page, Figure 4. **(1 mark)**

[Q3 Total: 20 marks]

Figure 4: The recover password page

4. The **forgot_trainer.html** page has a hidden employee field with a value of "trainer." When the Recover button is clicked, it triggers the **`findLostPassword()`** function. This function calls the **`verifyEmail`** function described above. If the email is valid, **`findLostPassword`** checks for an entry in the localStorage ***trainers*** dataset if the value of the hidden employee field is "trainer." If an entry is found, it displays a CSS popup message "Recovery link sent" on the page. If no entry is found, then the CSS popup message says "No account associated with email: xxx@xxx.com" where xxx is the email address entered.

[Q4 Total: 10 marks]

5. Figure 5 shows the Gym Manager Login page. The HTML file for this page must be named **admin_login.html** and it must be placed in a *subdirectory* called **admin**. This page is ***not*** accessible from the main console page but must be accessed by typing the URL into the browser directly. Include a hidden field called *employee* with a value of "*manager.*" **(3 marks)**

   a. The hamburger menu in the top left corner of the page, when clicked, shows a dropdown menu with two non-clickable menu items: Change Location and Gym Schedule. **(1 mark)**

   b. The page requires two pieces of information: the manager's email address and password. The data format for the email address and password must be validated by two respective JavaScript functions, which are invoked when the Log In button is clicked. These functions can use JavaScript string manipulation and other functions, ***but no regular expressions***.

Figure 5: The Gym Manager login page

i. **`verifyEmail(`*`email`*`)`**. This is the same function as described for the trainer's login page. **(2 marks)**

ii. **`verifyPassword(`*`pwd`*`)`**. The same function as previously described. **(2 marks)**

c. Use the **`loginEmployee(`*`data`*`)`** function as described above to check the entries in the localStorage dataset called ***managers*** if the employee field is set to "manager". **(6 marks)**

d. The Forgot Password link, goes to the HTML page, **forgot_manager.html** page that has the same design as Figure 4. **(1 mark)**

[Q5 Total: 15 marks]

6. The **forgot_manager.html** page functions similarly to the forgot_trainer.html page except it has an employee hidden field with a value of "manager."

[Q6 Total: 5 marks]

7. The following pages are the console pages that each respective type of user sees after they login. These pages must use a function called **`checkLogin()`** to make sure that users cannot access them unless they are logged in as either a member, trainer or manager. The gym member console page, called **member_console.html**, is shown in Figure 6. The personal trainer console pages are shown in Figure 7, with the trainer's client page

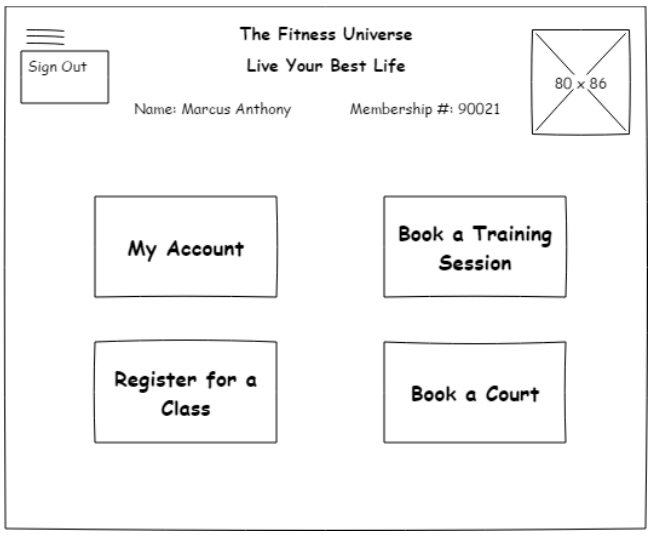

Figure 6: The gym member console page

(**trainer_client.html**) in Figure 7a and the trainer's schedule (**trainer_schedule.html**) in Figure 7b. The gym manager's console pages are the members page Figure 8a (**admin_members.html**) and trainer page Figure 8b (**admin_trainers.html**).
The functioning links are:

a. The **Sign Out** link, clears the respective entries from the sessionStorage object depending on the page. This will have the effect of logging the user out and returning them to their respective login page.
b. The **Client** link goes to **trainer_client.html**; the Training Schedule link goes to **trainer_schedule.html**; the Members link goes to **admin_members.html** and the Trainers link goes to **admin_trainers.html**. All other links remain on the page.

[Q7 Total: 40 marks]

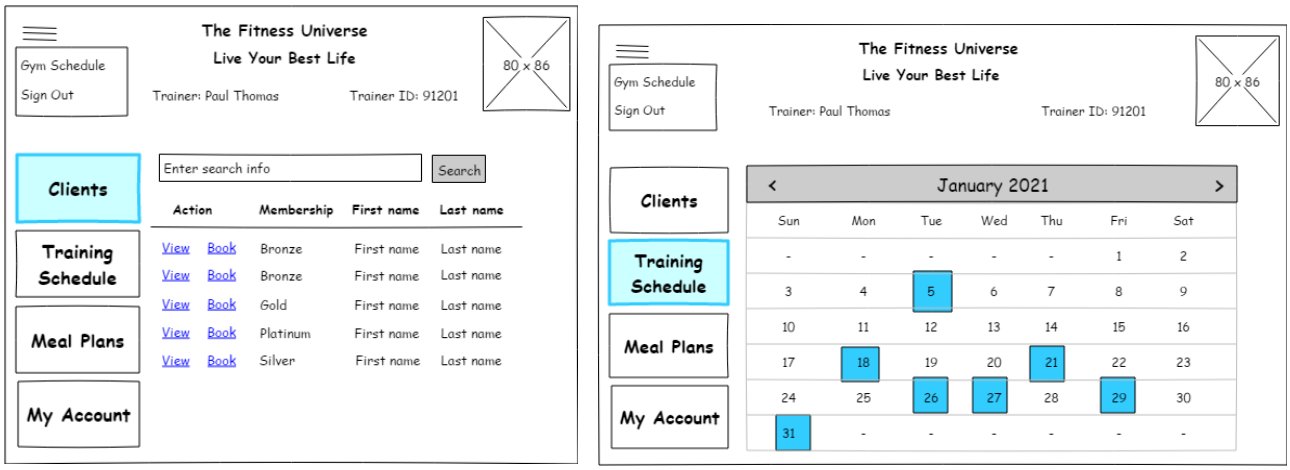

Figure 7: The Gym Trainer console pages

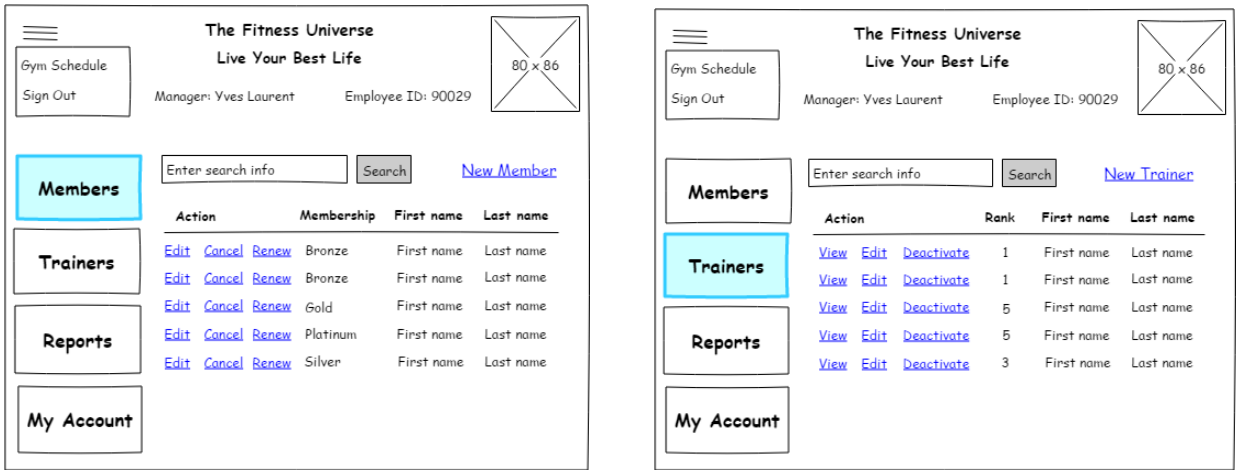

Figure 8: The Gym Manager console pages

8. A JavaScript function called **`loadData()`** will be provided that will load the required data. A brief explanation of each dataset will be given in the documentation. This function has to be invoked before a user can successfully log in.

[Q8 Total: 5 marks]

Implementation Tips

Here is a way to get started. This is just to help you get over the initial hump, not an exhaustive (or complete) process.

- Start with the page design and ensure it is mobile first responsive, that is, it displays properly on mobile and desktop devices. Pick any colour scheme/theme you like that is appropriate
- Write the JavaScript functions and make sure they produce the correct output. You can also create some function to display error messages to make your code more modular.
- Write the login in functions and make sure that the application pages are only accessible to a logged in user. This means you should create the checkLogin JavaScript function that checks to see if the user is logged in. You can implement the Sign Out functionality at this time if you want.