Autumn 2024: Web Application Assignment

Due by 7:00pm on Friday 31st May 2024 **Assessment Weight: 30%**

A. Requirements

a) ALL instructions given in this document MUST be followed to be **eligible** for full marks for the Web Application Assignment. This document has seven (7) pages in total including four (4) Appendices.
b) This assignment is **NOT** a group assignment; collusion, plagiarism, cheating, use of generative AI of any kind is not acceptable. As part of your submission, you MUST certify that all work submitted is your own and has not been written for you by any other person or AI tool. If you cannot honestly certify that the work is your own, then do not submit the assignment. Breaches of the Misconduct Rule will be dealt with according to the university policy (see the learning guide for more information).
c) All assignment submissions will be checked for academic misconduct using the MOSS program from Stanford University.
d) Design the web pages with ease of navigation and operation, attractiveness, and accessibility in mind. Images other than those provided in the assignment zip file, if any, may also be used in the assignment.
e) Your code must guard against SQL injection and Cross Site Scripting attacks. That is, sanitise all user input.
f) All assignment files are to be uploaded in the **project** folder in your TWA website on the TWA server as follows:
   - php and html files in the `project` folder
   - css files in the `project/css` folder
   - images in the `project/images` folder
   - javascript in the `project/javascript` folder.

   Note: Compressed archive files (eg, zip, tar etc) are not acceptable and will not count toward submission requirements.

g) Complete the full submission process before the due date and time. See section D for details of the submission process.
h) All styling and page layout must be achieved using CSS. The use of Bootstrap or other frameworks is **not** permitted.
i) jQuery or similar are not permitted.

For the problem definition described in section B you must
j) include your authorship details at the top of **each file** in coded comments.
k) **reference** all sources that you used for inspiration of your solution as per Section C of this document.
l) ensure that your web application renders correctly in Chrome and runs correctly from the TWA web server.

B. Web Application Assignment Details

B(i) - Background information and description

The aim of this project is to develop a simplified Exercise Tracker website that enables tracking of exercise workouts. Users can create an account, log their workouts, track their progress over time, and view basic statistics to check their progress. The details for the users and their workouts are stored in a MySQL database named `Fitness`. The Exercise Tracker website connects to this database to dynamically create the web page content. Additionally, the Exercise Tracker website will allow administrators to manage user accounts. The website will use PHP for server-side scripting and MySQLi for database interaction.

B(ii) – Functional Requirements

Your Web Application **must**

- be coded using HTML 5, CSS, JavaScript, and PHP as necessary. Note: any files described below that require any type of server-side functionality must be PHP files to achieve the functionality,
- Use MySQLi to utilise the `Fitness` database,
- provide consistent and easy-to-use navigation for all user types as appropriate. For example, navigation to a members-only page should not be present when the authenticated user is an administrator. Likewise, navigation to administrator-only pages should not be present when the authenticated user is a member.
- be designed for a 1366-pixel wide screen and be accessible,
- provide the following page content and functionality as described. Note: php pages other than those listed below may be needed for a full solution.

User Authentication and Authorisation
There are three types of users for the website: anonymous users, registered members, and website administrators.

Anonymous users do not need to register or login to use the website. However, they will not be able to access any pages designated as only for members or administrators.

As described in the preceding pages, certain parts of the website are to be accessed only by authenticated members or administrators. The same login form will be used by both members and administrators.

If a user tries to access a page for which they do not have permission, they will be automatically redirected to the site's index page.

To log in, members or administrators must provide their user credentials in a suitable form, which will be authenticated against the user details stored in the `Fitness` database. The login process **must** use a **postback** form. Upon successful login a registered member is to be redirected to the **'log workout'** page. Upon successful authentication a website administrator is to be redirected to the **'manage admin'** page.

Authenticated users, of any type, must be able to log off the system from any page of the website. When logging off, the `last_login` field of the `users` table should be updated with the current date, then redirect the user to the **index** page.

User Registration Page
To use the Exercise Tracker website, users are required to register with the site by completing a form on the registration page. The form requires users to enter their first name, last name, email address, mobile number, password, age, weight, and height. These details will be stored in the `Fitness` database. All form fields, except mobile number, age, weight, and height, must be completed in the registration form, and the provided email address must be unique to ensure successful registration. Since passwords should never be stored in plain text, the provided password must be hashed using the SHA256 algorithm before being stored in the database.

Upon successful validation of the form, the data must be stored in the database, and the user redirected to the login page.

Access Allowed:
- All user types.

Validation:
- **Client-side JavaScript** validation for mandatory fields. If any errors are detected, the form will display appropriate error messages using in-page DOM notifications and cannot submit. If no errors are detected, the form is submitted to the server where server-side validation is performed before updating the data in the `Fitness` database.
- **Server-side PHP** validation to check data types and sizes of input as necessary, guard against SQL injection and XSS attacks, and ensure the email address is unique. If any errors are detected, the form will display appropriate error messages using in-page DOM notifications, and the database is not updated. If no errors are detected, the data is updated in the database.

Manage Admin
Administrators of the Exercise Tracker website should be able to create new administrator accounts. An authenticated administrator will be able to use this page to create an administrator account by completing a registration form capturing the first name, last name, email address, mobile number, and password for the new administrator account. All these fields are mandatory, and the provided email address must be unique to create an administrator account. Since passwords should never be stored in plain text, the provided password must be hashed using the SHA256 algorithm before being stored in the database.

Upon successful validation of the form, the data must be stored in the database and a 'success' message displayed. The user is not redirected to a different page.

Note: an existing member account cannot be elevated to an administrator account.

Access Allowed:
- Administrators only.

Validation:

- **Client-side JavaScript** validation for mandatory fields. If any errors are detected, the form will display appropriate error messages using in-page DOM notifications and cannot submit. If no errors are detected, the form is submitted to the server where server-side validation is undertaken before the data is updated in the `Fitness` database.
- **Server-side PHP** validation to check data types and sizes of input as necessary, guard against SQL injection and XSS attacks, and ensure the email address is unique. If any errors are detected, the form will display appropriate error messages using in-page DOM notifications, and the database is not updated. If no errors are detected, the data is updated in the database.

All Pages

Display consistent and easy-to-use navigation for all user types as appropriate on each page of the website. The navigation presented to the user must be appropriate for their authentication status (i.e., anonymous user, authenticated member, authenticated administrator).

On every page that an authenticated member visits, display their name, current date, and date of the previous login.

On every page that an authenticated administrator visits, display their name, and current date.

Index Page

The index page should include appropriate headings, website logo, images, and description/overview[1] of the Exercise Tracker website, along with links to the ***'User Registration'***, and ***'Login'*** pages.

Access Allowed:

- All user types.

Log Workout Page

The ***log workout*** page should provide a form into which the authenticated member can enter data for their workout. On this page, the user will need to enter the date[2], type of exercise[3] (e.g., walking, running, cycling), duration of exercise, distance, and any workout notes. All form fields must be completed except for workout notes. This data needs to be validated before being saved in the database.

Access Allowed:

- Authenticated members only.

Validation:

- **Client-side JavaScript** validation for mandatory fields. If any errors are detected, the form will display appropriate error messages using in-page DOM notifications and cannot submit. If no errors are detected, the form is submitted to the server where server-side validation is undertaken before the data is updated in the `Fitness` database.
- **Server-side PHP** validation to check data types and sizes of input as necessary, guard against SQL injection and XSS attacks, and to ensure the date is not in the future. If any errors are detected, the form will display appropriate error messages using in-page DOM notifications and the database is not updated. If no errors are detected the data is updated in the database.

Workout History Page

The ***workout history*** page enables the user to view a history of their logged workouts. When the page first loads, display the date, type of exercise, duration, distance, and notes for each logged workout in reverse order of date (ie, the most recent workout at the start of the listed output). Since there will be many workouts over time, the user will need to be able to filter the output by date, and type of exercise. The user should also be able to sort the data on this page in ascending and descending order of date, type of exercise, and duration. The filters and sorting will need to be implemented using a postback form. None of the form fields are mandatory.

---

[1] A text file, `index_Text.txt` has been provided in the project zip file that includes suitable text that can be used in the index page to describe the website.

[2] The default value for the date should be the current server date. Users can change the date if the workout was on a different date, but it cannot be empty or a future date.

[3] The type of exercise input device must be generated directly from the `exercise` table of the database. This is to ensure that only valid exercise types are chosen.

Access Allowed:

- Authenticated members only.

Validation:

- **Server-side PHP** validation to check data types and sizes of input as necessary, and to guard against SQL injection and XSS attacks. If any errors are detected, the form will display appropriate error messages using in-page DOM notifications.

Statistics Page

The statistics page will allow the user to view basic statistics about their workouts. The page should display the total duration and distance of all workouts, as well as breakdowns for each type of exercise. Also, the page should calculate and display the average duration and distance for each exercise type. There is no need for any filtering or sorting of data on this page.

Access Allowed:

- Authenticated members only.

B(iii) – Database Description

1. Tables in the `Fitness` database are outlined in the **Data Dictionary in appendix 1**.

2. You have been provided with your own copy of the `Fitness` database on the TWA server. To access this database, you need to use a username and password. Details on how to connect to your copy of the database are in **appendix 2**.

3. The tables within the `Fitness` database have already been populated with some data. Use the `ViewAllTables.php` script to view the data (make sure you use the connection information as indicated above). This script is found in the project zip file.

4. A list of usernames and plain text decrypted passwords can be found in **User Credentials in appendix 3**.

B(iv) – HTML, CSS, JavaScript, PHP, and image files

CSS: All page styling is to be achieved using CSS. Create a CSS file called `project_Master.css`. Add your CSS rules to this file. You may create additional CSS files if you wish. All CSS files should be uploaded to the `project/css` folder of your TWA website.

JavaScript: Some functionality in the website will need to be achieved using JavaScript (eg, the client-side validation). Create a JS file called `project_Script.js`. Add your JavaScript to this file. You may create additional JavaScript files if you wish. The JavaScript files should be uploaded to the `project/javascript` folder of your TWA website.

PHP and HTML: Most pages listed in section B(ii) will need to be PHP files to achieve the required server-side functionality. You may also create additional PHP and HTML files as part of your solution if desired, **as long as doing so does not contradict or negate the stated page requirements** (for example, the login process must be achieved using a single php file that utilises postback). Upload PHP and HTML files to the `project` folder of your TWA website.

Images: All image files should be uploaded to the `project/images` folder of your TWA website.

C. Referencing

Referencing must follow the guidelines given in Section 3.2.3 of the TWA Subject Outline. Examples of how to implement appropriate referencing within code can be found in the FAQ in the TWA vUWS site.

D. Submission Instructions

To submit your Assignment, you must do the following by the due date and time specified on page 1 of this document.

1. Upload all assignment files in the **project** folder in your TWA website on the TWA server as follows:
    a. php and html files in the `project` folder
    b. css files in the `project/css` folder
    c. images in the `project/images` folder
    d. javascript in the `project/javascript` folder.

2. Run the submission script located at

   `http://twaaut.cdms.westernsydney.edu.au/submit/submit.asp`

   As part of the submission, you will be prompted for your TWA website username and password. You will then be asked to read the WSU policy on misconduct and certify that work submitted by you is your own work. This action will be logged in a database for future reference and is deemed to be evidence that you claim that your work is original. Next, you will need to select from a drop-down list the Assessment you are submitting, eg, Assignment 1, and click the *Submit Assessment* button. The web page will then display a listing of the files you have submitted along with a receipt number. You should print or screenshot this page for proof of submission.

E. Marking Criteria and Standards

The marking criteria and standards for the Web Application Assignment are published in Section 3.2.3 of the Subject Outline and will be used to assess your assignment submission according to the specific weightings identified in the table below:

CRITERIA	WEIGHT
CODE FUNCTIONALITY/CORRECTNESS	65%
WEB PAGE DESIGN	20%
FORM DESIGN	10%
CODE READABILITY	5%

Appendices

Appendix 1 – `Fitness` Data Dictionary

The `Fitness` database consists of 3 tables. Each table is outlined below. Primary and foreign keys are also indicated. A description of how to connect to your copy of the database is given in Appendix 2. Some notes on inserting and updating the tables are provided in Appendix 4.

Table Name: users

This table provides details about users that can access the system including their login credentials (email and password) and their personal details.

Note: passwords are encrypted using the SHA256 algorithm. For testing purposes, a copy of the plain-text passwords is given in Appendix 3.

Your database credentials provide **Select, Insert** and **Update** privileges for this table.

Column	Type	Null	Default	Comments
`user_id`	`int`	No		This is an auto incrementing number to uniquely identify a table row. **You do not insert this number into the database it is determined automatically.** Unique identifier for a user. ***Primary Key***
`first_name`	`varchar(50)`	No		user's first name
`last_name`	`varchar(50)`	No		user's last name
`email`	`varchar(100)`	No		Email address. This is used as the user's username for authentication. It must be unique for all users.
`mobile`	`varchar(10)`	Yes	`Null`	Mobile number for user
`password`	`varchar(100)`	No		user's password. This is used for authentication. The value stored in this field is encrypted using the SHA256 algorithm.

is_admin	tinyint(1)	No	0	Flag to indicate if the user is an administrator. 1 for Yes, 0 for No.
date_registered	timestamp	No	Current date time	Date that the user registered themselves on the website or that an Administrator created an Administrator account. Auto generated by default.
last_login	timestamp	Yes	Null	Date the member last logged into the website (prior to the current login).
age	int	Yes	Null	Age of the member in years.
height	int	Yes	Null	Height of the member in cm.
weight	float	Yes	Null	Weight of the member in kg.

Table Name: workout

This table is used to store data about exercise workouts for the website members.

Your database credentials have **Select** and **Insert** privileges for this table.

Column	Type	Null	Default	Comments
workout_id	int	No		This is an auto incrementing number to uniquely identify a workout. **You do not insert this number into the database it is determined automatically.** ***Primary Key***
user_id	int	No		Unique identifier of user that submitted the workout data. ***Foreign Key***
workout_date	timestamp	No	Current date time	The date the workout data was submitted. Auto generated by default.
exercise_id	int	No		The type of exercise for the workout. ***Foreign Key***
duration	int	No		Duration of the workout in minutes.
distance	float	No		Distance covered in the workout.
notes	text	Yes	Null	Any notes the member wants to record about the workout.

Table Name: exercise

This table stores data about the exercises that the website keeps track of.

Your database credentials have **Select** privileges for this table.

Column	Type	Null	Default	Comments
exercise_id	int	No		This is an auto incrementing number to uniquely identify an exercise type. **You do not insert this number into the database it is determined automatically.** ***Primary Key***
name	varchar(25)	No		The name of the type of exercise.
description	varchar(150)	No		The description of the type of exercise.

Appendix 2 – Connecting to your `Fitness` Database

You have your own copy of the `Fitness` database. To access this database, you use a MySQL username and password. The following generic connection information can be used to connect to your `Fitness` database from your php scripts:

Database name: ***Fitness###***
Username: `twa###`
Password: `twa###XX`
Server: `localhost`

where **###** is your twa site **number**, and **XX** refers to the first two characters of your twa site password.

For example, if your TWA site is `twa999`, and your password is **ab**`cd7890`, then the following would be your connection information:
Database name: ***Fitness999***
Username: `twa999`
Password: `twa999ab`
Server: `localhost`

Hence, to connect to the ***Fitness999*** database from your php script you would require code similar to the following:

```
$dbConn = new mysqli('localhost', 'twa999', 'twa999ab', 'Fitness999');
if ($dbConn->connect_error) {
    die('Connection error (' . $dbConn->connect_errno . ')'
    . $dbConn->connect_error);
}
```

Note: The tables within the database have already been populated with sample data. Use the `ViewAllTables.php` script to view the data (make sure you use the connection information as indicated above in the script to be able to view the data in your copy of the database).

Appendix 3 – User Credentials for the Exercise Tracker website

The passwords stored in the **password** field of the **users** table are encrypted using the SHA256 algorithm. Below are the plain-text passwords for these users.

email	Plain-text **password**	Administrator
s.stevenson@gmail.com	IamTheSonOfStevens	0
RosieBloom@icloud.com	roseColouredGlasses	1
BobbyBoy@gmail.com	myExercisePassword	0
KittyKat@yahoo.com	catsRBest	0

Appendix 4 – Notes about inserting and updating records in your `Fitness` database

- When inserting into a table do not supply the primary key value since the RDBMS will automatically generate this when the insert takes place.
- When inserting into a table, if a field has an automatic default value do not supply your own value for this field unless you want it to be different to the default value.

  For example, when inserting into the `users` table do not supply the `is_admin` field value since it will default to `0` which is the required value for a registered member. Only administrators have the value of `1`, which is never assigned via the member registration form.
- When updating the `users` table, the only field that will need updating is the `last_login` field. See the User Authentication and Authorisation section for details on when to update.