SCHOOL OF ENGINEERING AND TECHNOLOGY

ASSIGNMENT / PROJECT SUBMISSION FORM

PROGRAMME : BSc (Hons) Information Technology (481BIT1)

SEMESTER : Jan / Apr / Aug 2025

SUBJECT : WEB 2202 Web Programming

DEADLINE : 12 MARCH 2025 23:59 MYT

INSTRUCTIONS TO CANDIDATES

- This is a(n) individual / ~~group~~ project.

IMPORTANT
The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

- Coursework submitted after the deadline but within 1 week will be accepted for a maximum mark of 40%.
- Work handed in following the extension of 1 week after the original deadline will be regarded as a non-submission and marked zero.

Lecturer's Remark (Use additional sheet if required)
List down the name of the group members and the student IDs here.

I, Lee Ming Hui Isaac (Student's Name), 22057301 (Student ID), received the assignment and read the comments.

leeminghuiisaac, 12 March 2025 (Signature/Date)

Academic Honesty Acknowledgement

"I, Lee Ming Hui Isaac (Student's Name) verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties *(refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme)* for any kind of copying or collaboration on any assignment."

leeminghuiisaac, 12 March 2025 (Student's signature / Date)

Data Protection
The protection of personal data is an important concern to Sunway University and any personal data collected on this form will be treated in accordance with the Personal Data Protection Notice of the institution.
http://sunway.edu.my/pdpa/notice_english (English version)
http://sunway.edu.my/pdpa/notice_bm (Malay version)

1. Reflection

[Prepare a one-page (equivalent to about 500-words) reflection statement on your individual learning experience and insight from building (i.e., design and development) the website.]

First and foremost, I would like to **express my heartfelt gratitude to Dr Aaliya Sarfaraz** for assigning an assignment topic that challenges me to create an **e-commerce website closely linked to the United Nations' Sustainable Development Goal (SDG) 3: Good health and well-being**. Traditionally, e-commerce websites are related to other SDGs, such as goal 8: Decent work and economic growth, goal 9: Industry, innovation, technology and infrastructure, as well as goal 12: Responsible consumption and production. However, this unique project has pushed me to **think "out-of-the-box" beyond traditional web development approaches** and incorporate elements that promotes healthy living and accessibility among Malaysians, requiring me to have a stable balance between robust technical features and advocating for health and wellness.

One of the most confusing challenges I encountered was **implementing proper form validation that ensures data exchanged with the server is up to date**. This issue stemmed from **discrepancies in user profile updates**, where initially after users modified their shipping details, the changes should appear on their screens, but upon refreshing the page or proceeding to checkout, the old information was still displayed instead. To resolve this issue, **POST requests were implemented** to **validate and sanitise the inputs** using PHP code to ensure data is submitted appropriately, while **GET requests** were implemented to **obtain the most recent data from the server for the information displayed being the latest** and **received without missing packets**. As such, this allows my website being able to **securely process every input** with the latest information alongside **upholding data integrity**.

```php
$search_name = isset($_GET['name']) ? $_GET['name'] : '';
$search_category = isset($_GET['category']) ? $_GET['category'] : '';
$search_min_price = isset($_GET['min_price']) ? $_GET['min_price'] : '';
$search_max_price = isset($_GET['max_price']) ? $_GET['max_price'] : '';
```

(*A screenshot of my code showing the GET requests being handled by retrieving and sanitizing query parameters before using them to filter and display items from the database.)*

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $first_name = trim($_POST['first_name']);
    $last_name = trim($_POST['last_name']);
    $sex = trim($_POST['sex']);
    $birthdate = trim($_POST['birthdate']);
    $phone_number = trim($_POST['phone_number']);
    $street = trim($_POST['street']);
    $district = trim($_POST['district']);
    $city = trim($_POST['city']);
    $postal_code = trim($_POST['postal_code']);
    $state = trim($_POST['state']);
    $email = trim($_POST['email']);
    $password = trim($_POST['password']);
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);
```

(A screenshot of my code showing the POST requests being handled by retrieving and sanitising form data before inserting them into the database.)

Furthermore, **not understanding the hashed passwords** in the database after insertion was another challenge I have faced during this project. At first, I was confused by the **unintelligible string values that replaced the plain text passwords**; but after performing further studies into PHP's password hashing techniques, I realised that these complex strings are **intentional by design to maintain proper security**. Functions like **password_hash()** generate one-way encrypted values that make it nearly impossible to retrieve the original password, thereby **protecting user data** even if the database is compromised. Also, by integrating **password_verify()** into the login process, user credentials can be validated against these hashed values to ensure that the **authentication process**

remains secure and reliable. Consequently, this challenge furthered my understanding in PHP's role in safeguarding our sensitive information.

(A screenshot from my database of the hashed password "1234567890" where administrators do not know what the actual password is, hence protecting confidential information.)

Additionally, **adopting object-oriented programming (OOP) techniques** in my code was also another challenge I encountered. Initially, I adopted a **procedural approach** because I assumed the project would remain small-scale. However, as the website's complexity increased, **managing numerous files became increasingly challenging**, and it became clear that supporting new features and ensuring long-term scalability required a maintainable approach. **Incorporating OOP fundamentals**, such as encapsulation, inheritance, and polymorphism allowed me to **isolate different parts of the website functions** into separate classes that **improved code readability**, making **understanding, debugging, testing, and improvements efficient**. As a result, this further **increased my experience in integrating OOP techniques to tackle complicated problems and projects**, ensuring that the website can evolve as new features are added.

```php
<?php
class Item {
    private $db;

    public function __construct($db) {
        $this->db = $db;
    }

    public function getItems($filters = []) {
        $query = "SELECT * FROM items WHERE 1=1";
        $params = [];

        if (!empty($filters['name'])) {
            $query .= " AND name LIKE :name";
            $params[':name'] = '%' . $filters['name'] . '%';
        }
        if (!empty($filters['category'])) {
            $query .= " AND category = :category";
            $params[':category'] = $filters['category'];
        }
        if (!empty($filters['min_price'])) {
            $query .= " AND price >= :min_price";
            $params[':min_price'] = $filters['min_price'];
        }
        if (!empty($filters['max_price'])) {
            $query .= " AND price <= :max_price";
            $params[':max_price'] = $filters['max_price'];
        }

        $stmt = $this->db->prepare($query);
        $stmt->execute($params);
        return $stmt->fetchAll();
    }
}
?>
```

(A screenshot of my code showing the `Item` class, which uses encapsulation to manage database operations for retrieving items based on filters.)

(Total word count (excluding picture captions): 500 words)

2. Explanation and Description of Code

[Prepare a one-page (equivalent to about 500-words) explanation and description on a specific section or feature that the website that you have worked on.]

The specific section of the website that I have worked on is the checkout.php file, as it is responsible for managing the checkout process and serving as the final step in the user's shopping journey. This section of the website integrates multiple crucial functionalities, such as user authentication, session management, dynamic data retrieval, cart items display, secure order processing, and many more. Hence, I have determined that this is the most complex and feature-rich section of my grocery-themed e-commerce website.

Session Management

The section starts with session management through setting a timeout configuration of an hour (3600 seconds), where session data will be eligible for deletion when the session cookie expires. For session_start(), this line starts a new session or resumes an existing one before accessing or modifying any session variables. To illustrate:

```
2   // Set session timeout to an hour
3   ini_set('session.gc_maxlifetime', 3600);
4   session_set_cookie_params(3600);
5
6   session_start();
```

(This code configures the PHP session to expire after one hour by setting the garbage collection lifetime and cookie parameters to 3600 seconds, then initializing the session with session_start().)

The reason behind implementing this feature is to reinforce the security of the application by reducing the risk of unauthorised access if the user forgets to log out or leaves their session open on a shared or public device. This also frees up memory and storage space on the server by managing resources efficiently.

```php
// Check if the session has expired
if (isset($_SESSION['LAST_ACTIVITY']) && (time() - $_SESSION['LAST_ACTIVITY'] > 3600)) {
    session_unset();
    session_destroy();
    header('Location: login.php');
    exit();
}
$_SESSION['LAST_ACTIVITY'] = time();
```

(This code snippet implements session timeout functionality. It checks if the time elapsed since the user's last activity exceeds one hour (3600 seconds). If so, it un-sets and destroys the session, redirecting the user to the login page; otherwise, it updates the last activity timestamp to the current time.)

User Authentication

Then, it checks if the user is already logged in by verifying the presence of `$_SESSION['user_id']`. If the user is not logged in, they are then redirected to the login page (login.php); and if they have done so, they are brought back to the checkout.php page. This ensures that only authenticated users can proceed to checkout and prevent miscommunication. To show:

```php
// Check if the user is logged in
if (!isset($_SESSION['user_id'])) {
    // Store the current page in the session to redirect back after login
    $_SESSION['redirect_to'] = 'checkout.php';
    header('Location: login.php');
    exit();
}
```

(This code checks whether a user is logged in—if authenticated, they remain on checkout.php; if not, they are redirected to login.php to sign in, then they are redirected back to checkout.php.)

If the user is an administrator, the user will be redirected to another file named administration.php, where they can add new products, remove existing products, and view all products. Here is the view:

(A screenshot of the administration panel where administrators can add new products, remove existing products, and view all products.)

Database Connection

For a successful database connection to happen, a separate file is connected for providing secure and efficient data upload and retrieval. This file is named db_connect.php which contains the necessary credentials and connection logic for interaction with the MySQL database. In checkout.php, it is stated briefly like this:

```
25    // Include the database connection file
26    include('includes/db_connect.php');
```

(This code includes the database connection file, establishing a secure PHP Data Objects (PDO) connection for database interactions.)

And db_connect.php contains these lines of code to establish a secure PDO connection to the MySQL database. It sets the connection parameters (the server's name, username, password, database's name, and charset), configures PDO options for error handling and data retrieval,

and uses a try-catch block to manage connection errors:

```php
<?php
// includes/db_connect.php

// Database connection parameters
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "simply_fresh_foods";
$charset = 'utf8mb4';

// Data Source Name (DSN) string that contains the information required to connect to the database
$dsn = "mysql:host=$servername;dbname=$dbname;charset=$charset";

// PDO options to configure the PDO connection
$options = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
];

try {
    // Attempt to create a new PDO instance with the provided DSN and options
    $pdo = new PDO($dsn, $username, $password, $options);
} catch (PDOException $e) {
    // If the connection fails, catch the PDOException and display an error message
    die("Connection failed: " . $e->getMessage());
}
?>
```

(This code sets up the database connection by configuring the connection parameters and initialising a secure PDO instance with robust error handling for reliable MySQL interactions.)

Fetching User Details

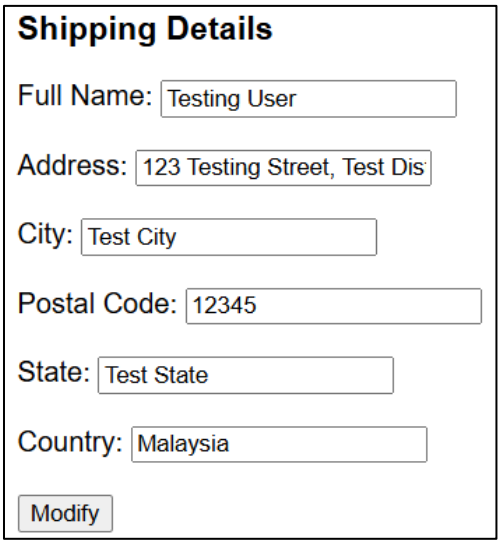
Moving on, the logged in user's shipping details are fetched from the database using their user ID (user_id, configured as primary key) stored in the session. This information is used to pre-fill the shipping details, which they have the option to modify it if they so choose, providing a seamless checkout experience. To demonstrate:

```php
// Fetch the logged-in user's shipping details
$user_id = $_SESSION['user_id'];
$sql = "SELECT first_name, last_name, street, district, city, postal_code, state FROM users WHERE user_id = ?";
$stmt = $pdo->prepare($sql);
$stmt->execute([$user_id]);
$userDetails = $stmt->fetch(PDO::FETCH_ASSOC);
```

(This code retrieves the logged-in user's shipping details from the database using the user ID stored in the session. It uses a PDO prepared statement to securely query for fields such as

first name, last name, and address information, which are then fetched as an associative array to pre-fill the checkout form.)

Sample view:



(This section renders a pre-filled shipping details form using PHP to insert user data from the database. All fields—full name, address, city, postal code, state, and country—are initially set to read only (readonly) to prevent unintended changes, while the "Modify" button allows users to unlock the form for editing as needed (The design is not final and will be improved.).)

(A screenshot of the relevant data retrieved from the database.)

Fetching Cart Items

Moreover, the cart items are fetched from the previous session using `$_SESSION['cart']`, allowing the website to display the items the user has added into their cart and providing a summary of their order. This allows the website to scale more effectively through using sessions to store cart items where session data is managed by the server, which

can handle multiple users and their respective carts without any performance degradation.

```
35    // Fetch the cart items from the session
36    $cart_items = isset($_SESSION['cart']) ? $_SESSION['cart'] : [];
```

(This code retrieves the shopping cart items from the session. It checks if the 'cart' key exists in $_SESSION; if it does, its value is assigned to $cart_items, otherwise, $cart_items is set as an empty array.)

Rewards/Points Awarding System

The primary goal of the rewards/points system is to build customer loyalty through awarding points for every purchase. Consequently, customers are incentivised to return and make additional purchases to accumulate more points. To do so, this function updates the user's point balance in the database by adding the specified number of points to their current balance. It uses a prepared statement to securely execute the SQL query, ensuring that the user's point balance is accurately updated. Here's the relevant code:

```
38    // Function to add reward points to the user's account
39    function addRewardPoints($user_id, $points) {
40        global $pdo;
41        $sql = "UPDATE users SET point_balance = point_balance + ? WHERE user_id = ?";
42        $stmt = $pdo->prepare($sql);
43        $stmt->execute([$points, $user_id]);
44    }
```

(This screenshot shows the `addRewardPoints` function, which updates the user's point balance in the database by adding the specified number of points to their current balance using a prepared statement for secure execution.)

and this, which calculates the total amount spent during the purchase and awards reward points based on the amount spent:

```php
// Example usage of the functions after a successful purchase
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['payment_method'])) {
    // Calculate total amount spent
    $total_amount_spent = 0;
    foreach ($cart_items as $item) {
        $total_amount_spent += $item['item_price'] * $item['item_quantity'];
    }

    // Add reward points (1 point for every RM1 spent)
    $points = $total_amount_spent;
    addRewardPoints($user_id, $points);
```

(This screenshot shows the process of calculating the total amount spent after a successful purchase and awarding reward points accordingly. The code iterates through the cart items to determine the total cost and then calls the addRewardPoints function to award one point for every RM1 spent.)

Notification Functions

The role of notifications in this e-commerce website is to provide immediate feedback and announcements to users after they have completed important actions, like making a purchase. For example, after a successful transaction, the system will call the sendNotification() function to insert a confirmation message into the notifications database, which informs the user that their purchase has been processed, and the reward points have been added to their account. Henceforth, notifications not only enhance the user experience by sending real-time confirmation messages but also act as evidence to ensure transparency.

```php
function sendNotification($user_id, $message) {
    global $pdo;
    $sql = "INSERT INTO notifications (user_id, message) VALUES (?, ?)";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([$user_id, $message]);
}
```

(This code defines the sendNotification function, which uses a PDO prepared statement to securely insert a notification message for a specified user into the notifications table.)

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['payment_method'])) {
    $total_amount_spent = 0;
    foreach ($cart_items as $item) {
        $total_amount_spent += $item['item_price'] * $item['item_quantity'];
    }

    $points = $total_amount_spent;
    addRewardPoints($user_id, $points);
    sendNotification($user_id, "Thank you for your purchase! You have earned $points reward points.");
    $email = $userDetails['email'];
    $subject = "Order Confirmation";
    $message = "<p>Thank you for your purchase!</p><p>You have earned $points reward points.</p>";
    sendEmail($email, $subject, $message);

    header('Location: confirmation.php');
    exit();
}
```

(This screenshot displays how the code handles the purchase process by calculating the total amount spent from the cart items, awarding reward points accordingly via addRewardPoints(), and sending a notification to the user with sendNotification(). It also dispatches an order confirmation email through sendEmail() before redirecting the user to the confirmation page.)

Email Functions

The email functions in this e-commerce website are to provide a formal, verifiable record of user actions. For instance, the system calls the sendEmail() function to send an order confirmation email after a successful transaction, which contains important details of the purchase, and the reward/points earned, ensuring users receive detailed communication regarding their transaction.

```php
function sendEmail($to, $subject, $message) {
    $headers = "From: no-reply@simplyfreshfoods.com\r\n";
    $headers .= "Content-Type: text/html; charset=UTF-8\r\n";
    mail($to, $subject, $message, $headers);
}
```

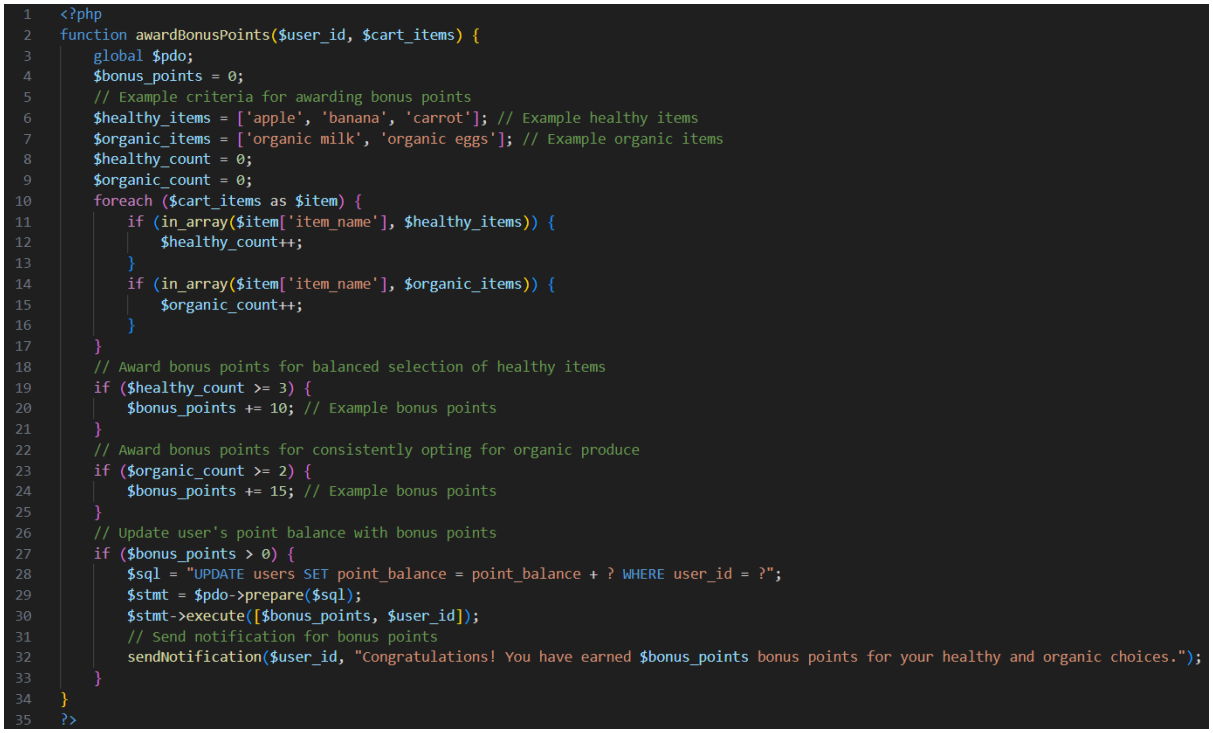
(This code defines the sendEmail function, which sends an email using PHP's mail() function. It sets the appropriate headers for a no-reply sender and Hyper Text Markup Language (HTML) content, ensuring that order confirmations or notifications are sent in a secure and properly formatted manner.)

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['payment_method'])) {
    $total_amount_spent = 0;
    foreach ($cart_items as $item) {
        $total_amount_spent += $item['item_price'] * $item['item_quantity'];
    }

    $points = $total_amount_spent;
    addRewardPoints($user_id, $points);
    sendNotification($user_id, "Thank you for your purchase! You have earned $points reward points.");
    $email = $userDetails['email'];
    $subject = "Order Confirmation";
    $message = "<p>Thank you for your purchase!</p><p>You have earned $points reward points.</p>";
    sendEmail($email, $subject, $message);

    header('Location: confirmation.php');
    exit();
}
```

(This screenshot shows the website retrieving the user's email from their details, defines an order confirmation subject and HTML message that includes reward points information, and then calls the sendEmail() function to dispatch the confirmation email to the user.)

Gamification and Rewards Challenges

Having special tasks for users to complete and earn extra rewards/points strengthens engagement with customers through motivating them by adding competitive, fun elements to make the shopping experience interactive. Besides that, this initiative also promotes users to make healthy choices that align with this project's goal: good health and well-being while incentivising them to return and continue purchasing from the site. To attain this, the `awardBonusPoints` function rewards points based on their purchases of healthy and organic items. It checks the cart for specific items and awards 10 bonus points for at least three healthy items and 15 bonus points for at least two organic items. These points are then added to the user's account, and a notification is sent to inform them of their earned bonus points.



```php
<?php
function awardBonusPoints($user_id, $cart_items) {
    global $pdo;
    $bonus_points = 0;
    // Example criteria for awarding bonus points
    $healthy_items = ['apple', 'banana', 'carrot']; // Example healthy items
    $organic_items = ['organic milk', 'organic eggs']; // Example organic items
    $healthy_count = 0;
    $organic_count = 0;
    foreach ($cart_items as $item) {
        if (in_array($item['item_name'], $healthy_items)) {
            $healthy_count++;
        }
        if (in_array($item['item_name'], $organic_items)) {
            $organic_count++;
        }
    }
    // Award bonus points for balanced selection of healthy items
    if ($healthy_count >= 3) {
        $bonus_points += 10; // Example bonus points
    }
    // Award bonus points for consistently opting for organic produce
    if ($organic_count >= 2) {
        $bonus_points += 15; // Example bonus points
    }
    // Update user's point balance with bonus points
    if ($bonus_points > 0) {
        $sql = "UPDATE users SET point_balance = point_balance + ? WHERE user_id = ?";
        $stmt = $pdo->prepare($sql);
        $stmt->execute([$bonus_points, $user_id]);
        // Send notification for bonus points
        sendNotification($user_id, "Congratulations! You have earned $bonus_points bonus points for your healthy and organic choices.");
    }
}
?>
```

(This screenshot shows the `awardBonusPoints` function, which calculates and awards bonus points based on the user's cart items. It defines criteria for healthy and organic items, updates the user's point balance if the criteria are met, and sends a notification to inform the user of their earned bonus points.)

Conclusion

In a nutshell, the checkout.php section proves the importance of a secure session management system, dynamic data retrieval, and attractive rewards pool can enhance an e-commerce experience. Through utilising both PHP and PDO for smooth database interactions, this component improves user satisfaction while also reinforcing data integrity and system security alongside client-side updates for real-time cart management. Understanding accessible designs and implementation of said features not only demonstrates my commitment to continuous improvement of my skills in this subject but also promoting good health and well-being for aligning my technical experience with society's social impact.

(Total word count (excluding picture captions): 966 words)

(Overall total word count (excluding picture captions): 1466 words)