
# UECS2094/2194 WEB APPLICATION DEVELOPMENT

# 2024 JANUARY TRIMESTER

# ASSIGNMENT REPORT

Lecturer: Dr. Faranak Nejati and Dr. Chia Kai Lin

| Name: | Student Id: | Course: | Practical: |
|---|---|---|---|
| Amanda Grace Seow Chia Wern | 2106302 | AM | P3 (UECS2094) |
| Cheryl Ng Zi Qie | 2207022 | AM | P3 (UECS2094) |
| Low Si Qing | 2104872 | SE | P1 (UECS2194) |
| Tan Wan Xuen | 2207214 | AM | P2 (UECS2094) |

# Table of Contents

## 1. Introduction

According to the task given, we have decided to develop a comprehensive and responsive online web application for our salon, Z23 Beauty Salon, in hopes of meeting both our needs as well as our customer's needs. We have managed to develop the website through Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript, Hypertext Preprocessor (PHP), and Structured Query Language (SQL). We had also ensured to encompass both static and dynamic elements while utilizing a range of client-side and server-side technologies.

Our aim in designing and developing the Z23 Beauty Salon website was to allow our customers to be able to have access and see what we have to offer them from the comfort of their homes. This includes viewing available services and products through the website. We also aimed to allow our customers to book any desired appointments and place orders for any products. Customers are also able to submit a form for enquiries and complaints.

Hence, we designed our website with a simple, elegant home page accompanied with a secure login system allowing users to log in, log out and create an account, with an intuitive navigation bar that allows seamless navigation to the desired webpages, ensuring a user-friendly experience. In our Z23 Beauty Salon website, we also provide service booking as well as sell some of the products we used. The types of services we provide include hair treatment, facials, manicures and pedicures, and massage treatments. On the other hand, the types of products we sell correlate back to our services, like hair oils, nail polishes and facial products. Two comprehensive lists of services and products are clearly shown with transparent pricing and detailed information.

Users will also be able to book any appointments through the service booking system while they can add products, they are interested in into their wish list. Once they finished scrolling through our website, the user will be able to purchase their desired products through the checkout system. It is also notable that the checkout system will not be used in the booking system as customers will only be required to pay after their booking is completed. Also, customers with any enquiries or complaints are welcome to leave their thoughts on our contact us page. Overall, the Z23 Beauty Salon website provides users with a seamless, responsive, and convenient web browsing experience.

## 2. Flow Chart

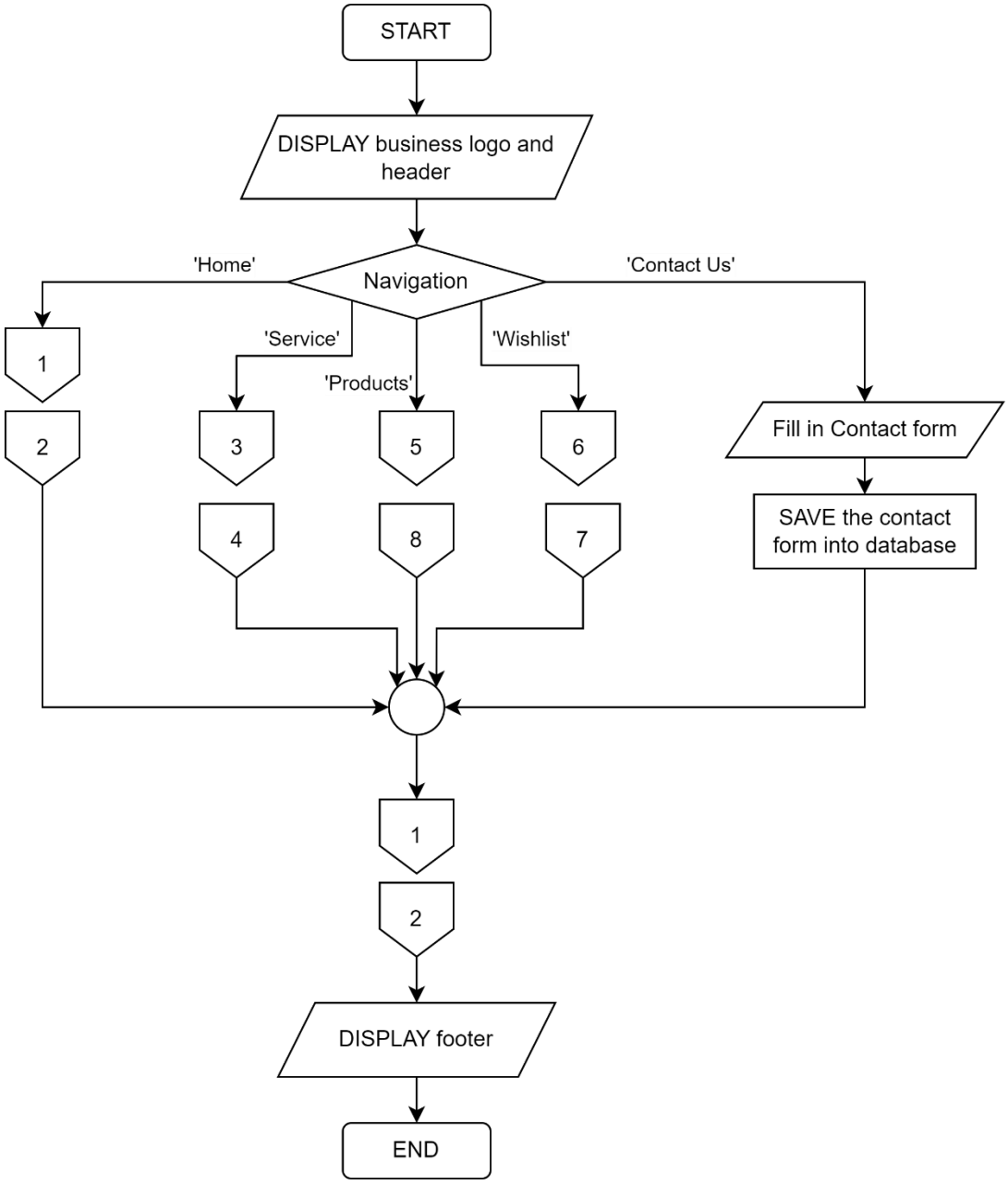


*Figure 2.1: Flow Chart (Part 1)*

The flow chart shown in Figure 2.1 shows the overall progress flow of the website. At the starting point, it initiates the websites by displaying our salon's logo and header. Following the header and the salon's logo, the website displays a navigation system that allows users to move through the

different webpages. It is also notable that the header, logo, and navigation bar has been linked and programmed to display on every page. The navigation system allows users to move through distinct webpages and explore our website with ease.

From the navigation bar, users can maneuver through the home page, service page, products page, wish list, and contact us page. When users click on the contact us page, they will be directed to a contact form. The completed form will then be submitted and saved into our database and into the “contact_info” table. After the form is submitted, the webpage will display and thank you message and redirect the users back to the home page.

It is notable that after each action and after being directed through each page, there will always be a footer displayed at the end of the webpage. Hence, that is the overall flow for our simple, interactive, and responsive website below. However, we shall further elaborate on the other functions of the navigation bar in the report below.

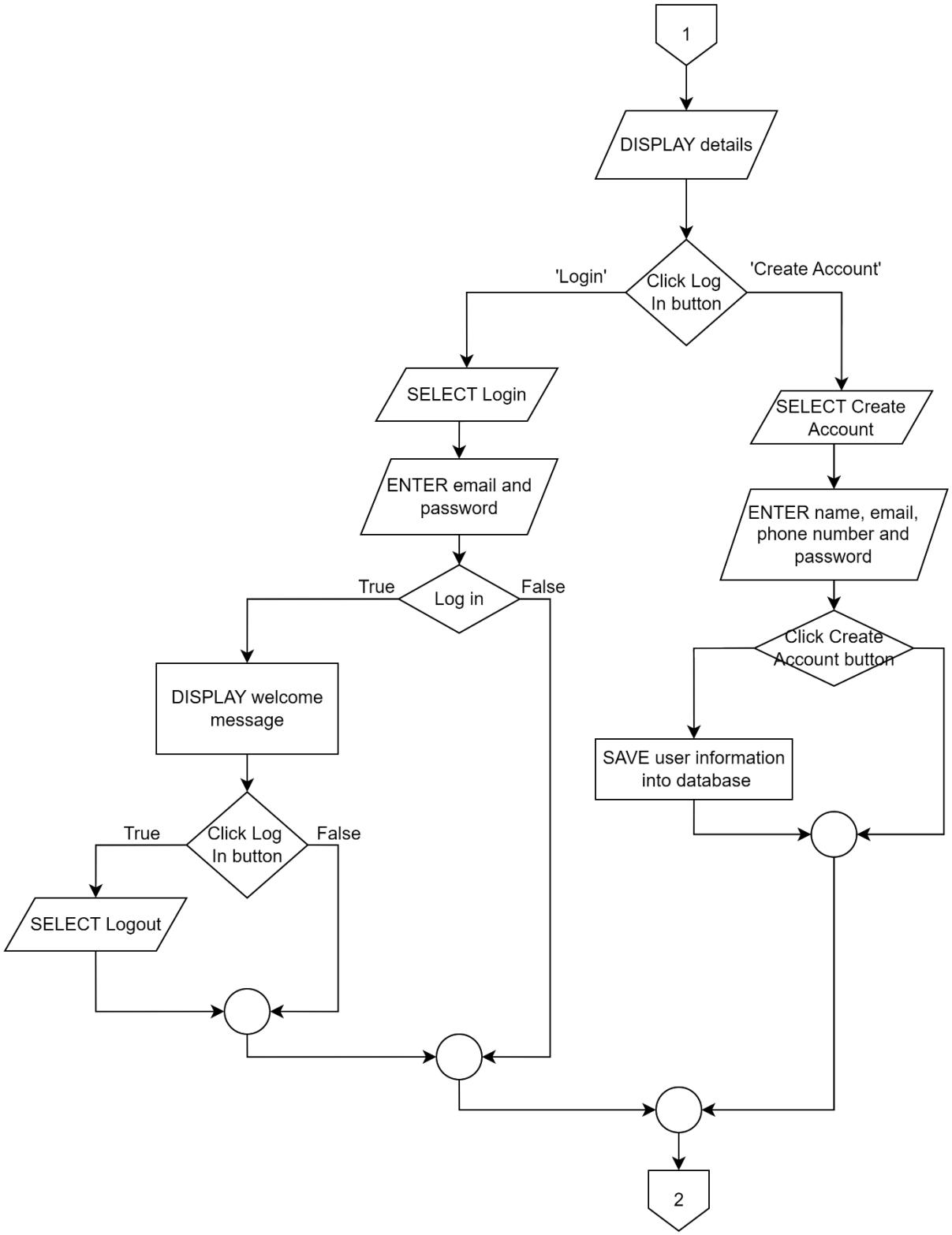


*Figure 2.2: Flowchart (Part 2)*

From Figure 2.2, we can see that when the user arrives at the home page, they are greeted with an overview of the website's details. Upon clicking the login button, users are presented with the option to either "Login" or "Create Account". When users opt for the "Login" action, the user is prompted to enter their email and password. Upon successful login, a session will start, and the webpage will display a welcome message with their email, indicating their successful authentication. It should be noted that once a user logs in, the login button will no longer give them the option to login or to create an account. Instead, they are given the option to log out of their account whenever they deem necessary. Hence, once they click the "Login" button again and choose the "Log Out" button, the session will be terminated, and they will be redirected back to the home page and are given the option to log in and create an account once again.

Alternatively, when users select the "Create Account" option, they are directed to a registration form where they are prompted to enter their name, email, phone number, and password. After filling in these details, users can click the "Create Account" button. Once that happens, the webpage will display the user's submission details, notifying the user of the successful account creation process. Simultaneously, the user's information is saved into the database for future logins and account management into the "users" table.

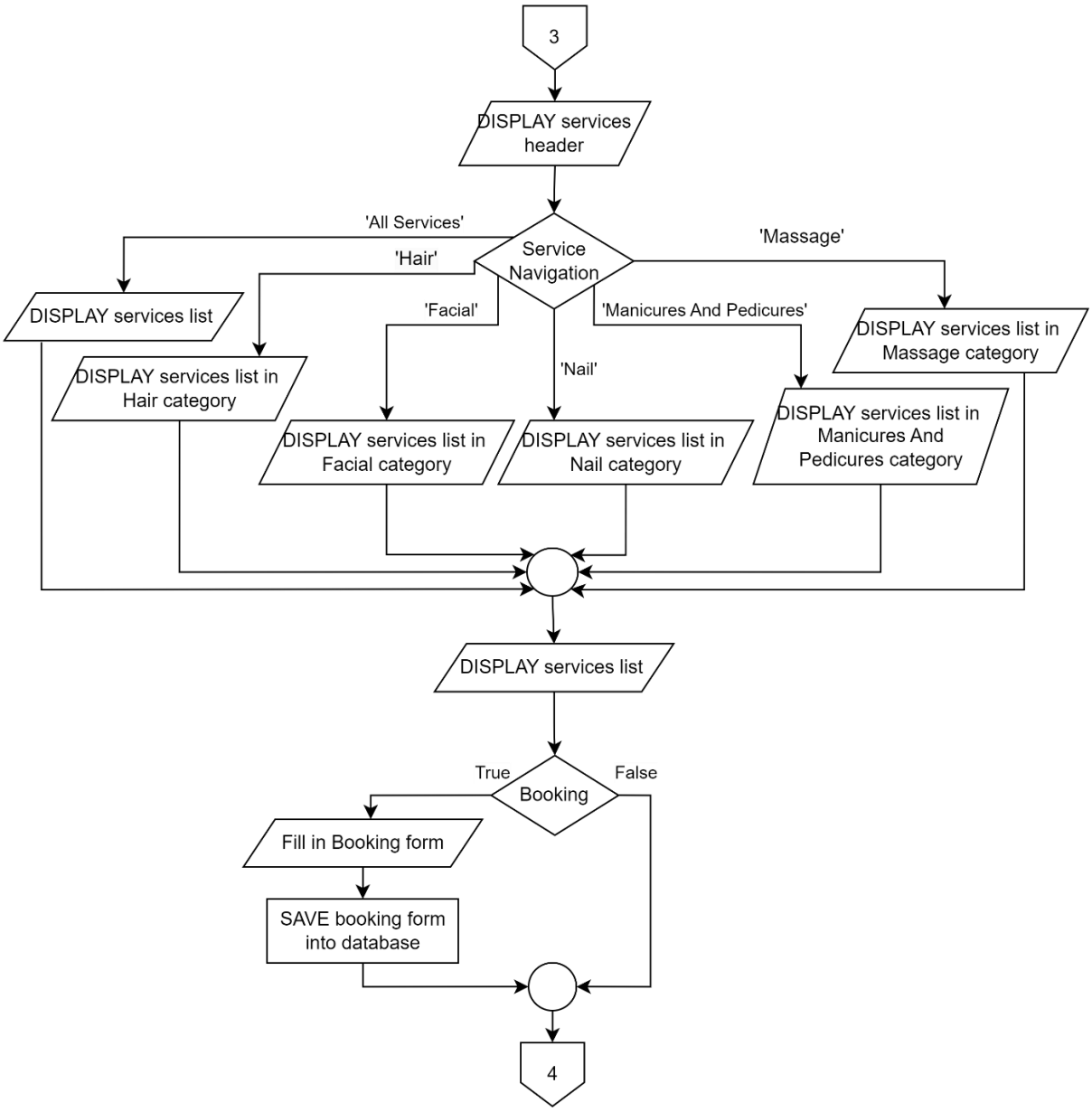


*Figure 2.3: Flowchart (Part 3)*

When the user selects the “Services” keyword on the navigation bar, they are redirected to the “Service” page, accompanied with a service header and a service navigation. On this page, the service navigation bar allows users to explore the different service categories the salon offers such as “Hair”, “Facial”, “Nail”, “Manicures and Pedicures”, and “Massage”. Each category provides specific services, accompanied by photos and price ranges.

When the user clicks on the “All services” option, they users can also click on the “All Services” button to see the complete list of all the provided services. Alternatively, selecting

“Hair”, “Facial”, “Nail”, “Manicures and Pedicures”, and “Massage” will display the provided services specific to that category.

After viewing the service list, users can proceed to make a booking by clicking the “Booking” button. This action redirects them to a booking form where they must fill in their details. Once the form is submitted, a confirmation message informs the user that no further changes can be made to the submitted details. The completed form is then saved into the database. Upon completion of these steps, users will see the service details and can click “Done” to return to the homepage.

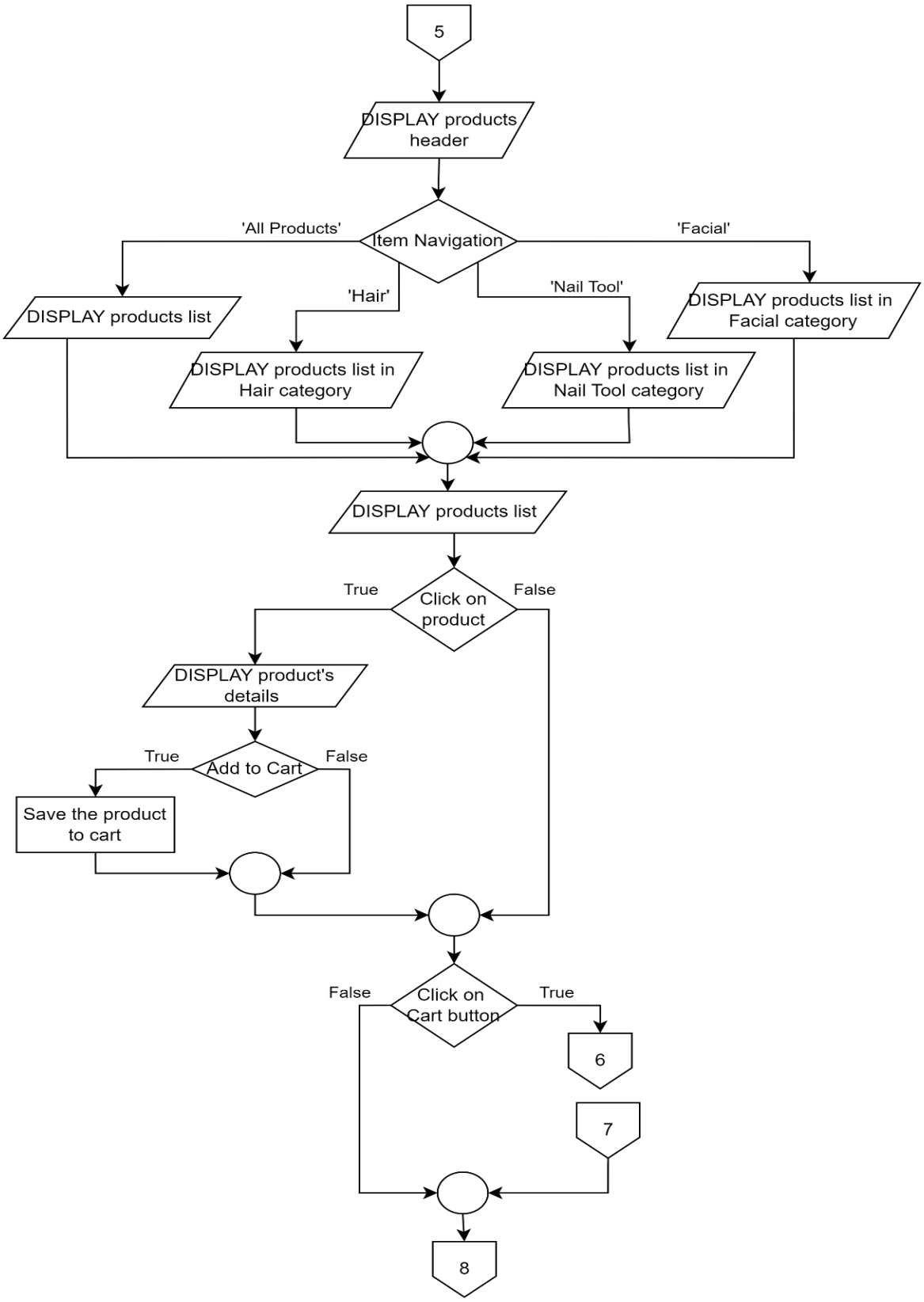


*Figure 2.4: Flowchart (Part 4)*

Moving on, we can see the flowchart of the product list page through Figure 2.4. The figure shows the progress flow of displaying product listings within various product categories. These listings include the ID, name, image, and price of each product.

The product navigation bar allows users to view the different products corresponding to its respective category. When the user chooses to click on ‘All Products’, a complete list of all the products are displayed to the users. On the other hand, when the user clicks on the different categories such as ‘Hair’, ‘Nail Tool’ and ‘Facial’, the webpage will display the products specific to that category.

When the users click on the product, a pop-up window appears to display the details of the product. If the users wish to buy the product, they may choose to press the ‘Add to Cart’ button. The product will then be saved into their checkout cart. In addition, if users want to view the items in the cart, they may click on the Cart button. Once that happens, the users will be directed to the wish list page.

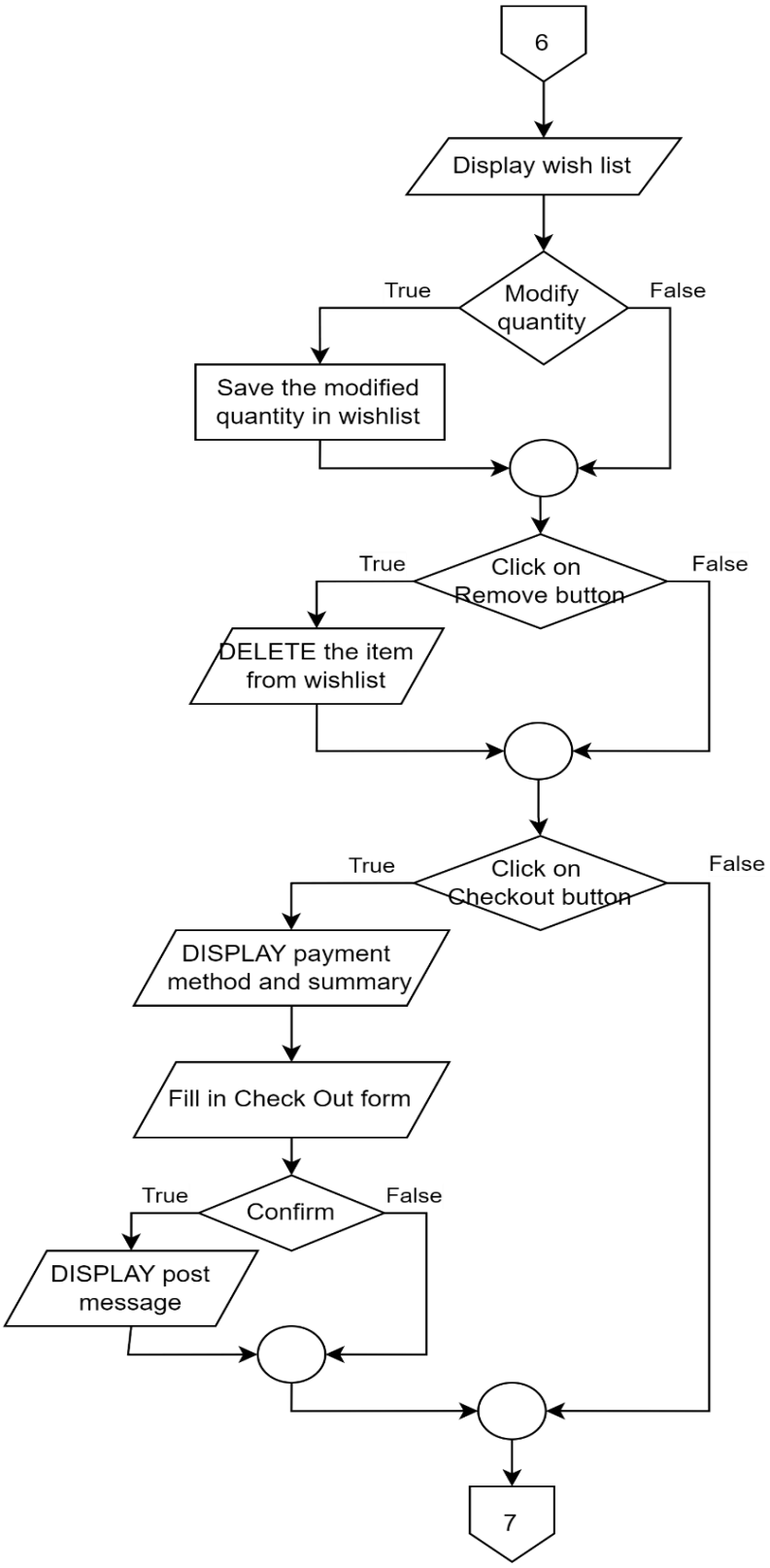


*Figure 2.5: Flowchart (Part 5)*

Next, we can observe the process of managing their wish list and the checkout process through the flow chart shown in Figure 2.5. The webpage first displays the salon's logo, header, and navigation bar. The display of items is added into the cart. To handle the items selected, it allows users to modify the quantity of the items in their cart. If the quantity of the items is modified, it will save the modified quantity in the wish list. If users choose to remove the selected item, by clicking on the Remove button, the items will be deleted from the wish list.

Next, if the user chooses to click on the Checkout button, it will then redirect users to the product check out page displaying the payment method and summary of products selected. In this page, they are required to fill up the checkout form and press the Confirm button to complete the checkout process. If the Confirm button is pressed and the information filled out is valid, it will redirect to the post message page by displaying a post message with appreciation. Upon completion, users will be redirected back to the homepage.

## 3. Detailed Explanation and Implementation of Code

### 3.1. database.sql

*Important Note: It is required for us to import the following SQL file into the localhost/phpMyAdmin before running the webpage.*

First and foremost, we create a SQL file for and use the following SQL command to create a new database named "Z23_Beauty_Salon" with the specified character set "utf8" and collation "utf8_general_ci".

```
-- Creating the database
CREATE DATABASE Z23_Beauty_Salon CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Using the "Z23_Beaty_Salon" database, we create a table named “users” for our user login system. A record will be inserted into the database whenever a user decides to create an account on our website. The table stores the user's id, name, email, phone number and password. The user id had been set to an auto increment column to increase with each account. We set the email column with a unique key as the user’s email will function as their username when they login to their account in the future and we wanted to avoid instances of a user creating multiple accounts with one email. The code for creating the users table is as follows:

```
USE Z23_Beauty_Salon;

-- create users table
CREATE TABLE users(
    id INT(11) AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    email VARCHAR(255) UNIQUE KEY,
    phone VARCHAR(255),
    password VARCHAR(255)
);
```

Next, we created a table for all the products we sell in our salon. We named the table as “item” to represent our products and consists of the columns, category, id, image_url, title, type, description, price, and quantity. The "id" column is specified as the primary key for the "item" table. It is also notable that we've standardized the price values to display only two decimal places through the database.

```
-- Creating a ITEM table
CREATE TABLE item(
    category TEXT,
    id VARCHAR(255) PRIMARY KEY,
    image_url VARCHAR(255),
    title TEXT,
    type TEXT,
    description TEXT,
    price DECIMAL(10, 2),
    quantity INT
);
```

In the "item" table, 32 items have been created, spanning four different categories: hair, nail tool, and facial. Each item constitutes of the same structure as described in the provided code snippet below.

```
-- Insert value to ITEM table
INSERT INTO item (category, id, image_url, title, type, description, price, quantity)
    VALUES('HAIR','TC01','https://www.lorealparis.com.my/-/media/project/loreal/brand-
sites/oap/apac/my/local-products/hair-care/hair-care/elseve-hyaluron/ha-night-cream-
150ml/ha-night-cream-
150ml.png?w=360&rev=de285d1770b34454a48f58c5fea20872&hash=71D15895F8301EFA284FBF6C2BDC969E9
8F9AE0C',
    'HydraLuxe Hyaluronic Night Renewal Cream - 150ml',
    'Treatment Cream/Oil', 'Experience the ultimate overnight hair treatment infused with
hyaluronic acid for deep hydration and sealed cuticles.
    Wake up to soft, detangled, and frizz-controlled hair, replumped with 4X moisture.',
26.89, 20);
```

It is also notable that the images of items in the HAIR category are obtained from Hair Care - Hair Products and Advice (n.d.), while the images of items in the NAIL TOOL and FACIAL categories had been sourced from Innisfree MY | Skincare and Beauty Products Inspired by Jeju (n.d.).

Similarly, a table named “service” had been created to represent the three main services our salon offers. The table consists of three columns, category, type, and priceRange.

```
-- Creating a SERVICE table
CREATE TABLE service(
    category TEXT,
    type TEXT,
    priceRange VARCHAR(255)
);
```

We then submit the total 15 services we provide through the specified SQL code provided below. All 15 services were inserted into the service table in a similar manner.

```
-- Insert value for SERVICE table
INSERT INTO service(category, type, priceRange)
    VALUES('HAIR','Hair Cut','RM10++');
```

Furthermore, a ‘checkout’ table is created to store the information filled in by the users in the checkout form. In consists of the columns id, buyer_name, buyer_email, buyer_phone, buyer_address, reference, and bank. The references column is used to store a user’s payment reference number. Hence, the column has been set to hold the value of a unique key as it is important that such numbers would not be entered twice into the system.

```
CREATE TABLE checkout(
    id INT AUTO_INCREMENT PRIMARY KEY,
    buyer_name VARCHAR(255) NOT NULL,
    buyer_email VARCHAR(255) NOT NULL,
    buyer_phone VARCHAR(15) NOT NULL,
    buyer_address VARCHAR(255) NOT NULL,
    reference VARCHAR(255) NOT NULL UNIQUE KEY,
    bank VARCHAR(255) NOT NULL
);
```

Moving on, we table “items_bought” was generated to store the items bought by the users, an ‘items_bought’ table is generated. It records the items purchased by the user according to the checkout id created in the ‘checkout’ table. That is done by indicating that the checkout_id column in the items_bought table references the id column in the checkout table through the last line of the code snippet provided below. The price per item column has also been set to only store its value with two decimal points.

```
CREATE TABLE items_bought (
    id INT AUTO_INCREMENT PRIMARY KEY,
    checkout_id INT,
    item_id VARCHAR(255),
    quantity INT,
    price DECIMAL(10, 2),
    FOREIGN KEY (checkout_id) REFERENCES checkout(id)
);
```

Furthermore, we created another table, "service_bookings" to store bookings made by users. The table stores the information for id, name, email, phone, num_pax, service_type, preferred_date, preferred_time and additional_message. The "id" column is specified as the primary key for the "service_bookings" table. Name, email, phone, num_pax, service_type, preferred_date and preferred_time are set “NOT NULL” which means the columns cannot be left empty. However, the additional_message has not been set to “NOT NULL” as it will be an optional column. That section would be elaborated further into the report.

```
CREATE TABLE service_bookings (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    num_pax INT NOT NULL,
    service_type VARCHAR(100) NOT NULL,
    preferred_date DATE NOT NULL,
    preferred_time TIME NOT NULL,
    additional_message TEXT
);
```

Last but not least, the "contact_info" table is created. The table was created to store the information of a user when they try to contact us through the form provided on our website. The table stores the information for id, salutation, enquiry, name, email, phone, subject and subscribe. The "id" column is specified as the primary key for the "contact_info" table. Salutation, enquiry, name, email, phone and subject are set “NOT NULL” which means the columns cannot be left empty.

```
CREATE TABLE contact_info (
    id INT AUTO_INCREMENT PRIMARY KEY,
    salutation VARCHAR(10) NOT NULL,
    enquiry VARCHAR(10) NOT NULL,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    subject TEXT NOT NULL,
    subscribe TINYINT(1) DEFAULT 0
);
```

Once all tables and columns have been created using SQL commands, we import the "database.sql" file into phpMyAdmin. The system will show the message, “*Import has been successfully finished*.”, as shown in Figure 3.1.1.

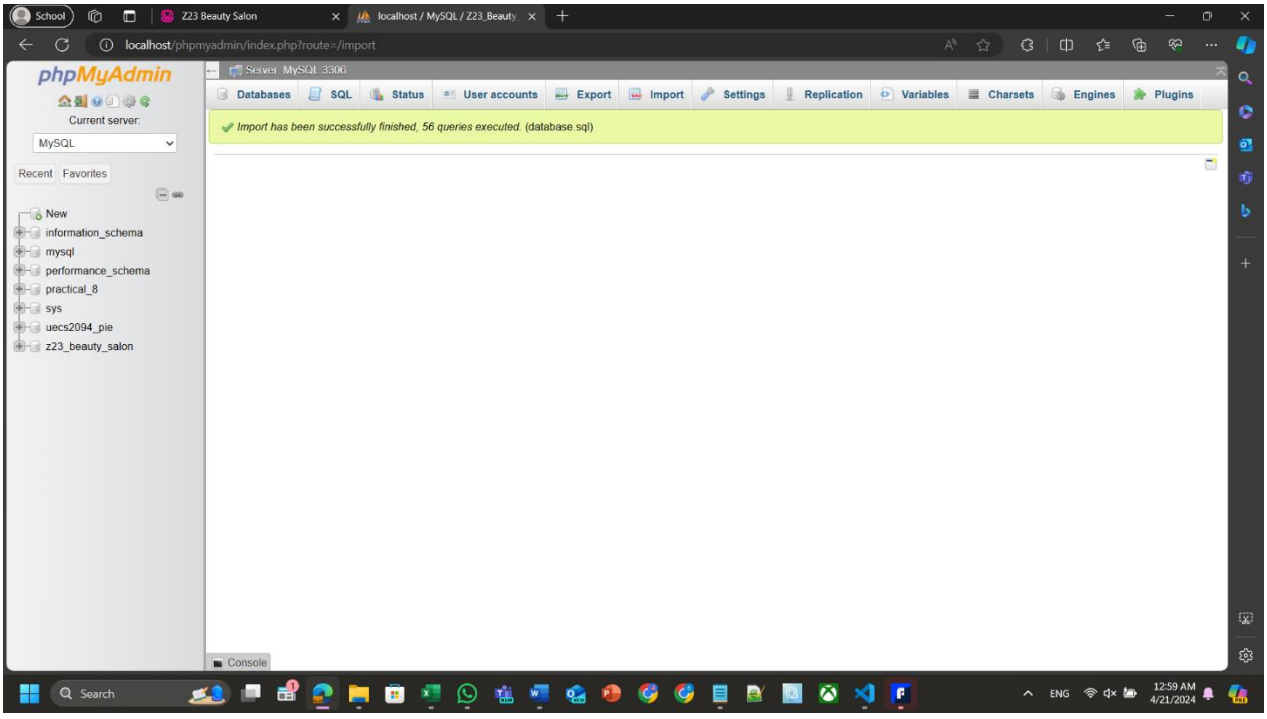

*Figure 3.1.1: Display of successful import message.*

After that, we can review all the tables and columns created and added through phpMyAdmin. We can see some examples of our view of the database and different tables through the figures below.

| | Table | Action | Rows | Type | Collation | Size | Overhead |
|---|---|---|---|---|---|---|---|
| ☐ | checkout | Browse Structure Search Insert Empty Drop | 0 | MyISAM | utf8mb3_general_ci | 4.0 KiB | - |
| ☐ | contact_info | Browse Structure Search Insert Empty Drop | 0 | MyISAM | utf8mb3_general_ci | 1.0 KiB | - |
| ☐ | item | Browse Structure Search Insert Empty Drop | 32 | MyISAM | utf8mb3_general_ci | 20.2 KiB | - |
| ☐ | items_bought | Browse Structure Search Insert Empty Drop | 1 | MyISAM | utf8mb3_general_ci | 3.0 KiB | - |
| ☐ | service | Browse Structure Search Insert Empty Drop | 15 | MyISAM | utf8mb3_general_ci | 1.6 KiB | - |
| ☐ | service_bookings | Browse Structure Search Insert Empty Drop | 1 | MyISAM | utf8mb3_general_ci | 2.1 KiB | - |
| ☐ | users | Browse Structure Search Insert Empty Drop | 4 | MyISAM | utf8mb3_general_ci | 9.3 KiB | - |
| | 7 tables | Sum | 53 | MyISAM | utf8mb3_general_ci | 41.2 KiB | 0 B |

*Figure 3.1.2: General view of all our tables.*

| | id | name | email | phone | password |
|---|---|---|---|---|---|
| ☐ Edit Copy Delete | 1 | Amanda | Amandagraceseow@gmail.com | 0178661101 | manda01 |
| ☐ Edit Copy Delete | 2 | Travis Lion King | travislion@gmail.com | 0162229812 | tt |
| ☐ Edit Copy Delete | 3 | Vincy Xue Hua | vincy999huaaa@hotmail.my | 0197654324 | xuehuapiaopiao |
| ☐ Edit Copy Delete | 4 | Shirley Elsa Lim | shirley@omg.com | 0175435242 | letitgoooooo |

*Figure 3.1.3: View of our "users" table*

| | id | buyer_name | buyer_email | buyer_phone | buyer_address | reference | bank |
|---|---|---|---|---|---|---|---|
| ☐ Edit Copy Delete | 1 | Amanda | Amandagraceseow@gmail.com | 0178883636 | 23, Jalan Sungai Long, 43000 Kajang, Cheras. | 38755 | Public bank |
| ☐ Edit Copy Delete | 2 | Travis Lim | travislion@gmail.com | 0196979889 | 44, Lorong Seksyen 2/2, 50000 Kuala Lumpur. | 38EB267 | Bank Islam |
| ☐ Edit Copy Delete | 3 | Vincy | vincy999huaaa@omg.my | 0134720909 | Evergreen Park Cypress, Persiaran SL 1, 43000 Kaja... | 4768922341 | May Bank |

*Figure 3.1.4: View of our "checkout" table*

| | | id | checkout_id | item_id | quantity | price |
|---|---|---|---|---|---|---|
| ☐ | Edit Copy Delete | 1 | 1 | CN01 | 1 | 25.90 |
| ☐ | Edit Copy Delete | 2 | 2 | CN01 | 1 | 25.90 |
| ☐ | Edit Copy Delete | 3 | 2 | NP02 | 1 | 13.20 |
| ☐ | Edit Copy Delete | 4 | 3 | CM02 | 4 | 15.00 |
| ☐ | Edit Copy Delete | 5 | 3 | NC01 | 2 | 15.00 |
| ☐ | Edit Copy Delete | 6 | 3 | SP01 | 1 | 22.50 |
| ☐ | Edit Copy Delete | 7 | 3 | TR02 | 1 | 75.00 |

*Figure 3.1.5: View of the "items_bought" table*

| | | id | name | email | phone | num_pax | service_type | preferred_date | preferred_time | additional_message |
|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | Edit Copy Delete | 1 | Amanda | Amandagraceseow@gmail.com | 0178883636 | 2 | Hair Coloring, HydraFacial | 2024-04-26 | 03:00:00 | |
| ☐ | Edit Copy Delete | 2 | Travis Lim | travislion@gmail.com | 0196979889 | 1 | Brightening Facial, Foot Massage, Back Massage | 2024-04-30 | 11:00:00 | |

*Figure 3.1.6: View of the "service_bookings" table*

| | | id | salutation | enquiry | name | email | phone | subject | subscribe |
|---|---|---|---|---|---|---|---|---|---|
| ☐ | Edit Copy Delete | 1 | Ms | General En | Amanda | Amandagraceseow@gmail.com | 0178883636 | Good morning. May I know is the salon offering any... | 0 |
| ☐ | Edit Copy Delete | 2 | Mdm | Complaints | Vincy | vincy999huaaa@omg.my | 0134720909 | Good morning. I would like to draw your attention ... | 1 |

*Figure 3.1.7: View of the "contact_info" table*

### 3.2. index.php

The home page is first initialized by the following code. The home page is given the title of the salon's name and linked to a CSS file that contains the entire CSS codes required for the website for easier standardization. After moving on to the body, we equipped our homepage with a header (header.php) and a navigation file (nav.php) that will be further elaborated during part 3.8.

```
<!DOCTYPE html>
<html>
<head>
       <title>Z23 Beauty Salon</title>
       <link rel="stylesheet" href="style/webstyles.css">
</head>
<body>
<?php include('includes/header.php'); ?>
<?php include('includes/nav.php'); ?>
```

Following the implementations of the header and navigation file, we proceed to a implement our user login system which links it to another file. The file would be further elaborated in part 3.3.

```
<div class="login-container">
       <?php include("users/login.php")?>
</div>
```

After implementing the user login system, we proceed to check if the user is logged in to their account or not. If the user has managed to login to their account, the home page will display a huge welcome message at the top of the page.

```
<?php
    // Check if welcome message is set in session
    if(isset($_SESSION['welcome_message'])){
        echo "<h1>" . $_SESSION['welcome_message'] . "</h1>";
        // Unset the welcome message after displaying it to avoid displaying it again
        unset($_SESSION['welcome_message']);
    }
?>
```

Moving on, we then proceed to display some information about our salon to our users. The information includes a simple backstory, our salon's vision, and our salon's mission. The brief introductory of our salon is represented through a section of the webpage by using the function 

```
<div class="aboutUs">
	<div class="overlay">
	<h3>ABOUT US:</h3>
	<p>Your beauty is our priority. We started small as a family back in 2001 with the
goal of allowing everyone to become the best versions of themselves.</p>
	<h3>OUR VISION:</h3>
	<p>To help everybody channel their inner beauty and transform it into a profound
confidence. We also see ourselves as a save space with no judgement for men and women
alike.</p>
	<h3>OUR MISSION: </h3>
	<p>We aim to empower men and women alike with their beauty. We want them to thrive
in confidence and feel comfortable in their own skin.</p>
	</div>
</div>
```

Then, we moved on to the photos of our salon. Through the code snippet, it is evident that we used the  to highlight the information. We then separate the specific area into 4 slots and display the photos of our salon. The photos have also been named. It is also notable that a function enlarge(this) will run when the photo is clicked.

```
div class="ourSalon">
	<h2>Our Salon: </h2>
	<div class="row">
	  <div class="column">
	       <img src="photos/salon.png" alt="Waiting Area" onclick="enlarge(this);">
	  </div>
	  <div class="column">
	       <img src="photos/feetMassage.jpg" alt="Foot Massages"
onclick="enlarge(this);">
	  </div>
	  <div class="column">
	       <img src="photos/spaRoom.jpg" alt="Facial Room" onclick="enlarge(this);">
	  </div>
	  <div class="column">
	       <img src="photos/maniPedi.jpeg" alt="Manicure and Pedicures"
onclick="enlarge(this);">
	  </div>
	</div>
```

According to the code snippet below, the  acts as a container for the expanded image and its associated text. The × represents a close button and allows us to close the enlarged photos when clicked upon. The following code is then used to display the expanded

image and its respective text and has been dynamically updated using JavaScript and will be elaborated further below.

```
<!-- The expanding image container -->
       <div class="container">
         <!-- Close the image -->
         <span onclick="this.parentElement.style.display='none'"
class="closebtn">&times;</span>

         <!-- Expanded image -->
         <img id="expandedImg" style="width:100%">

         <!-- Image text -->
         <div id="imgtext"></div>
       </div>
```

The following codes consists of JavaScript, and it creates the function enlarge(imgs). The function is then tasked to retrieve the image and expand it before returning it to the system for display.

```
       <script>
       function enlarge(imgs) {
              // Get the expanded image
              var expandImg = document.getElementById("expandedImg");
              // Get the image text
              var imgText = document.getElementById("imgtext");
              // Use the same src in the expanded image as the image being clicked on from
              grid
              expandImg.src = imgs.src;
              // Use the value of the alt attribute as text inside the expanded image
              imgText.innerHTML = imgs.alt;
              // Show the container element (hidden with CSS)
              expandImg.parentElement.style.display = "block";
       }
       </script>
</div>
```

Then, the final part of the body consists of a container named “Location.” The container displays a picture, a photo of our salon’s location and our details beside it. The details include our salon’s opening hours, our contact number, and our exact location. The “row” container is what allows us to display our photo beside our salon’s details with the help of some CSS.

```
<div class="location">
	<div class="row">
		<div class="columnLocation">
			<img src="photos/location.png" width="350px" height="350px">
		</div>
		<div class="columnLocation">
			<h3>Opening Hours:</h3>
			<p>9:00 am - 9:00 pm</p>
			<h3>Contact Us:</h3>
			<p>012-7389779</p>
			<h3>Location:</h3>
			<p>Sungai Long,</p>
			<p>23 GR FL, Persiaran SL3/2,</p>
			<p>Bandar Sungai Long,</p>
			<p>43000 Kajang, Selangor</p>
		</div>
	</div>
</div>
```

Last but not least, we insert our footer into our page to display our specially designed footer that will be further elaborated in another part (part 3.8).

```
<?php include('includes/footer.php'); ?>
</br>
</body>
</html>
```

### 3.3. users

#### 3.3.1. users/login.php

First and foremost, we started off our login page by starting a session. The php file will then connect to a database before checking if the users email and password tally with each other. If the password and email tallies a welcome message will be saved in a $welcome_message variable that had been defined right after initializing a session. However, the system will echo our that the login had failed if the email and password given does not tally with each other.

```
<?php

	session_start();

	// Define a variable to store the welcome message
	$welcome_message = '';

	if($_SERVER['REQUEST_METHOD'] == 'POST'){
		$email = $_POST['email'];
		$password = $_POST['password'];

		$conn = mysqli_connect("localhost", "root", "", "Z23_Beauty_Salon");

		// Check connection
		if ($conn->connect_error) {
			die("Connection failed: " . $conn->connect_error);
		}

		$query = "SELECT * FROM users WHERE email = '$email' AND
password='$password'";

		$result = mysqli_query($conn,$query);

		if (mysqli_num_rows($result)===1){
			$_SESSION['email'] = $email;
			$welcome_message = "Welcome " . htmlspecialchars($email) . "!";
			$_SESSION['welcome_message'] = $welcome_message;
			header('Location: /Z23BeautySalon/index.php');
			exit();
		}else{
			$error = 'Invalid email or password. Login failed.';
			echo $error;
		}
		mysqli_close($conn);
	}
?>
```

Moving on, the following HTML codes create a dropdown button and set it as the Log In button. Through the following if-else code, the user will be able to see the Login option and the create account option if the user has not logged into the account. On the other hand, the user will only see the logout option if the user has already logged into their account.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login and Create Account</title>
       <link rel="stylesheet" href="../style/webstyles.css">
</head>
<body>
    <div class="dropdown">
        <button onclick="toggleDropdown()" class="dropbtn">Log in</button>
        <div id="myDropdown" class="dropdown-content">
                    <?php
            // Check if user is logged in
            if(isset($_SESSION['email'])) {
                // If logged in, display logout option
                echo '<a href="#" onclick="logout()">Log Out</a>';
            } else {
                // If not logged in, display login option
                 echo '<a href="#" onclick="showLoginForm()">Login</a>';
                 echo '<a href="#" onclick="createAccount()">Create Account</a>';
            }
            ?>
        </div>
    </div>
```

Furthermore, the container form with the id, “login-form” is used to display the form for the users to key in their email and password. The create account container does the same function except it redirects users to the createAccount.php to display the sign-up form.

```
<div class="container" id="login-form">
        <h2>Login</h2>
        <form id="loginform" method="post">
            <input type="text" name="email" placeholder="Email" required>
            <input type="password" name="password" placeholder="Password" required>
            <button type="submit" value="login">Login</button>
        </form>
    </div>

    <div class="container" id="createAccount">
        <link href = "users/createAccount.php">
    </div>
```

Moving on, the code below shows the JavaScript code that allows the function to run. The function toggleDropdown() allows the login in button to behave as a toggle button or rather known as a dropdown button. The showLoginForm() functions similarly in name and allows the login form to be displayed to the users on the website. The logout() and createAccount() function behaves in a similar manner excepts it works by directing users to another page. The logout() function will redirect users to the logout.php, whereas the createAccount() function would redirect users to the createAccount.php.

Furthermore, the window.onclick = function(event) functions in a manner where users will be able to close the login button by clicking anywhere outside of the container. This allows users to close the button easily in case of any misclicks. The code runs together with the function closeDropdowns() to ensure a smooth process for users to close the login button.

```
<script>
		function toggleDropdown() {
			var dropdownContent = document.getElementById("myDropdown");
			if (dropdownContent.classList.contains("show")) {
				dropdownContent.classList.remove("show");
			} else {
				closeDropdowns(); // Close other dropdowns if open
				dropdownContent.classList.add("show");
			}
		}

		function showLoginForm() {
			document.getElementById("login-form").style.display = "block";
		}

		function logout() {
			// Implement and run logout.php
			window.location.href = "/Z23BeautySalon/users/logout.php";
		}

		function createAccount() {
			// Redirect to createaccount.php
			window.location.href = "users/createaccount.php";
		}

		// Close the dropdown if the user clicks outside of it
		window.onclick = function(event) {
			if (!event.target.matches('.dropbtn')) {
				closeDropdowns();
			}
		}

		function closeDropdowns() {
			var dropdowns = document.getElementsByClassName("dropdown-content");
			for (var i = 0; i < dropdowns.length; i++) {
				var openDropdown = dropdowns[i];
				if (openDropdown.classList.contains('show')) {
					openDropdown.classList.remove('show');
				}
			}
		}
	</script>
```

### 3.3.2. users/createAccount.php

For the following codes in the createAccount.php, the php file will first initialize the used variables and an array to hold error messages. Then the page will perform validation by checking if the created account form is empty and show the relevant error message if applicable.

```
<body>
	<?php include('../includes/header.php'); ?>
	<?php include('../includes/nav.php'); ?>
	<?php include('../includes/database.php'); ?>

	<?php
		if ($_SERVER['REQUEST_METHOD'] === 'POST') {
			// Initialize variables
			$name = $_POST['name'] ?? '';
			$email = $_POST['email'] ?? '';
			$phone = $_POST['phone'] ?? '';
			$password = $_POST['password'] ?? '';

			// Initialize an array to hold error messages
			$errors = [];

			// Perform validation
			if (empty($name)) {
				$errors['name'] = 'Your name is required.';
			}
			if (empty($email) || !filter_var($email, FILTER_VALIDATE_EMAIL)) {
				$errors['email'] = 'A valid email is required.';
			}
			$phone = str_replace([' ', '-', '(', ')'], '', $phone); // Strip
common formatting characters
			if (empty($phone)) {
				$errors['phone'] = 'Phone number is required.';
			} elseif (!ctype_digit($phone)) {
				$errors['phone'] = 'Phone number must be numeric.';
			}
			if (empty($password)) {
				$errors['password'] = 'A password of your choice is
required.';
			}
```

Once all the validation code has been filtered through, the user's data is then inserted into our database into our user's table. Once the data is inserted, a success message will be executed where the users can view the details they entered before being asked to return to the homepage to login to the system with their new account. If an error occurs halfway through the codes, the corresponding error message will be displayed to the users. If not, the page will display the login form instead.

```
        // Check if there are any errors
        if (empty($errors)) {
                $stmt = $conn->prepare("INSERT INTO users (name, email, phone,
                password) VALUES (?, ?, ?, ?)");
                $stmt->bind_param("ssss", $name, $email, $phone, $password);

                if ($stmt->execute()) {
                  // Success message

                echo '<div id="contentWrapper" class="content">';
                echo "<h2>Submission Details</h2>";
                echo "Name: " . htmlspecialchars($name) . "<br>";
                echo "Email: " . htmlspecialchars($email) . "<br>";
                echo "Phone: " . htmlspecialchars($phone) . "<br>";
                echo "Password: " . htmlspecialchars($password). "<br><br>";

                echo "Thank you for signing up with us! Your account has been
succesfully created!";
                echo "<br>";
                echo "Please return to the home page and login into your new
account!";
                echo "<br><br>";
                echo '<button><a href="/Z23BeautySalon/index.php">Return to Home
Page</a></button>';
                echo "</div>";
                } else {
                        // Error message
                        echo "Error: " . $stmt->error;
                }

                } else {
                        // Redisplay the form, with error messages
                        include('form.php');
                        }
                } else {
                        // Display the form for the first time
```

### 3.3.3. users/form.php

The following codes are used to create the form for users to create an account. It has also been linked to the createAccount.php. The form consists of 3 inputs[type=”text”] for the user’s name, email, and password. However, the input type for the user's phone number has been set to as input[type=”tel”]. It is notable that each input field will display an error message related to the “name” field if errors occur. The error message will be retrieved from the $errors array and displayed within the container for users to view.

```
<!-- Form for users to create account -->
<div id="contentWrapper" class="content">
    <form id="createAccount" action="createAccount.php" method="post">

        <label for="nam">Name: </label>
        <input type="text" id="nam" name="name" value="<?= htmlspecialchars($name) ?>">
        <div id="nameError" class="error"><?= $errors['name'] ?? '' ?></div><br>

        <label for="email">E-mail: </label>
        <input type="text" id="email" name="email" value="<?= htmlspecialchars($email) ?>">
        <div id="emailError" class="error"><?= $errors['email'] ?? '' ?></div><br>

        <label for="phone">Phone Number: </label>
        <input type="tel" id="phone" name="phone" value="<?= htmlspecialchars($phone) ?>">
        <div id="phoneError" class="error"><?= $errors['phone'] ?? '' ?></div><br>

        <label for="password">Password: </label>
        <input type="text" id="password" name="password" value="<?=
htmlspecialchars($password) ?>">
              <div id="passwordError" class="error"><?= $errors['password'] ??
'' ?></div><br>

        <input type="submit" value="Create Account">
    </form>

</div>
```

### 3.3.4. users/logout.php

This file is used when the users decide to log out of their account. The session_destroy() function will destroy all data registered to the following session and effectively logging the user out of their account. Once the session is destroyed, the header() function directs that user back to the home page once the logout process is completed. Last but not least, the exit() function terminates the script execution once the user is redirected. This allows us to ensure that no further code is executed after redirection and prevents any unexpected behavior from occurring. The code snippet from the logout.php can be seen below:

```
<?php
session_start();

session_destroy();

header("Location: /Z23BeautySalon/index.php");
exit();
?>
```

### 3.4. services

#### 3.4.1. index.php

The code snippet initializes parameters like servername, username, password, and dbname for establishing a connection with the SQL database. Afterward, it checks the connection status to ensure there are no errors encountered during the connection process. In addition, it displays the system header and navigation by including relevant files.

```php
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "Z23_Beauty_Salon";

    //Connect the database
    $conn = new mysqli($servername, $username ,$password ,$dbname);

    //Check connection
    if($conn->connect_error){
        die("Connection failed: ".$conn->connect_error);
    }
    include("../includes/header.php");
    include("../includes/nav.php");
?>
```

After setting up the SQL connection, the code will show a button allowing users to click on it, redirecting them to "booking.php." Once redirected, the header of the service list will be displayed. Subsequently, the code will include PHP files to showcase the service list title and navigation. The service list navigation operates akin to the item list navigation, assisting users in navigating through various service categories.

```php
<div id="container">
    <!-- Display a BOOKING button-->
    <button class='button'><a href='booking.php'>Booking</a></button>
    <!-- Display Title -->
    <h2 class="title">
        Services
    </h2>
    <!-- Display Services Navigation -->
    <?php include('serviceNav.php'); ?>
```

Following that, it will display the categories of services such as HAIR, FACIAL, MANICURES AND PEDICURES, and MASSAGE. Then, it will showcase the pictures retrieved from the "photos" file. All photos are from Explore Pinterest (n.d.). Afterward, it will present the types of services provided by retrieving the data from the database. All information, including type and price range from the specific category, will be echoed out.

```
<!-- Display All Hair -->
    <h3 class="itemsList-title">
        HAIR
    </h3>

    <div class="service-container">
        <?php
            // Fetch items from the database
            $sql = "SELECT * FROM service WHERE category='HAIR'";
            $result = $conn->query($sql);

            // Generate HTML markup for each item
            // Display every items
            if ($result->num_rows > 0) {

                //Initialize item pic number count
                $serviceNum = 1;

                // Output data of each row
                while($row = $result->fetch_assoc()) {
                    echo "<div class='item'>";
                    $image_url = "../photos/serviceHair".$serviceNum.".jpeg";
                    echo "<img src='" . $image_url ."' width='140' height='200'>";
                    echo "<h5>" . $row['type'] . "</h4>";
                    echo "<p>Price Range: " . $row['priceRange'] . "</p>";
                    echo "</div>";
                    $serviceNum++;
                }
            } else {
                echo "0 results";
            }
        ?>
    </div>
```

After iterating to display all services in each category, it will close the database connection.

```
            // Close the database connection
            $conn->close();
        ?>
    </div>
```

At the end of this service list page, it will display the footer.

```
    <!-- Display Footer -->
    <?php include('../includes/footer.php'); ?>
    </br>
</body>
```

### 3.4.2. service/serviceFacial/php | serviceHair.php | serviceMAP.php | serviceMassage.php

For these four PHP files, they execute the same task: displaying the service list for a specific category. Other functionalities, such as the system header, system navigation, system footer, and the booking button function, remain consistent with service/index.php. The only distinction between them is the code snippet indicating the category of service.

```
<?php
                // Fetch items from the database
                $sql = "SELECT * FROM service WHERE category='MASSAGE'";
                $result = $conn->query($sql);

                // Generate HTML markup for each item
                // Display every items
                if ($result->num_rows > 0) {

                    //Initialize item pic number count
                    $serviceNum = 1;

                    // Output data of each row
                    while($row = $result->fetch_assoc()) {
                        echo "<div class='item'>";
                        $image_url = "../photos/serviceMassage".$serviceNum.".jpeg";
                        echo "<img src='" . $image_url."' width='140' height='200'>";
                        echo "<h5>" . $row['type'] . "</h4>";
                        echo "<p>Price Range: " . $row['priceRange'] . "</p>";
                        echo "<button class='button'><a
href='index.php'>Booking</a></button>";
                        echo "</div>";
                        $serviceNum++;
                    }
                }else {
                    echo "0 results";
```

### 3.4.3. service/serviceNav.php

This navigation is employed to facilitate users in swiftly viewing specific categories of services.

```
<nav id="serviceNav">
    <hr>
    <a href="../service/">All Services</a>
    |
    <a href="../service/serviceHair">Hair</a>
    |
    <a href="../service/serviceFacial">Facial</a>
    |
    <a href="../service/serviceMAP">Manicures and Pedicures</a>
    |
    <a href="../service/serviceMassage">Massage</a>
    <hr>
</nav>
```

### 3.4.4. service/booking.php

```
<!-- Service booking form -->
<div class="serviceContainer">
    <h1>Service Booking</h1>
    <form id="serviceBookingForm" class="checkout-form" action="servicePostMessage.php" method="post">
        <div class="form-group">
            <label for="bookingName">Name:</label>
            <input type="text" id="bookingName" name="name">
            <div id="bookingNameError" class="error"></div>
        </div>
        <div class="form-group">
            <label for="bookingEmail">E-mail:</label>
            <input type="email" id="bookingEmail" name="email">
            <div id="bookingEmailError" class="error"></div>
        </div>
        <div class="form-group">
            <label for="bookingPhone">Phone Number:</label>
            <input type="tel" id="bookingPhone" name="phone">
            <div id="bookingPhoneError" class="error"></div>
        </div>
        <div class="form-group">
            <label for="numPax">Number of pax:</label>
            <input type="number" id="numPax" name="numPax" min="1" max="5" value="1">
            <div id="numPaxError" class="error"></div>
        </div>

<div class="form-group">
    <label for="serviceType"><b>Service Type:</b></label>
        <ul class="no-bullets"><i><b>HAIR:</b></i><br>
        <li><input type="checkbox" name="serviceType[]" value="Hair Cut"> Hair Cut </li>
        <li><input type="checkbox" name="serviceType[]" value="Hair Coloring"> Hair Coloring </li>
        <li><input type="checkbox" name="serviceType[]" value="Hair Treatment"> Hair Treatment </li></ul>
        <ul class="no-bullets"><i><b>FACIAL:</b></i><br>
        <li><input type="checkbox" name="serviceType[]" value="Classic Facial"> Classic Facial </li>
        <li><input type="checkbox" name="serviceType[]" value="Acne Facial"> Acne Facial </li>
        <li><input type="checkbox" name="serviceType[]" value="HydraFacial"> HydraFacial </li>
        <li><input type="checkbox" name="serviceType[]" value="Brightening Facial"> Brightening Facial </li>
        <li><input type="checkbox" name="serviceType[]" value="HydraBrightening Facial"> HydraBrightening Facial </li>
        <li><input type="checkbox" name="serviceType[]" value="AntiAging Facial"> AntiAging Facial </li></ul>
        <ul class="no-bullets"><i><b>MANICURES/PEDICURES:</b></i><br>
        <li><input type="checkbox" name="serviceType[]" value="Manicures"> Manicures </li>
        <li><input type="checkbox" name="serviceType[]" value="Pedicures"> Pedicures </li>
        <li><input type="checkbox" name="serviceType[]" value="Manicures And Pedicures"> Manicures And Pedicures</li></ul>
        <ul class="no-bullets"><i><b>MASSAGE:</b></i><br>
        <li><input type="checkbox" name="serviceType[]" value="Foot Massage"> Foot Massage </li>
        <li><input type="checkbox" name="serviceType[]" value="Back Massage"> Back Massage </li>
        <li><input type="checkbox" name="serviceType[]" value="Head Massage"> Head Massage </li></ul>
        <div id="serviceTypeError" class="error"></div>
</div>

<br>
<div class="form-group">
    <label for="bookingDate">Preferred Date:</label>
    <!-- Set the default value -->
    <?php
        // Check if the date is submitted and not empty
        $defaultDate = isset($_POST['booking-date']) ? $_POST['booking-date'] : '';

        // Get the current date
        $currentDate = date("Y-m-d");
    ?>
    <input type="date" id="bookingDate" name="booking-date" value="<?php echo $defaultDate; ?>" min="<?php echo $currentDate; ?>">
    <!-- Error message display -->
    <div id="bookingDateError" class="error"></div>
</div>
```

```
<div class="form-group">
    <label for="time">Preferred Time:</label>
    <select id="time" name="time">
        <option value="" selected disabled>Please choose a time</option>
        <option value="9:00 AM">9:00am</option>
        <option value="10:00 AM">10:00am</option>
        <option value="11:00 AM">11:00am</option>
        <option value="12:00 PM">12:00pm</option>
        <option value="1:00 PM">1:00pm</option>
        <option value="2:00 PM">2:00pm</option>
        <option value="3:00 PM">3:00pm</option>
        <option value="4:00 PM">4:00pm</option>
        <option value="5:00 PM">5:00pm</option>
        <option value="6:00 PM">6:00pm</option>
        <option value="7:00 PM">7:00pm</option>
    </select>
    <div id="timeError" class="error"></div>
</div>

    <div class="form-group">
        <label for="bookingMessage">Additional Message:</label>
        <textarea id="bookingMessage" name="booking-message" rows="3" cols="3"></textarea>
    </div>

    <div class="button-container">
        <input type="button" value="Confirm" class="styled-button" onclick="validateAndSubmit()">
    </div>
    </div>
</form>
```

The main container “serviceContainer” holds the form with an action to “servicePostMessage.php” where the form data will be submitted. Users input their name, email, phone number, and the number of people(numPax) for the appointment. The form features checkboxes for selecting various service types under categories such as “HAIR”, “FACIAL”, “MANICURES/PEDICURES”, and “MASSAGE”. Users can choose their preferred date and time from a dropdown list. Additional instructions can be added in the “bookingMessage” textarea.

Upon clicking the “Confirm” button, the “validateAndSubmit()” function handles client-side validation for required fields and submits the form to “servicePostMessage.php”. Error messages appear if validation fails, ensuring accurate submissions and enhancing the user experience.

```
function validateAndSubmit() {
    var isValid = validationForm();
    if (!isValid) {
        return; // Prevent form submission if validation fails
    }
    // Display alert message
    window.alert("Thank you for choosing us.\nNote that no change can be made.
                \nFeel free to contact us if you wish to make any changes!");

    // Form submission logic here
    document.getElementById("serviceBookingForm").submit();
}
```

The validateAndSubmit() function is triggered when the “Confirm” button is clicked. It first calls “validationForm()” to perform client-side form validation. If the “validationForm()” function returns true, it indicates all fields are valid, therefore, an alert message is displayed thanking the user for choosing the service. After that, the form “serviceBookingForm” is submitted by using “document.getElementById("serviceBookingForm"). submit()”.

```
function validationForm() {
    var isValid = true;
    var form = document.getElementById('serviceBookingForm');

    // Clear previous error messages
    document.querySelectorAll('.error').forEach(function (error) {
        error.textContent = '';
    });

    // Validate name
    if (form['bookingName'].value.trim() === '') {
        document.getElementById('bookingNameError').textContent = 'Name is required.';
        isValid = false;
    }

    // Validate email
    if (form['bookingEmail'].value.trim() === '') {
        document.getElementById('bookingEmailError').textContent = 'Email is required.';
        isValid = false;
    } else if (!(/^\S+@\S+\.\S+$/.test(form['bookingEmail'].value))) {
        document.getElementById('bookingEmailError').textContent = 'Email is not valid.';
        isValid = false;
    }

    // Validate phone number
    if (form['bookingPhone'].value.trim() === '') {
        document.getElementById('bookingPhoneError').textContent = 'Phone number is required.';
        isValid = false;
    } else if (!/^\d+$/.test(form['bookingPhone'].value)) {
        document.getElementById('bookingPhoneError').textContent = 'Phone number must be numeric.';
        isValid = false;
    }
```

```
// Validate number of pax
var numPax = parseInt(form['numPax'].value);
if (isNaN(numPax) || numPax < 1 || numPax > 5) {
    document.getElementById('numPaxError').textContent = 'Number of pax must be between 1 and 5.';
    isValid = false;
}

// Validate service type
var serviceTypes = form.querySelectorAll('input[name="serviceType[]"]:checked');
if (serviceTypes.length === 0) {
    document.getElementById('serviceTypeError').textContent = 'Please select at least one service type.';
    isValid = false;
}
```

```
// Validate preferred date
var bookingDateInput = document.getElementById('bookingDate');
var bookingDateValue = bookingDateInput.value.trim();

if (bookingDateValue === '') {
    document.getElementById('bookingDateError').textContent = 'Preferred date is required.';
    isValid = false;
} else {
    // Convert the selected date string to a Date object
    var selectedDate = new Date(bookingDateValue);

    // Get today's date
    var currentDate = new Date();

    // Compare the selected date with today's date
    if (selectedDate <= currentDate) {
        document.getElementById('bookingDateError').textContent = 'Please select a valid date.';
        isValid = false;
    } else {
        // Clear any previous error messages
        document.getElementById('bookingDateError').textContent = '';
    }
}


// Validate preferred time
if (form['time'].value.trim() === '') {
    document.getElementById('timeError').textContent = 'Preferred time is required.';
    isValid = false;
}

return isValid;
```

The validationForm() function handles the actual form validation. It sets “isValid” to true initially. Then, it retrieves the form element and clears any previous error messages. It proceeds to validate name, email, phone number, number of pax, service type, preferred date, and preferred time.

Subsequently, it validates each input field which is ensuring the name is not empty, verifying a valid email and phone number format, checking if at least one service type is selected, limiting the number of pax between 1 and 5, confirming a future date selection, and validating the chosen time slot.

For each validation, if an error is found, it sets “isValid” to false and updates the corresponding error message displayed to the user. At the end, it returns “isValid”, which is used in “validateAndSubmit()” to determine whether to submit the form. This ensures that all required fields are filled out correctly and provides immediate feedback to the user if there are any validation errors, helping them to correct their inputs before submitting the form.

### 3.4.5. service/servicePostMessage.php

```php
<div class="service-details">
    <h2>Thank you for choosing us!</h2>
    <p>Your service details are:</p>
    <?php
        $servername = "localhost";
        $username = "root";
        $password = "";
        $dbname = "Z23_Beauty_Salon";

        //Connect the database
        $conn = new mysqli($servername, $username, $password, $dbname);

        //Check connection
        if($conn->connect_error){
            die("Connection failed: ".$conn->connect_error);
        }

        // Retrieve form data
        $name = $_POST['name'];
        $email = $_POST['email'];
        $phone = $_POST['phone'];
        $numPax = $_POST['numPax'];
        $serviceType = implode(", ", $_POST['serviceType']);
        $date = $_POST['booking-date'];
        $time = $_POST['time'];
        $message = $_POST['booking-message'];
```

We retrieve the form data submitted via POST method, which includes the user's name, email, phone number, number of persons (numPax), service type, preferred date, preferred time, and additional message. For service type, we imploded into a comma-separated string if multiple options are selected.

```php
// SQL query to insert data into the table
$sql = "INSERT INTO service_bookings (name, email, phone, num_pax,
service_type, preferred_date, preferred_time, additional_message)
VALUES ('$name', '$email', '$phone', '$numPax', '$serviceType',
        '$date', '$time', '$message')";
```

Next, we construct an SQL INSERT query to insert this form data into the service_bookings table, mapping each form field to its corresponding column in the table.

```php
// Execute the query
if ($conn->query($sql) === TRUE) {
    // Build the query string
    $queryString = http_build_query([
        'name' => $name,
        'email' => $email,
        'phone' => $phone,
        'numPax' => $numPax,
        'serviceType' => $serviceType,
        'date' => $date,
        'time' => $time,
        'message' => $message
    ]);

    // Redirect to the thank you page with the query string
    header('Location: thank-you.php?' . $queryString);
    exit();
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

// Close the database connection
$conn->close();
```

Upon successful insertion, we build a query string using “http_build_query()” to pass the form data to “thank-you.php” for displaying the order details. The user is then redirected to thank-you.php using “header ('Location: thank-you.php?' . $queryString)”. In case of any errors during this process, an error message is displayed. Lastly, we close the database connection.

### 3.4.6. service/thank-you.php

```
<div class="service-details">
    <div class="head">
        <span class="icon">
            <i class="fa fa-heart"></i>
        </span>
        <h2>Thank you for choosing us!</h2>
    </div>
    <p><center>Your service details are:</center></p>
    <div class="order-details">
        <p>Name: <?php echo $_GET['name']; ?></p>
        <p>Email: <?php echo $_GET['email']; ?></p>
        <p>Phone: <?php echo $_GET['phone']; ?></p>
        <p>Number of Pax: <?php echo $_GET['numPax']; ?></p>
        <p>Service Type:
            <?php
            if (isset($_GET['serviceType']) && is_array($_GET['serviceType']) && count($_GET['serviceType']) > 0) {
                echo implode(', ', $_GET['serviceType']);
            } else {
                echo 'None selected';
            }
            ?>
        </p>

        <p>Date: <?php echo $_GET['date']; ?></p>
        <p>Time: <?php echo $_GET['time']; ?></p>
        <p>Message: <?php echo $_GET['message']; ?></p>
    </div>

    <div class="button-container">
        <input type="button" class="styled-button" onclick="window.location.href='../index.php'" value="Done">
    </div>
</div>
```

After submitting a service booking form, users are presented with a confirmation message about the order details. The service-details div contains a header section (head) that expresses gratitude to the user for choosing the service.

Within the order-details div, all specific details entered by the user are displayed. This includes their name, email, phone number, number of persons, selected service type, chosen date, time, and any additional message. If the user selects multiple service types, they are listed with commas using PHP's implode function, ensuring a clear and organized presentation of their inputs.

Hence, a “Done” button in the button-container allows users to easily return to the home page after reviewing their submitted service details. This convenient navigation feature enhances the user experience, providing a seamless way to continue exploring the website.

### 3.5. items

#### 3.5.1. item/index.php

Reviewing the provided code snippet below, it displays the system header and navigation through the inclusion of files. Subsequently, the servername, username, password, and dbname for the database are initialized for SQL connection purposes. The code then verifies the connection status to ascertain whether any errors occurred during the process.

```
        <?php
        // Start a session
        session_start();

        include("../includes/header.php");
        include("../includes/nav.php");

        $servername = "localhost";
        $username = "root";
        $password = "";
        $dbname = "Z23_Beauty_Salon";

        //Connect the database
        $conn = new mysqli($servername, $username ,$password ,$dbname);

        //Check connection
        if($conn->connect_error){
            die("Connection failed: ".$conn->connect_error);
        }
    ?>
```

Following that, the code seamlessly integrates the cart logo, enhancing user experience by providing convenient access to review their cart contents.

```
        <div id="cart-logo">
           <a href="../wishlist/index.php">
               <svg class="w-6 h-6 text-gray-800 dark:text-white" aria-hidden="true"
xmlns="http://www.w3.org/2000/svg" width="24" height="24" fill="none" viewBox="0 0 24 24">
                   <path stroke="currentColor" stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" d="M5 4h1.5L9 16m0 0h8m-8 0a2 2 0 1 0 0 4 2 2 0 0 0 0-
4Zm8 0a2 2 0 1 0 0 4 2 2 0 0 0 0-4Zm-8.5-3h9.25L19 7H7.312"/>
               </svg>
           </a>
       </div>
```

Next, the code proceeds to display the title and navigation of the item list. For this navigation part, it retrieves from **itemNav.php**. Within the item list navigation, users can effortlessly navigate to specific product categories simply by clicking on them, streamlining the browsing experience.

```
<!-- Display Title -->
<h2 class="title">
        Products
</h2>

<!-- Display Item Navigation -->
<?php include('itemNav.php'); ?>
```

Even if users do not interact with the navigation, all products will still be displayed in the item list. The item list organizes items according to their categories (Eg. Hair and Facial). For instance, when the Hair category is selected, the system retrieves the necessary data from the item table and filters it based on the 'HAIR' category. Using a while loop, the system displays each item one by one.

```
<!-- Display All Hair Product -->
        <h3 class="itemsList-title">
            HAIR</h3>

        <div class="items-container">
            <?php
                // Fetch items from the database
                $sql = "SELECT id , image_url ,title, type, price, quantity FROM item
WHERE category='HAIR'";
                $result = $conn->query($sql);

                // Generate HTML markup for each item
                // Display every items
                if ($result->num_rows > 0) {
                    // Output data of each row
                    while($row = $result->fetch_assoc()) {
                        echo "<div class='item' item-id='" . $row['id'] . "'>";
                        echo "<h4>". $row['id'] . "</h4>";
                        echo "<h5>" . $row['type'] . "</h5>";
                        echo "<img src='" . $row['image_url'] . "' alt='" .
$row['title'] . "' width='140' height='200'>";
                        echo "<h6>" . $row['title'] . "</h6>";
                        echo "<p>Price: RM" . $row['price'] . "</p>";
                        echo "<p>Quantity: " . $row['quantity'] . "</p>";
                        echo "</div>";
                    }
                } else {
                    echo "0 results";
                }
            ?>
        </div>
```

For the item preview section, it is split into a specific file called **itemPreview.php**.

```
<div id="items-preview">
        <?php include('itemPreview.php');?>
</div>
```

The webpage concludes by invoking the JavaScript file, **itemScript.js**, and a footer section.

```
    <script src="itemScript.js"></script>
    <?php include('../includes/footer.php'); ?>
    <br>
</body>
</html>
```

### 3.5.2. item/itemFacial.php | item/itemHair.php | item/itemNail.php

The `itemHair.php` file is crafted to filter and showcase products solely from the hair category, ensuring that only items designated to the hair category are exhibited on this specific webpage. This procedure will be replicated in the `itemNail.php` and `itemFacial.php` files as well, with the only variation being the category of products to display.

```
<!-- Display All Shampoo -->
        <h3 class="itemsList-title">
            HAIR</h3>

        <div class="items-container">
            <?php
                // Fetch items from the database
                $sql = "SELECT id , image_url ,title, type, price, quantity FROM
item WHERE category='HAIR'";
                $result = $conn->query($sql);

                // Generate HTML markup for each item
                // Display every items
                if ($result->num_rows > 0) {
                    // Output data of each row
                    while($row = $result->fetch_assoc()) {
                        echo "<div class='item' item-id='" . $row['id'] . "'>";
                        echo "<h4>". $row['id'] . "</h4>";
                        echo "<h5>" . $row['type'] . "</h5>";
                        echo "<img src='" . $row['image_url'] . "' alt='" .
$row['title'] . "' width='140' height='200'>";
                        echo "<h6>" . $row['title'] . "</h6>";
                        echo "<p>Price: RM" . $row['price'] . "</p>";
                        echo "<p>Quantity: " . $row['quantity'] . "</p>";
                        echo "</div>";
                    }
                } else {
                    echo "0 results";
                }
```

### 3.5.3. item/itemNav.php

This navigation facilitates users in promptly discerning the available product categories. When clicked, it functions as a link, directing users to the corresponding categories within the item list.

```
<nav id="itemNav">
    <hr>
    <a href="../item/itemList">All Products</a>
    |
    <a href="../item/itemHair">Hair</a>
    |
    <a href="../item/itemNail">Nail Tool</a>
    |
    <a href="../item/itemFacial">Facial</a>
    <hr>
</nav>
```

The functionality of these four files is the same, with the only difference being the items listed. The "All Products" page displays all items from the three categories, while the "Hair" page exclusively displays products from the hair category. Similarly, the "Nail Tool" page only displays items from the nail tool category, and the "Facial" page exclusively displays items from the facial category.

### 3.5.4. item/itemPreview.php

This PHP script begins by establishing a connection to the SQL database, following the same steps as in the itemList.php file. It then proceeds to select all information related to items from the item table in the database.

```
// Fetch items from the database
    $sql = "SELECT * FROM item";
    $result = $conn->query($sql);
```

Before displaying the item's details, this script presents a button that lets the user exit the item details section. This button provides users with the option to navigate away from the item details if they choose not to view it further. Using the retrieved data from the database, this script displays various details of the item to the user, such as the item ID, item type, image, item title, item description, price, and quantity. Additionally, it includes a button that allows the user to add the item to their cart. When the user clicks this button, it triggers an action that connects to **wishlist/index.php**.

```
// Generate HTML markup for each item
    // Display every item
    if ($result->num_rows > 0) {
        // Output data of each row
        while($row = $result->fetch_assoc()) {
            echo "<div class='preview' target-id='" . $row['id'] . "'>";
            echo "<div class='close-button' onclick='closePreview()'>X</div>";
            echo "<h4>". $row['id'] . "</h4>";
            echo "<h5>" . $row['type'] . "</h5>";
            echo "<img src='" . $row['image_url'] . "' alt='" . $row['title'] . "'
width='140' height='200'>";
            echo "<h6>" . $row['title'] . "</h6>";
            echo "<p>" . $row['description'] . "</p>";
            echo "<p>Price: RM" . $row['price'] . "</p>";
            echo "<p>Quantity: " . $row['quantity'] . "</p>";
            echo "<form class='addToCartForm' action='../wishlist/index.php'
method='post'>";
            echo "<input type='hidden' name='item-id' value='" . $row['id'] . "'>";
            echo "<button type='button' class='button addToCartBtn'>Add to Cart</button>";
            echo "</form>";
            echo "</div>";
        }
    } else {
        echo "0 results";
    }
```

Finally, the script concludes by closing the connection with the SQL database, ensuring that all database resources are properly released, and the connection is gracefully terminated.

```
    // Close the database connection
    $conn->close();
?>
```

### 3.5.5. item/itemScript.js

By selecting the element with the ID "items-preview" and setting its display attribute to "none," this JavaScript function effectively hides the element from view. It is designed to activate in response to a user-initiated click of the 'X' button, offering a way to end the item details preview area.

```
function closePreview() {
    document.querySelector('#items-preview').style.display = 'none';
}
```

The elements with the class "item" inside the container with the class "items-container" are all selected by this JavaScript code. It gives every item a click event listener. Upon clicking an item, its unique item ID is retrieved. It then searches through all items in the "items-preview" container that have the class "preview" and iterates through them. A preview displays the item preview container and the related preview element if its target ID matches the ID of the targeted item; if not, the preview element is hidden. By using this approach, users can click on an item to view its details.

```
document.querySelectorAll('.items-container .item').forEach(product => {
    product.addEventListener('click', () => {
        let id = product.getAttribute('item-id');
        let previews = document.querySelectorAll('#items-preview .preview');
        previews.forEach(preview => {
            if (preview.getAttribute('target-id') === id) {
                document.querySelector('#items-preview').style.display = 'block';
                preview.style.display = 'block';
            } else {
                preview.style.display = 'none';
            }
        });
    });
});
```

This line of code selects all HTML elements with the class "addToCartBtn" and stores them in the variable "addToCartButtons", allowing further manipulation or event handling with those buttons.

```
 // Get all "Add to Cart" buttons
var addToCartButtons = document.querySelectorAll('.addToCartBtn');
```

In this part, it iterates over each "Add to Cart" button on the webpage and attaches a click event listener to it. An XMLHttpRequest is used to locate the closest form element with the class "addToCartForm," retrieve the form data, and send it via a POST request to "../wishlist/index.php" when a button is clicked. It shows an alert message after it receives a response, informing the user whether the item was successfully added to the cart or if there was a problem. If the item is added successfully, it also adds an event listener to a link with the ID "nextPageLink" so that when the link is clicked, the user is redirected to the cart page.

```
// Loop through each button and attach event listener
addToCartButtons.forEach(function(button) {
    button.addEventListener('click', function() {
        var form = this.closest('.addToCartForm');
        var formData = new FormData(form);

        var xhr = new XMLHttpRequest();
        xhr.open('POST', '../wishlist/index.php', true);
        xhr.onreadystatechange = function() {
            if (xhr.readyState === XMLHttpRequest.DONE) {
                if (xhr.status === 200) {
                    // Item successfully added to cart
                    alert('Item added to cart!');

                    // Redirect to the next page when the cart logo is clicked
                    document.getElementById('nextPageLink').addEventListener('click', function() {
                        window.location.href = '../wishlist/index.php';
                    });
                } else {
                    alert('Failed to add item to cart. Please try again later.');
                }
            }
        };
        xhr.send(formData);
    });
});
```

### 3.6. wishlist

#### 3.6.1. wishlist/index.php

The index.php is used to show the items that were added to cart by the user in the item list. A session is started to continue the session initiated in the previous page. It is then connected to the database to retrieve the item details. Also, to improve the user's experience in browsing the webpages, the index.php is linked with the header.php and nav.php.

```php
<?php
session_start();

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Z23_Beauty_Salon";

// Connect to the database
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Include header and navigation
include("../includes/header.php");
include("../includes/nav.php");
?>
```

The index.php is then continued with the initialization of cartItems array to handle the items added in the shopping cart. Submitting item ID and its existence in the wish list are then determined.

```php
<?php

// Initialize cartItems array
if (!isset($_SESSION['cartItems'])) {
    $_SESSION['cartItems'] = [];
}

// Check if an item ID is submitted
if (isset($_POST['item-id'])) {
    $itemId = $_POST['item-id'];

    // Check if the item already exists in the cart
    if (array_key_exists($itemId, $_SESSION['cartItems'])) {
        $_SESSION['cartItems'][$itemId]['quantity']++;
    } else {
        $_SESSION['cartItems'][$itemId] = ['quantity' => 1];
    }
}

$totalPrice = 0;
?>
```

If the cartItems array is empty, it will let the user return to the item list to pick their desirable items. By pressing the ‘Go Shopping’ button, the web page will be redirected to the ‘../item/index.php’.

```
        <?php
        if (empty($_SESSION['cartItems'])) {
            ?>
            <div id="wishlist-product-col">
                <br><svg class="w-6 h-6 text-gray-800 dark:text-white" aria-hidden="true"
xmlns="http://www.w3.org/2000/svg" width="50" height="50" fill="none" viewBox="0 0 24 24">
                  <path stroke="currentColor" stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" d="M4 4h1.5L8 16m0 0h8m-8 0a2 2 0 1 0 0 4 2 2 0 0 0 0-
4Zm8 0a2 2 0 1 0 0 4 2 2 0 0 0 0-4Zm.75-3H7.5M11 7H6.312M17 4v6m-3-3h6"/>
                </svg>
                <p>There are no items in the cart</p>
                <a href="../item/index.php" class="shopping-button">Go Shopping</a>
            </div>
        <?php
        } else {

        ?>
```

If the cartItems array is not empty, it will first retrieve the item information from the database and display them one by one in the wish list. Note that there is quantity control for users to select their desire quantity of items to buy and a remove function if they wish to remove the selected items. Then, the total price is calculated with the increase and decrease of quantity simultaneously when the '+'and '-' buttons are pressed. The quantity should not be below zero according to the value set.

```
<!-- Display the items added to cart -->
<div class="wishlist">
<?php
    // Retrieve selected items
    $cartItems = $_SESSION['cartItems'];

    // Fetch details of items in the cart
    $itemIds = "'" . implode("','", array_keys($cartItems)) . "'";
    $sql = "SELECT id, title, price, image_url FROM item WHERE id IN ($itemIds)";
    $result = $conn->query($sql);

    // Display items in the cart
    if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
            $quantity = isset($cartItems[$row['id']]['quantity']) ? $cartItems[$row['id']]['quantity'] : 0;
            $quantity = max(0, $quantity);

                ?>
                <div class='cart-item' id='cart-item-<?php echo $row['id']; ?>'>
                    <p><b><?php echo $row['id'];?></b></p>
                    <img src='<?php echo $row['image_url']; ?>' alt='<?php echo $row['title']; ?>' width='100' height='250'>
                    <h4><?php echo $row['title']; ?></h4>
                    <p>Price: RM<?php echo $row['price']; ?></p>
                    <div class="quantity-selector">
                        <button id="quantity-input- " class="decrement-btn" data-item-id="<?php echo $row['id']; ?>" onclick="decreaseItem('<?php echo $row['id']; ?>')">-</button>
                        <input type="number" class="quantity-input" id="quantity-input-<?php echo $row['id']; ?>" value="<?php echo $quantity; ?>">
                        <button id="quantity-input- " class="increment-btn" data-item-id="<?php echo $row['id']; ?>" onclick="increaseItem('<?php echo $row['id']; ?>')">+</button>
                    </div>
                    <button class='remove-btn' onclick='removeItem("<?php echo $row['id']; ?>", event)'>Remove</button>
                </div>
                <?php

                $totalPrice += $row['price']*$quantity;
            }
        }
    }
?>
</div>
```

The webpage not only shows the selected items, but it also displays the total price and the checkout button. Upon clicking on the 'Checkout' button, the form data will be submitted to the 'productCheckOut.php'.

```
        <form action="productCheckOut.php" method="post">
        <?php
        // Check if the cart is not empty
        if (!empty($_SESSION["cartItems"])) {
        ?>
            <!-- Display the total price and checkout button -->
            <div class='wishlist-product-col' id="total-price-section">
                <div class='total-price'>
                    <p>Total: <span class='fw-bold' id="total-price-value">RM <?php echo
number_format($totalPrice, 2); ?></span></p>
                </div>
                <div class='text-center'>
                    <input type="hidden" name="cartItems" value='<?php echo
json_encode($_SESSION["cartItems"]); ?>'>
                    <input type="submit" class="checkout-btn" value="Checkout">
                </div>
            </div>
```

All the functions in the index.php are written in the "wishlist.js".

```
<script src="wishlist.js"></script>
```

### 3.6.2. wishlist/decrementItem.php | incrementItem.php

In the incrementItem.php, a session has started. The item ID is checked. If it is existing in the cart, then the quantity of the item will increase by 1. Otherwise, error message will be notified.

```php
<?php
session_start();
// Check if the item ID is provided in the request
if (isset($_GET['id'])) {
    $itemId = $_GET['id'];
    if (isset($_SESSION['cartItems'][$itemId])) {
        // Increment the quantity of the item by 1
        $_SESSION['cartItems'][$itemId]['quantity']++;
        echo json_encode(array('success' => true, 'quantity' =>
$_SESSION['cartItems'][$itemId]['quantity'], 'price' => $totalPrice));
    } else {
        echo json_encode(array('error' => 'Item not found in cart'));
    }
} else {
    echo json_encode(array('error' => 'Item ID not provided'));
}
?>
```

In the decrementItem.php, a session has started. The item ID is checked. If it is existing in the cart, then the quantity of the item will decrease by 1. Otherwise, error message will be notified.

```php
<?php
session_start();
// Check if the item ID is provided in the request
if (isset($_GET['id'])) {
    $itemId = $_GET['id'];
    if (isset($_SESSION['cartItems'][$itemId])) {
        // Increment the quantity of the item by 1
        $_SESSION['cartItems'][$itemId]['quantity']--;
        echo json_encode(array('success' => true, 'quantity' =>
$_SESSION['cartItems'][$itemId]['quantity'], 'price' => $totalPrice));
    } else {
        echo json_encode(array('error' => 'Item not found in cart'));
    }
} else {
    echo json_encode(array('error' => 'Item ID not provided'));
}
?>
```

### 3.6.3. wishlist/removeItem.php

A session has started. The item ID is checked. If it exists in the cart, then the session array will be unset and removed from the cart. Otherwise, error message will be notified.

```php
<?php
session_start();

// Check if the item ID is provided in the request
if (isset($_GET['id'])) {
    $itemId = $_GET['id'];
    if (isset($_SESSION['cartItems'][$itemId])) {
        // Get the quantity of the removed item
        $quantity = $_SESSION['cartItems'][$itemId]['quantity'];
        unset($_SESSION['cartItems'][$itemId]);
        // Return the success status
        echo json_encode(array('success' => true));
    } else {
        echo json_encode(array('error' => 'Item not found in cart'));
    }
} else {
    echo json_encode(array('error' => 'Item ID not provided'));
}
?>
```

### 3.6.4. wishlist/productCheckOut.php

The product checkout page initiates with the connection of database.

```php
<?php
    session_start();

    $servername = 'localhost';
    $username = 'root';
    $password = '';
    $dbname = 'Z23_Beauty_Salon';

    $conn = mysqli_connect($servername, $username, $password, $dbname);

    // Check connection
    if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
    }
?>
```

The payment method and information are then displayed.

```
    <div class="payment-info">
    <article>
        <p>PLEASE READ THIS BEFORE PROCEED:</p>
        <p>Kindly note that all payments should be transferred to the following bank
account:</p>
        <p><strong>Z23 Beauty Salon</strong></p>
        Account Number: 738-888-0054<br>
        Bank: Public Bank Berhad
    </article>
    </div>
    <div class="payment-instruction">
    <article>
        <p>Upon completion of the transfer, we will send you a confirmation email within 3
days. If you wish to modify or cancel
        your order, kindly contact us at 012-7389779.
        Your cooperation is highly appreciated.</p>
    </article>
    </div>
```

Upon confirming the chosen items in the index.php, the selected items will be displayed under the summary section.

```php
<?php
   // Initialize cartItems array
    if (!isset($_SESSION['cartItems'])) {
        $_SESSION['cartItems'] = [];
    }

    $totalPrice = 0;

    // Check whether the cart is not empty
    if (!empty($_SESSION['cartItems'])) {
        // Retrieve selected items
        $cartItems = $_SESSION['cartItems'];

        // Fetch details of items in the cart
        $itemIds = "'" . implode("','", array_keys($cartItems)) . "'";
        $sql = "SELECT id, title, price, image_url FROM item WHERE id IN ($itemIds)";
        $result = $conn->query($sql);

        // Track the quantity of each item
        $itemQuantities = [];

        // Store the quantity of each item
        foreach ($cartItems as $itemId => $item) {
            $itemQuantities[$itemId] = $item['quantity'];
        }

        // Display items in the cart
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $quantity = isset($itemQuantities[$row['id']]) ? $itemQuantities[$row['id']] : 0;

                // Show each product once with its respective quantity
                if ($quantity > 0) {
                    echo "<div class='cart-item-co' id='cart-item-" . $row['id'] . "'>";
                    echo "<p>x" . $quantity . "</p>";
                    echo "<img src='" . $row['image_url'] . "' alt='" . $row['title'] . "' width='70' height='180'>";
                    echo "<h4>" . $row['title'] . "</h4>";
                    echo "<p id='pricePerItem'>RM" . $row['price'] . "</p>";
                    echo "</div>";

                    $totalPrice += $row['price'] * $quantity;
                }
            }
        }
    }

    // Close database connection
    mysqli_close($conn);
?>
```

The total price with 2 decimal places is then displayed too.

```
        <!--Display the total price-->
        <div class='total-price'>
            <br><br><p class='h5'><b>Total: <span class='fw-bold' id="total-price-value">RM
<?php echo number_format($totalPrice, 2); ?></b></span></p>
        </div>
```

A checkout form is displayed, and users are required to fill in. To protect the privacy of the user from filling up the form, the inserted information will not be saved in the browser by turning off

the autocomplete. The code snippet below shows one of the labels for the checkout form. It provides a blank space for the user to fill in his or her name with the hints given ‘Enter your name’. There will be an error message showing if the user leaves the blank empty below the blank space. Moreover, the users are also required to fill in their email, phone number, shipping address, payment reference number and bank name.

```
    <!--Display check out form-->
    <div class="checkout-form">
        <h2>Check Out</h2>
        <form id="checkoutForm" action="" method="post">
            <div class="form-group-co">
                <br>
                <label for="checkoutName" class="input-label-co">Name:</label>
                <input type="text" id="checkoutName" name="name" class="input-field-co"
placeholder="Enter your name" autocomplete="off">
                <div id="checkoutNameError" class="error-message"></div>
            </div>
```

The validation of form data is checked upon the user clicking on the ’Confirm’ button.

```
<div class="button-container">
       <input type="submit" value="Confirm" class="styled-button-co"
onclick="validateForm(event)" name="submit-co">
</div>
```

All the functions in the poductCheckOut.php are written in “wishlist.js”.

```
<script src="wishlist.js"></script>
```

### 3.6.5. wishlist/saveCheckout.php

This file is connected to the database. Upon pressing the 'Confirm' button, the information filled in the checkout form by the user is inserted into the database.

```php
<?php
session_start();

$servername = 'localhost';
$username = 'root';
$password = '';
$dbname = 'Z23_Beauty_Salon';

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Retrieve check out form data
$name = $_POST['name'];
$email = $_POST['email'];
$phone = $_POST['phone'];
$shippingAddress = $_POST['shipping-address'];
$reference = $_POST['reference'];
$bank = $_POST['bank'];

// Insert checkout information into the database
$sqlCheckout = "INSERT INTO checkout (buyer_name, buyer_email, buyer_phone,
buyer_address, reference, bank) VALUES (?, ?, ?, ?, ?, ?)";
$stmtCheckout = $conn->prepare($sqlCheckout);
$stmtCheckout->bind_param("ssssss", $name, $email, $phone, $shippingAddress,
$reference, $bank);
```

At the same time, the id generated in the 'checkout' table is retrieved and used in the 'items_bought' table to link every item that has been purchased to the relevant checkout process. Then, all the items bought by the user are retrieved by using the session variables. Hence, item details like item id, quantity selected, and price are retrieved from the chosen items in the "../wishlist/index.php". And now, they can be inserted into the database. Otherwise, it will show the error message. The connection of database is then closed.

```php
if ($stmtCheckout->execute()) {
    // Get the checkout ID
    $checkoutId = $stmtCheckout->insert_id;

    // Retrieve selected items
    $cartItems = $_SESSION['cartItems'];

    // Fetch details of items in the cart
    $itemIds = "'" . implode("','", array_keys($cartItems)) . "'";
    $sqlItems = "SELECT id, title, price FROM item WHERE id IN ($itemIds)";
    $resultItems = $conn->query($sqlItems);

    // Insert bought items into the database
    $sqlItemsBought = "INSERT INTO items_bought (checkout_id, item_id, quantity, price) VALUES (?, ?, ?, ?)";
    $stmtItemsBought = $conn->prepare($sqlItemsBought);

    if ($resultItems->num_rows > 0) {
        while ($row = $resultItems->fetch_assoc()) {
            $itemId = $row['id'];
            $quantity = isset($cartItems[$itemId]['quantity']) ? $cartItems[$itemId]['quantity'] : 0;
            $price = $row['price'];

            $stmtItemsBought->bind_param("isdd", $checkoutId, $itemId, $quantity, $price);
            $stmtItemsBought->execute();
        }
        $stmtItemsBought->close();
        echo "Checkout information and items saved successfully.";
    } else {
        echo "No items found in the database.";
    }
} else {
    echo "Error inserting checkout information: " . $conn->error;
}

// Close connections
$stmtCheckout->close();
$conn->close();
?>
```

### 3.6.6. wishlist/postMessage.php

After the checkout process, the session is destroyed as the user successfully checked out their desired items.

```
<?php
    session_start();

    //Destroy the session as the user successfully checked out
    session_destroy();
?>
```

It will prompt a thank you message to the user.

```
<body class="post-body">
    <div class="order-details">
        <svg class="w-6 h-6 text-gray-800 dark:text-white" aria-hidden="true"
xmlns="http://www.w3.org/2000/svg" width="24" height="24" fill="currentColor" viewBox="0 0
24 24">
          <path d="M17.133 12.632v-1.8a5.406 5.406 0 0 0-4.154-5.262.955.955 0 0
0 .021-.106V3.1a1 1 0 0 0-2 0v2.364a.955.955 0 0 0 .021.106 5.406 5.406 0 0 0-4.154
5.262v1.8C6.867 15.018 5 15.614 5 16.807 5 17.4 5 18 5.538 18h12.924C19 18 19 17.4 19
16.807c0-1.193-1.867-1.789-1.867-4.175ZM6 6a1 1 0 0 1-.707-.293l-1-1a1 1 0 0 1 1.414-
1.414l1 1A1 1 0 0 1 6 6Zm-2 4H3a1 1 0 0 1 0-2h1a1 1 0 1 1 0 2Zm14-4a1 1 0 0 1-.707-1.707l1-
1a1 1 0 1 1 1.414 1.414l-1 1A1 1 0 0 1 18 6Zm3 4h-1a1 1 0 1 1 0-2h1a1 1 0 1 1 0 2ZM8.823
19a3.453 3.453 0 0 0 6.354 0H8.823Z"/>
        </svg>
        <h2>Thank you for choosing us!</h2>
        <p>Your order has been confirmed. Kindly check your mail box that we had drop a
confirmation email to you.</p>
        <button id="return">Return to Homepage</button>
    </div>
```

As the ‘Return to Homepage’ button is pressed, it will redirect users to the home page.

```
<script>
    document.getElementById("return").onclick = function() {
        window.location.href = "../index.php";
    };
</script>
```

### 3.6.7. wishlist/wishlist.js

The java script code in the wishlist.js includes actions like removing items, increment of items, decrement of items and update of total price for index.php. Other than that, it also incorporates form validation for 'productCheckOut.php'. Firstly, according to the image below, it shows the removal of items by sending an AJAX request. Upon removal of items, the total price will be updated simultaneously.

```
function removeItem(itemId, event) {
        event.preventDefault(); // Prevent the default form submission behavior

        // Send an AJAX request to remove the item from the session
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "removeItem.php?id=" + itemId, true);
        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4) {
                if (xhr.status == 200) {
                    var response = JSON.parse(xhr.responseText);
                    if (response.success) {
                        // Remove the item from the page
                        var itemElement = document.getElementById('cart-item-' + itemId);
                        if (itemElement) {
                            itemElement.parentNode.removeChild(itemElement);
                        }
                        // Refresh the total price after successful removal
                        updateTotalPrice(response.price, '-');
                        alert('Item removed successfully');
                    } else {
                        // Handle error response
                        alert('Error: ' + response.error);
                    }
                } else {
                    // Handle HTTP error
                    alert('Error: ' + xhr.status);
                }
            }
        };
        xhr.send();
    }
```

To make both increase and decrease button functionable, event listener is added to both buttons. When the increase button is pressed once, the value in the box will increase by one. The same goes for the decrease button, however, the value in the box will decrease by one. When both buttons are pressed, they call the increaseItem and decreaseItem functions to ensure that the items added to cart are aligned with the latest quantity.

```
document.addEventListener('DOMContentLoaded', function () {
    var incrementButtons = document.querySelectorAll('.increment-btn');
    var decrementButtons = document.querySelectorAll('.decrement-btn');

    incrementButtons.forEach(function (button) {
        button.addEventListener('click', function (event) {
            event.preventDefault();
            var input = this.closest('.quantity-selector').querySelector('.quantity-input');
            var value = parseInt(input.value, 10);
            value = isNaN(value) ? 0 : value;
            if (value < 10) {
                value++;
                input.value = value;

                // Send an AJAX request to increase items
                var itemId = this.closest('.cart-item- ').getAttribute('data-item-id');
                increaseItem(itemId);
            }
        });
    });

    decrementButtons.forEach(function (button) {
        button.addEventListener('click', function (event) {
            event.preventDefault();
            var input = this.closest('.quantity-selector').querySelector('.quantity-input');
            var value = parseInt(input.value, 10);
            value = isNaN(value) ? 0 : value;
            if (value > 1) {
                value--;
                input.value = value;

                // Send an AJAX request to decrease items
                var itemId = this.closest('.cart-item- ').getAttribute('data-item-id');
                decreaseItem(itemId);
            }
        });
    });
});
```

For the decrease item function, while it gets the latest quantity, it will send an AJAX request to the decrementItem.php to update the latest quantity of item added into the cart and show the latest total price.

```
function decreaseItem(itemId) {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', 'decrementItem.php?id=' + itemId, true);
    xhr.onreadystatechange = function() {
        if (xhr.readyState === XMLHttpRequest.DONE) {
            if (xhr.status === 200) {
                var response = JSON.parse(xhr.responseText);
                if (response.success) {
                    // Update quantity input field
                    var quantityInput = document.getElementById('quantity-input-' + itemId);
                    quantityInput.value = response.quantity;

                    // Update total price
                    updateTotalPrice(response.price, '-');

                } else {
                    // Handle error response
                    alert('Error: ' + response.error);
                }
            } else {
                // Handle HTTP error
                alert('Error: ' + xhr.status);
            }
        }
    };
    xhr.send();
}
```

Moreover, for the increase item function, while it gets the latest quantity, it will send an AJAX request to the incrementItem.php to update the latest quantity of item added into the cart and show the latest total price.

```
function increaseItem(itemId) {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', 'incrementItem.php?id=' + itemId, true);
    xhr.onreadystatechange = function() {
        if (xhr.readyState === XMLHttpRequest.DONE) {
            if (xhr.status === 200) {
                var response = JSON.parse(xhr.responseText);
                if (response.success) {
                    // Update quantity input field
                    var quantityInput = document.getElementById('quantity-input-' + itemId);
                    quantityInput.value = response.quantity;

                    // Update total price
                    updateTotalPrice(response.price, '+');
                } else {
                    // Handle error response
                    alert('Error: ' + response.error);
                }
            } else {
                // Handle HTTP error
                alert('Error: ' + xhr.status);
            }
        }
    };
    xhr.send();
}
```

The total price is updated by using the below function. And, to update the total price simultaneously, the page is reloaded once user clicks on the increase, decrease or remove buttons.

```
function updateTotalPrice(price, operation) {
    var totalPriceElement = document.getElementById('total-price-value');
    var totalPrice = parseFloat(totalPriceElement.innerText.replace('RM ', ''));
    var itemPrice = parseFloat(price);
    if (operation === '+') {
        totalPrice += itemPrice;
    } else if (operation === '-') {
        totalPrice -= itemPrice;
    }
    totalPriceElement.innerText = 'RM ' + totalPrice.toFixed(2);

    // Reload the page after updating the total price
    window.location.reload();
}

// Get all buttons on the page
var buttons = document.querySelectorAll('button');
buttons.forEach(function(button) {
    button.addEventListener('click', function() {
        // Reload the page
        location.reload();
    });
});
```

The validation of the form is done by ensuring that the user has filled in all the blanks and provided the valid information. If all the information is valid, the form will be submitted, and an AJAX request is sent to connect to the database. So that the checkout form details can be recorded in the database.

```
/*for productChecOut.php validation*/
function validateForm(event) {
        event.preventDefault(); // Prevent default form submission behavior

        var form = document.getElementById('checkoutForm');
        var isValid = true;

        // Clear previous error messages
        document.querySelectorAll('.error-message').forEach(function (error) {
            error.textContent = '';
        });

        // Validate name
        if (form['checkoutName'].value.trim() === '') {
            document.getElementById('checkoutNameError').textContent = 'Name is required.';
            isValid = false;
        }

        // Validate email
        if (form['checkoutEmail'].value.trim() === '') {
            document.getElementById('checkoutEmailError').textContent = 'Email is required.';
            isValid = false;
        } else if (!(/^\S+@\S+\.\S+$/.test(form['checkoutEmail'].value))) {
            document.getElementById('checkoutEmailError').textContent = 'Email is not valid.';
            isValid = false;
        }

        // Validate phone number
        if (form['checkoutPhone'].value.trim() === '') {
            document.getElementById('checkoutPhoneError').textContent = 'Phone number is required.';
            isValid = false;
        } else if (!/^\d+$/.test(form['checkoutPhone'].value)) {
            document.getElementById('checkoutPhoneError').textContent = 'Phone number must be numeric.';
            isValid = false;
        }

        // Validate shipping address
        if (form['checkoutShippingAddress'].value.trim() === '') {
            document.getElementById('checkoutShippingAddressError').textContent = 'Shipping address is required.';
            isValid = false;
        }
```

```
    //Validate reference number
    if(form['referenceNo'].value.trim()===''){
        document.getElementById('referenceError').textContent='Payment reference number is required.';
        isValid=false;
    }

    //Validate bank name
    if(form['checkoutBank'].value.trim()===''){
        document.getElementById('checkoutBankError').textContent='Bank name is required.';
        isValid=false;
    }

    if (isValid) {
    // Submit the form if all inputs are valid
    var xhr = new XMLHttpRequest();
    xhr.open('POST', 'saveCheckout.php', true);
    xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
    xhr.onreadystatechange = function () {
        if (xhr.readyState == XMLHttpRequest.DONE) {
            if (xhr.status == 200) {
                alert('Checkout information and items saved successfully.');
                window.location.href = 'postMessage.php';
            } else {
                // Error message
                alert('Error: ' + xhr.responseText);
            }
        }
    };
    xhr.send(new URLSearchParams(new FormData(form)));
}
}
```

### 3.7. contactus

#### 3.7.1. contactus/index.php

```
<div id="contact-form">
  <form action="contactPostMessage.php" method="post">
    <div class="form-group" id="salutationInput">
      <b>Salutation:</b><br>
      <input type="radio" id="mr" name="salutation" value="Mr">Mr
      <input type="radio" id="ms" name="salutation" value="Ms"> Ms
      <input type="radio" id="mrs" name="salutation" value="Mrs"> Mrs
      <input type="radio" id="mdm" name="salutation" value="Mdm"> Mdm<br>
      <div class="error"></div>
    </div>
    <div class="form-group" id="enquiryInput">
      <b>Type of Enquiry:</b><br>
      <input type="checkbox" name="enquiry[]" value="General Enquiry"> General Enquiry
      <input type="checkbox" name="enquiry[]" value="Complaints"> Complaints
      <input type="checkbox" name="enquiry[]" value="Suggestions"> Suggestions<br>
      <div class="error"></div>
    </div>
    <div class="form-group" id="nameInput">
      <label for="nam">Name:</label>
      <input type="text" id="nam" name="name">
      <div class="error"></div>
    </div>
    <div class="form-group" id="emailInput">
      <label for="email">E-mail:</label>
      <input type="email" id="email" name="email">
      <div class="error"></div>
    </div>
    <div class="form-group" id="phoneInput">
      <label for="phone">Phone Number:</label>
      <input type="tel" id="phone" name="phone">
      <div class="error"></div>
    </div>
    <div class="form-group" id="subjectInput">
      <label for="message">Subject:</label>
      <textarea id="message" name="message" rows="6"></textarea>
      <div class="error"></div>
    </div>
    <div class="checkbox-group">
      <input type="checkbox" id="subscribe" name="subscribe">
      <label for="subscribe">Subscribe to Newsletter</label>
    </div>
    <br>
    <input type="submit" value="Submit">
  </form>
</div>
```

This form is enclosed within a  for styling and organization. In this form, we aim to get the salutation, type of enquiries, name, email, phone number, subject which is message about the enquiry. The type of enquiry is set to an array as we accept multiple selection from users where there are few options which are general enquiry, complaints and suggestion. Besides, we do have checkboxes for users to subscribe to newsletter about our latest information.

Additionally, there are  elements within each form-group to display error messages if form validation fails.

Once the user has entered all valid information, the submit button helps the user to submit the form. Hence, the form submits data to “contactPostMessage.php” by using the POST method.

### 3.7.2. contactus/validation.js

```
const form = document.querySelector('form');
const salutationInput = document.getElementById('salutationInput');
const nameInput = document.getElementById('nam');
const emailInput = document.getElementById('email');
const phoneInput = document.getElementById('phone');
const enquiryInput = document.getElementById('enquiryInput');
const subjectInput = document.getElementById('subjectInput');
```

We select the necessary HTML elements required for validation, including user inputs for salutation, name, email, phone, type of enquiry, and subject.

```
// Add event listener to the form for submission
form.addEventListener('submit', function(event) {
  event.preventDefault();
  const errors = [];
```

An event listener is added to prevent the form from its default behaviour, which is submitting to a server, by using “event.preventDefault()”. This listener intercepts the form submission, performs validation, and decides whether to submit the form based on the validation results.

```
// Validate the salutation type input field
if (!salutationInput.querySelector('input[type="radio"]:checked')) {
  errors.push('Please select a salutation');
  salutationInput.lastElementChild.innerHTML = 'Please select a salutation';
  salutationInput.lastElementChild.style.color = 'red'; // set color to red
} else {
  salutationInput.lastElementChild.innerHTML = '';
}

// Validate the name input field
if (nameInput.value.trim() === '') {
  errors.push('Please enter your name');
  nameInput.nextElementSibling.innerHTML = 'Please enter your name';
  nameInput.nextElementSibling.style.color = 'red'; // set color to red
} else {
  nameInput.nextElementSibling.innerHTML = '';
}

// Validate the email input field
if (emailInput.value.trim() === '') {
  errors.push('Please enter your email address');
  emailInput.nextElementSibling.innerHTML = 'Please enter your email address';
  emailInput.nextElementSibling.style.color = 'red'; // set color to red
} else if (!isValidEmail(emailInput.value.trim())) {
  errors.push('Please enter a valid email address');
  emailInput.nextElementSibling.innerHTML = 'Please enter a valid email address';
  emailInput.nextElementSibling.style.color = 'red'; // set color to red
} else {
  emailInput.nextElementSibling.innerHTML = '';
}


// Validate the phone number input field for 9-11 digits
if (phoneInput.value.trim() === '') {
  errors.push('Please enter your phone number');
  phoneInput.nextElementSibling.innerHTML = 'Please enter your phone number';
  phoneInput.nextElementSibling.style.color = 'red'; // set color to red
} else if (!/^\d{9,11}$/.test(phoneInput.value.trim())) {
  errors.push('Please enter a valid phone number with 9 to 11 digits');
  phoneInput.nextElementSibling.innerHTML = 'Please enter a valid phone number with 9 to 11 digits';
  phoneInput.nextElementSibling.style.color = 'red'; // set color to red
} else {
  phoneInput.nextElementSibling.innerHTML = '';
}

// Validate the enquiry type input field
if (!enquiryInput.querySelector('input[type="checkbox"]:checked')) {
  errors.push('Please select at least one type of enquiry');
  enquiryInput.lastElementChild.innerHTML = 'Please select at least one type of enquiry';
  enquiryInput.lastElementChild.style.color = 'red'; // set color to red
} else {
  enquiryInput.lastElementChild.innerHTML = '';
}

// Validate the subject input field
if (subjectInput.querySelector('textarea').value.trim() === '') {
  errors.push('Please enter your message');
  subjectInput.lastElementChild.innerHTML = 'Please enter your message';
  subjectInput.lastElementChild.style.color = 'red'; // set color to red
} else {
  subjectInput.lastElementChild.innerHTML = '';
}
```

Validation is performed on each input field to ensure that all required fields are selected or filled correctly. For salutation, it checks if a radio button is selected. For name and subject, it verifies if

the input is not empty. The email validation checks if the email input is not empty and matches a valid email format using a regular expression. Phone validation ensures the input is not empty and matches a format of 9 to 11 digits. Type of enquiry validation confirms at least one checkbox is checked. This process ensures that all necessary fields are correctly filled before allowing form submission. If any field is invalid, it immediately notifies the user to correct their inputs.

```
// Submit the form if there are no errors
if (errors.length === 0) {
  form.submit();
}
```

When all validations pass with no error, the form is submitted using form.submit().

```
// Function to validate email using a regular expression
function isValidEmail(email) {
  const emailRegex = /^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$/;
  return emailRegex.test(email);
}
```

The “isValidEmail” function validates an email address using a regular expression, checking if the email matches a pattern of alphanumeric characters, hyphens, periods, and valid domain endings.

### 3.7.3. contactus/contactPostMessage.php

```php
<div class="contactus">
    <?php
    // Retrieve form data
    $salutation = $_POST['salutation'];
    $enquiry = implode(", ", $_POST['enquiry']);
    $name = $_POST['name'];
    $email = $_POST['email'];
    $phone = $_POST['phone'];
    $subject = $_POST['message'];
    $subscribe = isset($_POST['subscribe']) ? 1 : 0;

    // Database connection
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "Z23_Beauty_Salon";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    // Insert data into the database
    $sql = "INSERT INTO contact_info (salutation, enquiry, name, email, phone, subject, subscribe)
            VALUES ('$salutation', '$enquiry', '$name', '$email', '$phone', '$subject', '$subscribe')";

    if ($conn->query($sql) === TRUE) {
        echo "Thank you for your enquiries. We will get back to you as soon as possible.";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    // Close connection
    $conn->close();
    ?>
    <div class="button-container">
        <input type="button" class="styled-button" onclick="window.location.href='../index.php'" value="Done">
    </div>
</div>
```

This page displays a message thanking the user for their enquiries and informs them that they will be contacted soon. It retrieves the form data submitted by the user and inserts this data into the "contact_info" table in the "database.sql". If the insertion is successful, a success message is displayed. However, if there is an error during insertion, an error message with details is shown.

After displaying the message, the user is presented with a "Done" button. Clicking this button redirects the user back to the home page.

### 3.8. includes

#### 3.8.1. includes/header.php

```
<header id = "pageHeader">
	<img src="/Z23BeautySalon/photos/logo.png" width="100"
height="100">
	<h1>Z23 Beauty Salon</h1>

</header>
```

The code snippet above is used to standardize our salon's logo and title.

#### 3.8.2. includes/nav.php

```
<nav id="topNavigation">
	<hr>
	<a href="/Z23BeautySalon">Home</a>
	|
	<a href="/Z23BeautySalon/service/">Services</a>
	|
	<a href="/Z23BeautySalon/item/">Products</a>
	|
	<a href="/Z23BeautySalon/wishlist/">Wishlist</a>
	|
	<a href="/Z23BeautySalon/contactus">Contact Us</a>
	<hr>
</nav>
```

The code snippets above are used to display our website's navigation. All of the snippets have been linked in their respective pages with the href function. This allows users to have easy access to all our website's functionality.

#### 3.8.3. includes/footer.php

```
<footer id = "pageFooter"><footer id = "pageFooter">
	<br><hr>
	&copy; 2001 Z23 Beauty Salon. All rights reserved.
</footer>
```

The code snippets above are used to display our website's footer. It is displayed on every page for standardization purposes.

### 3.9. style/webstyles.css

Due to the numerous stylings in webstyles.css, we have decided to only include a small portion of our CSS codes to use as an example. For the code below, we used CSS to adjust the header. We had also standardized the headers font as Verdana, Sans-serif.

```
header#pageHeader
{
        margin: 0;
        padding: 0;
        text-align: center;
        font-family: Verdana, Sans-serif;
}

header#pageHeader .img-container {
    float: left;
}
```

On the other hand, the CSS styles here are to adjust the outlook of our navigation system. From the CSS code below, we can see that we set our navigation system's text to the center for a prettier outlook and all the links in the navigation no longer hold any text decoration. However, our cursor would change from an arrow to a pointer when hovering over a navigation link.

```
nav#topNavigation,
nav#serviceNav,
nav#itemNav{
  text-align: center
}

nav a {
       color: rgb(0, 0 , 0);
       cursor: pointer;
       text-decoration: none

}
```

Next, we have the CSS styles for our webpage's body, headers, and errors. By standardizing them together, the output on the website would be much neater with the same text alignment, font, and size.

```
body {
	padding: 10px;
	background-color: #F7F4F0;
	font-family: Verdana, Sans-serif;
}

h1,h2,h3,h4,h5,h6{
	color: rgb(0,0,0);
	font-weight: bold;
	line-height: 1.25em;
	padding: 0px;
	font-weight: bold;
}

h1,h2 {
	text-align: center;
}

h3,h4,h5,h6 {
	text-align: left;
}

.error {
    color: #ff0000;
	font-size:15px;}
```

## 4. Sample Output

### 4.1. Home Page

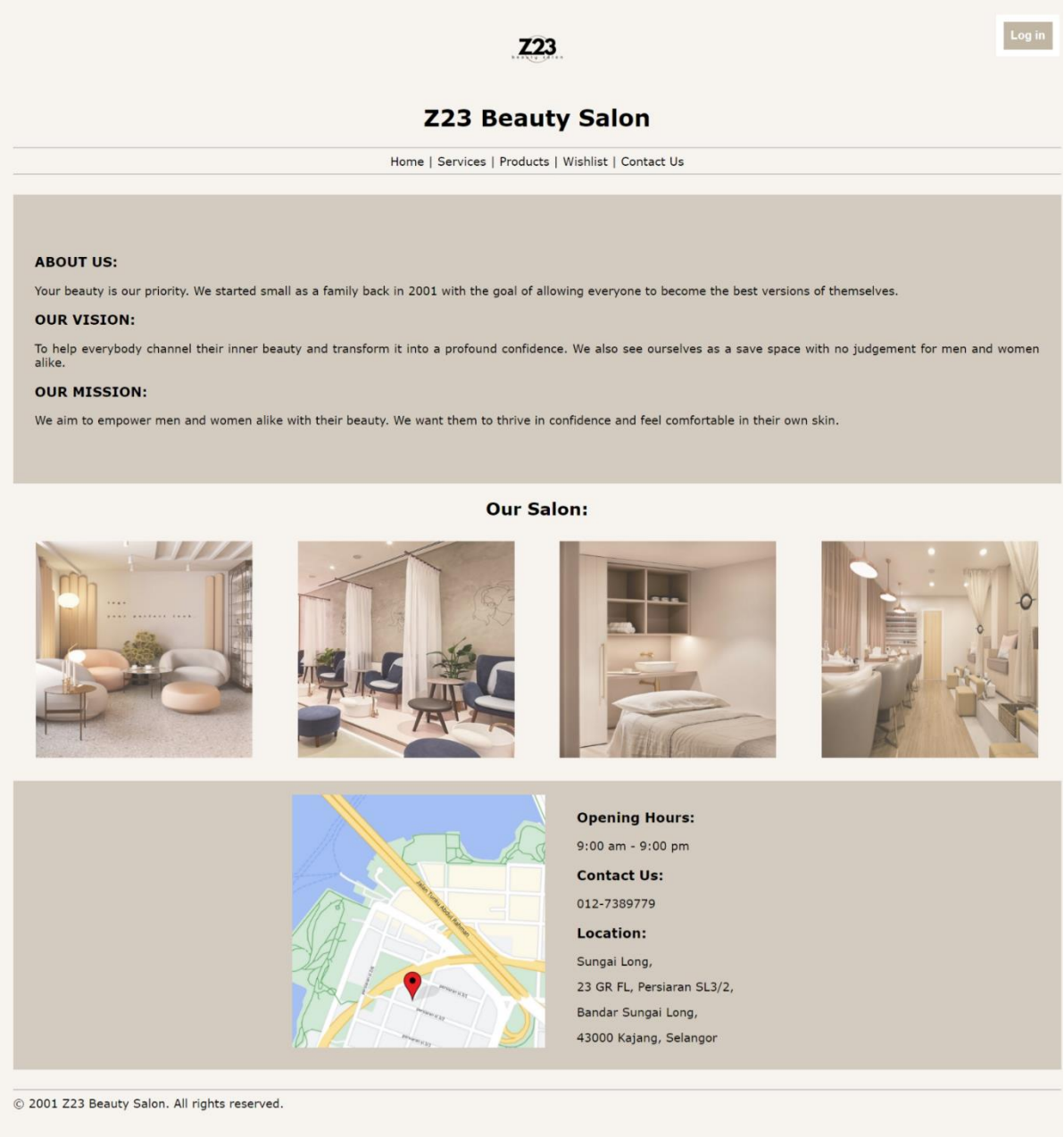


*Figure 4.1.1: Overview of our Home Page*

As shown in Figure 4.1.1, we can see a complete overview of our website's home page. From the home page, users are about to learn about our salon, view photos of our website and view our location too.

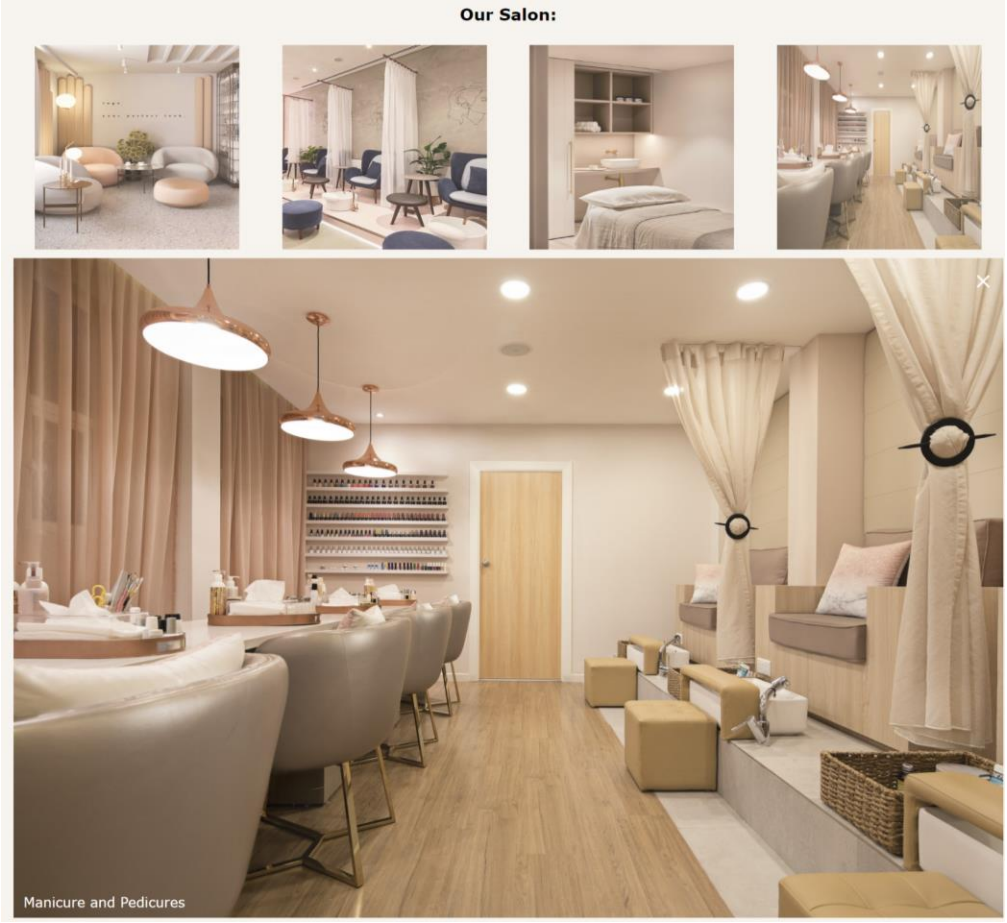


*Figure 4.1.2: Interactive webpage design*

According to Figure 4.1.2, we can see an overview of the “ourSalon” container. This container is an interactive container that allows users to enlarge a photo of clicking on it. When the photo is enlarged, the alt words will also be displayed alongside at the bottom left of the photo.

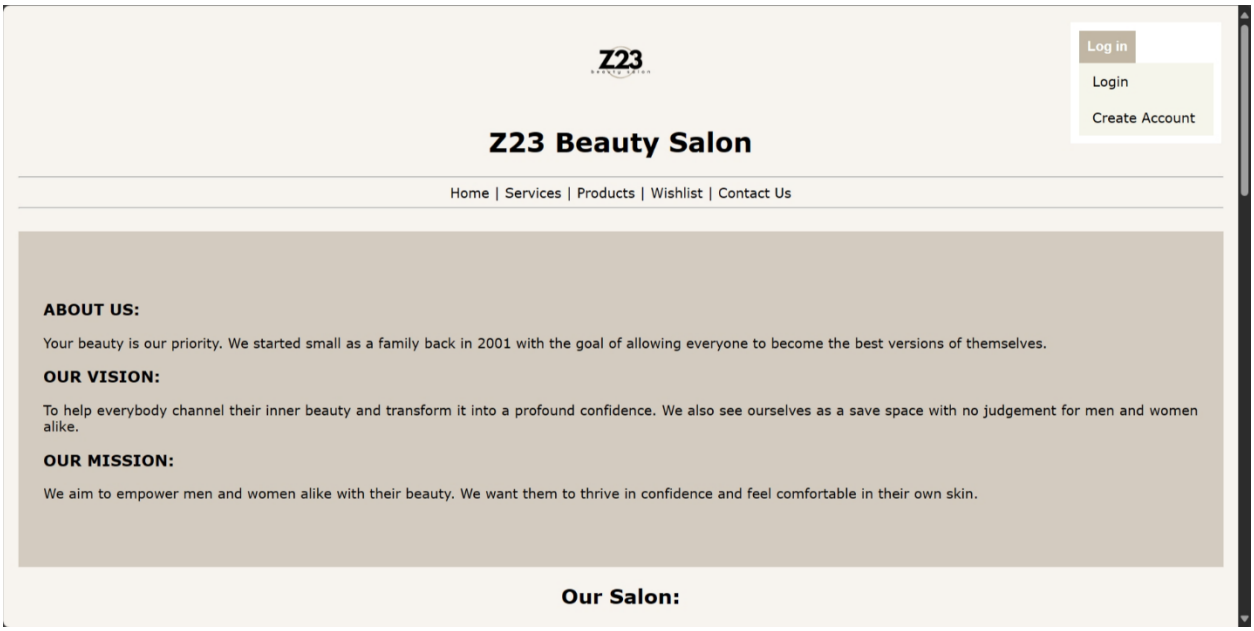


*Figure 4.1.3: Login Button*

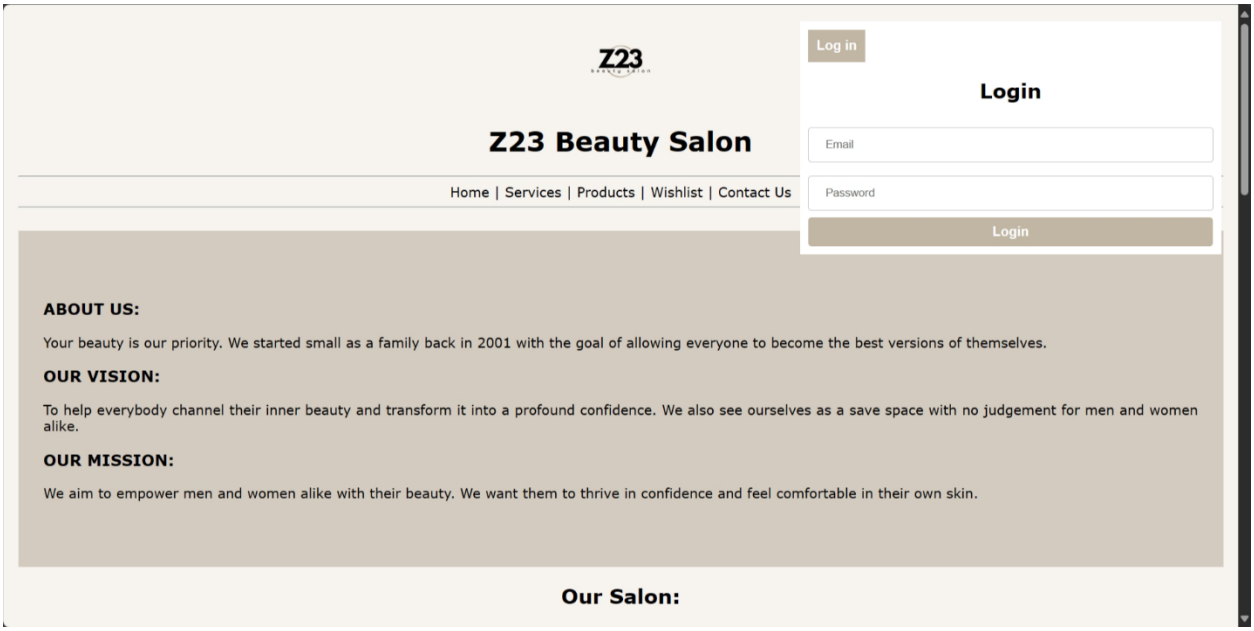


*Figure 4.1.4: Login Function*

Through the home page users are also given the option to login and to create an account. When users click on the login button, the login container will then expand to allow users to key in their account details as shown in Figure 4.1.3 and Figure 4.1.4.

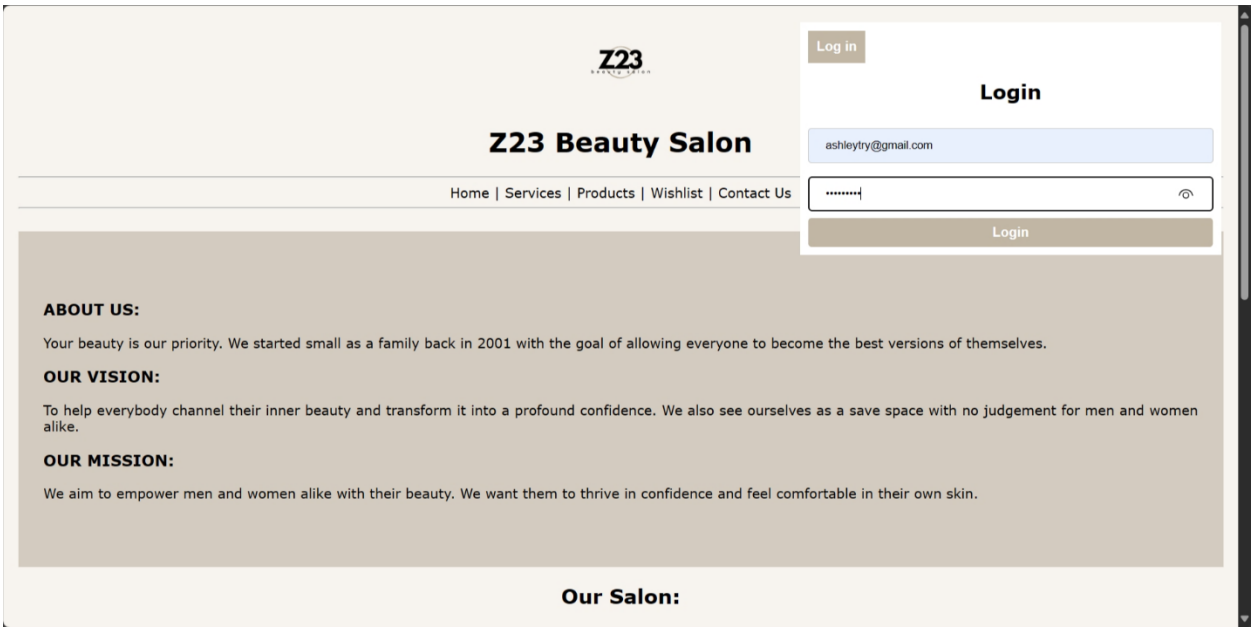


*Figure 4.1.5: Entering Account Details*

According to Figure 4.1.5, we can see how the form would look like once we've keyed in our details. In the password section of the form, they password is hidden. However, it can still be seen if the users clicked on the eye logo at the right side of the password column.

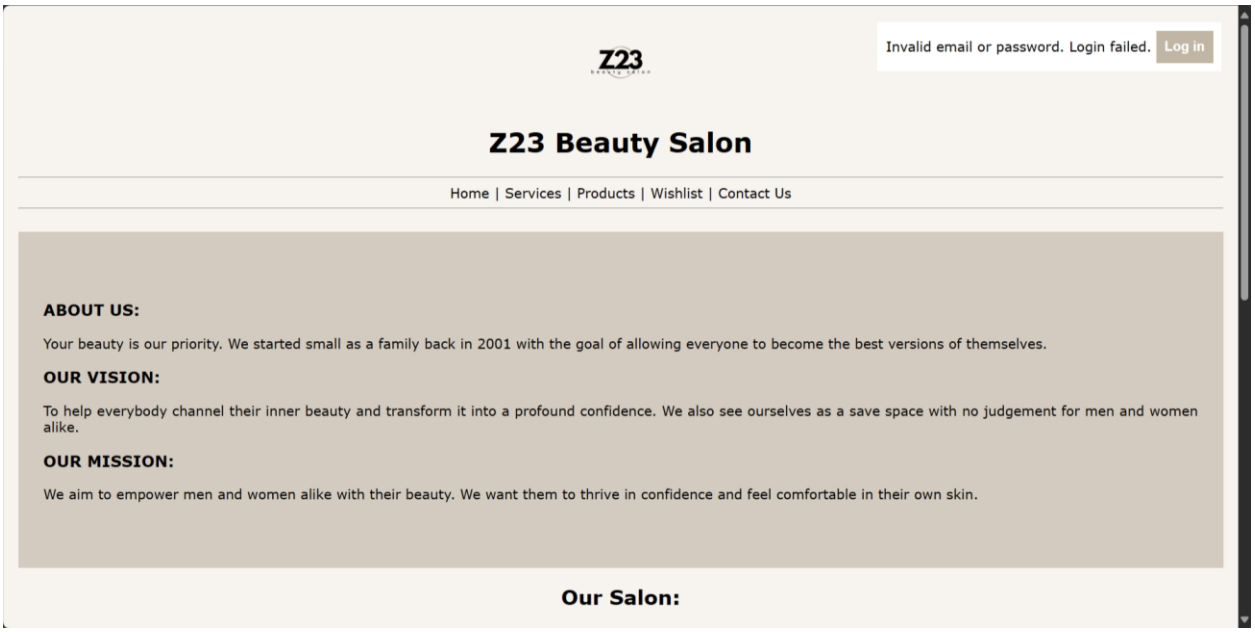


*Figure 4.1.6: After entering invalid details.*

On the other hand, Figure 4.1.6 shows the output given if a user keys in an invalid password. Such instances may occur if the user keyed in an invalid email or password into the form.

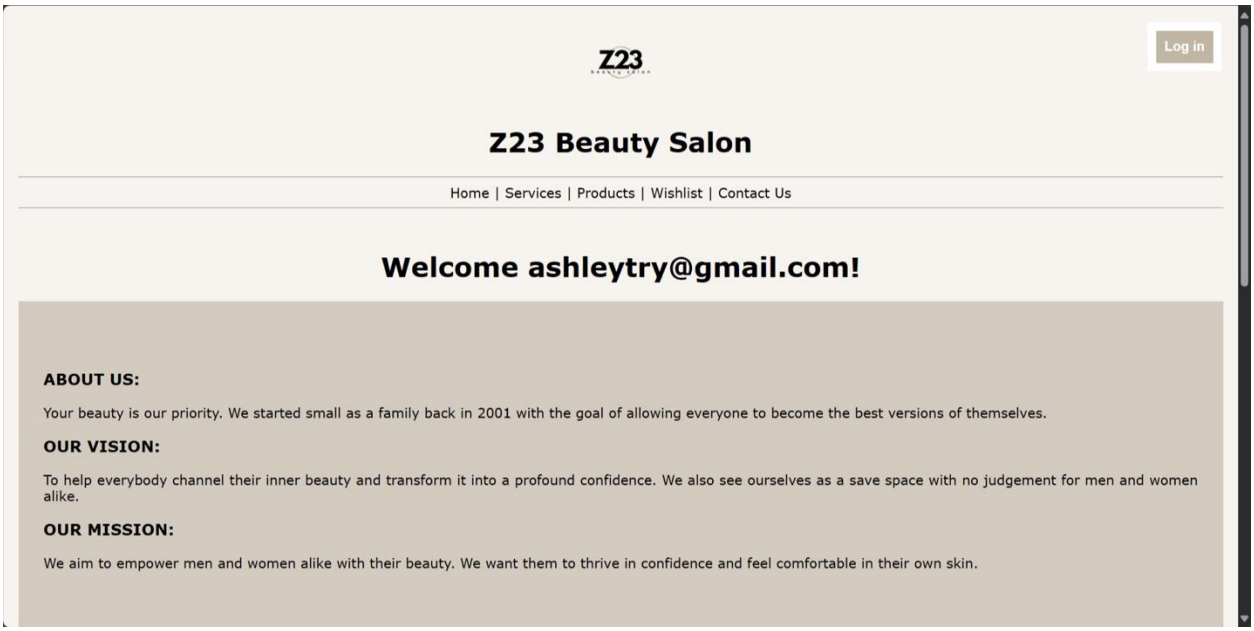


*Figure 4.1.7: Welcome Message after a successful login*

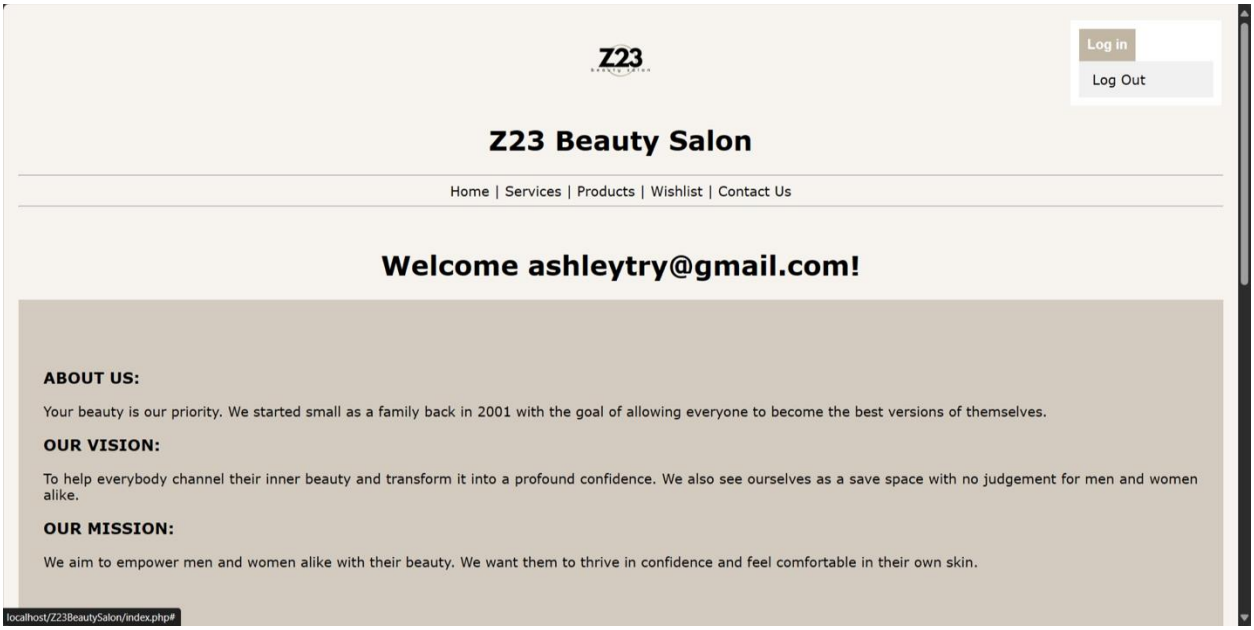


*Figure 4.1.8: Log In button only displays log out option*

From Figure 4.1.7, we can see that a welcome message is displayed once users managed to login in successfully. Furthermore, the dropdown login button changes and only shows the log out button for the user. The login and create account features are no longer available to the user. It is also notable that a session is initiated once a user manages to login into their account successfully.

## 4.2. Create Account Page

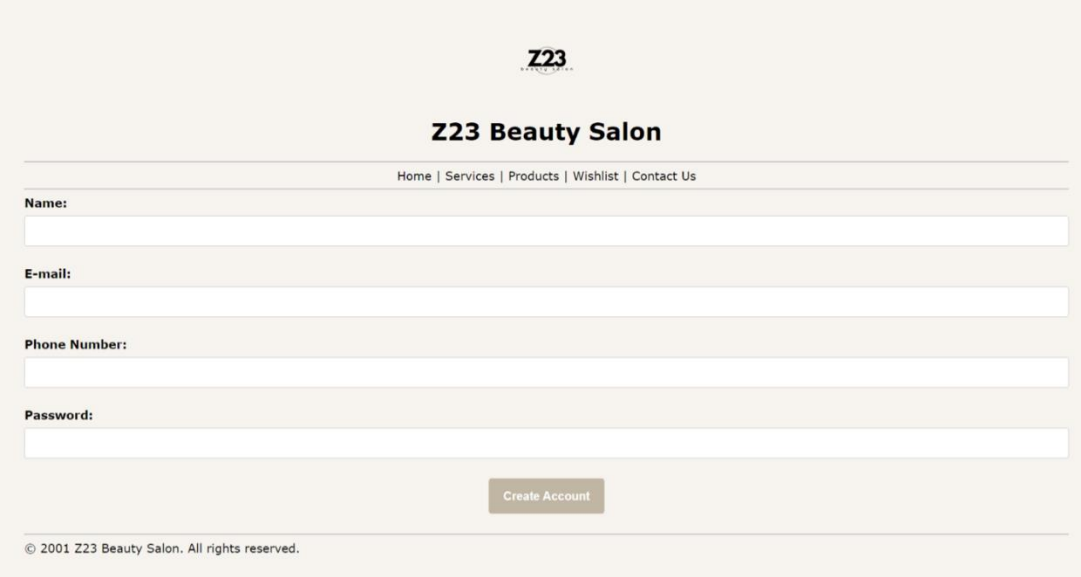


*Figure 4.2.1: Form for account creation*

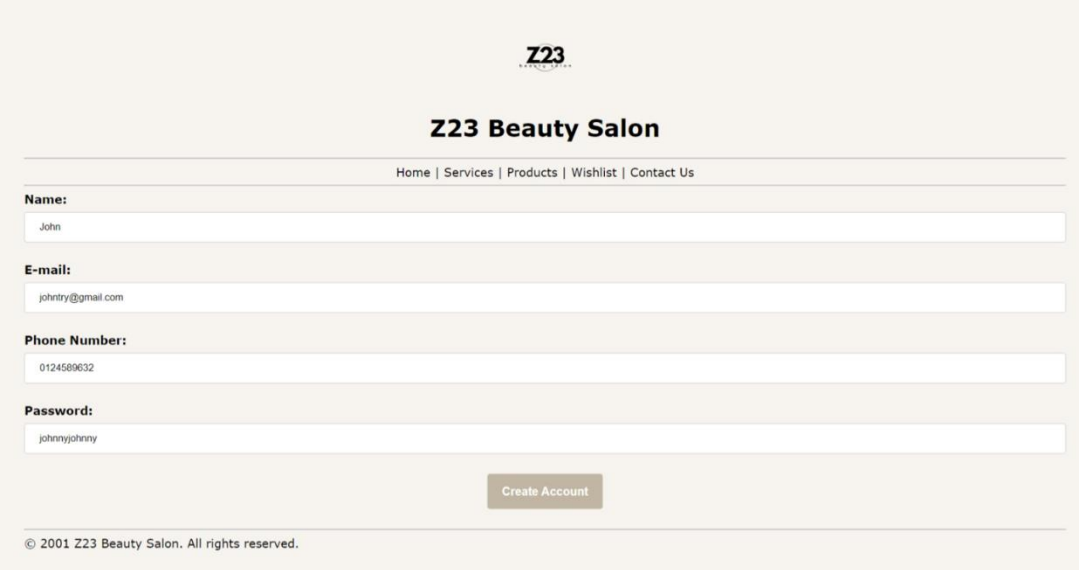


*Figure 4.2.2: Form after being filled*

From Figure 4.2.1 and Figure 4.2.2, we can see how the form looks like when it's empty and when it its filled. Users are required to fill in all fields of the forms before clicking on the create account button.

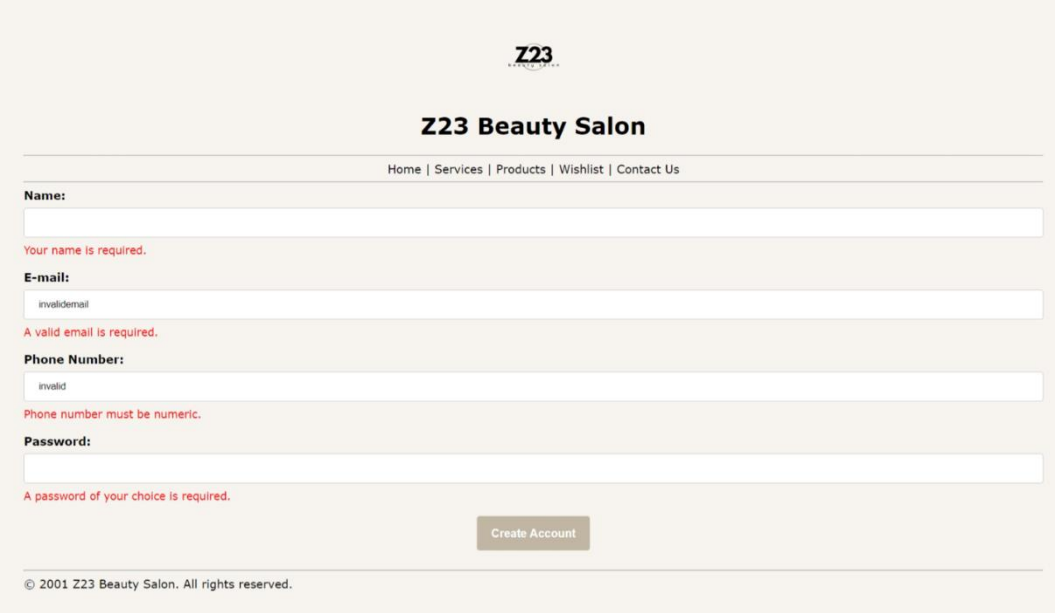


*Figure 4.2.3: Error message when empty form is submitted*

According to Figure 4.2.3, an error message will appear whenever a user tried to lift a field empty or tries to submit an empty form.

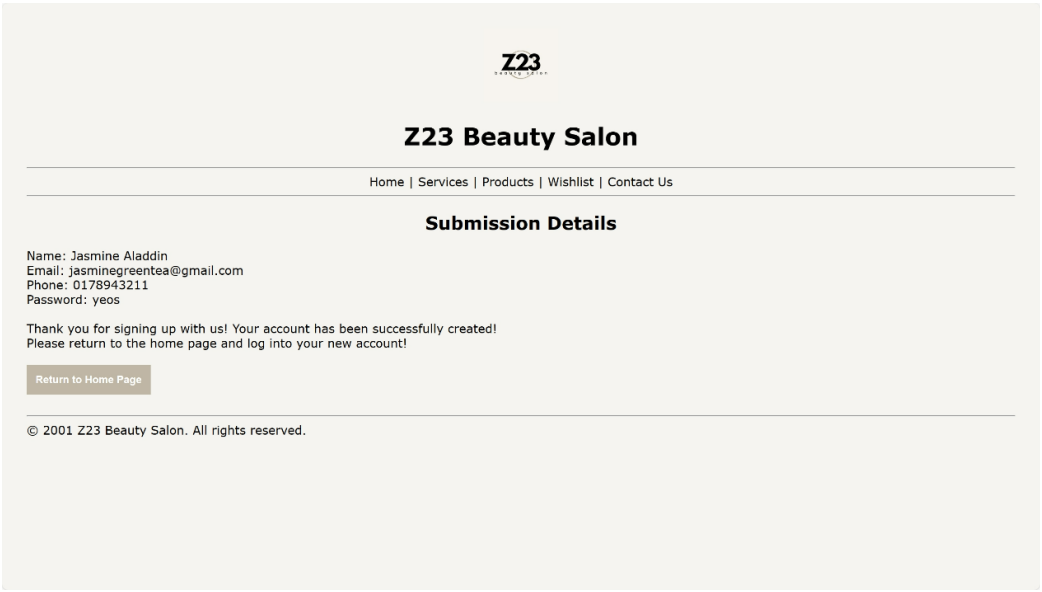


*Figure 4.2.4: Submission Details*

After a user manages to fill and submit the form, the users will be directed to a page where the users will be able to review their submitted details. Then, they'll have to enter the return to home button to be redirected back to the home page to login into their new account.

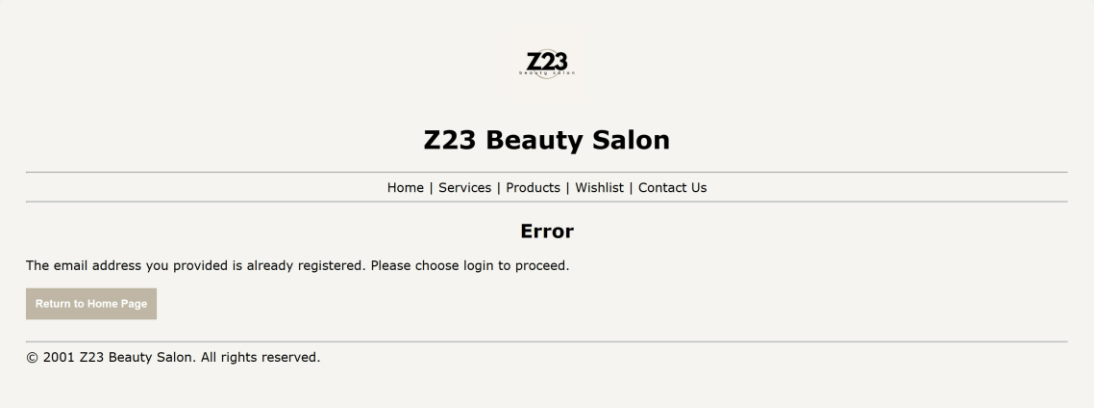


*Figure 4.2.5: Error message*

The error message seen in Figure 4.2.5 will occur when a user tries to create an account with an email that has already been registered into the system. When that error occurs, the create account process is terminated and the user can only return to the home page.

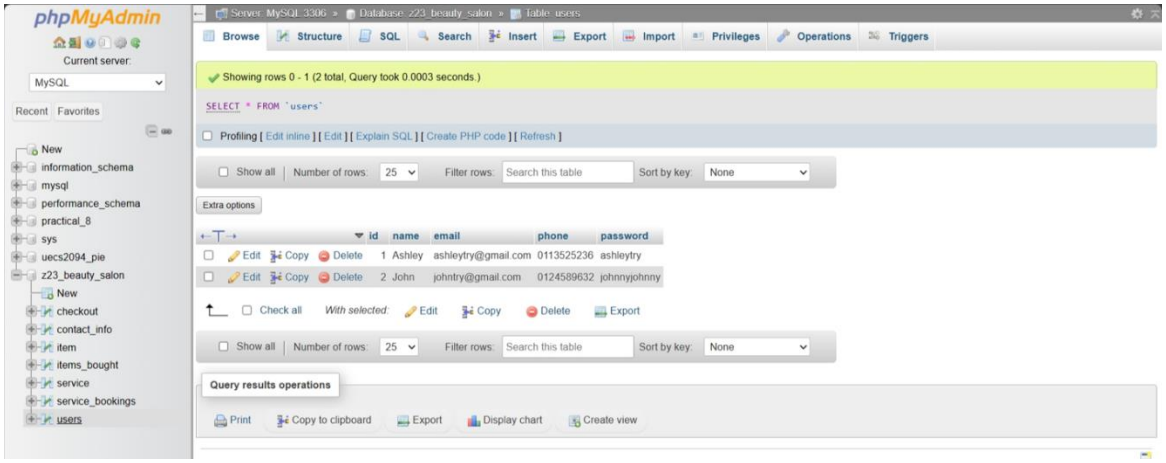


*Figure 4.2.5: Successful insert of users*

Figure 4.2.5 shows that the successful insert of information as user succesful created the account. It is recorded and saved successfully in the “users” table in the database.

### 4.3. Services

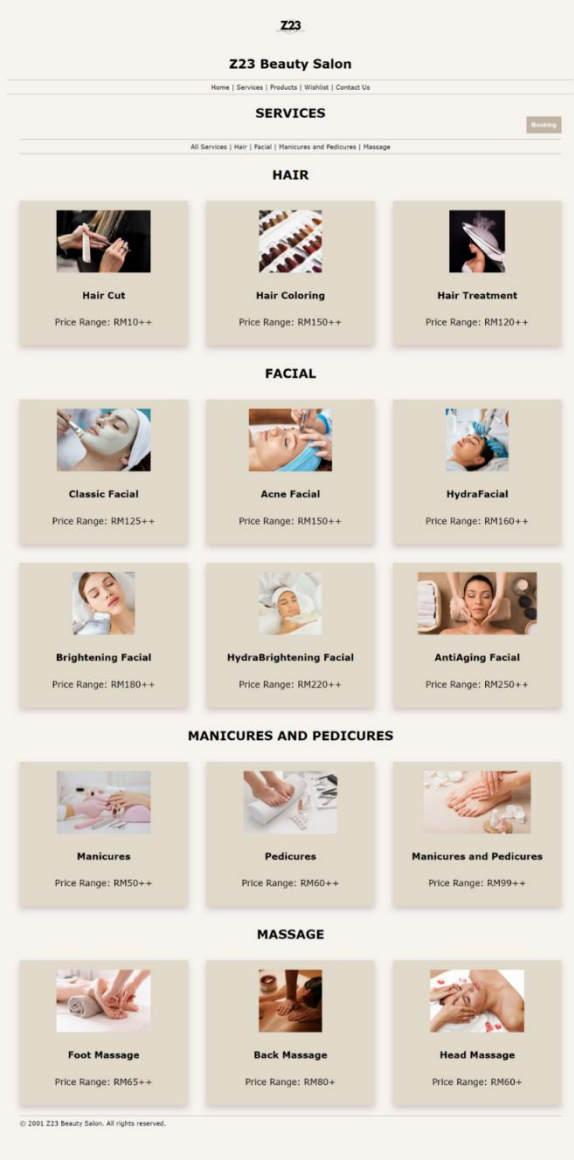


*Figure 4.3.1: Booking List*

The above figure shows the booking page. It includes the four categories of the services, which are hair, facial, manicures and pedicures, and massage. The price range and types of services are shown explicitly.

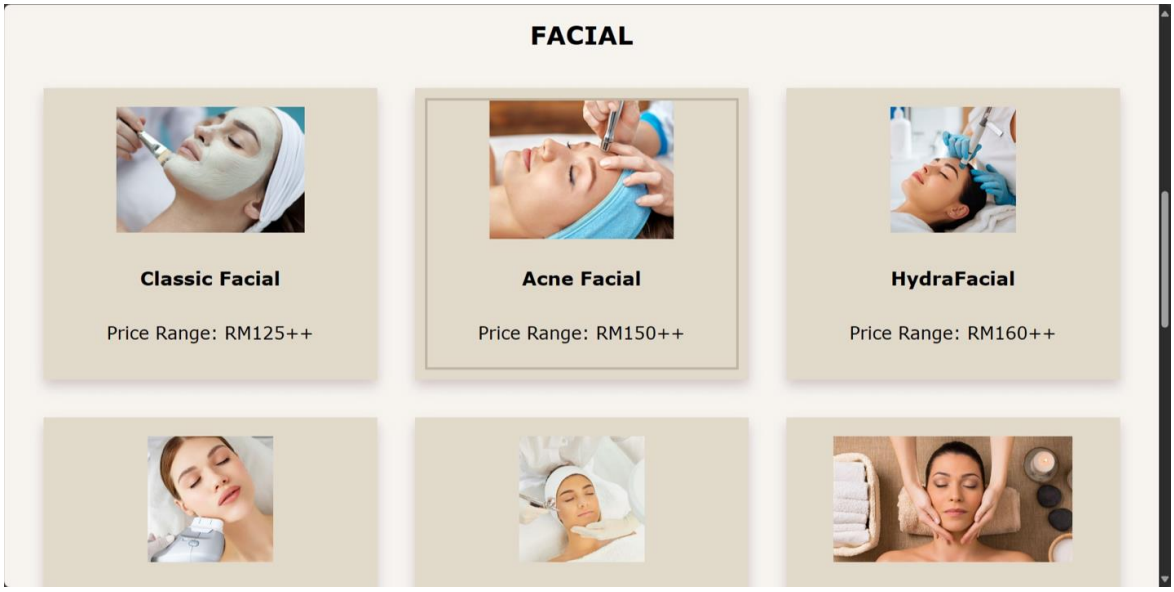


*Figure 4.3.2: Hover of each type of booking*

As the mouse moves through each booking container, there is a border display around it, providing visual interaction to the user.

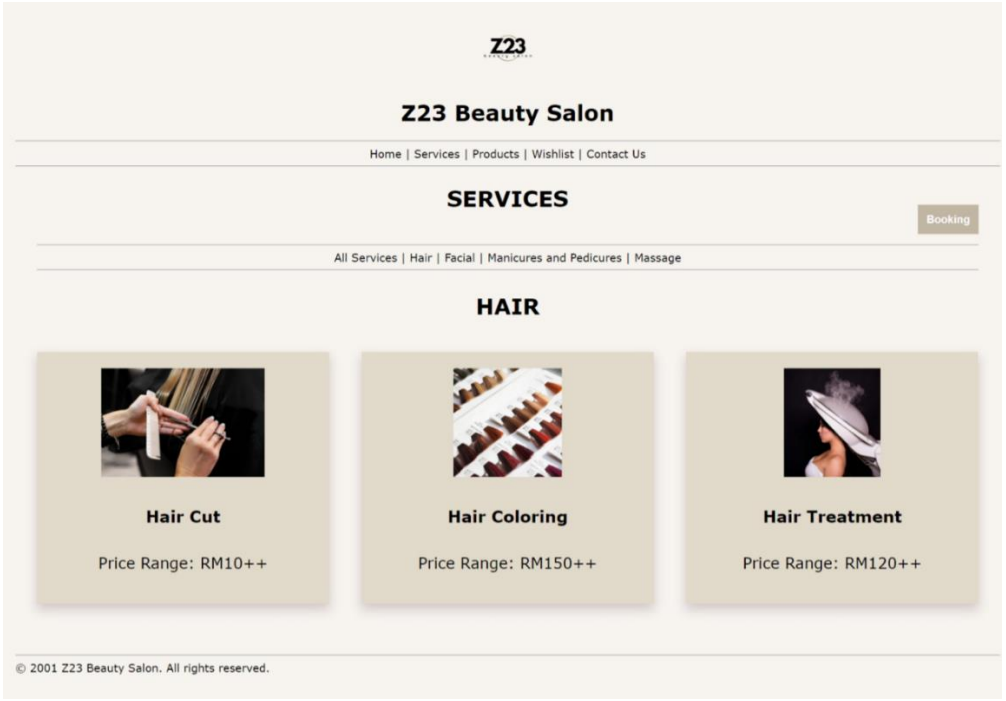


*Figure 4.3.4: Showcase of hair services*

In figure 4.3.4, according to the service navigation bar, the hair services are displayed when the keyword ‘Hair’ is clicked.

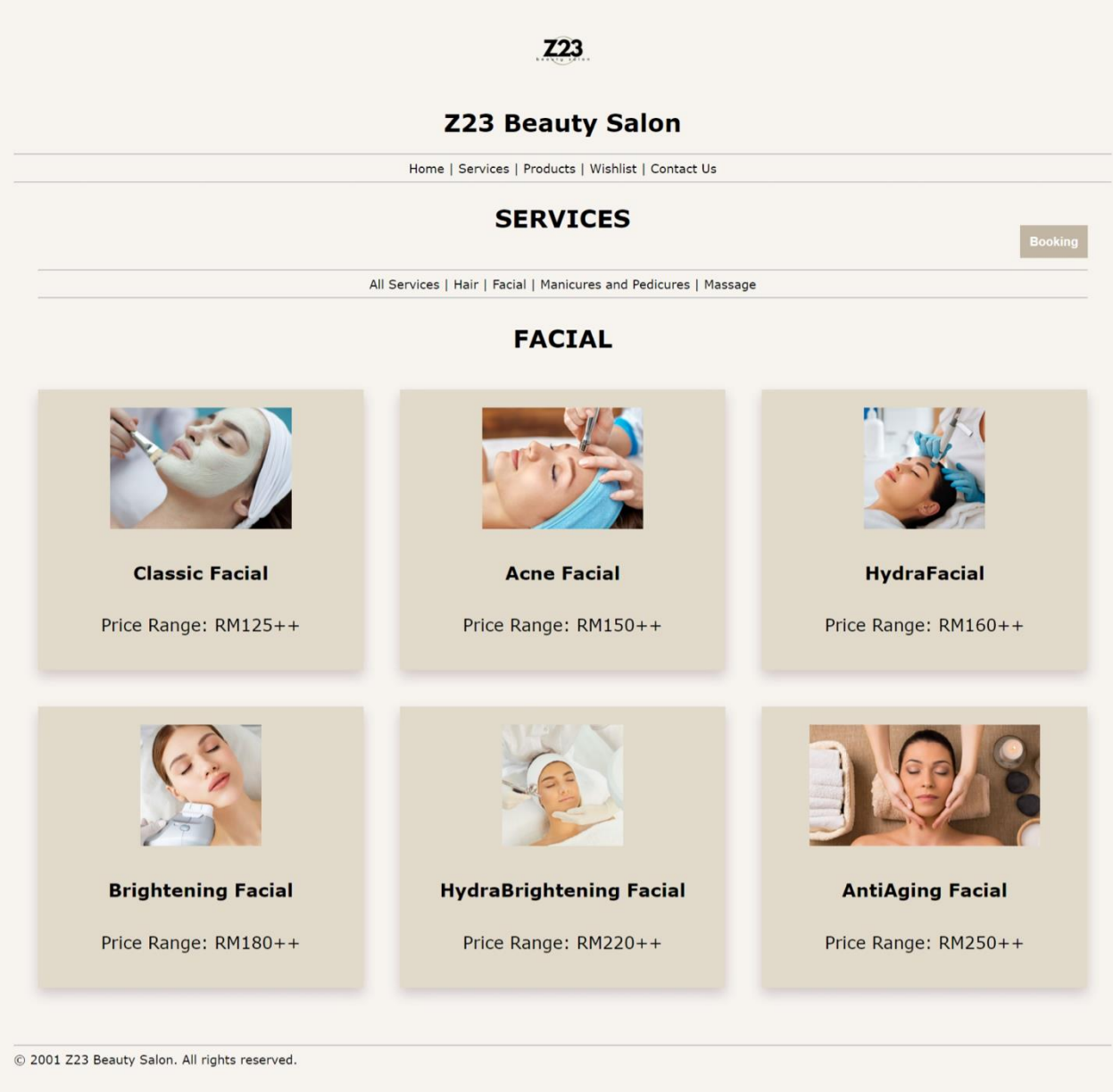


*Figure 4.3.5: Showcase of facial services*

In figure 4.3.5, according to the service navigation bar, the facial services are displayed when the keyword ‘Facial’ is clicked.

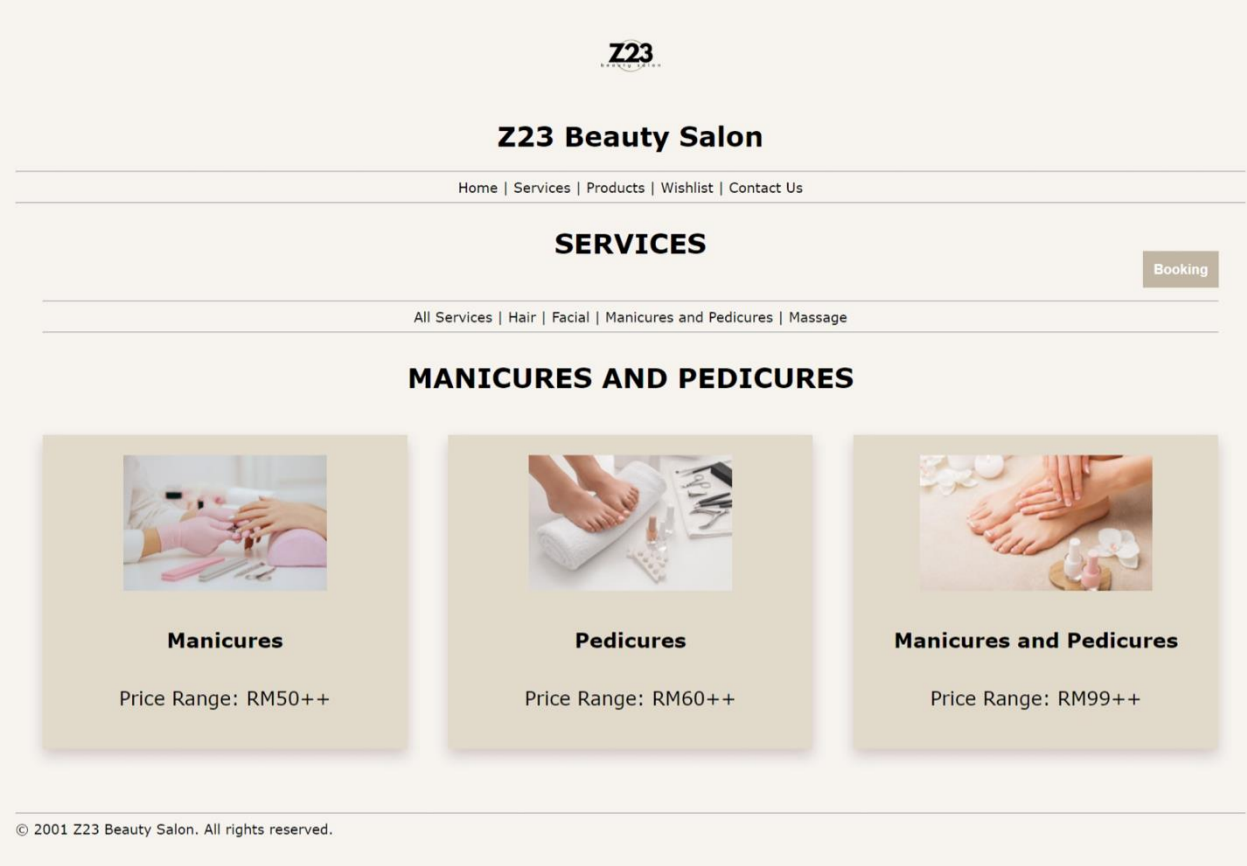


*Figure 4.3.6: Showcase of manicures and pedicures services*

In figure 4.3.6, according to the service navigation bar, manicures and pedicures services are displayed when the keyword ‘Manicures and Pedicures’ is clicked.

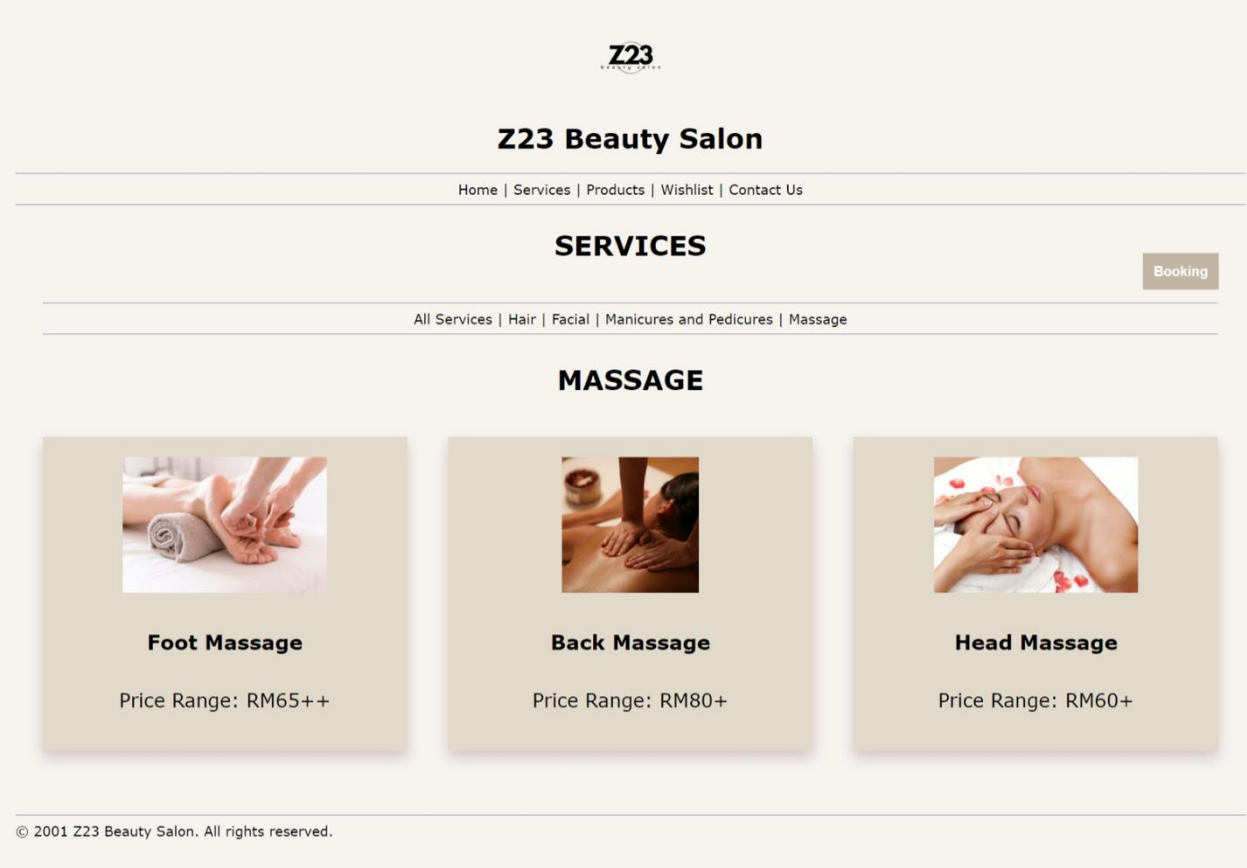


*Figure 4.3.7: Showcase of massage services*

In figure 4.3.7, according to the service navigation bar, the massage services are displayed when the keyword ‘Massage’ is clicked.

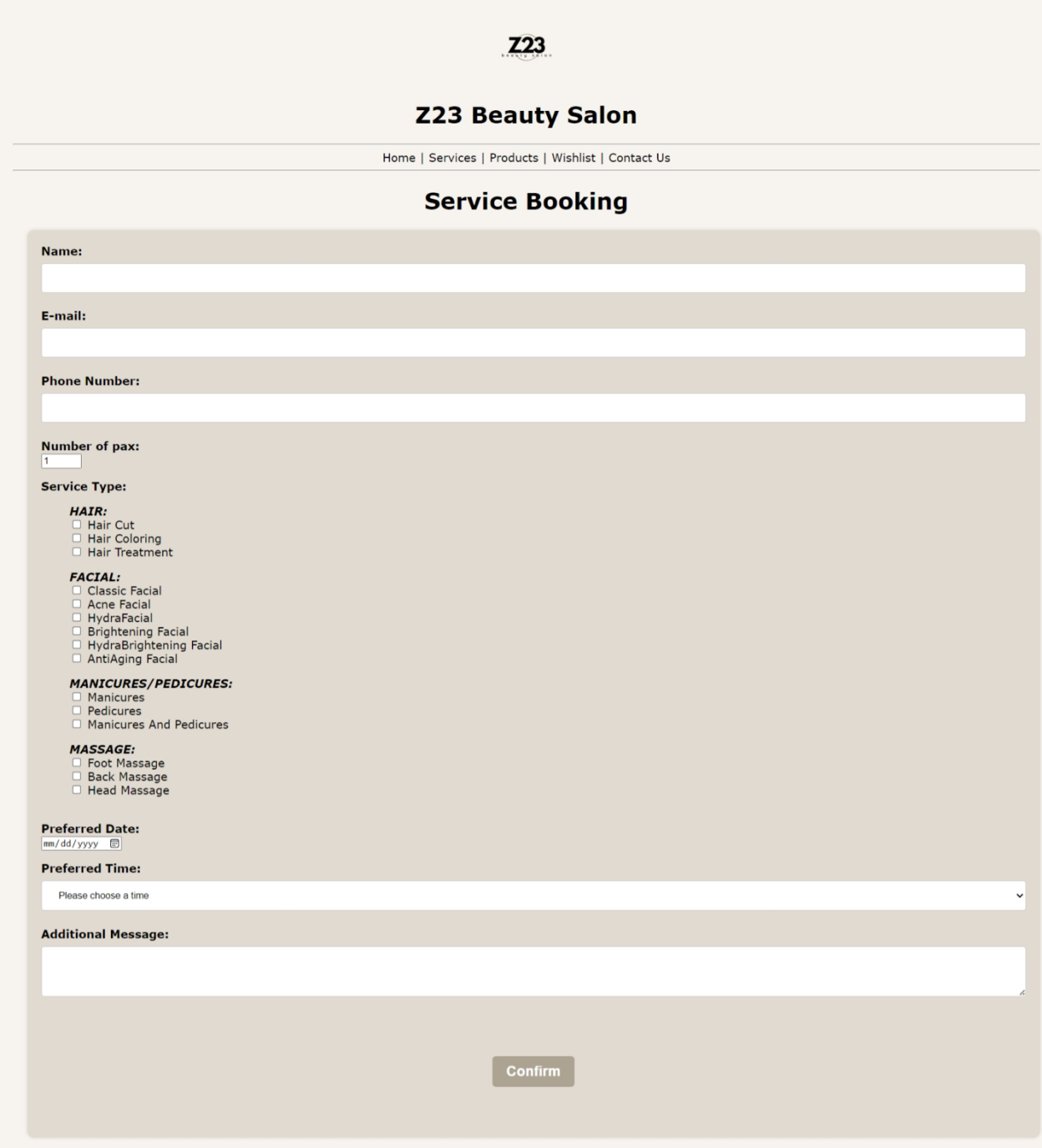


*Figure 4.3.8:Form for Service Booking*

Users must complete the form for service booking before they can continue on to the next step.

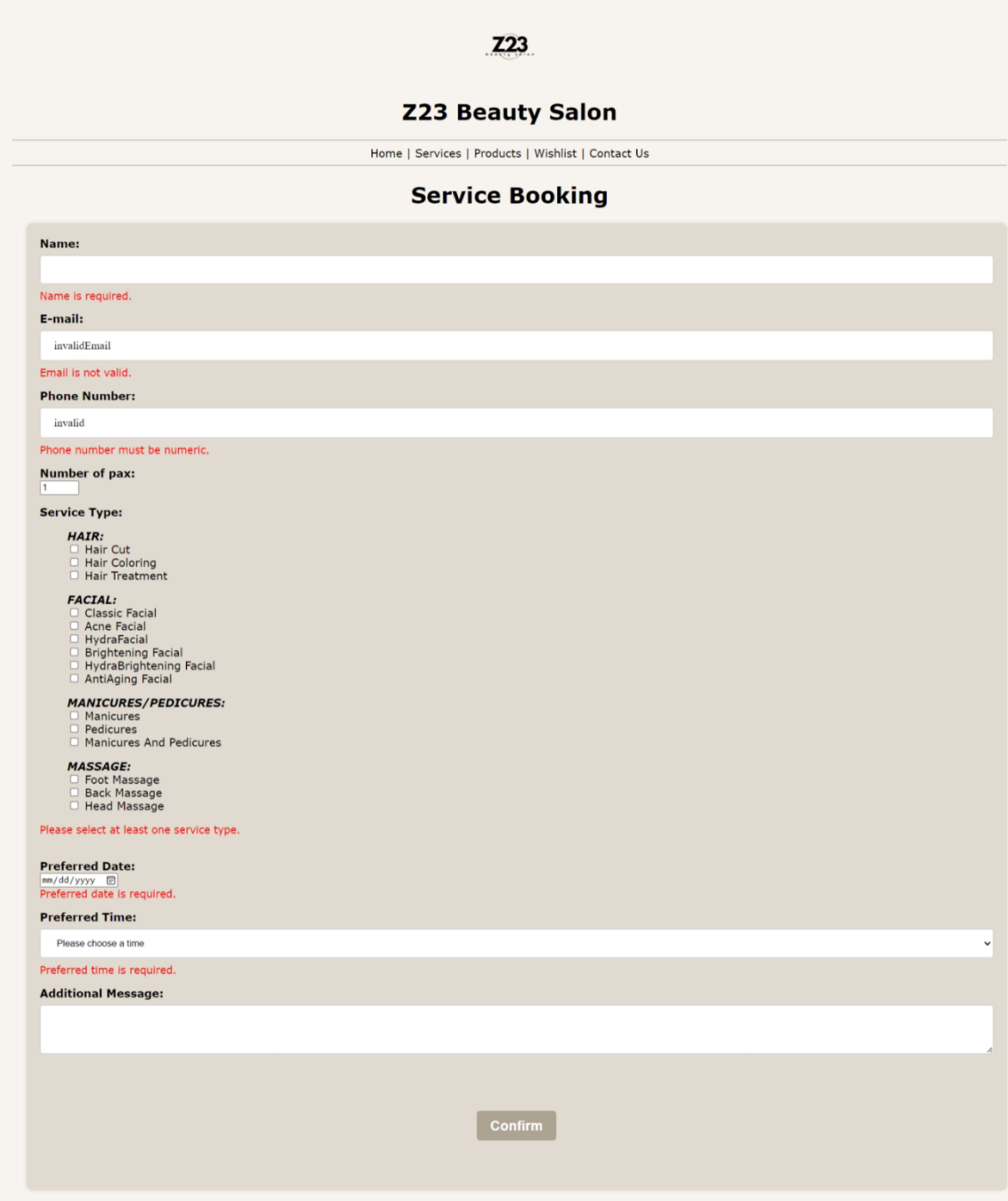


*Figure 4.3.9:Error Messages when invalid details submitted.*

According to *Figure 4.3.*9, error messages are implemented to guide users during the service booking process. When users submit the service booking form with empty fields or invalid details such as an incorrect email format or an invalid phone number, the system will display specific error messages. These error prompts ensure that users are alerted to any incomplete or erroneous inputs, prompting them to correct the information before proceeding.

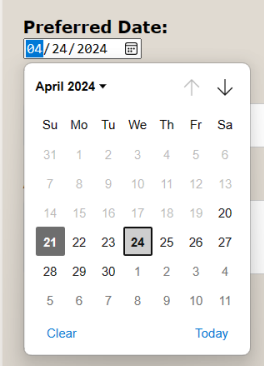


*Figure 4.3.10: Select Preferred Date.*

According to *Figure 4.3.10*, users are prompted to select a valid date for their appointment. The system ensures that the chosen date must be a future date, preventing users from selecting past dates.


*Figure 4.3.11: Form after being filled with valid details.*

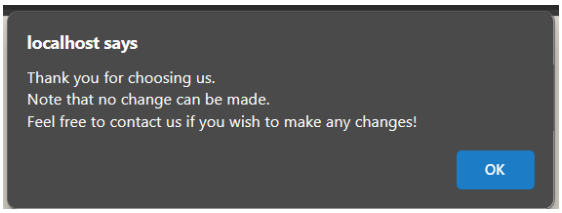


*Figure 4.3.12:Post message to inform users that no changes can be made once details submitted.*

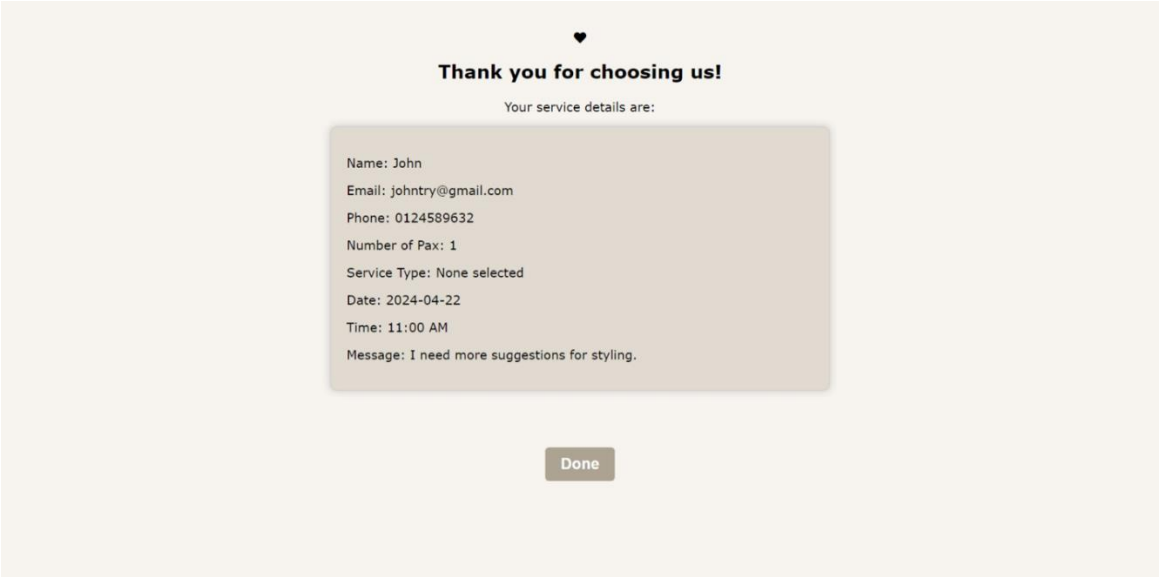


*Figure 4.3.13:Display user's service details upon successful submitted.*

According to *Figure 4.3.13*, upon successfully submitting their valid service details, users are presented with a summary of their booking information for further review. After reviewing their service details, users are provided with a convenient "Done" button that directs users back to the home page, allowing them to continue exploring the website or access other features.

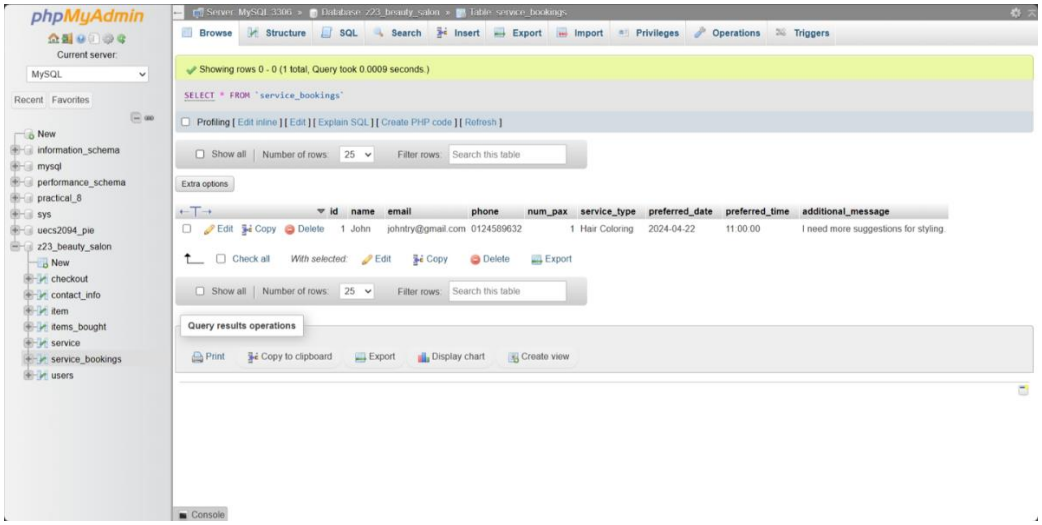


*Figure 4.3.14:Successful insert service details*

According to Figure 4.3.14, the successful submission of the service booking form results in the insertion of the user's service details into the “service_bookings” table within the database. This table serves as a repository for storing all the relevant information pertaining to service bookings made by users.

### 4.4. Product Page


*Figure 4.4.1: List of products*

Upon navigating into the product page, we may view the product list including all categories of products, hair categories, nail tool categories and facial categories with a transperant price list.

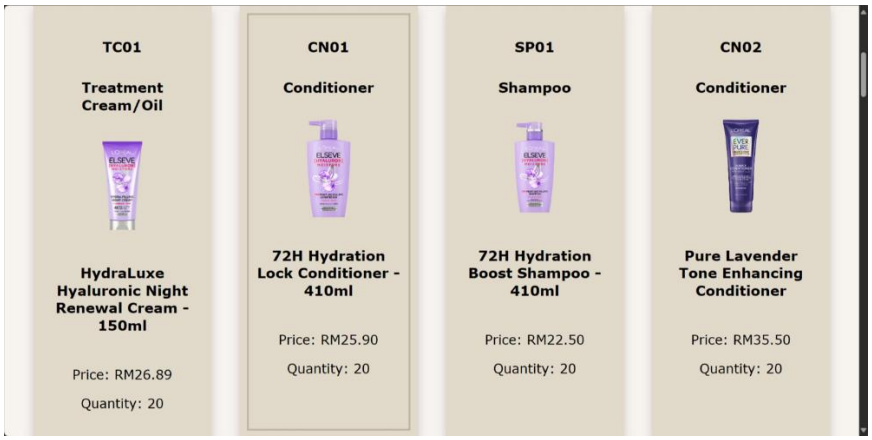


*Figure 4.4.2: Selection of product*

To view the details of each product, we may click on the image of the product to have further understanding about the particular product. For instance, the image of 72H Hydration Lock Conditioner- 410ml is pressed.

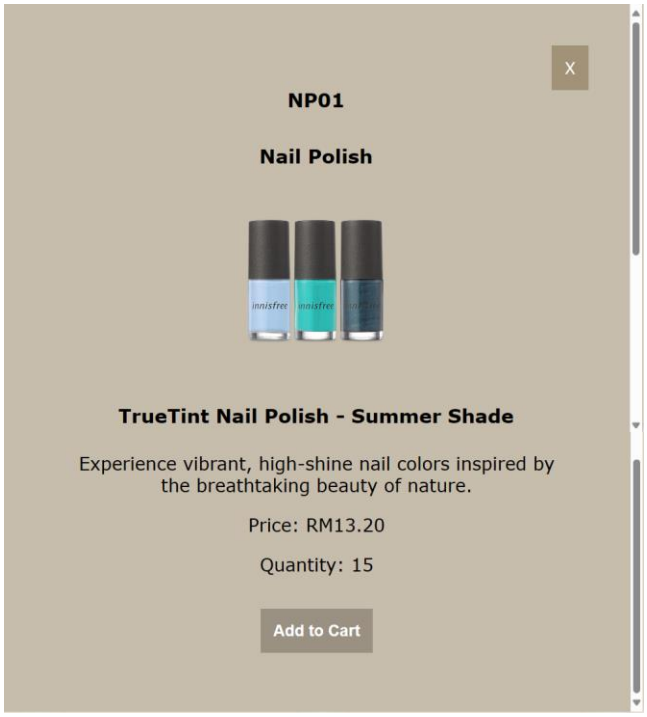


*Figure 4.4.3: Pop-up window*

There will be a pop-up window shown incorporating the item id, image, title, descriptions, price, quantity of items stored and an add to cart button. As we are interested to purchase the item, we may click on the add to cart button to add it into the wish list.

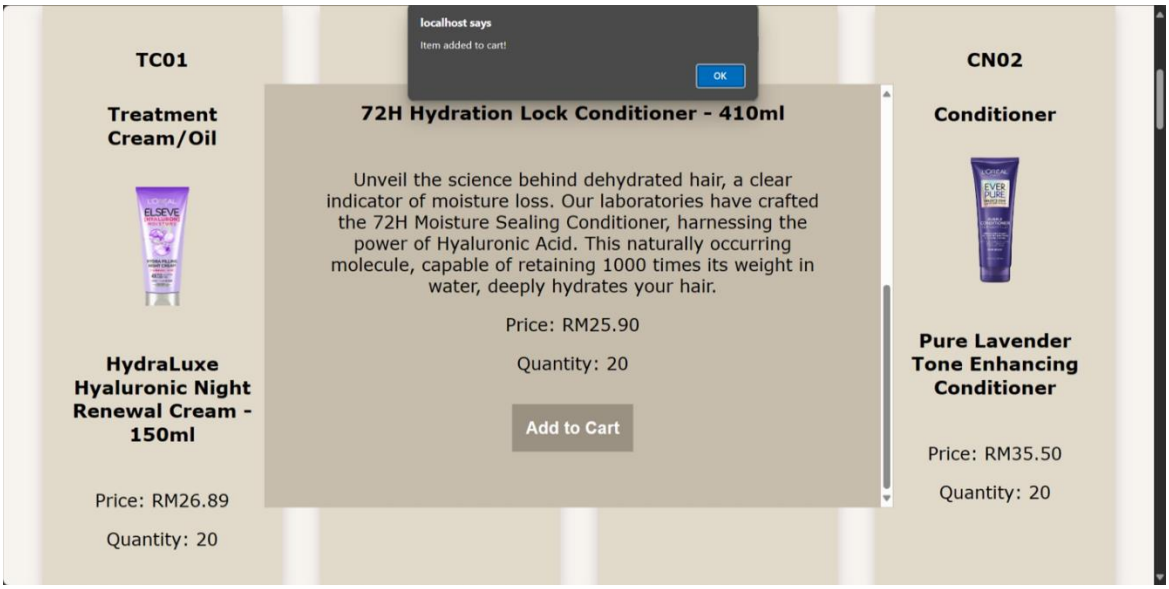


*Figure 4.4.4: Success of adding an item to the wish list*

After clicking on the add to cart button, it will shows an alert message to notify that the item is added to the cart. Then, we may move to the wish list to check the existance of the added item.

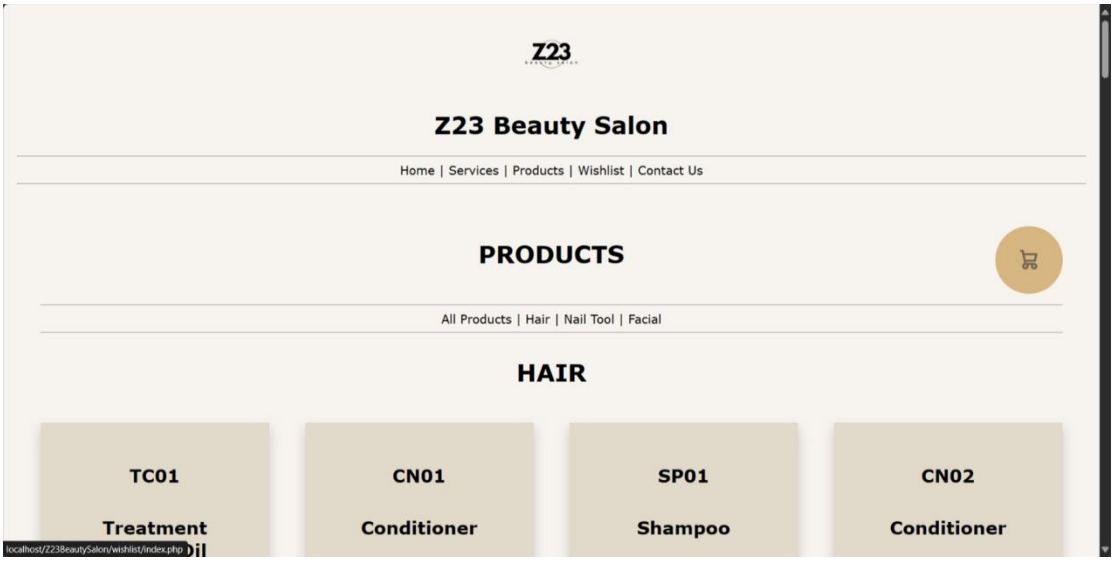


*Figure 4.4.5: Navigation to the wish list*

By moving our cursor, we may press on the cart image to navigate to the wish list which contains the items we had added to the cart just now. The cart button will show a different color as we move our cursor on it.

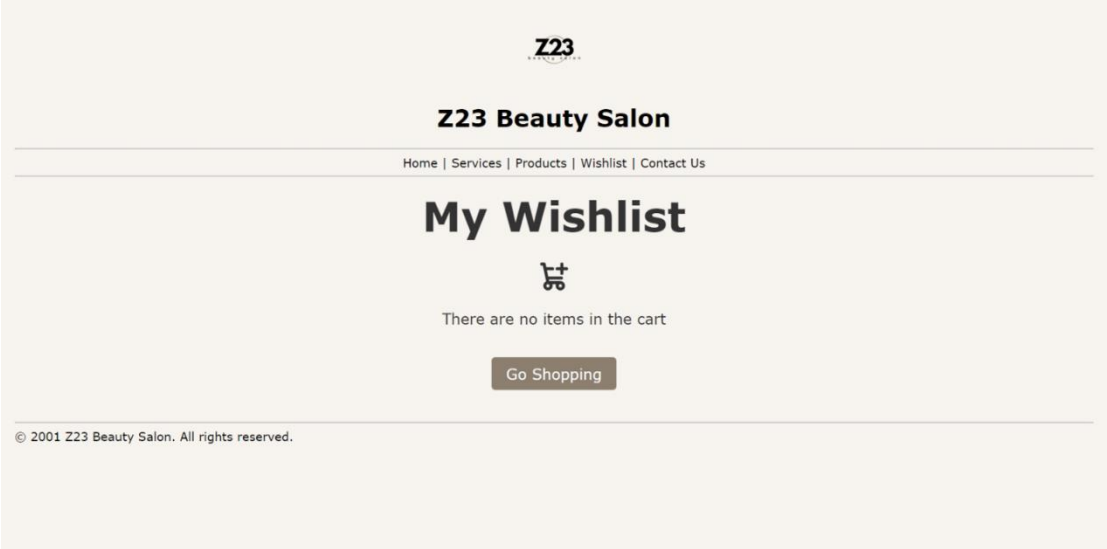


*Figure 4.4.6: Empty wish list*

When we navigated to the wish list, if there is no existane of items, it shows there are no items in the cart and encourage us to return to the product list by clicking on the go shopping button.

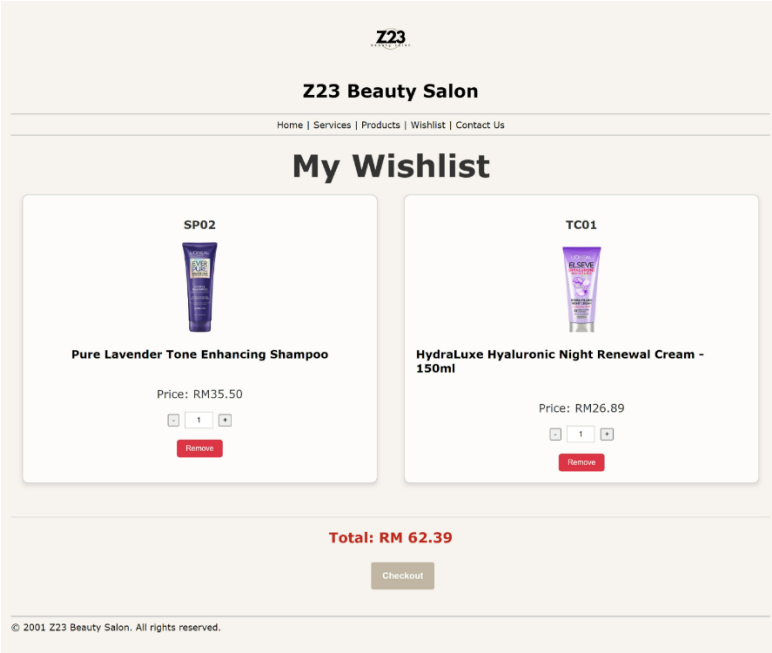


*Figure 4.4.7: Wish list with items added*

As we had added items into the cart just now, upon clicking on the cart image shown in Figure 4.4.5, the webpage will be redirected to the wish list. Here shows the two items added into the cart and the display of the total price.

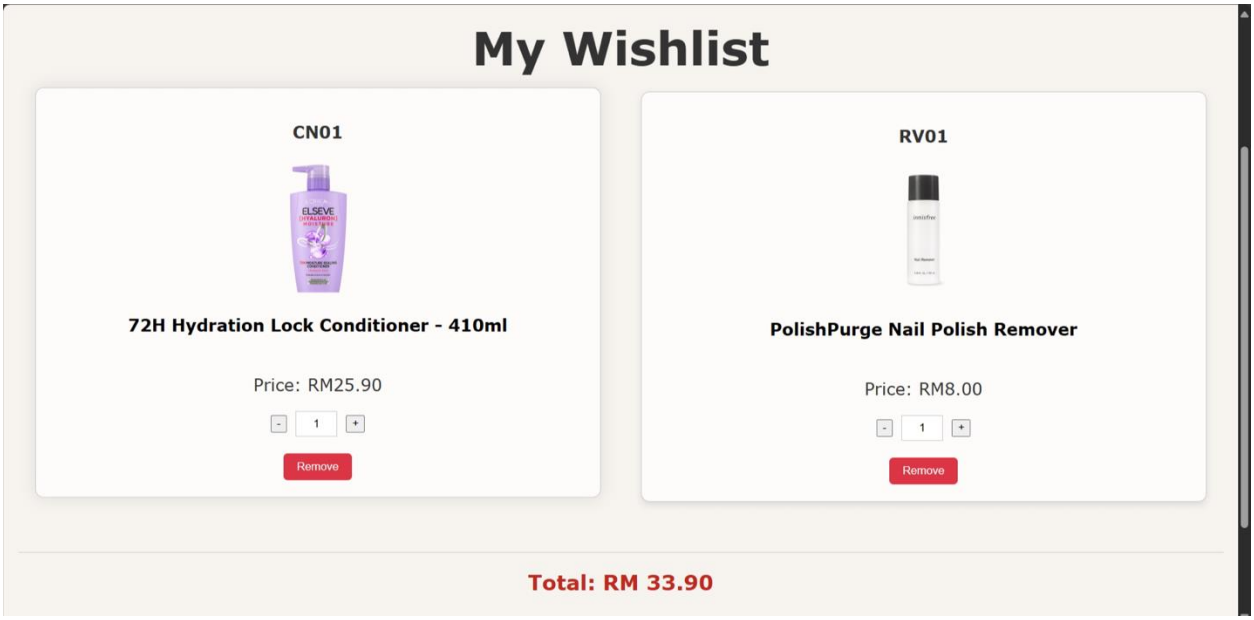


*Figure 4.4.8: Hover of each container*

When we move our mouse over all these containers, there is a hover effect to provide visual feedback and it increases the interactivity of elements to the webpage.

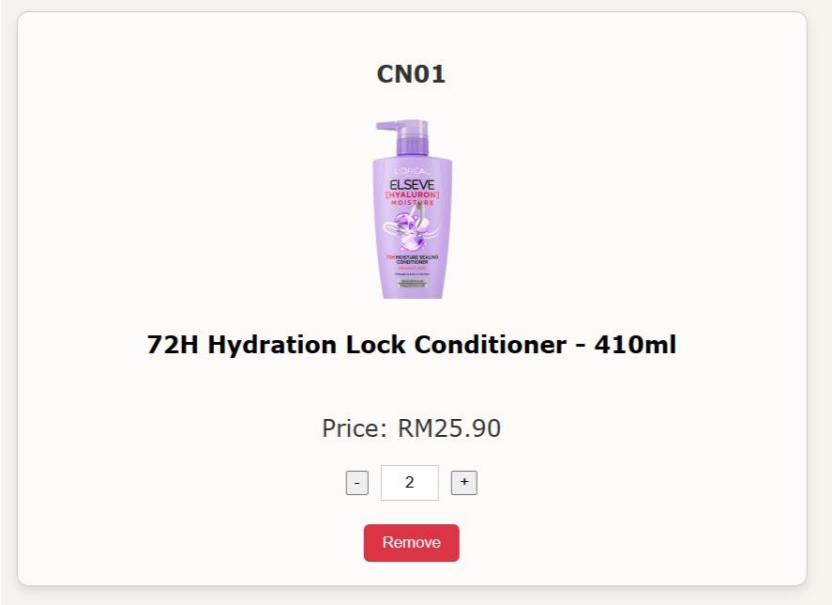


*Figure 4.4.9: Modify quantity of item selected*

We can increase the quantity of the item selected by pressing the '+' button. It will increase by one every time we press. If we want to decrease the quantity of the item chosen, we may click on the '-' button. Then, it will decrease the quantity by one upon clicking on it once.

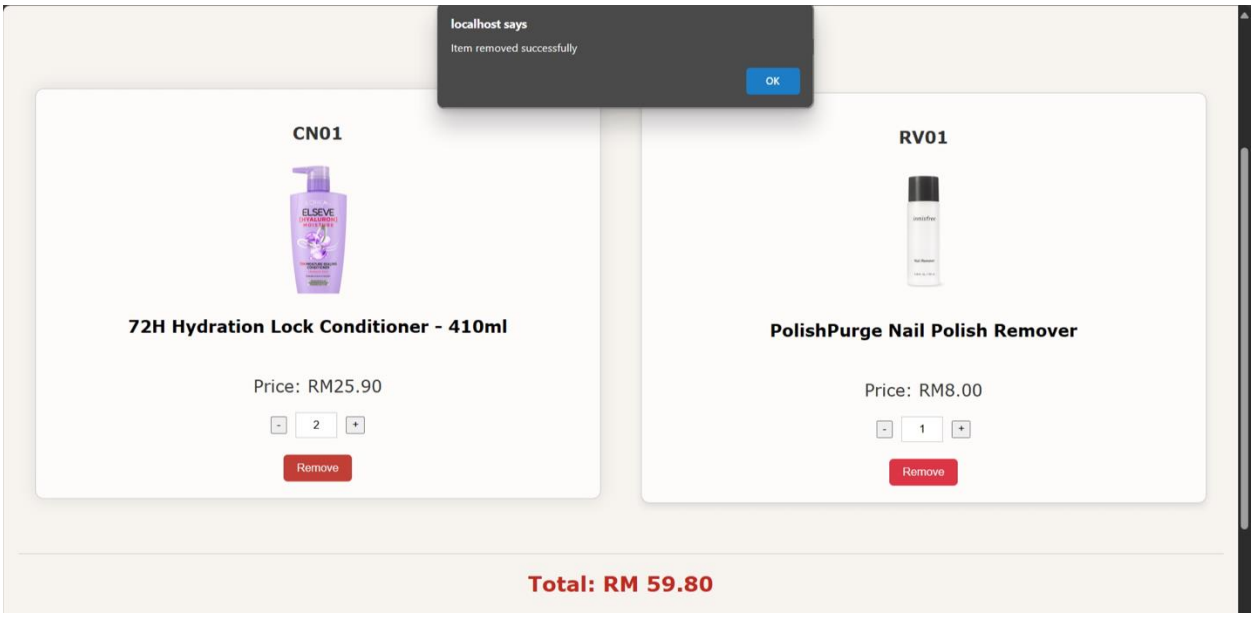


*Figure 4.4.10: Removal of unwanted item*

According to Figure 4.4.10, we would like to remove the 72H Hydration Lock Conditioner- 410ml. By pressing on the Remove button, the system will alert us by showing a message saying that the item removed successfully. Then, the container containing that removed item will be disappeared from the wish list.

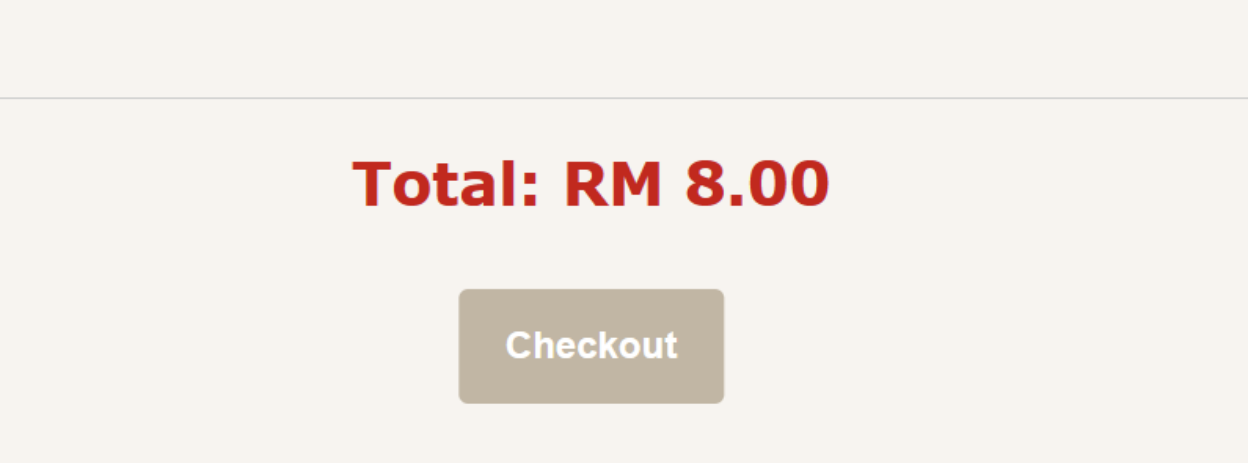


*Figure 4.4.11: Check out of products*

The price now becomes RM8.00 after the removal of the conditioner. As we had completed the process of managing items, we would like to proceed with the check out process. Before proceeding, the check out button is pressed and it will be redirected to the check out page.

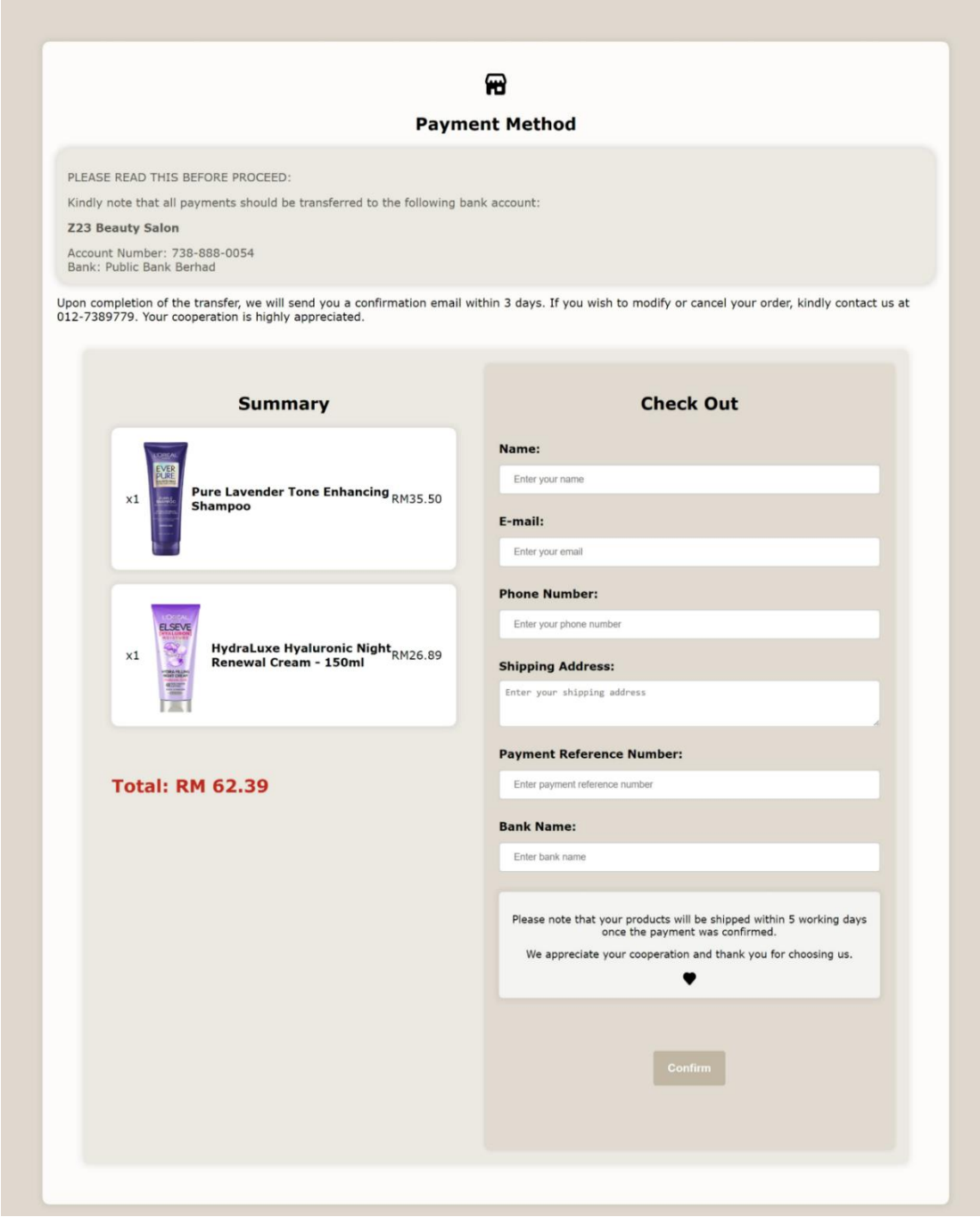


*Figure 4.4.12: Check out and payment page*

After pressing the check out button, we arrived at the check out and payment page. Payment method is shown at the top of the webpage. Several instructions are given to guide us upon filling in the check out form. The items confirmed to purchase will be show in the summary section of the webpage.

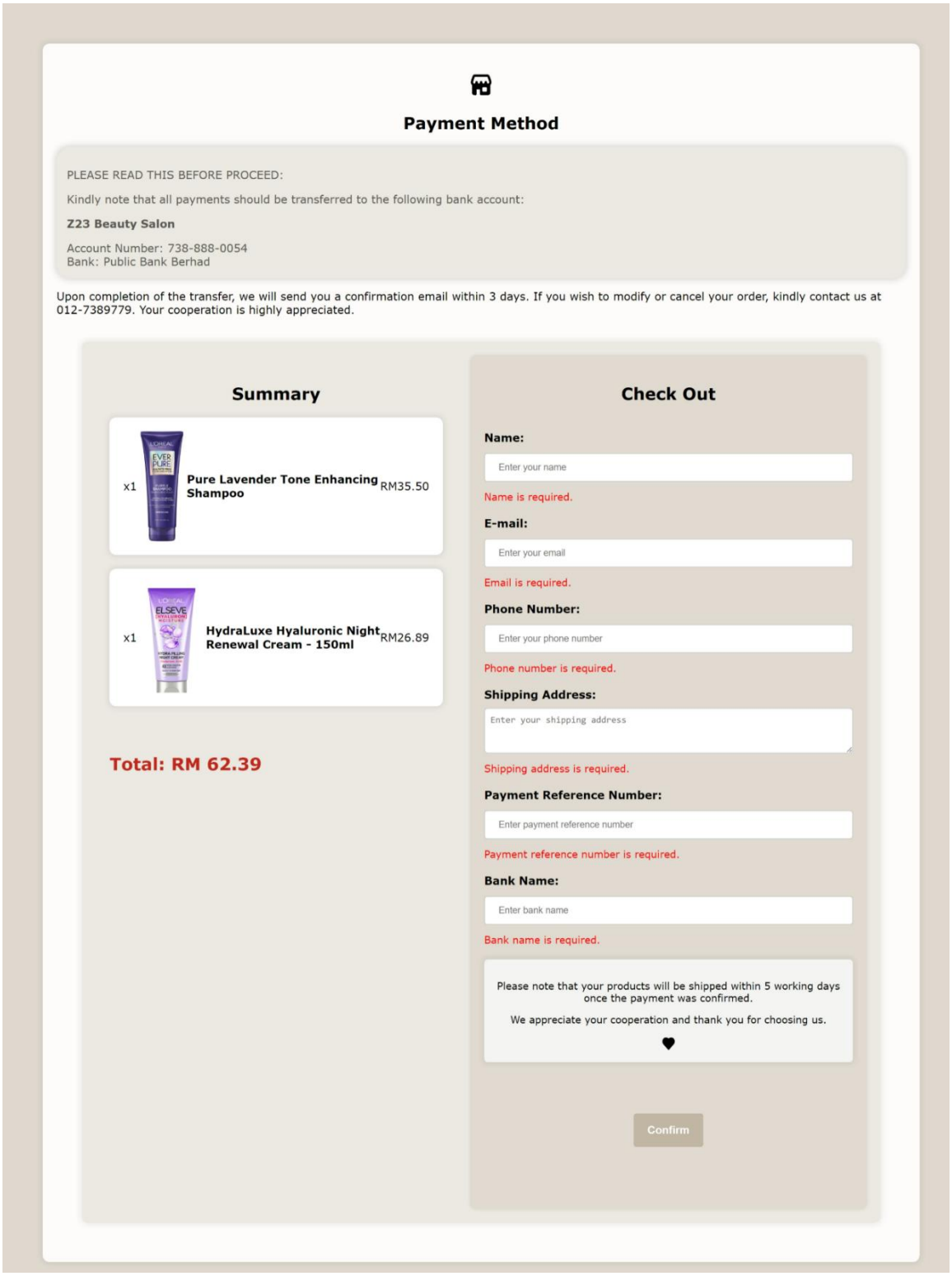


*Figure 4.4.12: Validation of check out form*

Every blanks in the check out form should be inserted correctly to complete the checkout process. The error messages provide a clear guide to us. Else, we may not check out any of these items.

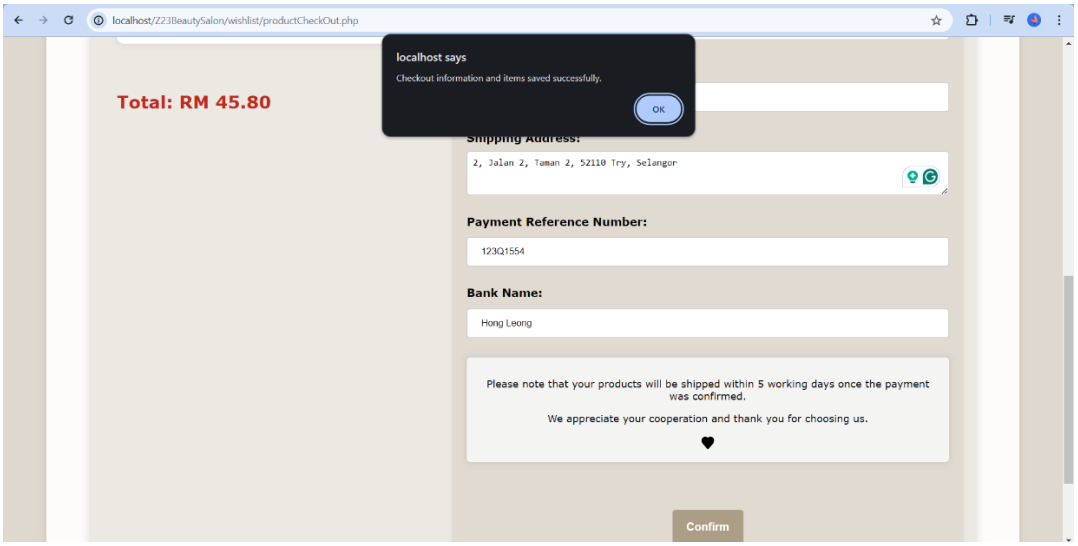


*Figure 4.4.13: Completion of check out process*

When we had completed the transaction and checkout form, the system will alert us that the cehckout information and items saved succesfully into the database. After clicking the 'OK' button, it will be redirected to the post message page.

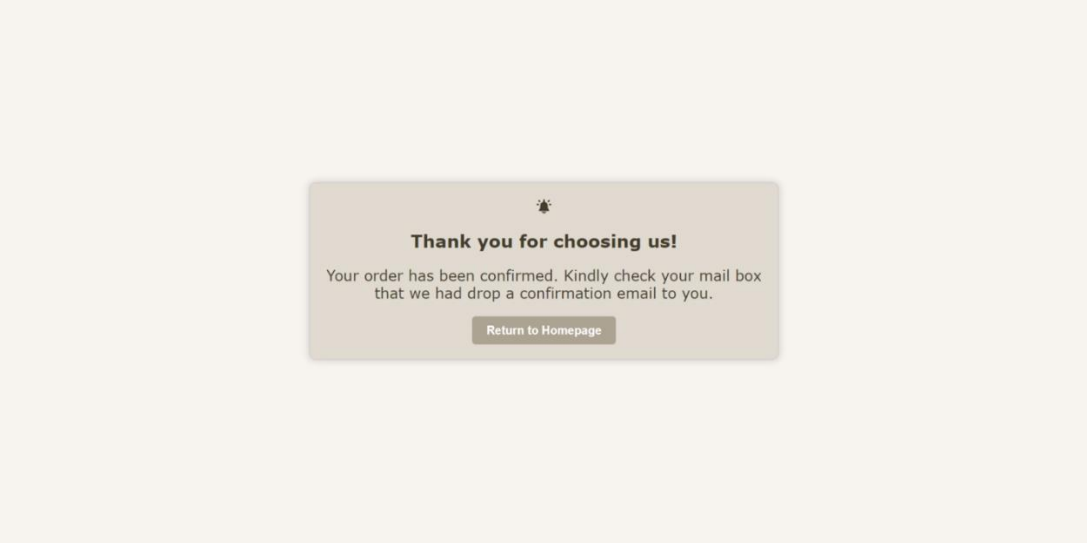


*Figure 4.4.14: Appreciation message*

An appreciation message is displayed after the successful completion of purchasing and checking out of the items. Now, we may click on the 'Return to Homepage' button to return to the home page and continue to explore more.

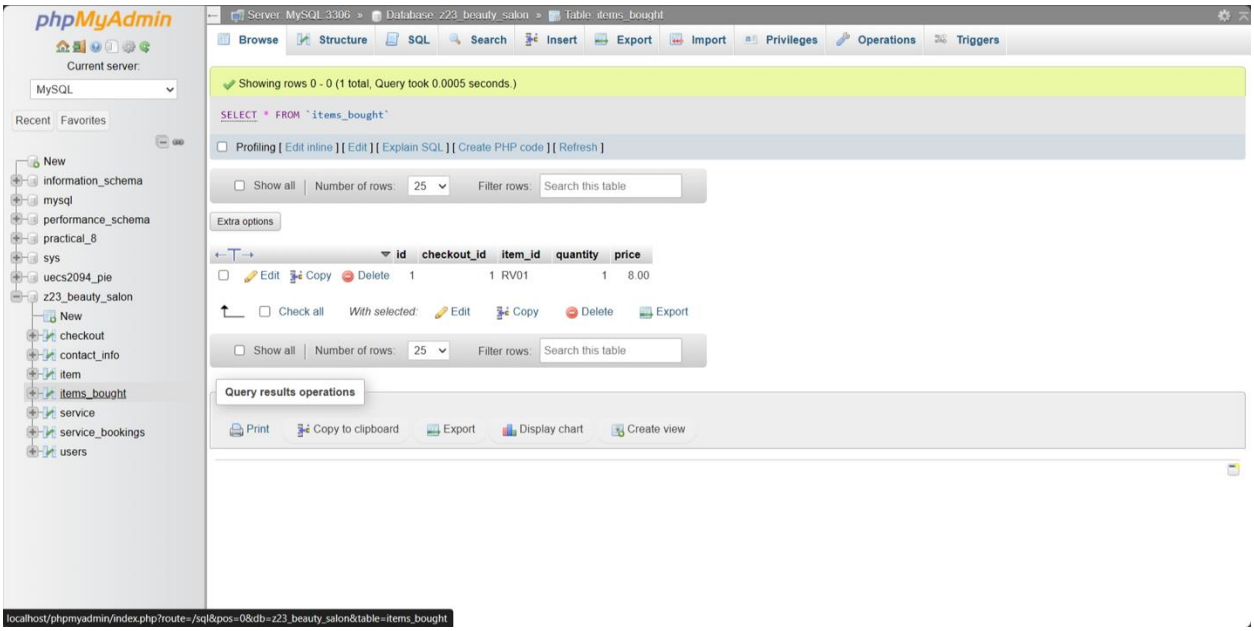


*Figure 4.4.15: Database for "items_bought" table*

The items bought are recorded into the database table namely "items_bought" allowing reference for the admins.

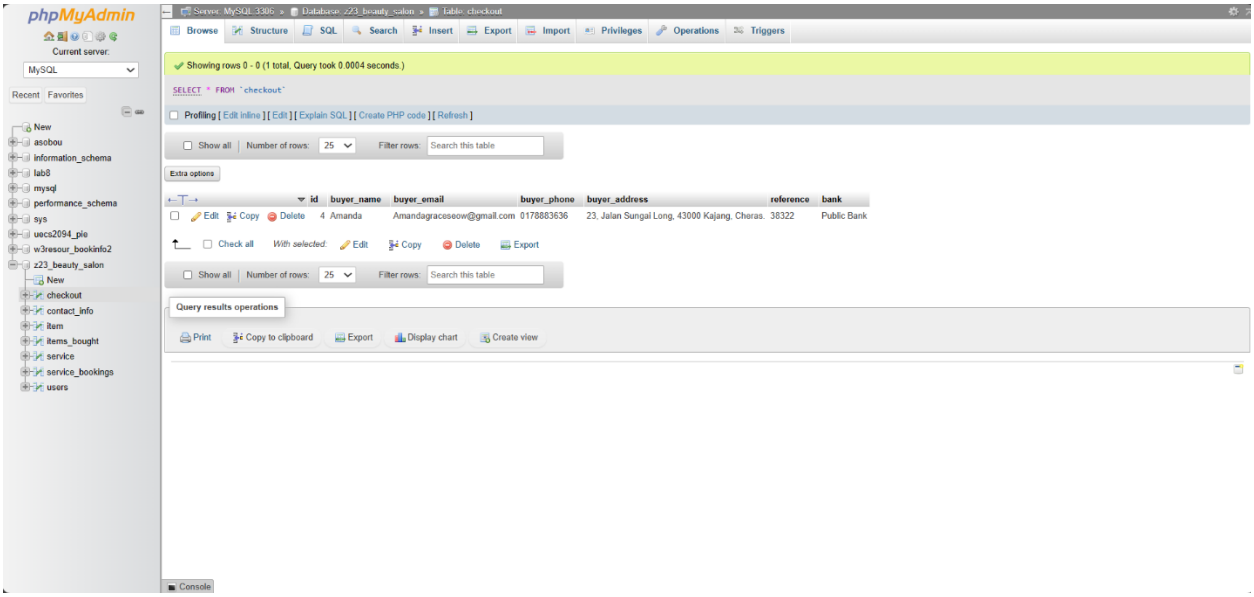


*Figure 4.4.15: Database for "checkout" table*

The checkout details are recorded into the database table namely "chekckout" allowing reference for the admins and shipping of items.

### 4.5. Contact Us Page

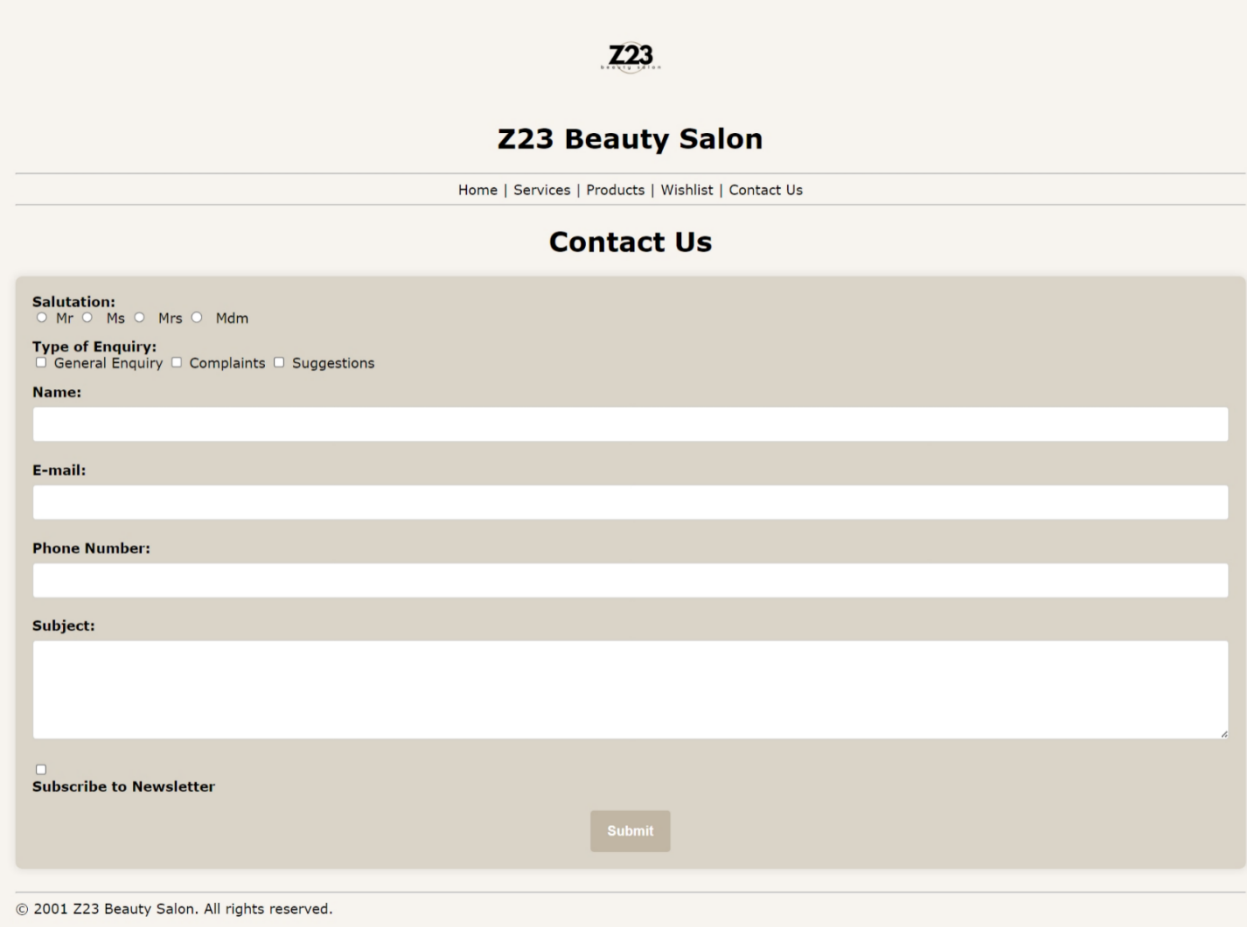


*Figure 4.5.1: Contact us page*

When we press the keyword ‘Contact Us’ at the navigation bar, it brings us to the contact us page attaching with a form for us to fill in our general enquiry, complaints and suggestions. We may fill up this form to sent a message to the beauty salon.

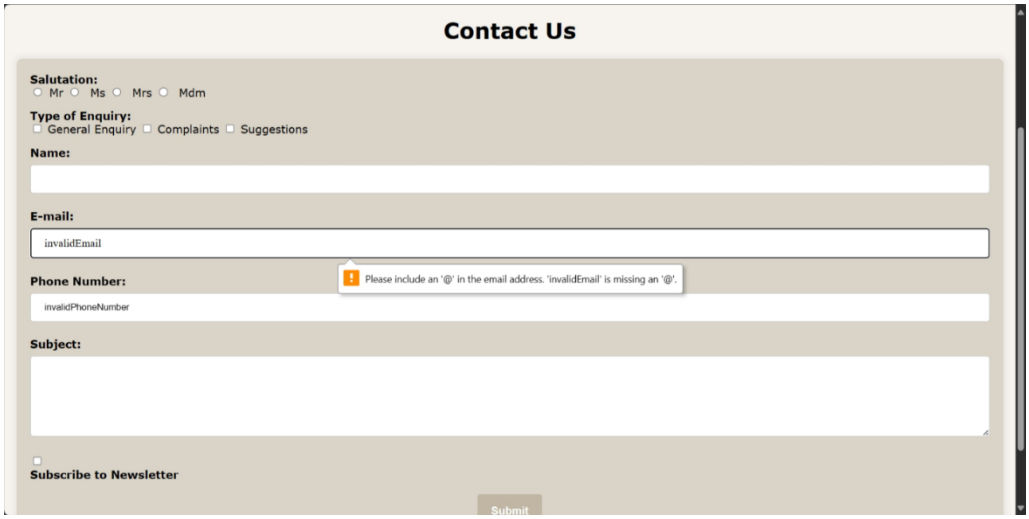


*Figure 4.5.2: Validation of email address*

A valid email address is required. If the email address is invaild, the form will not be submitted.

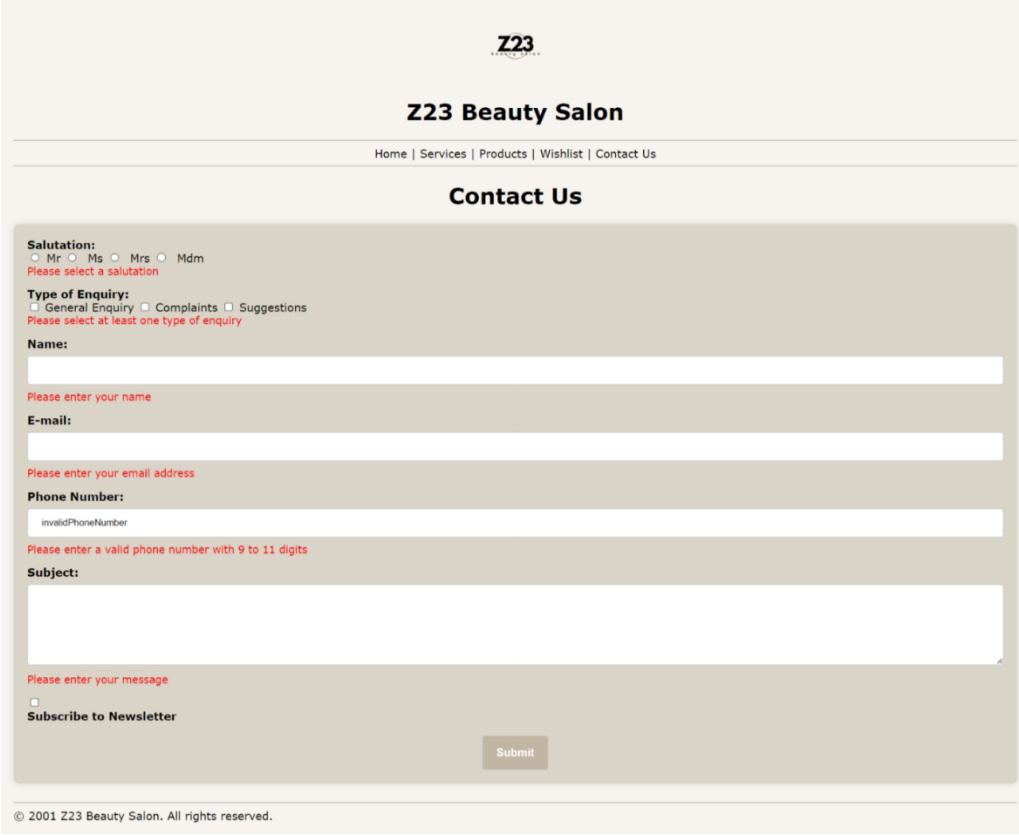


*Figure 4.5.3: Validation of  the contact us form*

To ensure the form is fully filled, validation is carried out. When there is any empty blanks detected, the error messages will be display. It provides a guide for us to complete the form correctly.

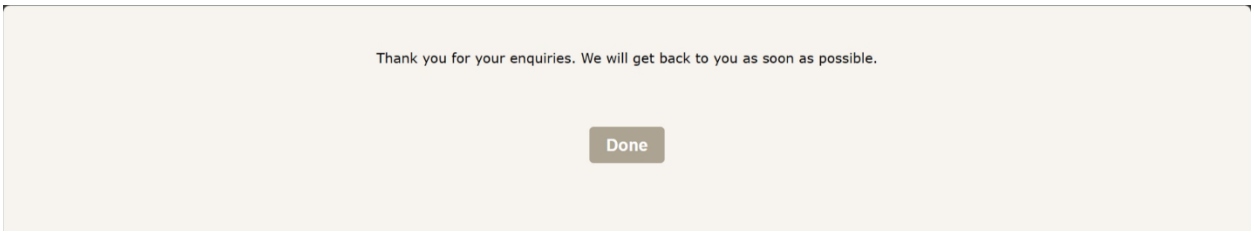


*Figure 4.5.4: Thank you message*

After completing the form and it can be successfully submitted, it will be redirect to the post message page showcasing a thank you message.

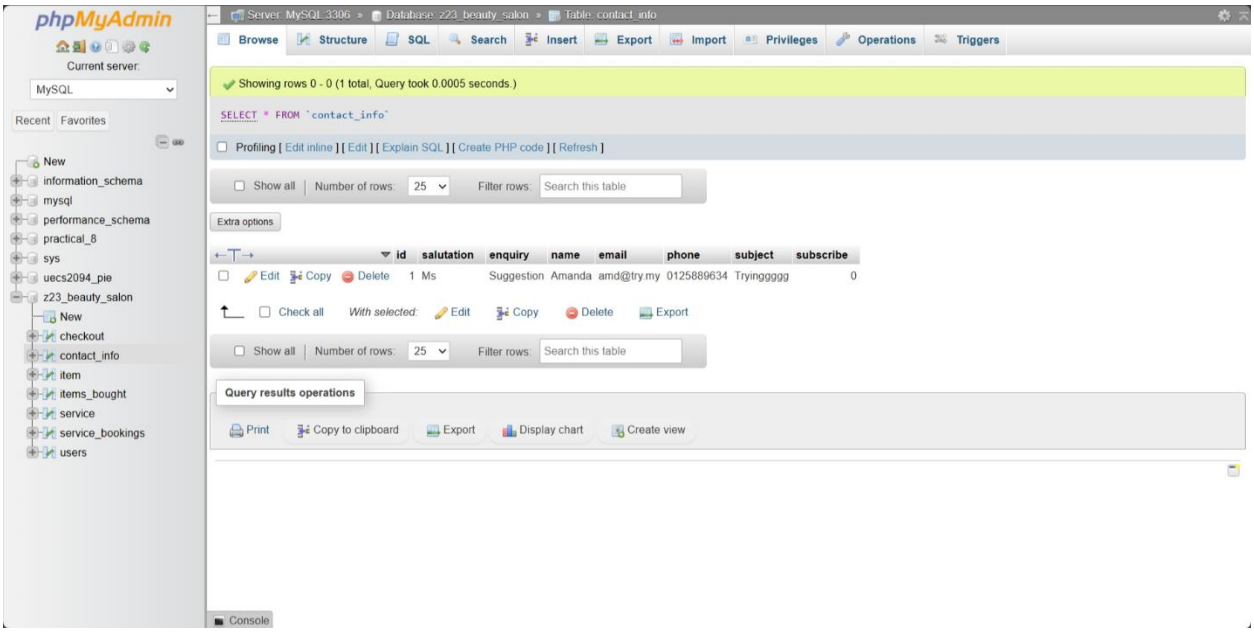


*Figure 4.5.5: Database for "contact_info" table*

Upon completion of the form, the information filled will be saved and recorded into the "contact_info" table in the database.

## 5. Conclusion

In conclusion, the development of Z23 Beauty Salon's web system aims to provide a seamless and user-friendly experience for both the salon and its clients. The website is equipped with a secure login system that allow user to login, logout and register a new account. The navigation bar enhances user experience, offering straightforward access to essential sections such as the home page, product listings, services, wishlist, and contact us.

The website boasts a comprehensive product listing, neatly categorized into Hair, Nail Tools, and Facial products. Users can effortlessly navigate through these categories, explore detailed product descriptions, and conveniently add items to their cart for a smooth shopping journey.

For those seeking for beauty services, the service section presents a diverse array of options including Hair, Facial, Manicures and Pedicures, and Massage. Each service category provides information about the offered services, accompanied by corresponding price ranges. The integrated booking system simplifies the process for users to schedule their desired services, allowing them to select preferred dates, times, and include any additional requests.

The wishlist functionality serves as a valuable tool for users, enabling them to save desired items for future reference. This feature streamlines the decision-making process before making a purchase, providing convenience and organization.

Additionally, the "Contact Us" section serves as a direct channel for users to communicate inquiries or feedback to the salon. The submission of the contact form ensures efficient handling of user queries by the salon's management.

Finally, the Z23 Beauty Salon web system offers a cohesive integration of features including a secure login system, navigation, product listings, booking processes, wishlist functionality, and seamless checkout procedures. This user-centric design aims to elevate the overall user experience, creating a convenient and enjoyable platform for beauty and wellness needs for both the salon and its clients.

**Workload Summary:**

| Members | Task Assigned + Done |
|---|---|
| **Amanda Grace Seow Chia Wern** | - Home Page<br>- User login system<br>- Code compilation<br>- Word File compilation |
| **Cheryl Ng Zi Qie** | - Home Page<br>- Contact Us Page<br>- Appointment Booking System and Page<br>- Word File compilation |
| **Low Si Qing** | - Service Page<br>- Item Page<br>- Flowchart<br>- Word File compilation |
| **Tan Wan Xuen** | - Wishlist Page<br>- Wishlist System<br>- Checkout and Payment Process<br>- Word File compilation |

**References:**

*Explore Pinterest*. (n.d.). Pinterest. https://www.pinterest.com/ideas/

Flowbite.com. (2023). Available at: https://flowbite.com/icons/.

*Hair Care - Hair Products & Advice*. (n.d.). L'Oréal Paris. Retrieved April 18, 2024, from https://www.lorealparis.com.my/hair/hair-care

*innisfree MY | Skincare & Beauty Products Inspired by Jeju*. (n.d.). Www.innisfree.my. Retrieved April 18, 2024, from https://www.innisfree.my/?utm_source=google&utm_medium=cpc&utm_campaign=sg_Innisfree&utm_term=&gad_source=1&gclid=Cj0KCQjwiYOxBhC5ARIsAIvdH50bKry6EHPTkLYlciNNPXCG0rVnW97nZTLRKGDelBIeqqZ9BvxjETgaAkwCEALw_wcB