SCHOOL OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTING AND INFORMATION SYSTEMS

WEB2202 WEB PROGRAMMING

ACADEMIC SESSION: AUGUST 2022

FINAL ASSESSMENT: WEB PROJECT **DUE DATE: 25 NOVEMBER 2022**

STUDENT NAME : **Lee Kah Hoong**

STUDENT ID : **21072996**

PROGRAMME : **Bachelor of Software Engineering (Hons)**

YEAR / SEMESTER : **Year 2 / Semester 4**

INSTRUCTIONS TO CANDIDATES

- This is an **Individual/Group** project.
- The **Web Project** and **Presentation** will contribute **50%** to the Final Assessment mark.
- The **Web Proposal** and **Web Report** will contribute **50%** of the Coursework mark.

IMPORTANT NOTICE

The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to all late submission of work. Project submitted after the deadline will be regarded as a non-submission and marked zero.

Academic Honesty Acknowledgement

"I Lee Kah Hoong 21072996 (student name / ID) 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 undergraduate programme student handbook)* for any kind of copying or collaboration on any assignment."

Lee. 24th November 2022 (Student's signature / Date)

Introduction, an overview of the project and rationale behind the creation of the website and the requirements.

In this project, we developed an e-commerce website using HTML, CSS, PHP, JavaScript and XAMPP web server to access MySQL Database and interpret PHP. The name of this website is Green Stuff. Green Stuff is a company that sells recycled products and it is inspired by the Sustainable Development Goals of the United Nations. While the aim of creating this website is to expand the marketplace of company so that sales will be maintained and even improved especially during this Covid-19 pandemic where everyone switches from in-store purchasing to online purchasing to avoid too much of contacts. Apart from that, it also promotes environmental awareness and allows more exposure to the importance of recycling in society. This website requires to allow customers to view and know more about the company like the latest products, the story behind them, store locations, contact methods, all products, and the price. It also allows customers to shop online. Customers can sign in or sign up for their account and add certain products they are interested in into their cart, while the subtotal and quantity added will be shown on the cart page. Customers are also allowed to remove the product they added to the cart or update the quantity they want on the cart page. On the other hand, the website should require to allow the admin to log in and edit or delete the product or user registered on the website. Security features like logging in with a highly secured password and session timeout are required too.

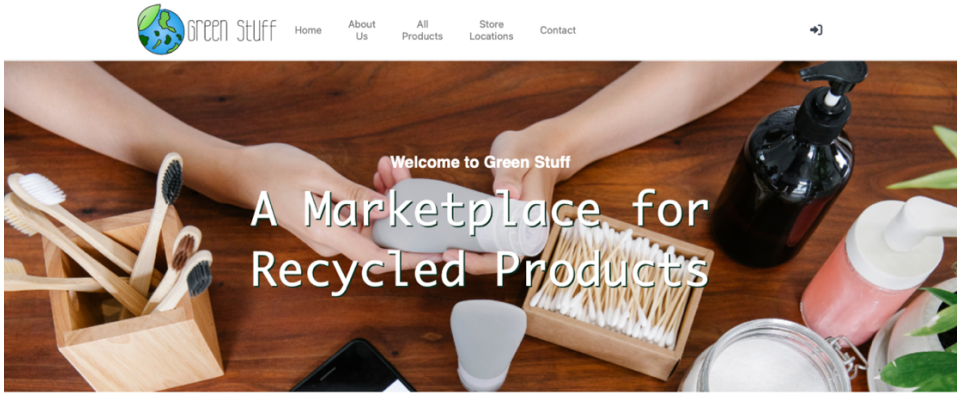


Figure 1: Home page with a welcome message

Figure 2: What's trending on the home page

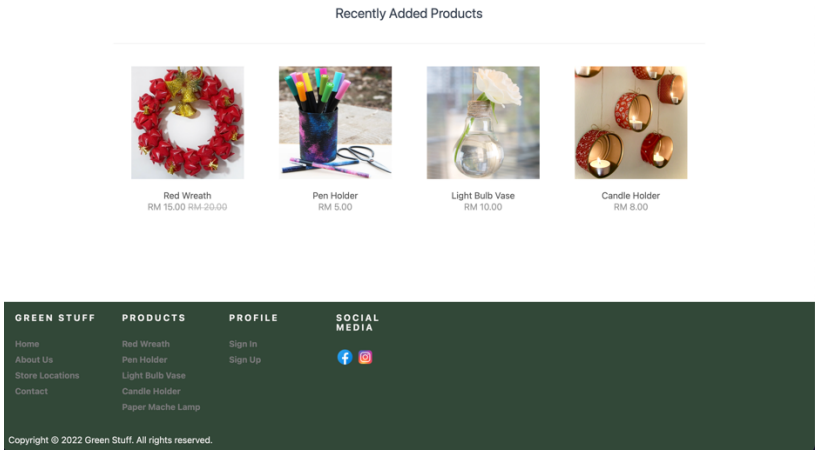


Figure 3: Recently added products on the home page

By referring to figure 1, on the main page of the website there will be a welcome message and short intro of the company, followed by an interesting video about recycling under the "What's Trending?" heading, shown in figure 2, and recently added products that show the latest product uploaded by the admin, shown by figure 3. In addition, on the home and product pages, users can check out the product by simply clicking on the title or picture to access the respective product page for more information. By referring to figures 1 and 3, on every page, there will be a header which includes the navigation bar, and also a footer that comes with links that can bring customers to the content pages including those popular products, sign in, sign up, etc. There are also two dummy icons to navigate customers to the social media page of

the company in the footer. While on the bottom of the footer, there should be a copyright message which will appear on every page of the website.

Next, on the all products page, customers will be able to see the amount and a small preview of products we selling shown in figure 4. By clicking the image or the title of the product, they will be directed to the details of the product. For example, by referring to figure 5, each product page will show the description and highlights of the product which include the materials used and the size of the product. Users can easily choose the quantity they want and then add it to the cart.

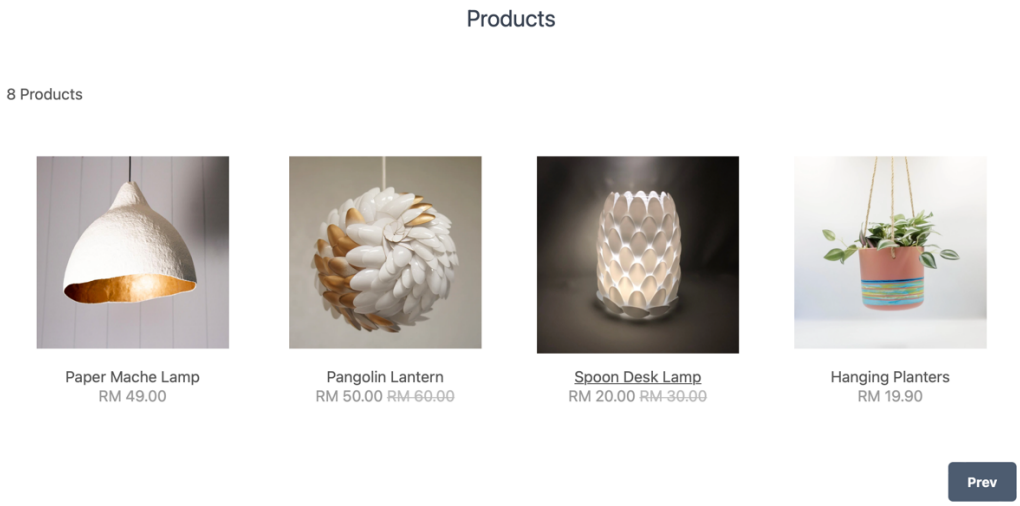


Figure 4: All products page

Figure 5: Product details page

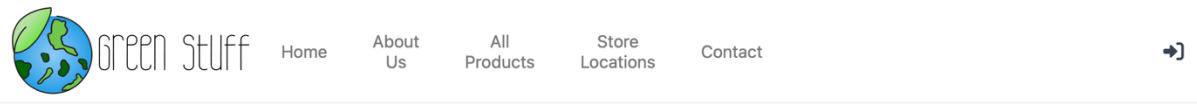


Figure 6: Navigation bar

In addition, customers can check the company's details by clicking the name of the page they wish to visit on the navigation bar like about us, products, store locations, contact and view cart/sign in / sign up pages shown in figure 6. Furthermore, on the about us page, there will be a video of an advertisement about Green stuff which is shown in figure 7. There will also be a story about the company under the video for customers to know more about it.

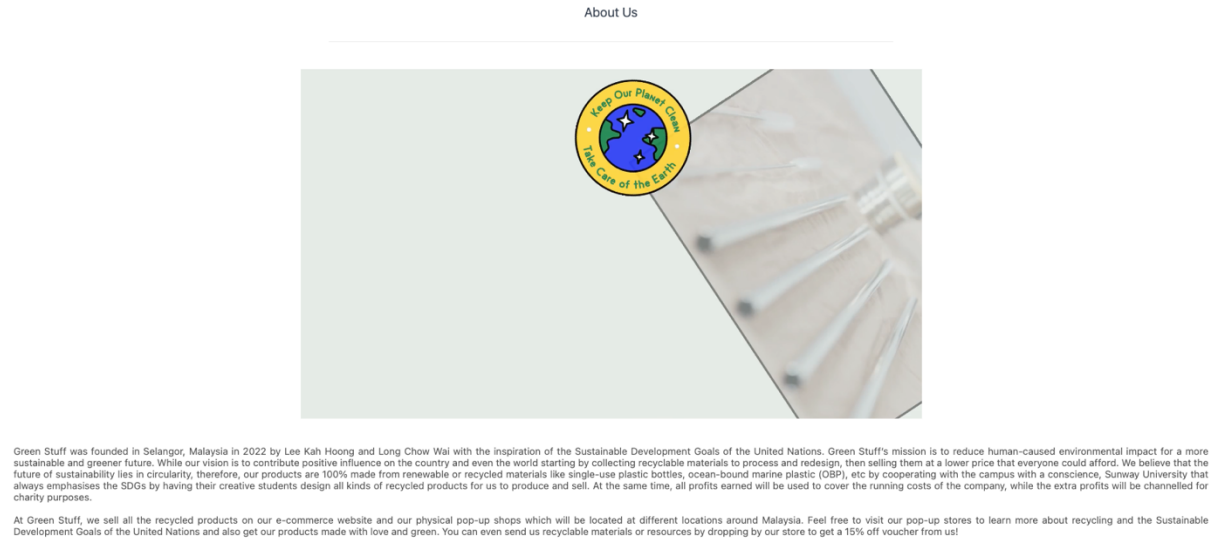


Figure 7: About us page

Next, customers can easily locate us by visiting the Store Locations page which includes the address and location of the branches, shown in figure 8, so that they can donate recyclable stuff they are not using to us or even purchase our products physically.

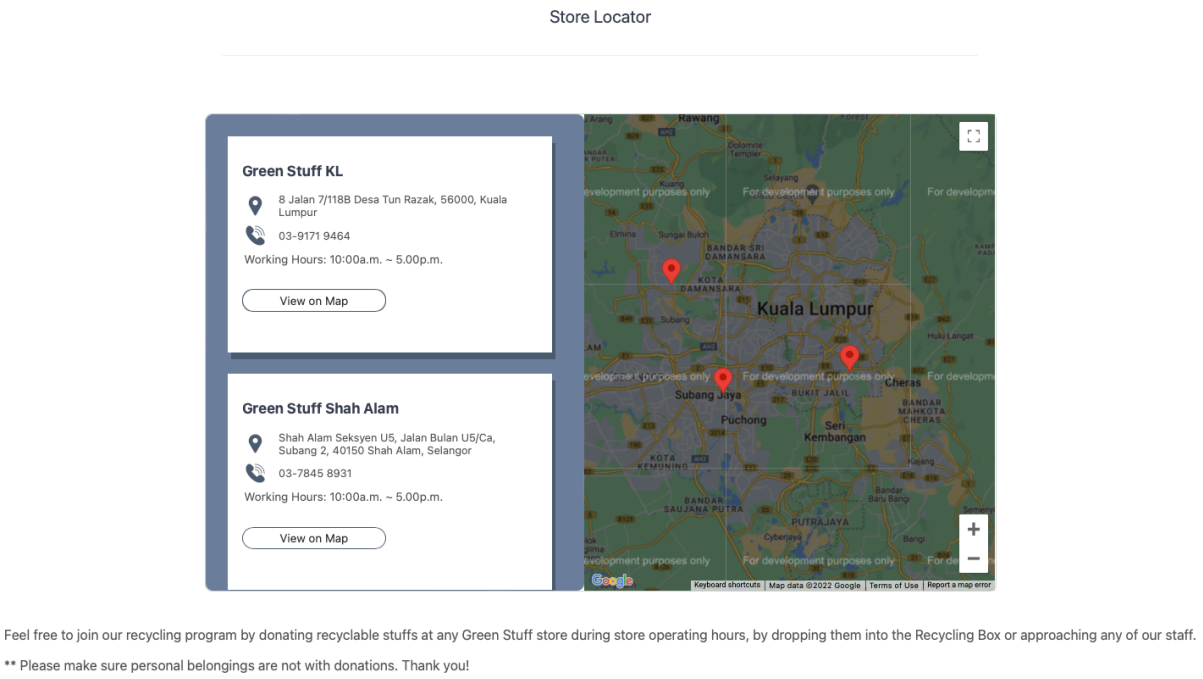


Figure 8: Store locations page

On the other hand, when customers have any enquiries, they can simply enter their details and message under the contact page or even contact us directly via the contact method we provided for further assistance, shown in figure 9.

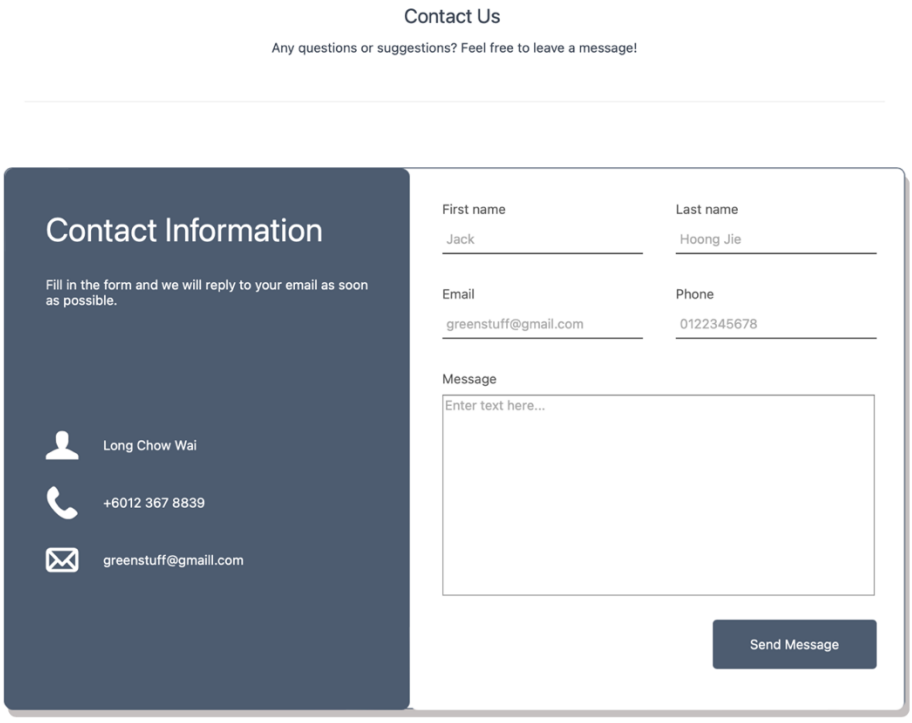


Figure 9: Contact page

Lastly, there are three types of interface on our website, which are guest mode, admin mode and member mode. Each of them provides a different function. In guest mode, the user only can view the website and products, they have to log in as member mode to add the product to the cart and place an order. Users will be prompted to sign in by entering their username and password on the login page as figure 10 shown, while those that don't have an account will be prompted to sign up for an account by filling in their information on the signup page as figure 11 shown.

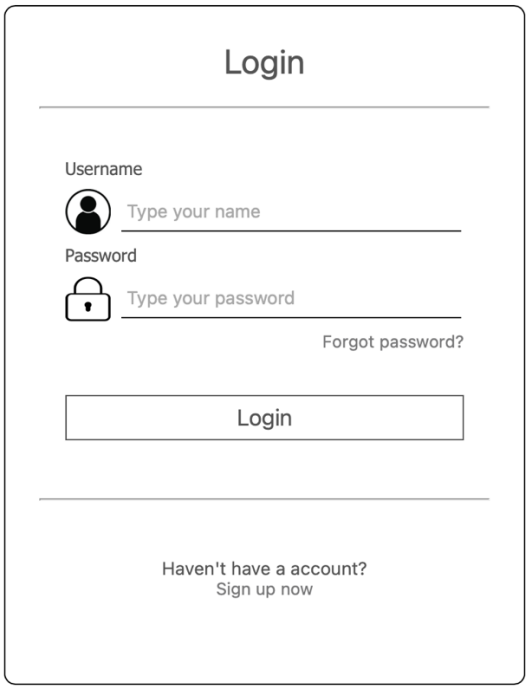


Figure 10: Login page

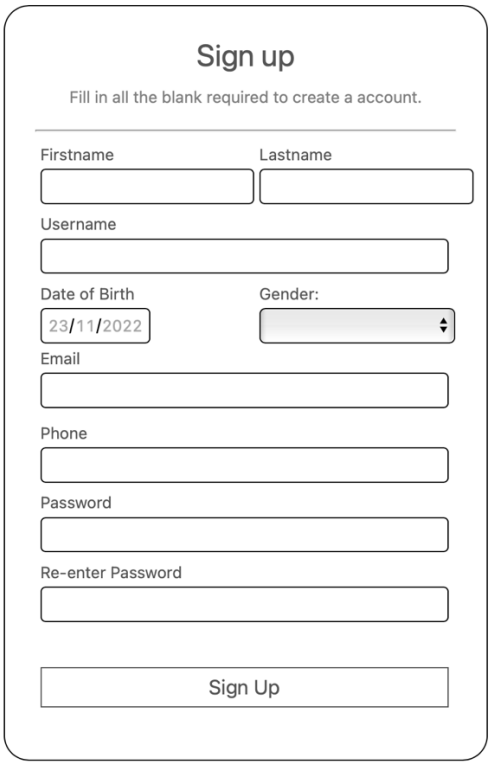


Figure 11: Sign-up page

By referring to figures 12, 13 and 14, for the admin mode, after the default admin has logged in, there are three extra functions available which include editing, add or remove customers or products and also add new admin account.

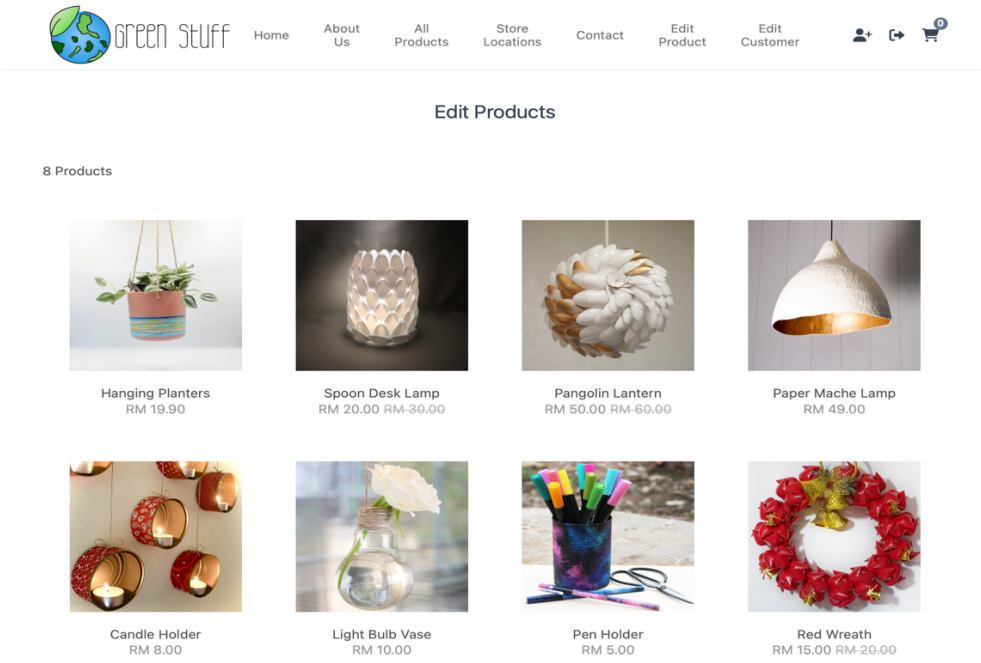


Figure 12: Edit products page

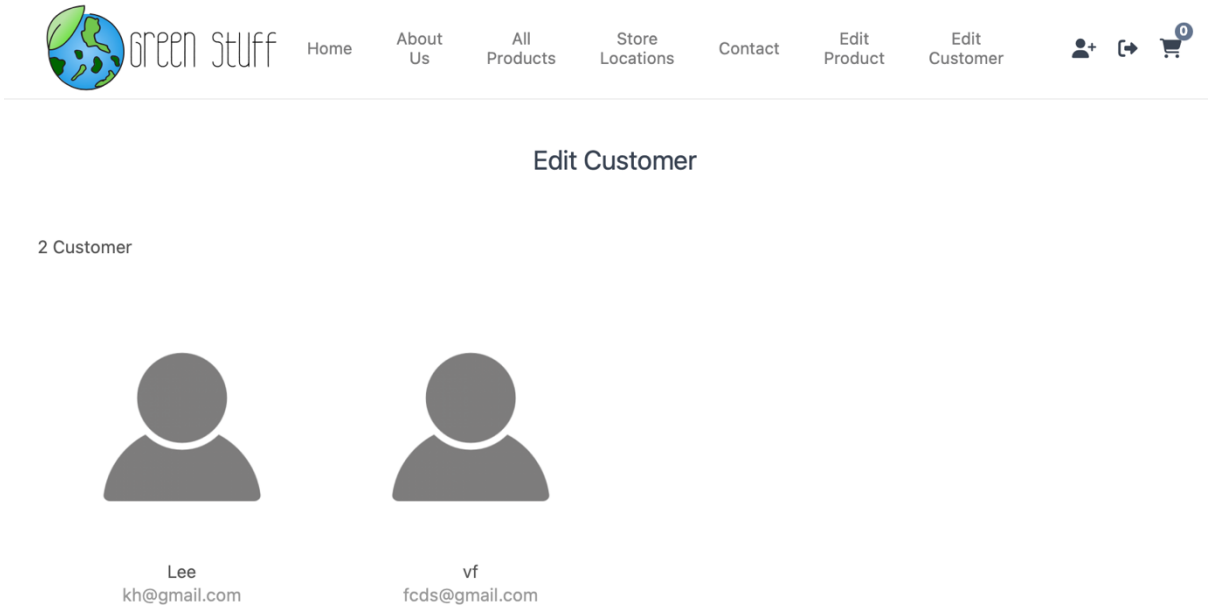


Figure 13: Edit customer page

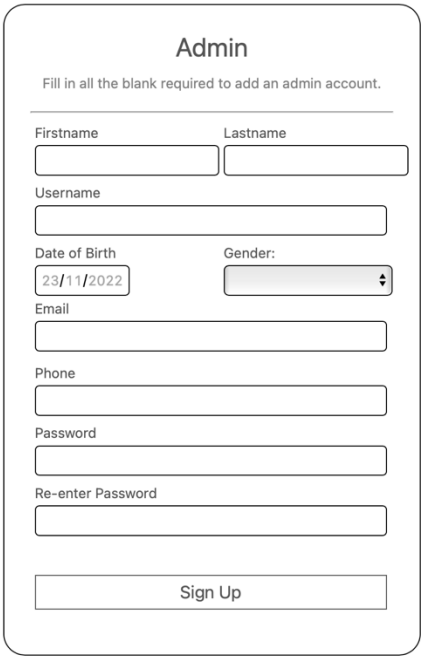


Figure 14: Add new admin page

After the user is in either admin or member mode, the session will be started. The session will be ended if the user logs out of their account or is inactive for more than five minutes. For example, if the user leaves the website for more than five minutes without touching anything, he or she will be logged out automatically and an alert box will pop out notifying the user that the session is expired and bring the user to the login page after closing the alert box, shown in figure 15.

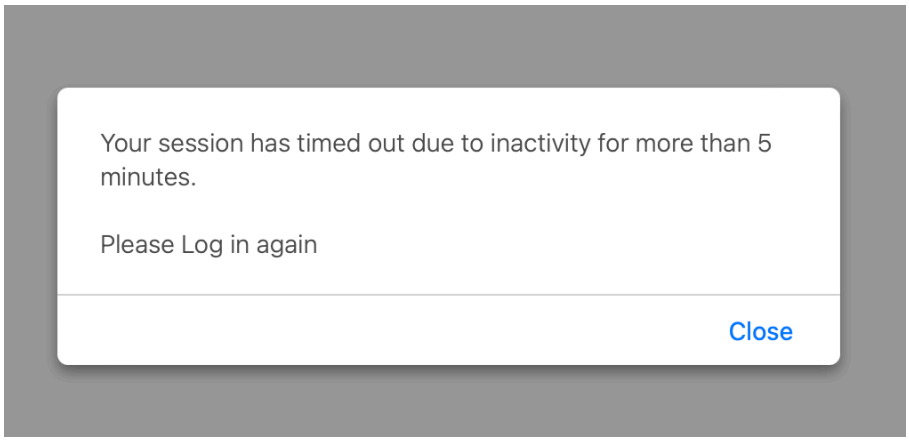


Figure 15: Session expired

On the other hand, by referring to figure 16, when the user is logged in, users will be able to see the quantity and products they added to the cart on the cart page. The total amount will also be shown on the page. Customers are allowed to update the quantity of product they wish to buy and even remove it from the cart by adjusting the quantity and clicking “Update” or just clicking remove below the product name. Then, by clicking “Place Order”, the order will be processed and the cart will be emptied. Conversely, if it is in guest mode, the user will be prompted to log in first before adding any products to the cart.

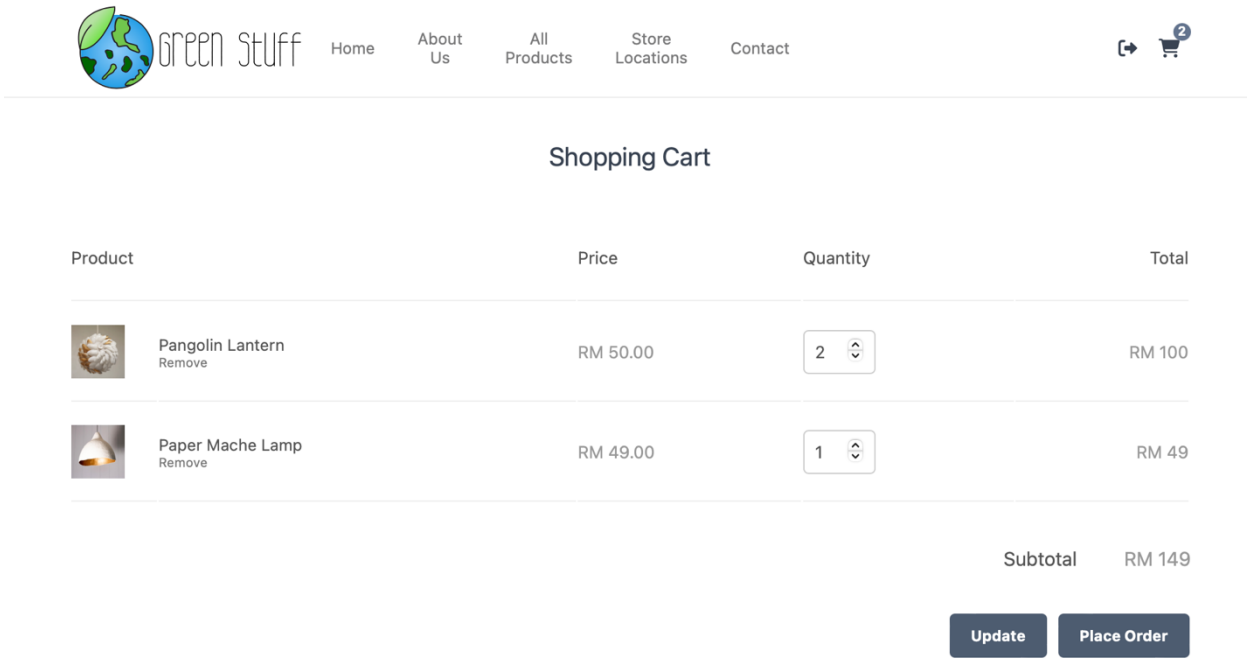


Figure 16: Cart page

Implementation of PHP functions and justification of the approaches

In this assignment, we applied HTML, CSS, PHP, JavaScript, and MySQL Database to the website to develop a user-friendly website so that it is attractive and easy for the user to use. The storyboard of the website started with gathering information, sketching the website interface, creating a database, implementing codes and product details, testing, enhancing, and upgrading.

First of all, after we are done gathering information and sketching the website interface, we started developing the website by creating basic PHP files which includes the index, home, functions, product, cart, and CSS files. Then, we created a new database called “web” with four tables inside to store product, admin and customer details and customer feedback using XAMPP to access to phpMyAdmin. After that, we inserted the products we sell into the “products” table with the details of their id, name, description, price, retail pricing and date added, shown in figure 17.

```
INSERT INTO products VALUES(5, 'Paper Mache Lamp', '<p>Pendant light, made of old newspaper.</p>
<h3>Highlights</h3>
<ul>
<li>Materials: old newspaper, glue, gold spray paint, led light bulb</li>
<li>Diameter: 42 centimetres</li>
<li>Height: 39 centimetres</li>
<li>Light Bulb: Requires LED bulb with E27 ending that is maximum 20W</li>
</ul>', '49.00','0.00',9,'1newslamp.jpeg','2022-10-13 16:36:14');
```

Figure 17: Inserting products into the database table

After we are done with the database, we first started editing the function.php file to enable the database connection function and also include the header and footer of each mode of the website. To enable the database connection function, we used the pdo_connect_mysql() function with the database details we used when creating the database, shown in figure 18. Of course, if there is an error with the connection, the script will stop with an error message using the catch (PDOException $exception) function.

```php
function pdo_connect_mysql(){
    $DATABASE_HOST = 'localhost';
    $DATABASE_USER = 'root';
    $DATABASE_PASS = '';
    $DATABASE_NAME = 'web';

    try{
        return new PDO('mysql:host=' . $DATABASE_HOST . ';dbname=' . $DATABASE_NAME . ';charset=utf8', $DATABASE_USER, $DATABASE_PASS);
    } catch (PDOException $exception) {
        exit('Unable to connect to database!');
    }
}
```

Figure 18: Database connection function

After that, we created the header and footer function so that every page of the website will display the same header and footer by using the function instead of repeating the layout code in every file. We can also easily change the layout by just editing this function file. The function name we used for the header and footer are function h_header($title) and function h_footer(), therefore on other pages, we will just have to execute the function name to apply the layout to the page. For example, the home page header, <?=h_header('Home')?>. Moreover, we added a more user-friendly design which is displaying the number of items in the customer's cart on the cart icon in the header by using the count($_SESSION['cart']) command to count the number of items in the cart and tag to display it on the icon, shown in figure 19 and 20.

```php
function h_header($title) {
    $num_items_in_cart = isset($_SESSION['cart']) ? count($_SESSION['cart']) : 0;
```

Figure 19: Implementation of header function and count($_SESSION['cart']) command

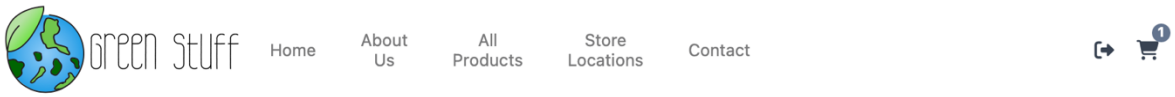


Figure 20: Cart Icon with the number of items in the cart

As we have three different modes of interface which are the guest, member and admin modes, so we separate the session using the $_SESSION['ID'] variable. The ID indicates a different type of user, 1 refers to an admin while 2 refers to a member as Figures 21 and 22 shown. For example, if the isset() function checks the "ID" variable matches 1 and so does the "ID", the specific code for admin mode will be executed which enables the function of editing product or customer, and adding admin to be

accessed by the user. While for member mode, they will be able to add products to the cart.

```
if(isset($_SESSION['ID']) && $_SESSION['ID']==1){
```

Figure 21: If statement when the user is an admin

```
if(isset($_SESSION['ID']) && $_SESSION['ID']==2){
```

Figure 22: If statement when the user is a member

Moving on to the index file shown in figure 23. It is the main file to access all the other pages so we set up basic routing and use GET requests to determine which page is which. We started the session using the “session_start” function to store the products that users adding to the cart. Then, we included the functions file to enable the database connection to MySQL using a PDO connection.

```
<?php
    session_start();
    include 'functions.php';
    $pdo = pdo_connect_mysql();
    $page = isset($_GET['page']) && file_exists($_GET['page'] . '.php') ? $_GET['page'] : 'home';
    include $page . '.php';
?>
```

Figure 23: Index file

Following by the basic routing method used to check if the GET request variable ($_GET['page']) exists. If not, the home page will be the default page, else it will be the requested page. For example, if users want to access the product page, they can simply navigate to http://localhost/webpro/index.php?page=products, conversely, if they navigate http://localhost/webpro/index.php, they will automatically be brought to the home page as default.

Moving on to the home page, For this page, we included a welcome message, an interesting video, and recently added products to have a better impression of our website during the customer’s first visit as it is the first-page customer will see. By using PDO’s prepared and execute statement shown in figure 24, an SQL query will be prepared to retrieve the four most recently added products that are ordered by the

date added to be shown on the home page, shown in figure 26. The result will then store as the $recently_added_products variable as an associated array. Then we used a PHP foreach loop, shown in figure 25 to loop the code for each product in the array to display the four recently added products, while if and endif are used to display the price of the product that is on promotion like Red Wreath in figure 26.

```php
<?php
include 'logtimeout.php';
$stmt = $pdo->prepare('SELECT * FROM products ORDER BY date_added DESC LIMIT 4');
$stmt->execute();
$recently_added_products = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>
```

Figure 24: PDO prepared and execute statement

```php
<?php foreach ($recently_added_products as $product): ?>
<a href="index.php?page=product&id=<?=$product['id']?>" class="product">
    <img src="imgs/<?=$product['img']?>" width="200" height="200" alt="<?=$product['name']?>">
    <span class="name"><?=$product['name']?></span>
    <span class="price">
        RM <?=$product['price']?>
        <?php if ($product['rrp'] > 0): ?>
        <span class="rrp">RM <?=$product['rrp']?></span>
        <?php endif; ?>
    </span>
</a>
<?php endforeach; ?>
```

Figure 25: PHP foreach loop & if, endif

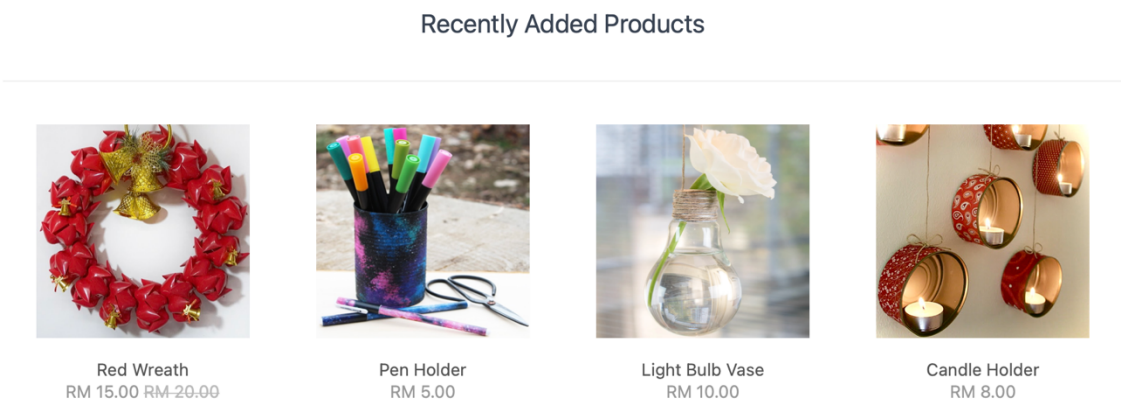


Figure 26: Recently added product

After that, we inserted the header and footer function to include the navigation bar and footer in the home page so that users can easily access other pages by executing the

function name, <?=h_header('Home')?> and <?=h_footer('Home')?> before and after the HTML code for the home page layout, shown in figure 27 and 28.

```
9   <?=h_header('Home')?>
10
11  <div class="featured">
12      <h2 style="font-family:Helvetica;">Welcome to Green Stuff</h2>
13      <p style="font-family:'Monaco';text-shadow: 2px 2px #023020">A Marketplace for Recycled Products</p>
```

Figure 27: Executing header function

```
58  </div>
59
60  <?=h_footer()?>
61
```

Figure 28: Executing footer function

Furthermore, by referring to figures 29 and 31, for the all products page, we displayed four products once and users can simply click “Next” for more products. We set the number of products to be shown on each page as four using the variable $num_products_on_each_page and order the products according to the date added using PDO prepared statement. To determine which page the user is on, we applied a GET request, in the URL this will appear as index.PHP?page=products&p=1, and in our PHP script the parameter *p* we can retrieve with the $_GET['p'] variable. Assuming the request is valid, the code will execute a query that will retrieve the limited products from our database.

```
$num_products_on_each_page = 4;
$current_page = isset($_GET['p']) && is_numeric($_GET['p']) ? (int)$_GET['p'] : 1;
$stmt = $pdo->prepare('SELECT * FROM products ORDER BY date_added DESC LIMIT ?,?');
```

Figure 29: All products page

Then, we also used the bindValue statement and PDO fetches all statements so that we can use an integer in the SQL statement to limit the number of products shown and fetch the products from the database and return the result as an array as figure 30 shown.

```php
$stmt->bindValue(1, ($current_page - 1) * $num_products_on_each_page, PDO::PARAM_INT);
$stmt->bindValue(2, $num_products_on_each_page, PDO::PARAM_INT);
$stmt->execute();
$products = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Figure 30: bindValue statement & PDO fetch all statement

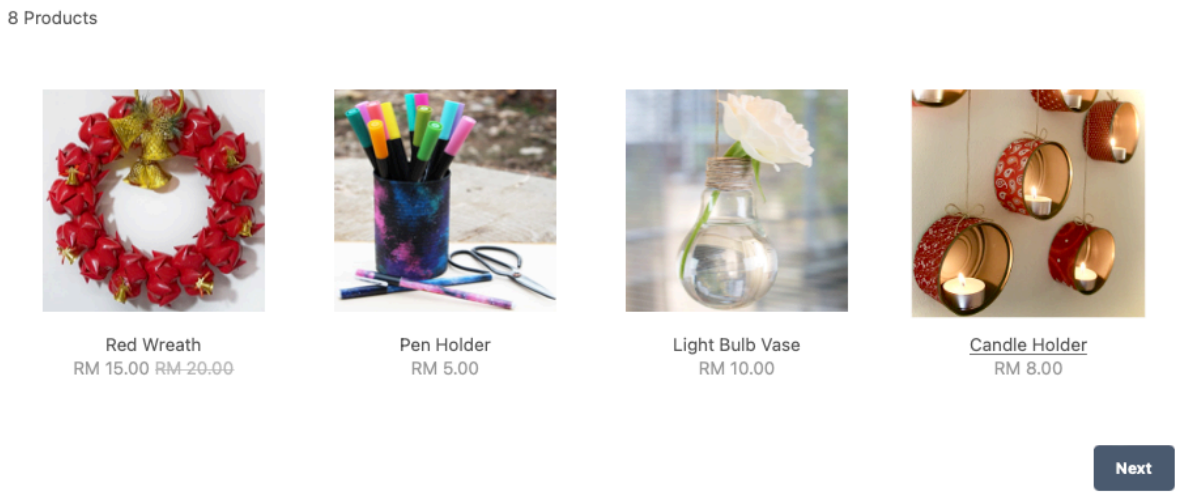


Figure 31: All products page

Apart from that, we applied PDO rowCount statement as figure 32 shown, to get the total number of products in our products table so that users can see the total number of products on the top of the products page as figure 31 shown.

```php
$total_products = $pdo->query('SELECT * FROM products')->rowCount();
```

Figure 32: PDO rowCount statement

Continuing with the layout HTML code of the all products page, we implemented a PHP foreach loop, shown in figure 33 to loop the code for each product in the array to display the four recently added products, while if and endif are used to display the price of the product that is on promotion like Red Wreath in figure 31.

```php
<?php foreach ($products as $product): ?>
<a href="index.php?page=product&id=<?=$product['id']?>" class="product">
    <img src="imgs/<?=$product['img']?>" width="200" height="200" alt="<?=$product['name']?>">
    <span class="name"><?=$product['name']?></span>
    <span class="price">
        RM <?=$product['price']?>
        <?php if ($product['rrp'] > 0): ?>
        <span class="rrp">RM <?=$product['rrp']?></span>
        <?php endif; ?>
    </span>
</a>
<?php endforeach; ?>
```

Figure 33: PHP foreach loop & if, endif for product display

At the same time, we applied another PHP if and endif to the other set of products display when a user clicked “Next”, shown in figure 34. The “Next” and “Prev” buttons will only be visible to the user if the total number of products is greater than the $num_products_on_each_page variable. While the page will be directed to the respected page due to the link set for the anchor tag like directing to the second page of the product page, index.PHP?page=products&p=2 page shown in figure 35 after clicking Next.

```php
<?php if ($current_page*$num_products_on_each_page >= $total_products): ?>
<a href="index.php?page=products&p=<?=$current_page-1?>">Prev</a>
<?php endif; ?>
<?php if ($current_page == 1): ?>
<a href="index.php?page=products&p=<?=$current_page+1?>">Next</a>
<?php endif; ?>
<?php if (($current_page != 1) && ($current_page*$num_products_on_each_page < $total_products)): ?>
<a href="index.php?page=products&p=<?=$current_page-1?>">Prev</a>
<a href="index.php?page=products&p=<?=$current_page+1?>">Next</a>
<?php endif; ?>
```

Figure 34: PHP if, endif for more products

Paper Mache Lamp
RM 49.00

Pangolin Lantern
RM 50.00 ~~RM 60.00~~

Spoon Desk Lamp
RM 20.00 ~~RM 30.00~~

Hanging Planters
RM 19.90

Prev

Figure 35: Second page of all products page

Moreover, the detailed product page displays all details for a specified product that is determined by the GET request ID variable. Users can also view the price, image, and description, and even able to change the quantity and add to the cart with a click of a button as figure 36 shown.

Figure 36: Detailed product page

By referring to figure 37, the GET request ID variable will check if the requested id variable exists. If it exists, the code will proceed to retrieve the product from the products table in the “web” database and return the result as an array using the PDO prepared, execute, and fetched all statements. At the same time, if the id for the product doesn't exist or wasn’t specified, an error message will be displayed using the if-else statement, while the exit() function will prevent further script execution and display the error, shown in figure 38.

```php
if (isset($_GET['id'])) {
    $stmt = $pdo->prepare('SELECT * FROM products WHERE id = ?');
    $stmt->execute([$_GET['id']]);
    $product = $stmt->fetch(PDO::FETCH_ASSOC);
```

Figure 37: Retrieve product from the database

```
if (!$product) {
    exit('Product does not exist!');
}
else {
    exit('Product does not exist!');
```

Figure 38: Id checking

Continuing with the layout code of the detailed product page, a form is created and the action attribute is set to the shopping cart page, index.php?page=cart along with the method set to post, shown in figure 39, so that the cart page will add the product to cart when a user clicked “Add To Cart”. On the other hand, by referring to figure 40, a maximum value is set to the product's quantity retrieved from the products table so that users will not add an amount that is larger than our stock to the cart, while the product id is also added to the form so that we will know which product the customer added to the cart. On the other hand, the product's name, description, etc shown in figure 36 will be taken from the products table in the database with the product ID, using the product variable from the products.php page shown in figure 41.

```
<form method="post"  action="index.php?page=cart">
```

Figure 39: Form with the action attribute

```
<input type="number" name="quantity" value="1" min="1" max="<?=$product['qty']?>" placeholder="Quantity" required>
<input type="hidden" name="product_id" value="<?=$product['id']?>">
```

Figure 40: Quantity & product id of the product

```
<div class="description">
    <?=$product['desc']?>
</div>
```

Figure 41: Description of product

Lastly, we also added a login warning function that will be triggered when they clicked on the “Add To Cart” button under $_SESSION[‘ID’], which is the guest mode shown in figure 42, to limit the guest user by prompting them to sign in before they can add products to the cart, shown in figure 43 and 44. When the user clicked on close, they

will be directed to the login page due to the location replace() method shown in figure 43 line 9.

```php
<?php if(!empty($_SESSION['ID'])){?>
<input type="submit" value="Add To Cart">
<?php }else{?>
<input type="button" value="Add To Cart"  class="addtocart" style="text-align:center;" onclick="return loginwarning();" >
<?php }?>
```

Figure 42: Add to cart in guest mode

```
5       function loginwarning() {
6           if(!confirm("Please login to add the item to cart")) {
7           return false;
8            }
9           window.location.replace("index.php?page=Login");
```

Figure 43: Login warning message & location replace() method

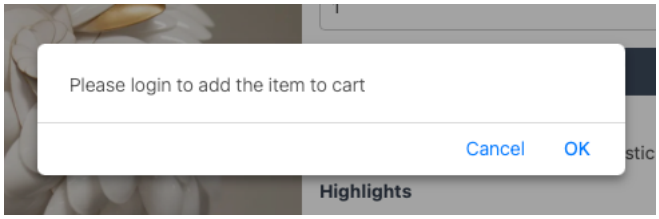


Figure 44: Login warning message

Moving on to the cart page, we use PHP sessions to remember the shopping cart products, for example, when a customer navigates to another page etc, the shopping cart will still contain the products previously added until the session expires. An isset() function is used when a user clicked the “Add To Cart” button on the product details page, to check the product id and quantity from the form data before adding to the cart, shown in figure 45. Then, we set the post variable of product id and quantity so that it is easier to identify them and also convert them into an integer, shown in figure 46. Continuing with using PDO prepared, execute statement, to prepare the SQL statement and check if the product exists in the database at the same time using the product id, shown in figure 47. Then only fetch the product from the database and return the result as an Array using the PDO fetch statement, shown in figure 48. These are to check the form values from the product.php file to ensure the product exists so that the users will not be adding non-existent products to the cart.

```php
if (isset($_POST['product_id'], $_POST['quantity']) && is_numeric($_POST['product_id']) && is_numeric($_POST['quantity'])) {
```

Figure 45: Check product id & quantity

```php
$product_id = (int)$_POST['product_id'];
$quantity = (int)$_POST['quantity'];
```

Figure 46: Set post variable

```php
$stmt = $pdo->prepare('SELECT * FROM products WHERE id = ?');
$stmt->execute([$_POST['product_id']]);
```

Figure 47: Check the product existence

```php
$product = $stmt->fetch(PDO::FETCH_ASSOC);
```

Figure 48: PDO fetch statement

Following by checking if the product was added to the cart using the if else statement shown in figure 49. If the product variable and quantity are bigger than zero shown in line 15, figure 49, means that the product exists in the database and therefore it can be proceeded to create or update the session variable for the cart. After the product id is checked through the PHP array_key_exists() function shown in figure 49, line 17 and it exists in the cart, it will just update the quantity in the cart shown in figure 49, line 19, else if the product is not in the cart, it will be added as line 22. While if there is no product in the cart, this will add the first product to the array and cart shown in line 25.

```php
15      if ($product && $quantity > 0) {
16          if (isset($_SESSION['cart']) && is_array($_SESSION['cart'])) {
17              if (array_key_exists($product_id, $_SESSION['cart'])) {
18
19                  $_SESSION['cart'][$product_id] += $quantity;
20
21              } else {
22                  $_SESSION['cart'][$product_id] = $quantity;
23              }
24          } else {
25              $_SESSION['cart'] = array($product_id => $quantity);
26          }
27      }
```

Figure 49: If else statement

The session variable cart will be an associated array of products, and with this array, users can add multiple products to the shopping cart. The array key will be the product id and the value will be the quantity. To prevent resubmission, the header location is applied shown in figure 50 which will redirect the page to the cart page.

```php
header('location: index.php?page=cart');
exit;
```

Figure 50: Header location

Apart from that, users are allowed to remove a product from the cart. By referring to figure 51, when the “remove” button is clicked, a GET request is used to determine which product to remove, for example, if the id of the product is 1, the following URL will remove it from the shopping cart, index.php?page=cart&remove=1 through unset function.

```php
if (isset($_GET['remove']) && is_numeric($_GET['remove']) && isset($_SESSION['cart']) && isset($_SESSION['cart'][$_GET['remove']])) {
    unset($_SESSION['cart'][$_GET['remove']]);
}
```

Figure 51: Remove the product from the cart

Furthermore, for the update product quantity function, when the user clicks the "Update" button on the shopping cart page, it will loop through the post data to update the quantities for every product in the cart using the foreach loop shown in figure 52.

```php
foreach ($_POST as $k => $v) {
    if (strpos($k, 'quantity') !== false && is_numeric($v)) {
        $id = str_replace('quantity-', '', $k);
        $quantity = (int)$v;
        if (is_numeric($id) && isset($_SESSION['cart'][$id]) && $quantity > 0) {
            $_SESSION['cart'][$id] = $quantity;
        }
    }
}
```

Figure 52: Foreach loop

Moreover, to place an order, the cart will be checked if it is empty shown in figure 53, if it is not empty, the cart will be emptied using the unset function and the user will be directed to the place order page as figure 54 shown.

```php
if (isset($_POST['placeorder']) && isset($_SESSION['cart']) && !empty($_SESSION['cart'])) {
    unset($_SESSION["cart"]);
    header('Location: index.php?page=placeorder');
    exit;
}
```

Figure 53: Place order

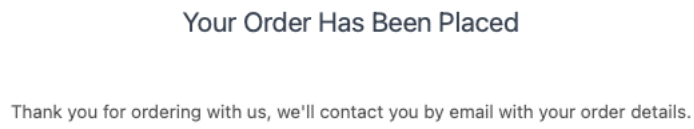


Figure 54: Place order page

At the same time, the session variable for products in the cart is checked as figure 55 shown. When there are products in the cart, those products will be retrieved from the products table in the database using the PDO prepared statement through the array_to_question_marks variable together with the product name, description, image, and price as the information did not store in the session variable before. Then we applied the execute statement for the array keys which are the id of the products followed by PDO fetch all statements to fetch products from the database as an Array. While the subtotal will be calculated by iterating the products and multiplying the price by the quantity using a foreach loop to notify users before they check out, shown in figure 56.

```php
$products_in_cart = isset($_SESSION['cart']) ? $_SESSION['cart'] : array();
$products = array();
$subtotal = 0.00;

if ($products_in_cart) {
    $array_to_question_marks = implode(',', array_fill(0, count($products_in_cart), '?'));
    $stmt = $pdo->prepare('SELECT * FROM products WHERE id IN (' . $array_to_question_marks . ')');
    $stmt->execute(array_keys($products_in_cart));
    $products = $stmt->fetchAll(PDO::FETCH_ASSOC);
    foreach ($products as $product) {
        $subtotal += (float)$product['price'] * (int)$products_in_cart[$product['id']];
    }
}
```

Figure 55: Check the session variable and calculate subtotal

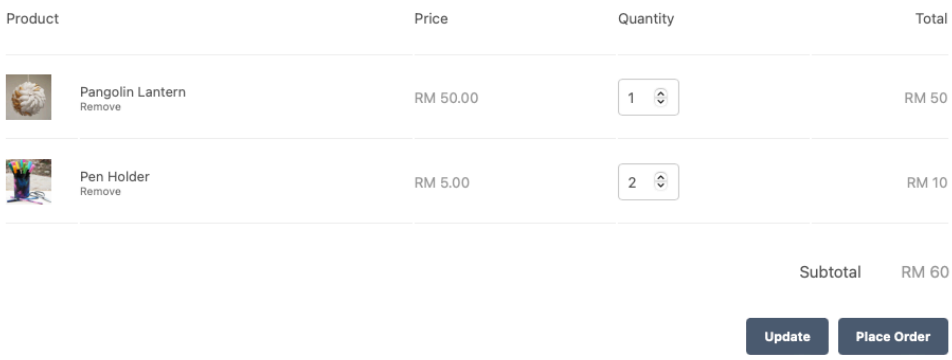


Figure 56: Cart page

Besides that, to access the admin mode, a default admin was created first in the database so that he or she can add another new admin. The admin can log in on the login page with the username and password registered in the database through the login page shown in figure 57. This login page also allows customers to log in as a member. When there is a user log in, their details will be verified through the code shown in figure 58 to check if the details match with details stored in the customer and admin table in the database. If the user does not exist, it will return a “User not found” message. While if the user exists in the customers' table, the details will be matched with the details fetched from the database through the password_verify() function shown in figures 59 & 60, if it matches the details, the session will be started and the user will be brought to the home page by the header location function. At the same time, the $_SESSION[‘ID’] will be set to 2 if the user is a customer which indicates it is in customer mode, while 1 indicates it is in admin mode as I mentioned earlier.

Figure 57: Login page

```php
if(isset($_POST['username']) && isset($_POST['password'])){

  $un_temp = sanitise($pdo,$_POST['username']);
  $pw_temp = sanitise($pdo,$_POST['password']);
  $query   = "SELECT * FROM cust_info WHERE Username=$un_temp";
  $result  = $pdo->query($query);

  if (!$result->rowCount()) {
    $query   = "SELECT * FROM `admin_info` WHERE Username=$un_temp";
    $result  = $pdo->query($query);
    if (!$result->rowCount()) {
    echo "*User not found";
    }
    else{
      $row = $result->fetch();
      $fn  = $row['First_name'];
      $ln  = $row['Last_name'];
      $un  = $row['Username'];
      $id = $row['AdminID'];
      $pw  = $row['Pwd'];
```

Figure 58: Validate the account

```php
else{
  $row = $result->fetch();
  $fn  = $row['First_name'];
  $ln  = $row['Last_name'];
  $un  = $row['Username'];
  $id = $row['AdminID'];
  $pw  = $row['Pwd'];

  if (password_verify($pw_temp, $pw)){
    session_start();

    $_SESSION['First_name'] = $fn;
    $_SESSION['Username'] = $un;
    $_SESSION['AdminID'] = $id;
    $_SESSION['ID'] = 1;

    setcookie($un,$id,time()+3600);

    header("Location: index.php");
  }
  else echo("*Invalid username/password combination");
}
```

Figure 59: Verify admin password

```php
else{
  $row = $result->fetch();
  $fn  = $row['First_name'];
  $ln  = $row['Last_name'];
  $un  = $row['Username'];
  $id = $row['CustID'];
  $pw  = $row['Pwd'];

  if (password_verify($pw_temp, $pw)){
    session_start();

    $_SESSION['First_name'] = $fn;
    $_SESSION['Username'] = $un;
    $_SESSION['CustID'] = $id;
    $_SESSION['ID'] = 2;

    setcookie($un,$id,time()+3600);

    header("Location: index.php");
  }
  else echo("*Invalid username/password combination");
}
```

Figure 60: Verify customer password

If the user does not have an account, he or she will be able to sign up by clicking sign up now shown in figure 57. While for admin, the admin will have to register the new admin on the add admin page as shown in figure 61. The member signup PHP file and add admin PHP file, both contain the function to ensure there are no blanks in the form to prevent the missing user details as shown in figure 62. If there is a blank, an alert message will be displayed as shown in figure 63. When all the blanks are filled in, the details will go through data validation shown in figure 64 and function valid() shown in figure 65.

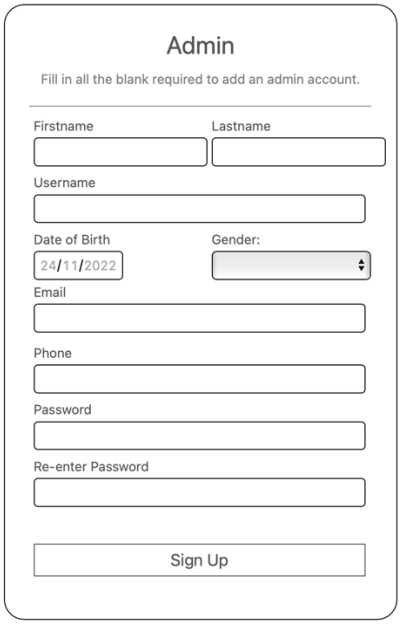


Figure 61: Add admin page

```
function NoEmptyForm() {
    var x = document.forms["Admin"]["FirstName"].value;
        if (x == "" || x == null) {
            alert("Firstname must be filled out");
            return false;
        }
    var x = document.forms["Admin"]["LastName"].value;
        if (x == "" || x == null) {
            alert("Lastname must be filled out");
            return false;
        }
    var x = document.forms["Admin"]["UserName"].value;
        if (x == "" || x == null) {
            alert("Username must be filled out");
            return false;
        }
    var x = document.forms["Admin"]["DateOfBirth"].value;
        if (x == "" || x == null) {
            alert("Date of Birth must be filled out");
            return false;
        }
    var x = document.forms["Admin"]["Gender"].value;
        if (x == "" || x == null) {
            alert("Gender must be filled out");
            return false;
        }
    var x = document.forms["Admin"]["Email"].value;
        if (x == "" || x == null) {
            alert("Email must be filled out");
            return false;
        }
    var x = document.forms["Admin"]["Phone"].value;
        if (x == "" || x == null) {
            alert("Phone must be filled out");
            return false;
        }
    var x = document.forms["Admin"]["Password"].value;
        if (x == "" || x == null) {
            alert("Password must be filled out");
            return false;
        }
    var x = document.forms["Admin"]["Reenterpassword"].value;
        if (x == "" || x == null) {
            alert("Re-enter Password must be filled out");
            return false;
```

Figure 62: Form checking

Figure 63: Alert for blank space

```
$valid = valid($_POST['FirstName'], "/^[A-Za-z ]+$/");
$valid .=valid($_POST['LastName'], "/^[A-Za-z]+$/" );
$valid .=valid($_POST['UserName'],"/^[A-Za-z\d]{2,10}/");
$valid .=valid($_POST['Email'],"/^[a-z0-9._%+-]+@[a-z]+\.[a-z]{2,}$/");
$valid .=valid($_POST['Phone'],"/^[0-9]+$/");
$valid .=valid($_POST['Password'], "/^(?=.*\W)(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,}$/");
```

Figure 64: Data validation

```
function valid($data,$data_pattern){
    if(!preg_match($data_pattern, $data)){
        return 1;
    }
```

Figure 65: Validation function

After validating all the details, the details will be compared with the user details in the database to ensure there is no repeated username, email or phone number as shown in figure 66. By using the PDO query statement, $stmt, information will be all fetched from the customers' table in the database, while $stmt1 will be checking with the admin table from the database. If both have no matches, the password will go through checking to ensure both passwords entered are the same, then the details will only be inserted into the table in the database which indicates sign up successfully as figure 66 shown. The same applies to admin registration.

```php
if($valid == ""){
    $stmt = $pdo->query('SELECT * FROM cust_info WHERE Username =\'. $myusername .'OR Email=\'. $email .'OR Phone=\'. $phone);
    $check = $stmt->fetchAll();
    $stmt1 = $pdo->query('SELECT * FROM admin_info WHERE Username =\'. $myusername .'OR Email=\'. $email .'OR Phone=\'. $phone);
    $check1 = $stmt1->fetchAll();
    if((count($check) == 0 ) && (count($check1)==0) && ($_POST['Password'] == $_POST['Reenterpassword'])){
        $query = "INSERT INTO $table (CustID,First_name,Last_name,Username,Dob,Gender,Email,Phone,Pwd) VALUES(NULL,$firstname,$lastname,$myusername,$dob,'$gende
        $result = $pdo->query($query);
        if(! $result){
            die('Error:' . mysqli_error());
        }
        header("location:index.php?page=Sucessfull");
    }
}
```

Figure 66: Check repeated user

On the other hand, if the details entered are in the wrong format, error messages will be shown in figure 67 due to the function datavalid() that will return the errormessage variable when the validation failed. This is because the datavalid() function is assigned to the variable, $validation, therefore when the format of details is wrong, the validation variable will not be empty as it contains the error message from datavalid(), which will echo the error message. Figure 69 shows an example of a validation variable echo error message, and it applies to other details like phone numbers too.

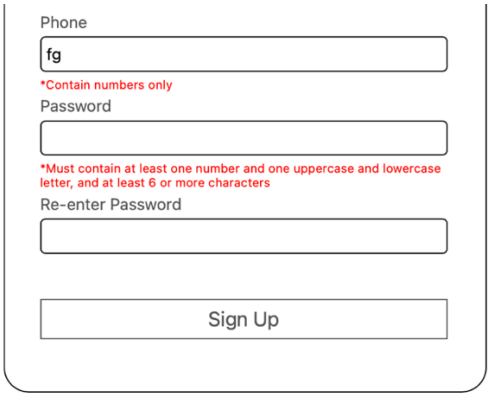


Figure 67: Wrong format

```php
function datavalid($data,$data_pattern,$errormessage){
    if(preg_match($data_pattern, $data)){
        return "";
    }
    else{
        return $errormessage;
    }
}
```

Figure 68: Data valid function

```php
if(isset($_POST['FirstName'])){
$validation = datavalid($_POST['FirstName'], "/^[A-Za-z ]+$/" , "*Contain letters and spacing only");
if($validation != ""){
echo $validation;
```

Figure 69: Validation variable

After successfully registering, the user will be directed to the successful page by the header location function which shows a successful message shown in Figures 70 & 71.

```php
header("location:index.php?page=Sucessfull");
```

Figure 70: header location function

You have successfully create a account. Please proceed to login page for login.

Figure 71: Sign up successful message

Moving on to edit and delete a product. This feature is only accessible by the admin to prevent unnecessary confusion. The PDO prepared, executes and fetches all statements that are used to get the products from the products table in the database to assign to the products variable shown in figure 72.

```php
$stmt = $pdo->prepare('SELECT * FROM products ORDER BY date_added');
$stmt->execute();
$products = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Figure 72: Fetch products

```php
<?php foreach ($products as $product): ?>
<a href="index.php?page=changeproduct&id=<?=$product['id']?>" class="product">
    <img src="imgs/<?=$product['img']?>" width="200" height="200" alt="<?=$product['name']?>">
    <span class="name"><?=$product['name']?></span>
    <span class="price">
        RM <?=$product['price']?>
        <?php if ($product['rrp'] > 0): ?>
        <span class="rrp">RM <?=$product['rrp']?></span>
        <?php endif; ?>
    </span>
</a>
<?php endforeach; ?>
```

Figure 73: Foreach loop

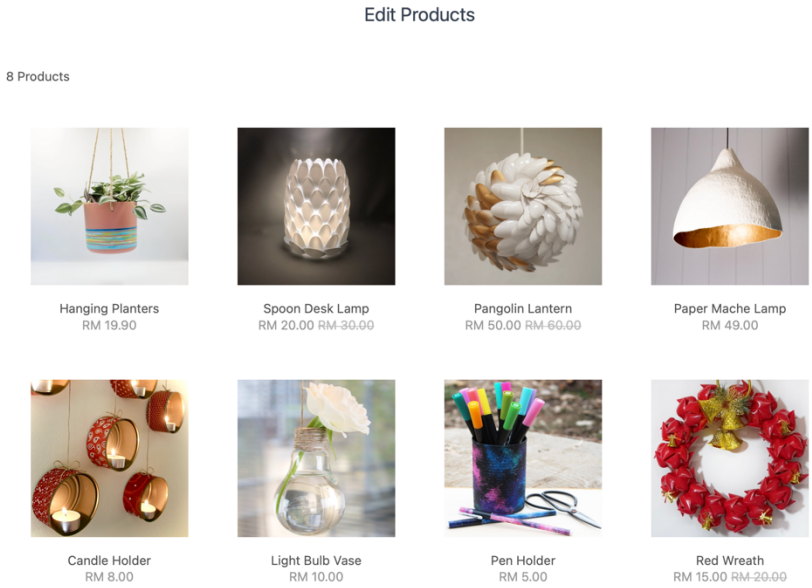


Figure 74: Edit products page

By using the foreach loop shown in figure 73 which is similar to the all products page, all products are displayed on the edit product page shown in figure 74. To edit the product, the admin can just click the title or image of the product that has an anchor that will direct the admin to the specific product detail page, the changeproduct.php file for modification according to the product id shown in figure 73.

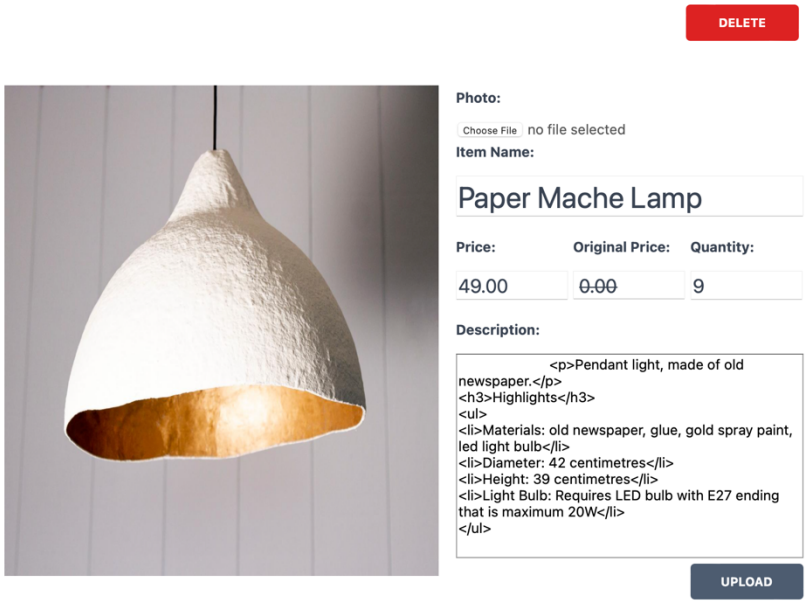


Figure 75: Edit product detail page

Continuing to the edit product detail page shown in figure 75, the admin is allowed to modify the product price, quantity, etc by just inserting the new details in the spaces. Of course, blank space checking will be done by the function StopEmptyForm() shown in figure 76 just like how the login and sign-up page works. An alert message will be given when there are empty spaces detected shown in figure 77.

```
function StopEmptyForm() {

    var x = document.getElementById("itemName").value;
        if (x == "" || x == null) {
            alert("Item name must be filled out");
            return false;
        }
    var x = document.getElementById("price").value;
        if (x == "" || x == null) {
            alert("Price must be filled out");
            return false;
        }
    var x = document.getElementById("originalPrice").value;
        if (x == "" || x == null) {
            alert("Original Price must be filled out");
            return false;
        }
        var x = document.getElementById("itemQuantity").value;
        if (x == "" || x == null) {
            alert("Item Quantity must be filled out");
            return false;
        }
    var x = document.getElementById("description").value;
        if (x == "" || x == null) {
            alert("Description must be filled out");
            return false;
        }
}
```

Figure 76: Function StopEmptyForm()

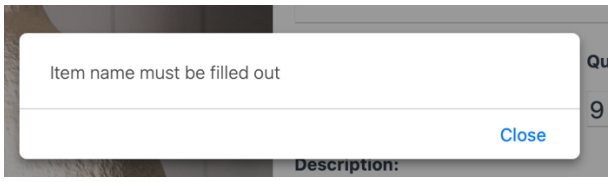


Figure 77: Empty space alert

At the same time, by referring to figure 78, if the photo of the product is re-uploaded, the PDO prepared and execute statement will update the photo id to the latest photo id in the products table while the switch case function will upload the photo in the directed folder or echo error message if the photo failed to upload.

```php
if (array_key_exists('uploadFile', $_FILES) && ($_FILES['uploadFile']['size'] != 0)) {
    $uploadDir = __DIR__ . DIRECTORY_SEPARATOR . 'imgs';
    $targetFilename = $uploadDir . DIRECTORY_SEPARATOR . $_FILES['uploadFile']['name'];
    $uploadInfo = $_FILES['uploadFile'];
    $img_name = $_FILES['uploadFile']['name'];
    $stmt = $pdo->prepare('UPDATE products SET img=? WHERE id=?');
    $stmt-> execute(array($img_name, $_POST['id']));
    switch ($uploadInfo['error']) {
        case UPLOAD_ERR_OK:
            mime_content_type($uploadInfo['tmp_name']);
            move_uploaded_file($uploadInfo['tmp_name'], $targetFilename);
        case UPLOAD_ERR_NO_FILE:
            echo 'No file was uploaded.';
            break;
        default:
            echo sprintf('Failed to upload [%s]: error code [%d].', $uploadInfo['name'], $uploadInfo['error']);
            break;
        }

    }
```

Figure 78: Upload product photo

After that, all information will undergo a validation process shown in figure 80, just like during the signup and login page. If the details entered by the admin match with the data format shown in figure 79, the valid variable will be empty and therefore the database will be updated with the new details according to the product id and shows a successful alert message shown in Figures 81 & 82.

```php
$valid = valid($_POST['itemName'], "/^[A-Za-z0-9 ]+$/");
$valid .= valid($_POST['price'], "/^[1-9]{1}[0-9]{0,}.[0-9]{2}$/");
$valid .= valid($_POST['originalPrice'], "/^[0-9]{1,}.[0-9]{2}$/");
$valid .= valid($_POST['itemQuantity'], "/^[0-9]+$/");
```

Figure 79: Data format

```php
function valid($data,$data_pattern){
    if(!preg_match($data_pattern, $data)){
        return 1;
    }
```

Figure 80: Valid function

```php
if ($valid ==""){
    $stmt = $pdo->prepare('UPDATE products SET `name`=?, `desc`=?, `price`=?, `rrp`=?, `qty`=? WHERE `id`=?');
    $stmt-> execute(array($_POST['itemName'],$_POST['description'],$_POST['price'],$_POST['originalPrice'],$_POST['itemQuantity'],$_POST['i
    echo '<script type="text/javascript">',
    'alert("You have sucessfully updated the product.");',
    '</script>';

}
```

Figure 81: Update details to the database

Figure 82: Successful update alert message

On the other hand, if the admin wishes to delete a product, he or she can do so by clicking the delete button and an alert box will pop out to confirm if the admin wants to delete the product shown in figure 83. After clicking “OK”, the admin will be directed to the deleteproduct.php due to the action tag shown in figure 84.

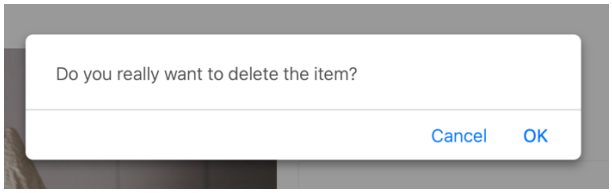


Figure 83: Delete product alert box

```
<form method="post" action="deleteproduct.php"  name="Product" onsubmit="return confirm('Do you really want to delete the item?');" >
    <input type="hidden" name="id" value="<?=$product['id']?>">
    <input type="submit" class="delete"style="width:130px;background:#de2222;"value="Delete">
</form>
```

Figure 84: Delete product form

After that, the $_POST[‘id’] will be assigned as the deleting product id, which then processes the PDO query statement which will be deleting the product from the table in the database shown in figure 85. After the deletion, an alert with successfully deleted product will be displayed to notify the admin, as shown in figure 86.

```
if (isset($_POST['id'])) {
    $stmt = $pdo->query('DELETE FROM products WHERE id='. $_POST['id']);
    if(! $stmt){
        die('Error:' . mysqli_error());
    }
```

Figure 85: Product deleting

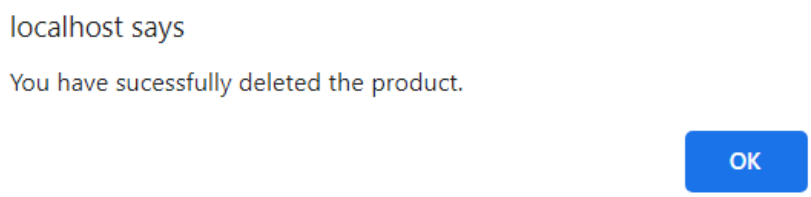


Figure 86: Alert message of successful deletion of product

Conversely, if the admin would like to add a new product to the website, he or she will be brought to the add product page shown in figure 88 by clicking the add product icon under the edit product page shown in figure 87. Admin will just need to fill in the details and upload the image of the product to add a new product. While the details of the product will undergo the blank space checking that is done by the function StopEmptyForm() shown in figure 89 just like editing a product. An alert message will be given when there are empty spaces detected shown in figure 77.

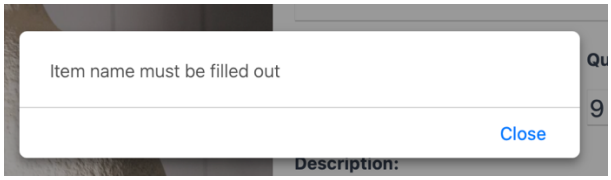


Figure 77: Empty space alert

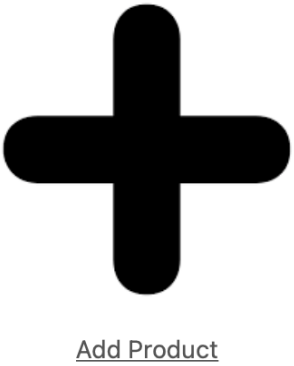


Figure 87: Add product icon

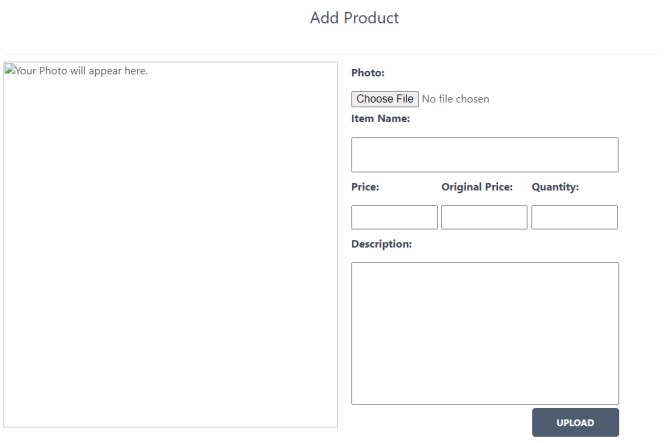
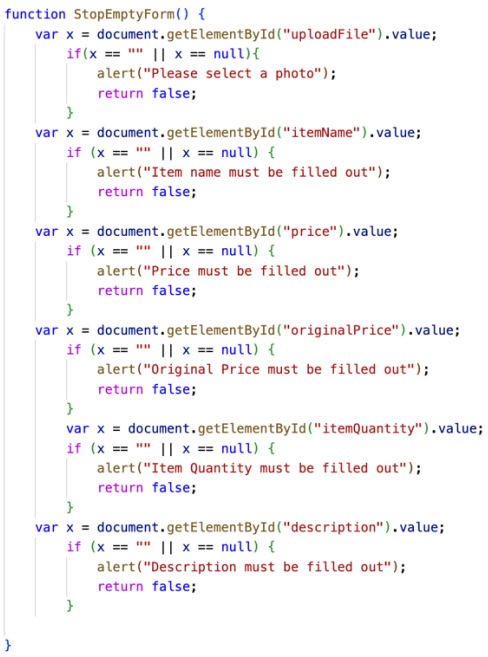


Figure 88: Add product page



```
function StopEmptyForm() {
    var x = document.getElementById("uploadFile").value;
        if(x == "" || x == null){
            alert("Please select a photo");
            return false;
        }
    var x = document.getElementById("itemName").value;
        if (x == "" || x == null) {
            alert("Item name must be filled out");
            return false;
        }
    var x = document.getElementById("price").value;
        if (x == "" || x == null) {
            alert("Price must be filled out");
            return false;
        }
    var x = document.getElementById("originalPrice").value;
        if (x == "" || x == null) {
            alert("Original Price must be filled out");
            return false;
        }
        var x = document.getElementById("itemQuantity").value;
        if (x == "" || x == null) {
            alert("Item Quantity must be filled out");
            return false;
        }
    var x = document.getElementById("description").value;
        if (x == "" || x == null) {
            alert("Description must be filled out");
            return false;
        }

}
```

Figure 89: Function StopEmptyForm()

Then, the data will goes through the same validation process as the editing product, shown in Figures 79 & 80. While the different part is that the product image is not allowed to be left empty, therefore, there will be an extra validation shown in figure 89.

```
$valid = valid($_POST['itemName'], "/^[A-Za-z0-9 ]+$/");
$valid .= valid($_POST['price'], "/^[1-9]{1}[0-9]{0,}.[0-9]{2}$/");
$valid .= valid($_POST['originalPrice'], "/^[0-9]{1,}.[0-9]{2}$/");
$valid .= valid($_POST['itemQuantity'], "/^[0-9]+$/");
```

Figure 79: Data format

```
function valid($data,$data_pattern){
    if(!preg_match($data_pattern, $data)){
        return 1;
    }
```

Figure 80: Valid function

If the data are validated, images and details of the products will be added to the database as figure 90 shown. Last but not least, there will be a successfully added product alert box shown to notify the user, shown in figure 91.

```php
if (isset($_POST['itemName'])){
$valid = valid($_POST['itemName'], "/^[A-Za-z0-9 ]+$/");
$valid .= valid($_POST['price'], "/^[1-9]{1}[0-9]{0,}.[0-9]{2}$/");
$valid .= valid($_POST['originalPrice'], "/^[0-9]{1,}.[0-9]{2}$/");
$valid .= valid($_POST['itemQuantity'], "/^[0-9]+$/");
    if ($valid ==""){
        $uploadDir = __DIR__ . DIRECTORY_SEPARATOR . 'imgs';
        $targetFilename = $uploadDir . DIRECTORY_SEPARATOR . $_FILES['uploadFile']['name'];
        $uploadInfo = $_FILES['uploadFile'];
        $img_name = $_FILES['uploadFile']['name'];
    switch ($uploadInfo['error']) {
        case UPLOAD_ERR_OK:
            mime_content_type($uploadInfo['tmp_name']);
            move_uploaded_file($uploadInfo['tmp_name'], $targetFilename);
        case UPLOAD_ERR_NO_FILE:
            echo 'No file was uploaded.';
            break;
        default:
            echo sprintf('Failed to upload [%s]: error code [%d].', $uploadInfo['name'], $uploadInfo['error']);
            break;
        }
        $stmt = $pdo->prepare('INSERT INTO `products` (`id`,`name`,`desc`,`price`,`rrp`,`qty`,`img`)VALUES(NULL,?,?,?,?,?,?)');
        $stmt-> execute(array($_POST['itemName'],$_POST['description'],$_POST['price'],$_POST['originalPrice'],$_POST['itemQuantity'],$img_name
        echo '<script type="text/javascript">',
        'alert("You have sucessfully added the product.");
        window.location.replace("index.php?page=EditProduct"); ',
        '</script>';
    }
}
```

Figure 90: Add product to the database

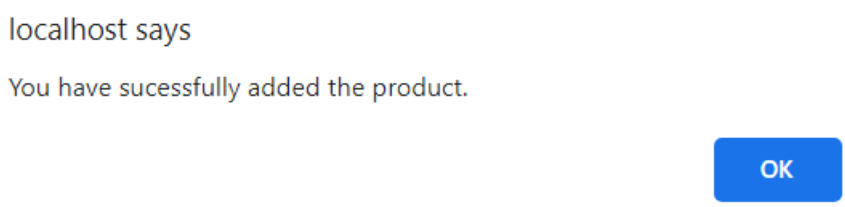


Figure 91: Success alert

Next, edit and delete the customer profile feature. This feature is also only accessible by the admin. The PDO prepared, executes and fetches all statements that are used to get the customer's details from the customers' table in the database to assign to the customers variable shown in figure 92. By using the foreach loop shown in figure 93 which is similar to the edit products page, all users are displayed on the edit customers page shown in figure 94. To edit the customer details, the admin can just click the username or icon of the customer that has an anchor that will direct the admin to the specific customer detail page, the changecustomer.php file for modification according to the CustID shown in figure 93.

```php
$stmt = $pdo->prepare('SELECT * FROM cust_info ORDER BY CustID');
$stmt->execute();
$customers = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Figure 92: Fetch customers

```php
<?php foreach ($customers as $customer): ?>
<a href="index.php?page=changecustomer&CustID=<?=$customer['CustID']?>" class="product">
    <img src="imgs/profile.png" width="200" height="200" alt="profile icon">
    <span class="name"><?=$customer['Last_name']?></span>
    <span class="price">
    <?=$customer['Email']?>
    </span>
</a>
<?php endforeach; ?>
```

Figure 93: Foreach loop

Figure 94: Edit customer page

Continuing to the edit customers detail page shown in figure 95, the admin is allowed to modify the name, username, phone number, etc by just inserting the new details in the spaces. Of course, blank space checking will be done by the function NoEmptyForm() shown in figure 96 just like how the login and sign-up page works. An alert message will be given when there are empty spaces detected shown in figure 97.

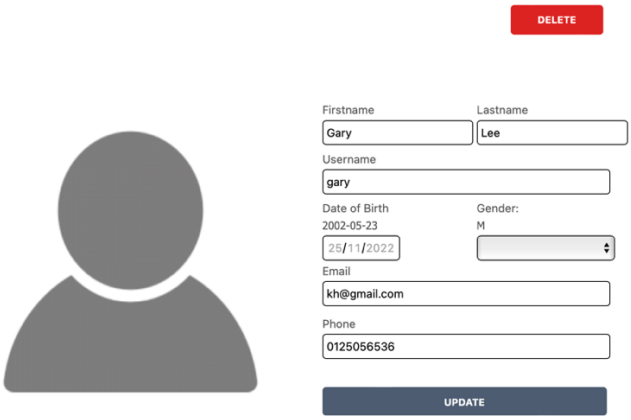


Figure 95: Edit customer detail page

```javascript
function NoEmptyForm() {
    var x = document.forms["Customers"]["FirstName"].value;
        if (x == "" || x == null) {
            alert("Firstname must be filled out");
            return false;
        }
    var x = document.forms["Customers"]["LastName"].value;
        if (x == "" || x == null) {
            alert("Lastname must be filled out");
            return false;
        }
    var x = document.forms["Customers"]["UserName"].value;
        if (x == "" || x == null) {
            alert("Username must be filled out");
            return false;
        }
    var x = document.forms["Customers"]["DateOfBirth"].value;
        if (x == "" || x == null) {
            alert("Date of Birth must be filled out");
            return false;
        }
    var x = document.forms["Customers"]["Gender"].value;
        if (x == "" || x == null) {
            alert("Gender must be filled out");
            return false;
        }
    var x = document.forms["Customers"]["Email"].value;
        if (x == "" || x == null) {
            alert("Email must be filled out");
            return false;
        }
    var x = document.forms["Customers"]["Phone"].value;
        if (x == "" || x == null) {
            alert("Phone must be filled out");
            return false;
        }
```

Figure 96: Function NoEmptyForm()

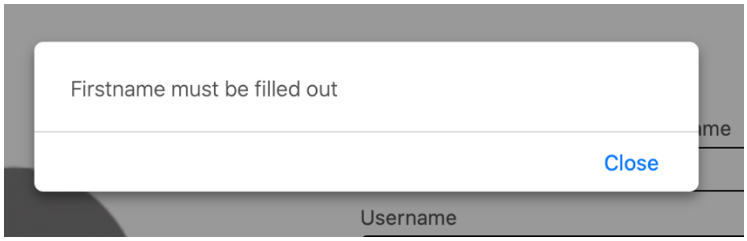


Figure 97: Empty space alert

```php
if(isset($_POST['Phone']) ){
    $stmt = $pdo->query('SELECT * FROM cust_info WHERE Phone =. $pdo->quote($_POST['Phone']).'AND CustID !=. $_POST['CustID']);
    $checkphone = $stmt->fetchAll();
    $stmt1 = $pdo->query('SELECT * FROM admin_info WHERE Phone =. $pdo->quote($_POST['Phone']));
    $checkphone1 = $stmt1->fetchAll();
    $validation = datavalid($_POST['Phone'],"/^[0-9]+$/","*Contain numbers only" );
    if($validation != ""){
        echo $validation;

       }
    elseif(count($checkphone)>0 || count($checkphone1)>0){
        echo '*Phone is used';

    }
    else{
        echo'';
    }
}
```

Figure 97.1: Check repeated information

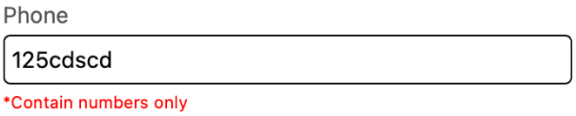


Figure 97.2: Wrong format

After that, all information will undergo a validation process shown in figure 99, just like during the signup and login page. On the other hand, after validating all the details, the details will be compared with the user details in the database to ensure there is no repeated username, email or phone number as shown in figure 97.1, the similar code applies to other details like email and username too. By using the PDO query statement, $stmt, information will be all fetched from the customers' table in the database, while $stmt1 will be checking with the customer table from the database. If both have no matches and the details entered by the admin match with the data format shown in figure 98, the valid variable will be empty and therefore the database will be updated with the new details according to the CustID and shows a successful alert message shown in figure 100 & 101. Conversely, if the details entered are in the wrong format, error messages will be shown in figure 97.2 due to the function datavalid() that will return the errormessage variable when the validation failed. This is because the datavalid() function is assigned to the variable, $validation, therefore when the format of details is wrong, the validation variable will not be empty as it contains the error message from datavalid(), which will echo the error message. Figure 97.1 shows an example of a validation variable echo error message, and it applies to other details like phone numbers too.

```
$valid = valid($_POST['FirstName'], "/^[A-Za-z ]+$/");
$valid .=valid($_POST['LastName'], "/^[A-Za-z]+$/" );
$valid .=valid($_POST['UserName'],"/^[A-Za-z\d]{2,10}/");
$valid .=valid($_POST['Email'],"/^[a-z0-9._%+-]+@[a-z]+\.[a-z]{2,}$/");
$valid .=valid($_POST['Phone'],"/^[0-9]+$/");
```

Figure 98: Data format

```
function valid($data,$data_pattern){
    if(!preg_match($data_pattern, $data)){
        return 1;
    }

}
```

Figure 99: Valid function

```php
if($valid == ""){
    $stmt = $pdo->query('SELECT * FROM cust_info WHERE (Username =\'. $myusername .'OR Email=\'. $email .'OR Phone=\'. $phone.') AND (CustID !=\'. $_POST['CustID'].')');
    $check = $stmt->fetchAll();
    $stmt1 = $pdo->query('SELECT * FROM admin_info WHERE Username =\'. $myusername .'OR Email=\'. $email .'OR Phone=\'. $phone);
    $check1 = $stmt1->fetchAll();
    if((count($check) == 0 ) && (count($check1) == 0)){
        $stmt = $pdo->prepare('UPDATE cust_info SET `First_name`=?, `Last_name`=?, `Username`=?, `Dob`=?, `Gender`=?, `Email`=?, `Phone`=? WHERE `CustID`=?');
        $stmt-> execute(array($_POST['FirstName'],$_POST['LastName'],$_POST['UserName'],$_POST['DateOfBirth'],$gender,$_POST['Email'],$_POST['Phone'], $_POST['CustID'])
        echo '<script type="text/javascript">',
        'alert("You have sucessfully updated the customer.");',
        '</script>';
    }
}
```

Figure 100:Update details to the database

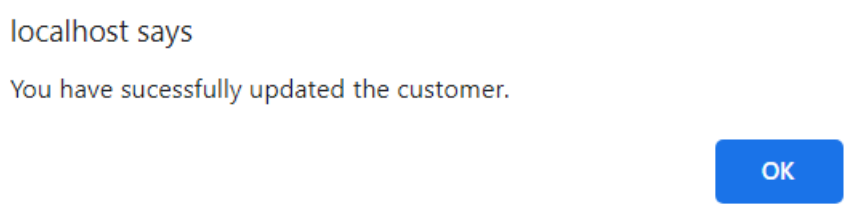


Figure 101: Successful update alert message

On the other hand, if the admin wishes to delete a customer profile, he or she can do so by clicking the delete button and an alert box will pop out to confirm if the admin wants to delete the customer profile shown in figure 102. After clicking “OK”, the admin will be directed to the deletecustomer.php due to the action tag shown in figure 103.

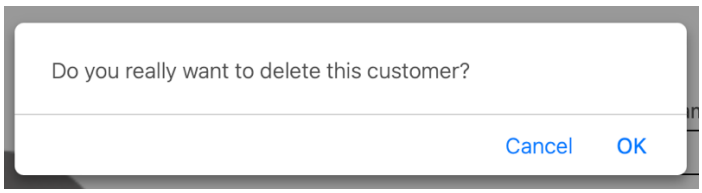


Figure 102: Delete customer alert box

```php
<form method="post" action="deletecustomer.php"  onsubmit="return confirm('Do you really want to delete this customer?');" >
    <input type="hidden" name="CustID" value="<?=$customer['CustID']?>">
    <input type="submit" class="delete"style="width:130px;background:#de2222;"value="Delete">
</form>
```

Figure 103: Delete customer form

After that, the $_POST[‘id’] will be assigned as the deleting CustID, which then processes the PDO query statement which will be deleting the customer from the table in the database shown in figure 104. After the deletion, an alert with successfully deleted product will be displayed to notify the admin, shown in figure 105.

```php
if (isset($_POST['CustID'])) {
    $stmt = $pdo->query('DELETE FROM cust_info WHERE CustID='. $_POST['CustID']);
    if(! $stmt){
        die('Error:' . mysqli_error());
    }
```

Figure 104: Deleting a customer

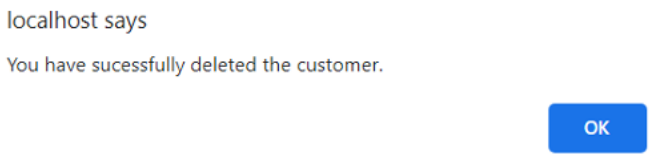


Figure 105: Alert message of successfully deleting customer

Furthermore, we had set session timeout for admin and customer mode when they are logged in, which will be automatically logged out after five minutes of inactivity by implementing the function destroy_session_and_data() for both admin and customer mode, shown in figure 106. The function will be triggered when the last activity time is more than 300 seconds which is five minutes as shown in figure 107. When the user gets back to the website after five minutes, an alert box will be shown in figure 108 which will redirect users to the login page to prompt them to log in again after they click “Close”. To implement this session timeout function, all PHP files for the website will need to include this logtimeout.php file to include the function except a few function files that are not necessary.

```php
function destroy_session_and_data_admin(){
  unset($_SESSION['First_name']);
  unset($_SESSION['Username']);
  unset($_SESSION['AdminID']);
  unset($_SESSION['ID']);
  $_SESSION = array();
  session_unset();
  setcookie(session_name(), '', time() - 2592000, '/');
  session_destroy();
  }


function destroy_session_and_data(){
  unset($_SESSION['First_name']);
  unset($_SESSION['Username']);
  unset($_SESSION['CustID']);
  unset($_SESSION['ID']);
  $_SESSION = array();
  session_unset();
  setcookie(session_name(), '', time() - 2592000, '/');
  session_destroy();
  }
```

Figure 106: Function destroy_session_and_data()

```php
if (isset($_SESSION['First_name']) && $_SESSION['ID']==2){

    if (isset($_SESSION['LAST_ACTIVITY']) && (time() - $_SESSION['LAST_ACTIVITY'] > 300)) {
        destroy_session_and_data();

        echo '<script type="text/javascript">',
        'alert("Your session has timed out due to inactivity for more than 5 minutes. \n\nPlease Log in again");
        window.location.replace("index.php?page=Login"); ',
        '</script>';

    }

    $_SESSION['LAST_ACTIVITY'] = time(); // update last activity time stamp
}
```

Figure 107: Session timeout

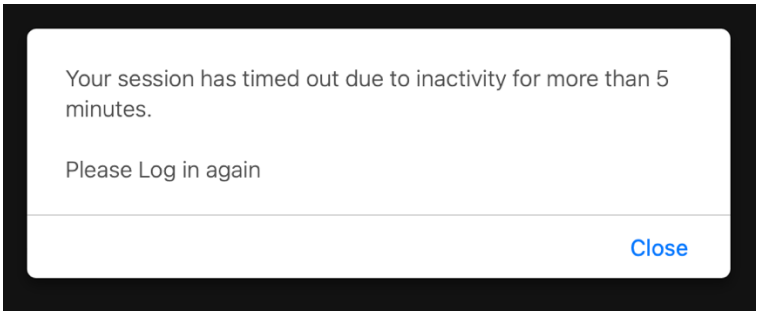


Figure 108: Alert box of session timeout

Last but not least, the logout feature is available when users are logged in, it is accessible by clicking the logout icon as figure 109 shown which is placed on the top right of the website. When the user clicked the logout icon, the $_SESSION[‘ID’] is retrieved, shown in figure 110 to identify if the user is an admin or customer to run the respective PHP function destroy_session_and_data() shown in figure 111.

Figure 109: Logout icon

```php
if (isset($_SESSION['First_name']) && $_SESSION['ID']==2){

  destroy_session_and_data();
  echo '<script type="text/javascript">',
  'alert("You have sucessfully logout.");
  window.location.replace("index.php"); ',
  '</script>';
}
if (isset($_SESSION['First_name']) && $_SESSION['ID']==1){

  destroy_session_and_data_admin();
  echo '<script type="text/javascript">',
  'alert("You have sucessfully logout.");
  window.location.replace("index.php"); ',
  '</script>';
}
```

Figure 110: Logout

```php
function destroy_session_and_data_admin(){
  unset($_SESSION['First_name']);
  unset($_SESSION['Username']);
  unset($_SESSION['AdminID']);
  unset($_SESSION['ID']);
  $_SESSION = array();
  session_unset();
  setcookie(session_name(), '', time() - 2592000, '/');
  session_destroy();
  }


function destroy_session_and_data(){
  unset($_SESSION['First_name']);
  unset($_SESSION['Username']);
  unset($_SESSION['CustID']);
  unset($_SESSION['ID']);
  $_SESSION = array();
  session_unset();
  setcookie(session_name(), '', time() - 2592000, '/');
  session_destroy();
  }
```

Figure 111: Function destroy_session_and_data()

An alert box will be shown in figure 112 when the user is successfully logged out.

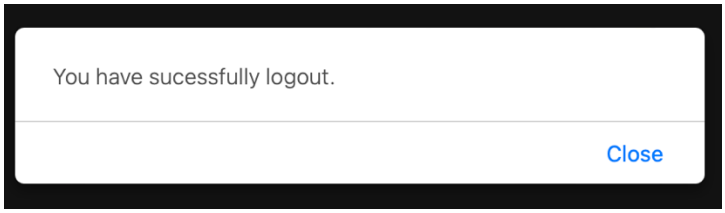


Figure 112: Logout alert box

Test cases and test results

Test Case ID	TC_001_01
Test Case Description	Login with the exists member username and password and verify the website response.
Objective	Test if the username and password validation is functioning for member mode.
Pre-requisites	Able to access the login page.
Condition	Enter the correct username and password.
Test Steps	1. Enter the username. 2. Enter the password. 3. Click the “Login” button.
Input Data	Username: leekahhoong Password: Mnbvcxz123!
Expected Result	The website should direct user to home page and show the cart icon and allow user to add products to cart.
Actual Result	The website direct user to home page and allow user to add products to the cart.
Author	Lee Kah Hoong
Status	Pass

Test Case ID	TC_001_02
Test Case Description	Login with the non-exists member and admin username and password and verify the website response.
Objective	Test if the username and password validation is functioning for member mode.
Pre-requisites	Able to access the login page.
Condition	Enter the non-exists username and password.
Test Steps	1. Enter the username. 2. Enter the password. 3. Click the “Login” button.
Input Data	Username: dewdwe Password: Teevcxz123!
Expected Result	The website should show *User not found message
Actual Result	The website shows *User not found message
Author	Lee Kah Hoong
Status	Pass
Test Case ID	TC_001_03
Test Case Description	Login with the exists admin username and password and verify the website response.
Objective	Test if the username and password validation is functioning for admin mode.
Pre-requisites	Able to access the login page.
Condition	Enter the correct username and password.

Test Steps	1. Enter the username. 2. Enter the password. 3. Click the “Login” button.
Input Data	Username: garylee Password: Poiuytrewq123!
Expected Result	The website should direct user to home page and show the cart icon and allow user to edit, add or remove products and edit or delete customers profile, and add admin.
Actual Result	The website direct user to home page and show the cart icon and allow user to edit, add or remove products and edit or delete customers profile, and add admin.
Author	Lee Kah Hoong
Status	Pass

Test Case ID	TC_001_04
Test Case Description	Sign up with the exists member email and phone number and verify the website response.
Objective	Test if the email and phone number validation is functioning for member mode.
Pre-requisites	Able to access the signup page.
Condition	Enter the repeated email and phone number.
Test Steps	1. Enter the details. 2. Click the “Sign Up” button.
Input Data	Email: mr.gary@outlook.my Phone Number: 0125250408
Expected Result	The website should show email and phone is used error message
Actual Result	The website show email and phone is used error message
Author	Lee Kah Hoong
Status	Pass

Test Case ID	TC_001_05
Test Case Description	Sign up with new member email and phone number and verify the website response.
Objective	Test if the email and phone number validation is functioning for member mode.
Pre-requisites	Able to access the signup page.
Condition	Enter new email and phone number.
Test Steps	1. Enter the details. 2. Click the “Sign Up” button.
Input Data	Email: kahhoong02@icloud.com Phone Number: 0125788884
Expected Result	The website should show “You have successfully create a account. Please proceed to login page for login.”

Actual Result	The website show “You have successfully create a account. Please proceed to login page for login.”
Author	Lee Kah Hoong
Status	Pass

Test Case ID	TC_001_06
Test Case Description	Add new admin to the website.
Objective	Test if the new admin adding function is functional.
Pre-requisites	Able to access the add admin page.
Condition	Enter the details correctly.
Test Steps	1. Enter the details. 2. Click the “Sign Up” button.
Input Data	Email: fishcake_@outlook.com Phone Number: 0173849534
Expected Result	The website should show added successful message
Actual Result	The website show added successful message
Author	Lee Kah Hoong
Status	Pass