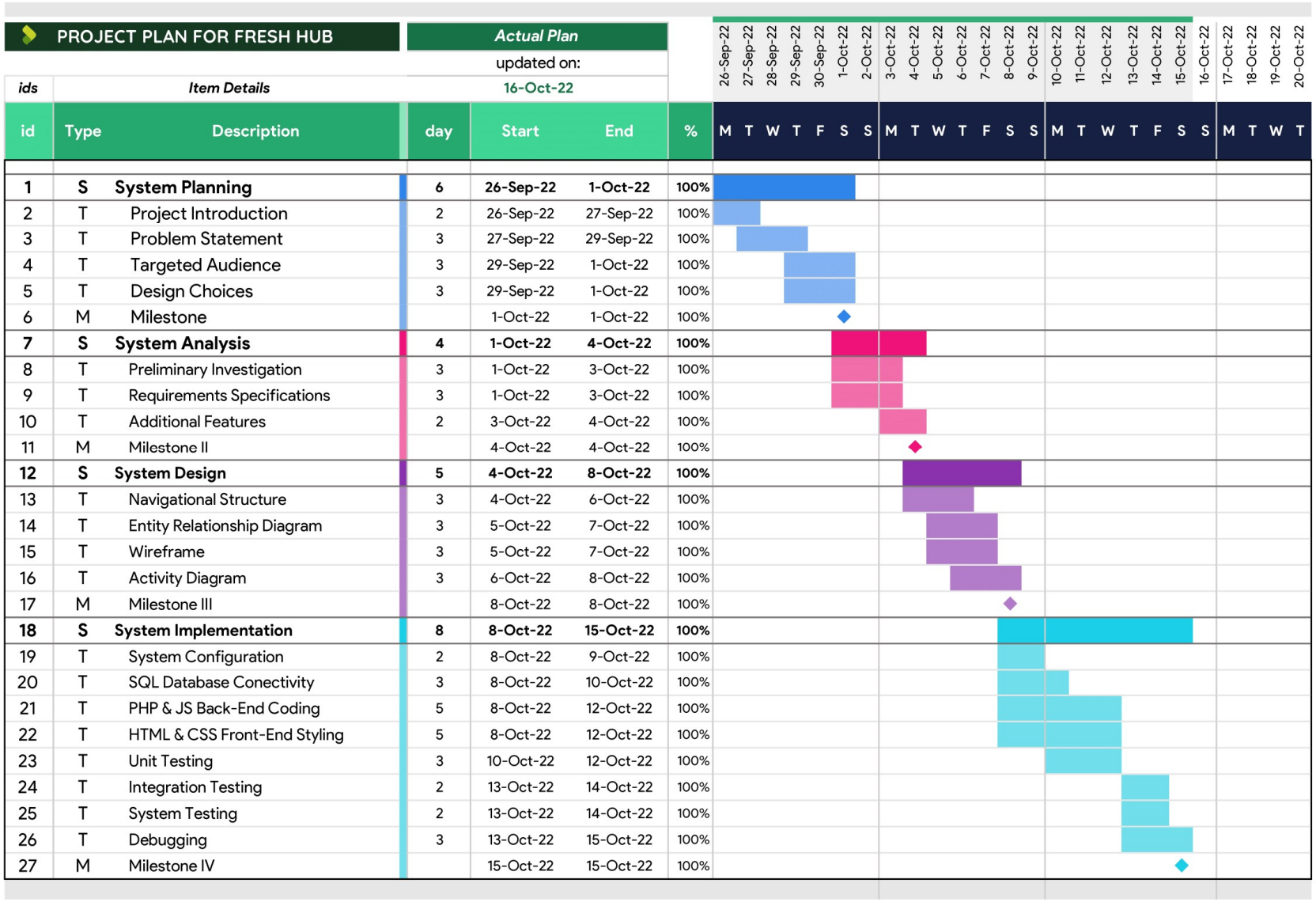
Module Code	:	AAPP009-4-2-WDT – Web Development
Intake Code	:	UCDF2104ICT(SE)
Lecturer Name	:	Daniel Mago Vistro
Hand in Date	:	16 October 2022
Lab No.	:	Lab 7
Group No.	:	19
Group Leader	:	Gan Ming Liang (TP063338)

Student ID	**Student Name**
TP063338	Gan Ming Liang
TP063248	Bryan Wong Win Kit
TP063403	Ian Joseph Lai Zi Jin

TABLE OF CONTENT

1.0 GANTT CHART



PROJECT PLAN FOR FRESH HUB

Actual Plan

updated on: 16-Oct-22

ids	Item Details						
id	Type	Description	day	Start	End	%
1	S	**System Planning**	6	26-Sep-22	1-Oct-22	100%
2	T	Project Introduction	2	26-Sep-22	27-Sep-22	100%
3	T	Problem Statement	3	27-Sep-22	29-Sep-22	100%
4	T	Targeted Audience	3	29-Sep-22	1-Oct-22	100%
5	T	Design Choices	3	29-Sep-22	1-Oct-22	100%
6	M	Milestone		1-Oct-22	1-Oct-22	100%
7	S	**System Analysis**	4	1-Oct-22	4-Oct-22	100%
8	T	Preliminary Investigation	3	1-Oct-22	3-Oct-22	100%
9	T	Requirements Specifications	3	1-Oct-22	3-Oct-22	100%
10	T	Additional Features	2	3-Oct-22	4-Oct-22	100%
11	M	Milestone II		4-Oct-22	4-Oct-22	100%
12	S	**System Design**	5	4-Oct-22	8-Oct-22	100%
13	T	Navigational Structure	3	4-Oct-22	6-Oct-22	100%
14	T	Entity Relationship Diagram	3	5-Oct-22	7-Oct-22	100%
15	T	Wireframe	3	5-Oct-22	7-Oct-22	100%
16	T	Activity Diagram	3	6-Oct-22	8-Oct-22	100%
17	M	Milestone III		8-Oct-22	8-Oct-22	100%
18	S	**System Implementation**	8	8-Oct-22	15-Oct-22	100%
19	T	System Configuration	2	8-Oct-22	9-Oct-22	100%
20	T	SQL Database Conectivity	3	8-Oct-22	10-Oct-22	100%
21	T	PHP & JS Back-End Coding	5	8-Oct-22	12-Oct-22	100%
22	T	HTML & CSS Front-End Styling	5	8-Oct-22	12-Oct-22	100%
23	T	Unit Testing	3	10-Oct-22	12-Oct-22	100%
24	T	Integration Testing	2	13-Oct-22	14-Oct-22	100%
25	T	System Testing	2	13-Oct-22	14-Oct-22	100%
26	T	Debugging	3	13-Oct-22	15-Oct-22	100%
27	M	Milestone IV		15-Oct-22	15-Oct-22	100%

Figure 1.0 – *Gantt Chart, Fresh Hub*

1.1 Project Plan

No.	Task Name	Start Date	End Date	Duration (Days)
1	**SYSTEM PLANNING**	26-Sep-22	1-Oct-22	6
1.1	Project Introduction	26-Sep-22	27-Sep-22	2
1.2	Problem Statement	27-Sep-22	29-Sep-22	3
1.3	Targeted Audience	29-Sep-22	1-Oct-22	3
1.4	Design Choices	29-Sep-22	1-Oct-22	3
2	**SYSTEM ANALYSIS**	1-Oct-22	4-Oct-22	4
2.1	Preliminary Investigation	1-Oct-22	3-Oct-22	3
2.2	Requirements Specification	1-Oct-22	3-Oct-22	3
2.3	Additional Features	3-Oct-22	4-Oct-22	2
3	**SYSTEM DESIGN**	4-Oct-22	8-Oct-22	5
3.1	Navigation Structure	4-Oct-22	6-Oct-22	3
3.2	Entity Relationship Diagram	5-Oct-22	7-Oct-22	3
3.3	Wireframe	5-Oct-22	7-Oct-22	3
3.4	Activity Diagram	6-Oct-22	8-Oct-22	3
4	**SYSTEM IMPLEMENTATION**	8-Oct-22	15-Oct-22	8
4.1	System Configuration	8-Oct-22	9-Oct-22	2
4.2	SQL Database Connectivity	8-Oct-22	10-Oct-22	3
4.3	PHP & JS Back-End Coding	8-Oct-22	12-Oct-22	5
4.4	HTML & CSS Front-End Coding	8-Oct-22	12-Oct-22	5
4.5	Unit Testing	10-Oct-22	12-Oct-22	3
4.6	Integration Testing	13-Oct-22	14-Oct-22	2
4.7	System TESTING	13-Oct-22	14-Oct-22	2
4.8	Debugging	13-Oct-22	15-Oct-22	3

1.2 Workload Matrix

ASIA PACIFIC UNIVERSITY OF TECHNOLOGY AND INNOVATON
AAPP009-4-2-WDT (Web Development)
Student Coursework Workload Matrix - Grades and Feedback Attachment

INTAKE: UCDF2104ICT(SE)	STUDENT NAME	Bryan Wong Win Kit	Gan Ming Liang	Ian Joesph Lai Zi Jin		-	-
System Name: Scout	TP NO.	TP063248	TP063338	TP063403		-	-

A. Group Component

CLO	ASSIGNMENT COMPONENT	ALLOCATED MARKS	CONTRIBUTION PERCENTAGE	CONTRIBUTION PERCENTAGE	CONTRIBUTION PERCENTAGE	CONTRIBUTION PERCENTAGE	CONTRIBUTION PERCENTAGE	CONTRIBUTION PERCENTAGE	TOTAL %
1	Homepage, About Us, Header, Footers	-	0.00	100.00	0.00				100
2	Login, Register, Forget Password	-	100.00	0.00	0.00				100
3	Products Page, Admin Panels (Add, Modify, Delete Products)	-	0.00	0.00	100.00				100
4	Contact Us, View Order History, View Inquiries	-	0.00	100.00	0.00				100
5	Profile Page & Edit User Profile	-	0.00	0.00	100.00				100
6	Shopping Cart, Checkout Page	-	100.00	0.00	0.00				100
7	ERD, Navigational Structure		34.00	33.00	33.00				100
8	Gantt Chart and Project Plan		33.00	34.00	33.00				100
9	Documentation & Recordings	-	33.00	33.00	34.00				100
	Total Marks and Contribution	-	34%	33%	33%	0%	0%	0%	

Signature						

Figure 1.1 – *Workload Matrix, Group 19*

2.0 INTRODUCTION

2.1 Introduction

Over the years, agriculture has seen a growth in development in terms of its popularity, as well as demand. The agricultural market requires strong support to improve current food security issues, as well as the nutrition provided by these foods. The current situation undeniable as these foods play a critical role, being the main source of income for the world's poorest folk. Food quality, quantity, and diversity need to elevate to drive the market further to aid economic transformation.

Technology in today's world is being used in almost all parts of life, and agriculture should be no different from that. With the help of digital technology, farmers can access a pool of information that they have never had before. Knowledge of marketing, current trends and even agricultural knowledge can help accelerate their business. Incorporating digital technology will greatly aid the rate of growth of the agriculture market, further decreasing scarcity of food available to those in need. As such, our team has completed the research and development required to produce an **Online Organic Food Market System**, Fresh Hub, to assist in accelerating growth of the agriculture industry and most importantly, supporting those in need.

2.2 Objectives

- Accelerating Growth of the Agriculture Food Industry
- Increasing the standards of food quality & quantity
- Helping Farmers & Food Sellers achieve their current goals & beyond
- Spreading food availability towards areas of scarcity and those in need
- Collaborating with top level industry firms, further ensuring strength and integrity to achieve our goals

2.2 Targeted Audience

Figure 2.0 – *About Us, Fresh Hub*

Fresh Hub is designed to improve food security in developing countries whilst promoting healthy-eating as part of our everyday life. As such, our targeted audience falls under **low** to **middle-income family groups** of developing nations that have a combined monthly salary of $1000 or less. By offering affordable organic produce as an alternative to low-quality street foods, we hope to create a positive impact on the long-term health of our targeted demographics.

2.3 Design choice

2.3.1 Logo Design

The Fresh Hub logo is created using **Adobe Illustrator**. The design is inspired by a **monocot leaf** that signify the **organic** nature of our produces, with a translucent green gradient that denotes the **authenticity** of our products. The goal of this minimal logo is to tell our audience that it's easy to build a healthy diet when shopping with Fresh Hub.

Figure 2.1 – *Logo, Adobe Illustrator*

2.3.2 Colour Palette

The colour palette that we used for our web application is derived from Colorhunt.co and inspired by Clerksy.co. The philosophy behind the colour choice are as follows (Chapman, 2021):

- **Green**: Abundance, Nature
- **Yellow**: Happiness, Hope
- **Orange**: Energy, Vitality
- **Dark Red:** Longing, Courage

Figure 2.2 – *Colour Palette, Colorhunt*

2.3.3 Font Choices

The fonts we have chosen for our websites are **Calistoga** and **Didact Gothic.** Calistoga is a serif typeface, while Didact Gothic is a sans-serif typeface. Having diverse letterforms not only helps to establish a strong visual hierarchy among contents, but also enhance the website's personality (Birch, 2020).

- Serif: **Calistoga**
- Sans-Serif: Didact Gothic

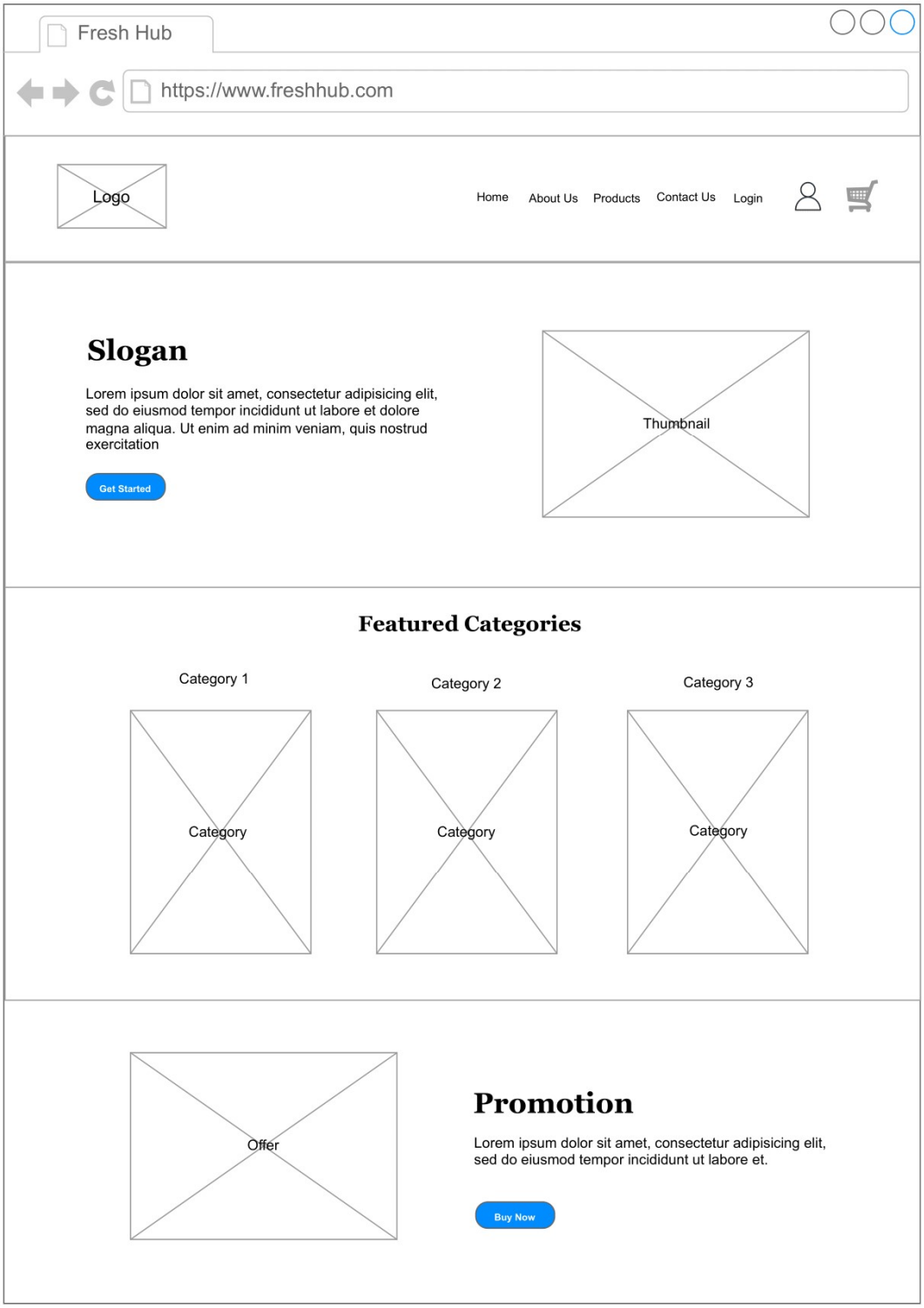
Figure 2.3 – *Calistoga, Google Fonts*

Figure 2.4 – *Didact Gothic, Google Fonts*

3.0 SYSTEM DESIGN

3.1 Wireframes

3.1.1 Homepage



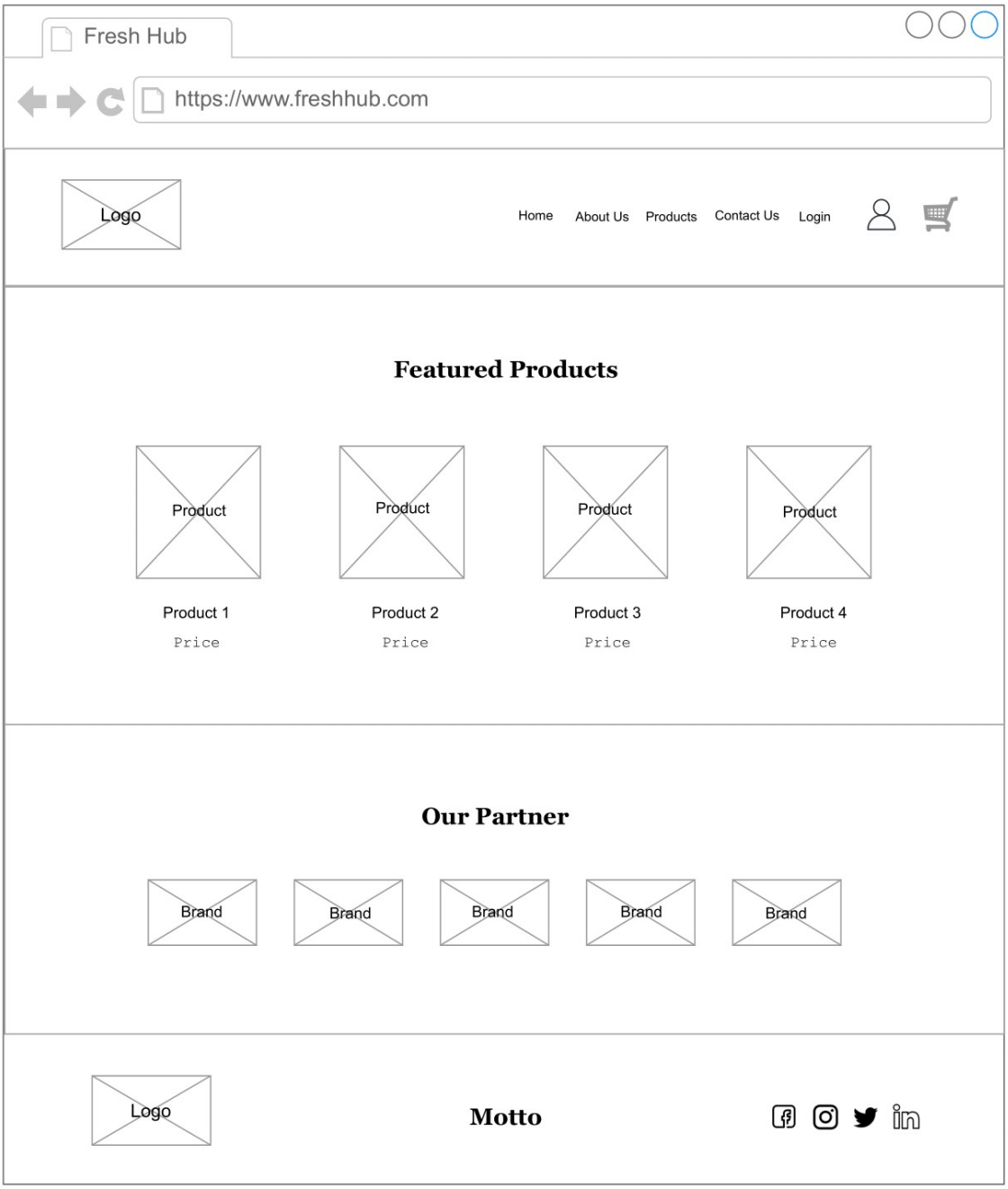


Figure 3.00 – *Homepage Wireframe, draw.io*

The homepage UI is designed to be functional and welcoming. There are lots of pictures to be found, with a sticky header that scrolls along with the page. Each section is separated into division and styled using a CSS box model guide by (Fitzgerald, 2022).

3.1.2 About Us

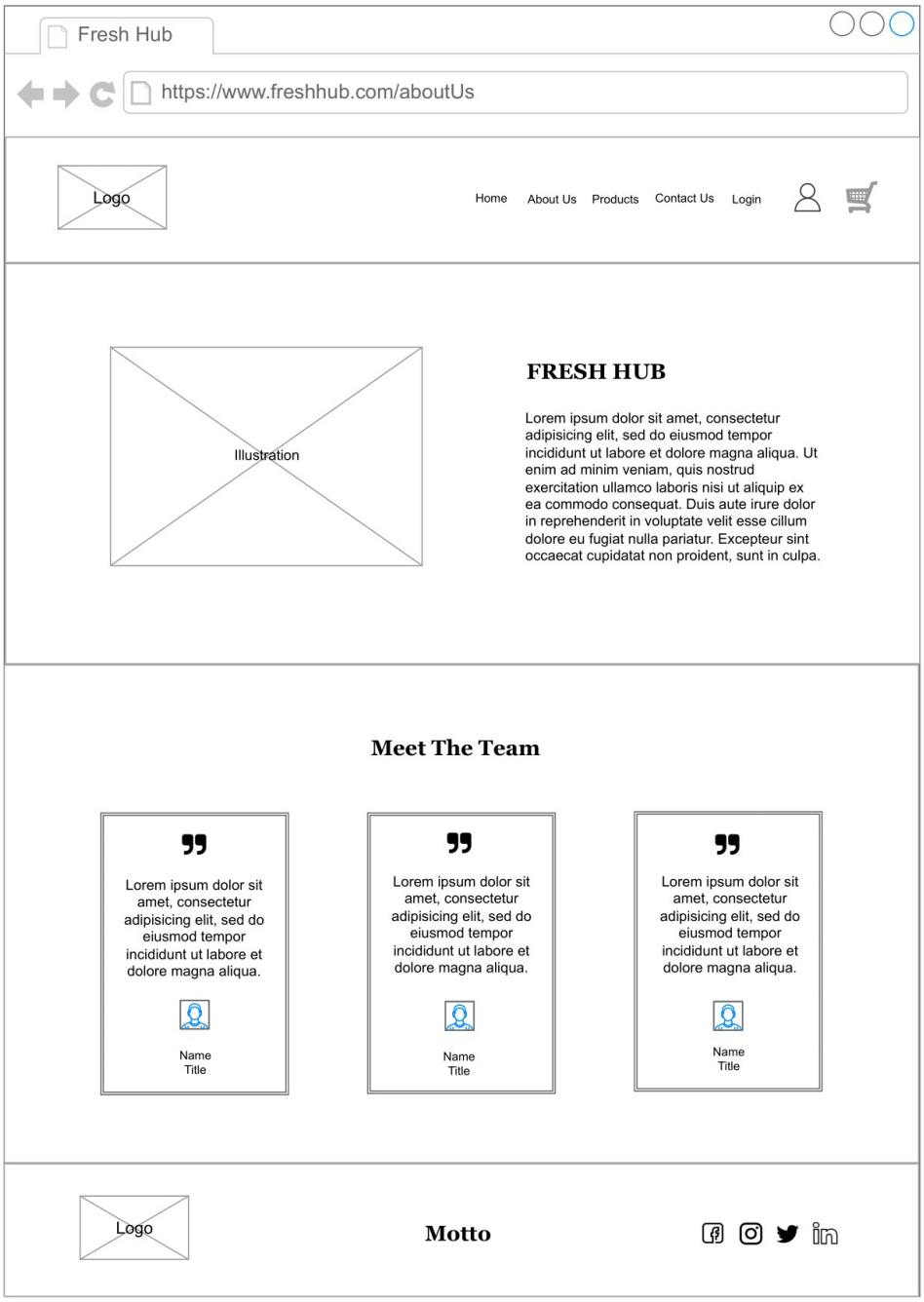


Figure 3.01 – *About Us Wireframe, draw.io*

The about us page is kept to a simple design of just two sections. The top division features a introduction to the company with an illustrator of the company. Whilst the bottom division features the 3 core Fresh Hub developers that have coded the website.

3.1.2 Contact Us

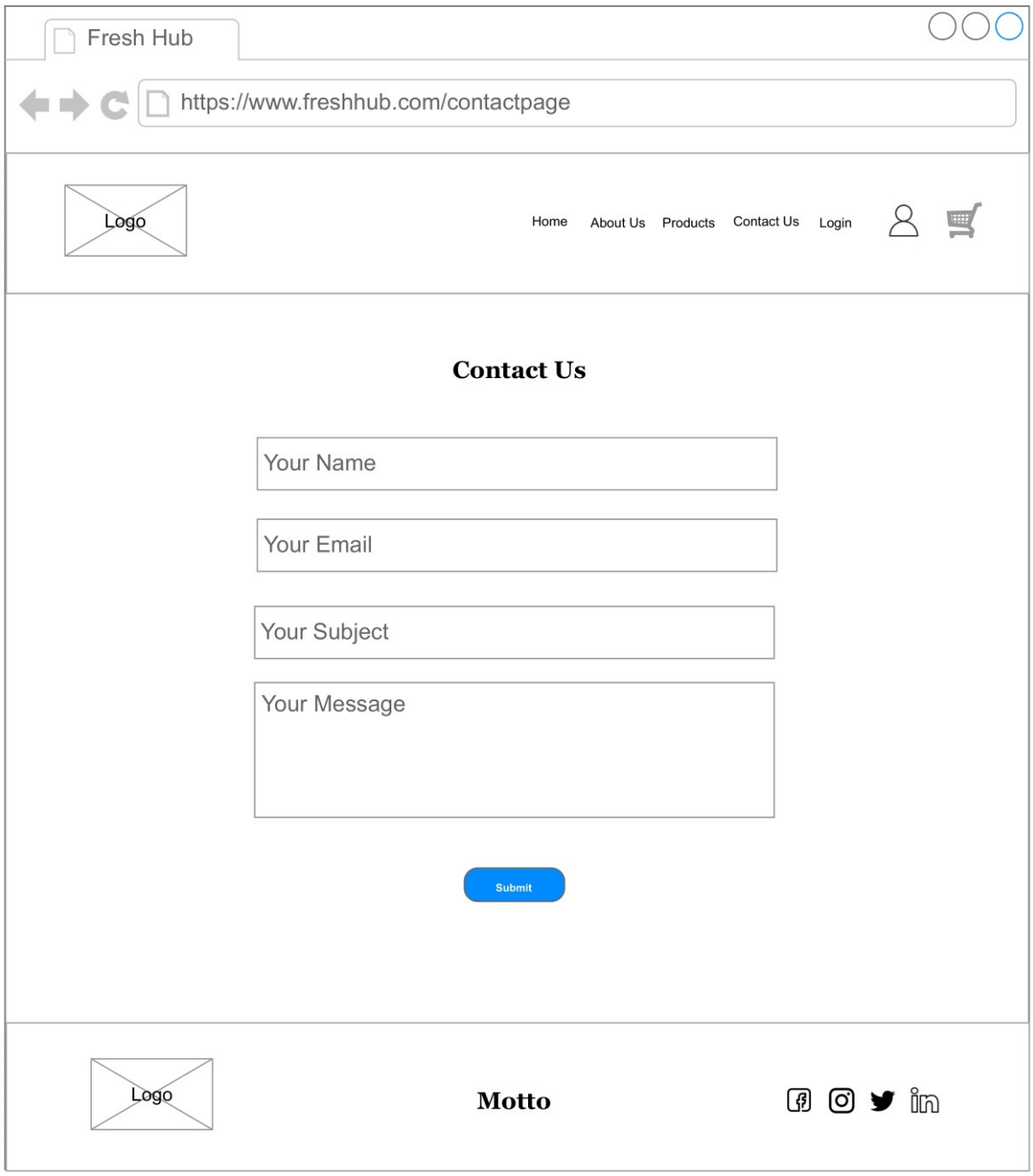


Figure 3.02 – *Contact Us Wireframe, draw.io*

The contact us page consist of a HTML form that has 4 input field. Its main purpose is to allow user to submit inquiries for enlightenment, recommendation for improvements, and complaints to alleviate their frustration.

3.1.3 View Inquiries

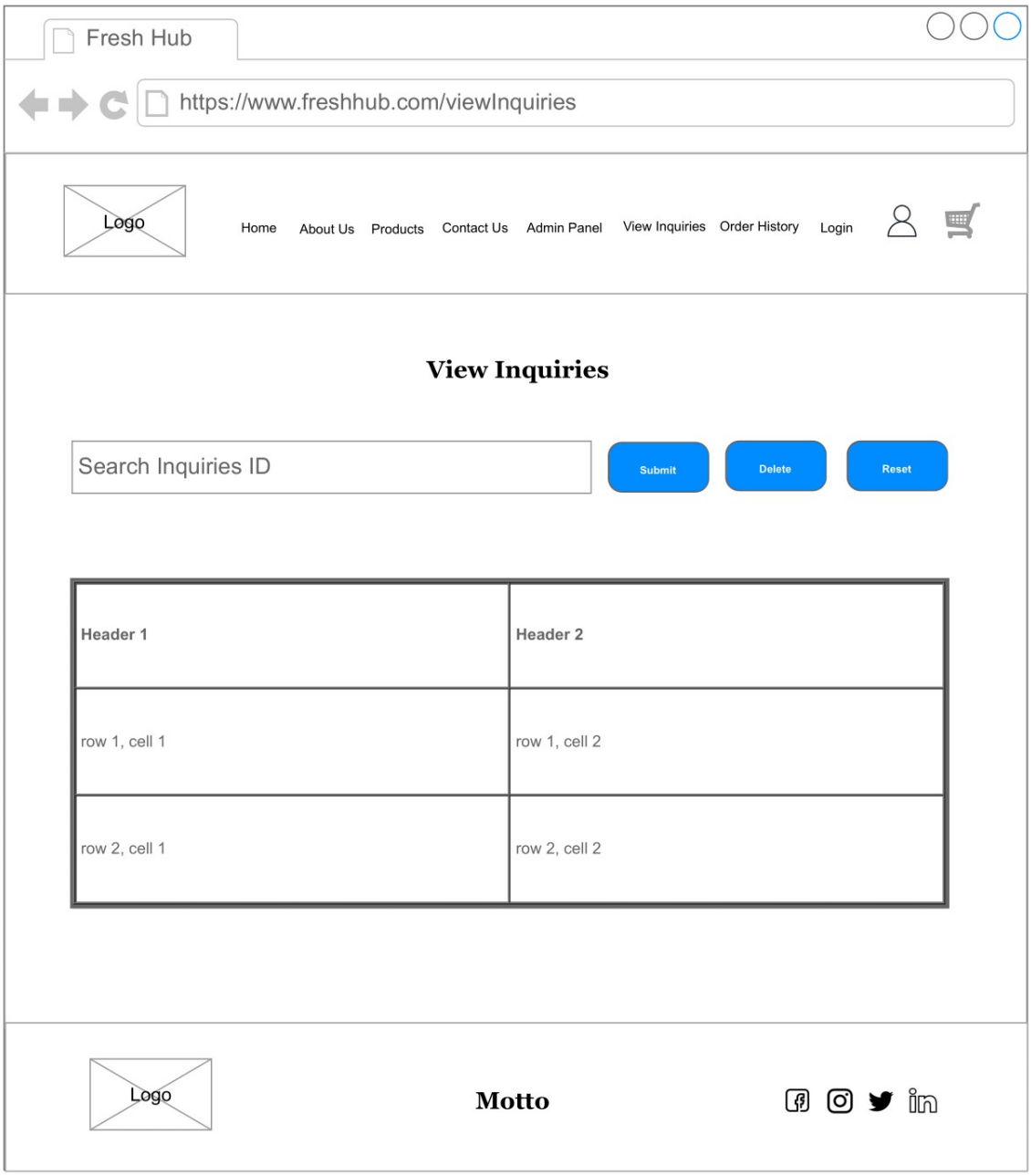


Figure 3.03 – *View Inquiries Wireframe, draw.io*

The inquiries page is an Admin exclusive page that features a search bar, a table, and 3 buttons titled ‘submit’, ‘delete’, and ‘reset’ respectively. The ‘submit’ button will **fetch the inquiries** from the database based on the **ID** that was inserted; the ‘delete’ button will delete the record, and ‘reset’ button will refresh the table for latest inquiries.

3.1.4 View Order History

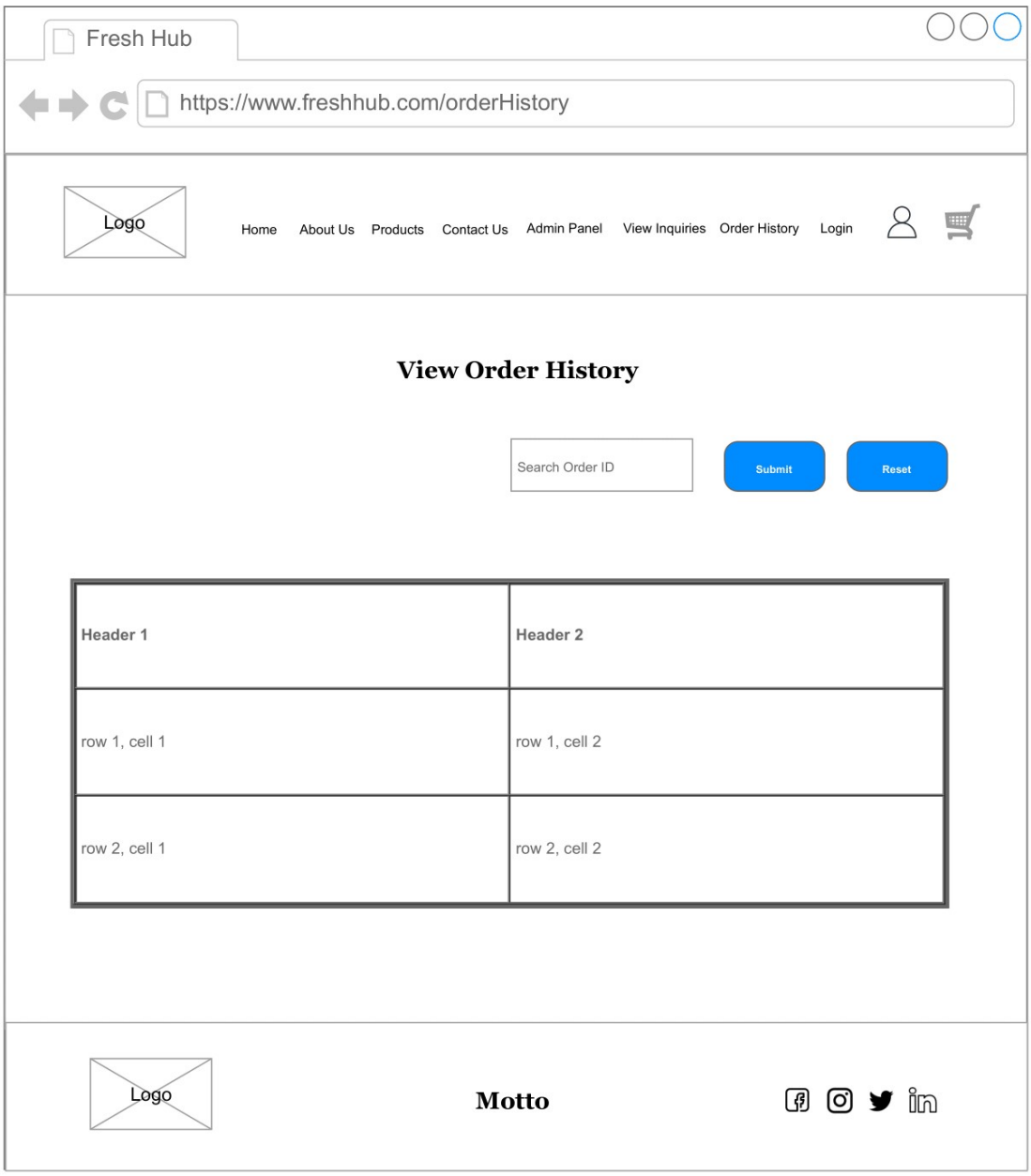


Figure 3.04 – *View Order History Wireframe, draw.io*

The order history page is accessible by all registered member regardless of roles. Its design features a simple search bar, a table, and two buttons. Customers or Admin may view all their order history, or search for a specific one by entering its Order ID.

3.1.5 Header Footer Detection

The headers are designed to change based on the roles of the user. There are a total of **3 designs**, one for Guest (site-visitors), Customer (registered-member), and Admin, each with unique features and functionality. Meanwhile, the footer is the same across all roles.

Figure 3.05 – *Guest Headers Wireframe, draw.io*

Figure 3.06 – *Customer Headers Wireframe, draw.io*

Figure 3.07 – *Admin Headers Wireframe, draw.io*

Figure 3.08 – *Footer Wireframe, draw.io*

3.1.6 Admin Panel

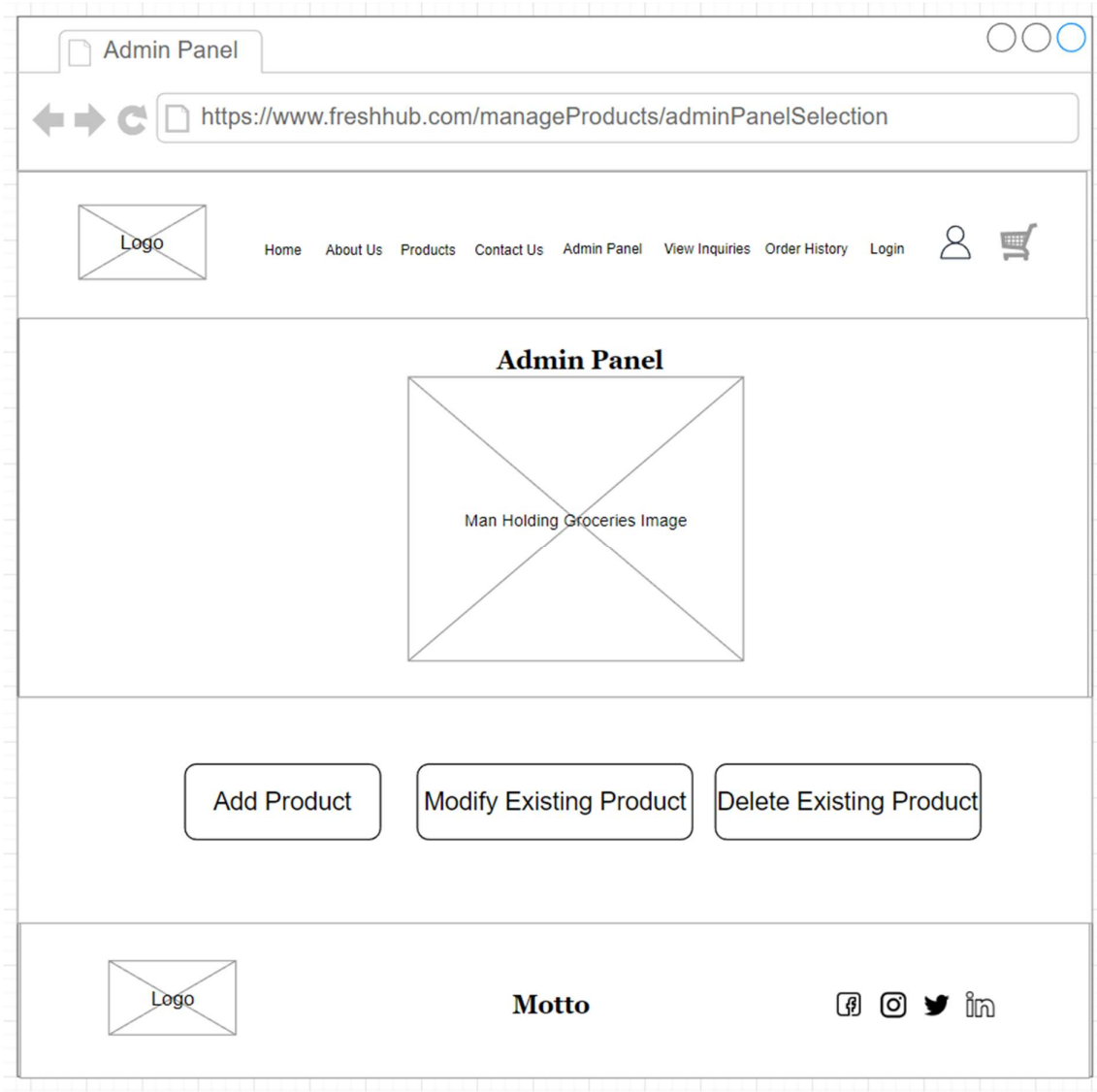


Figure 3.09 – *Admin Panel wireframe, draw.io*

The Admin Panel incorporates the Admin Header as shown previously in **Fig 3.07**, and the usual footer shown in **Fig 3.09**. All the admin's main functions are here, being able to access them by clicking the buttons. They include "Add Product", "Modify Existing Product", and "Delete Existing Product". Clicking on any of these buttons will bring the admin to the respective page.

3.1.7 Product Page

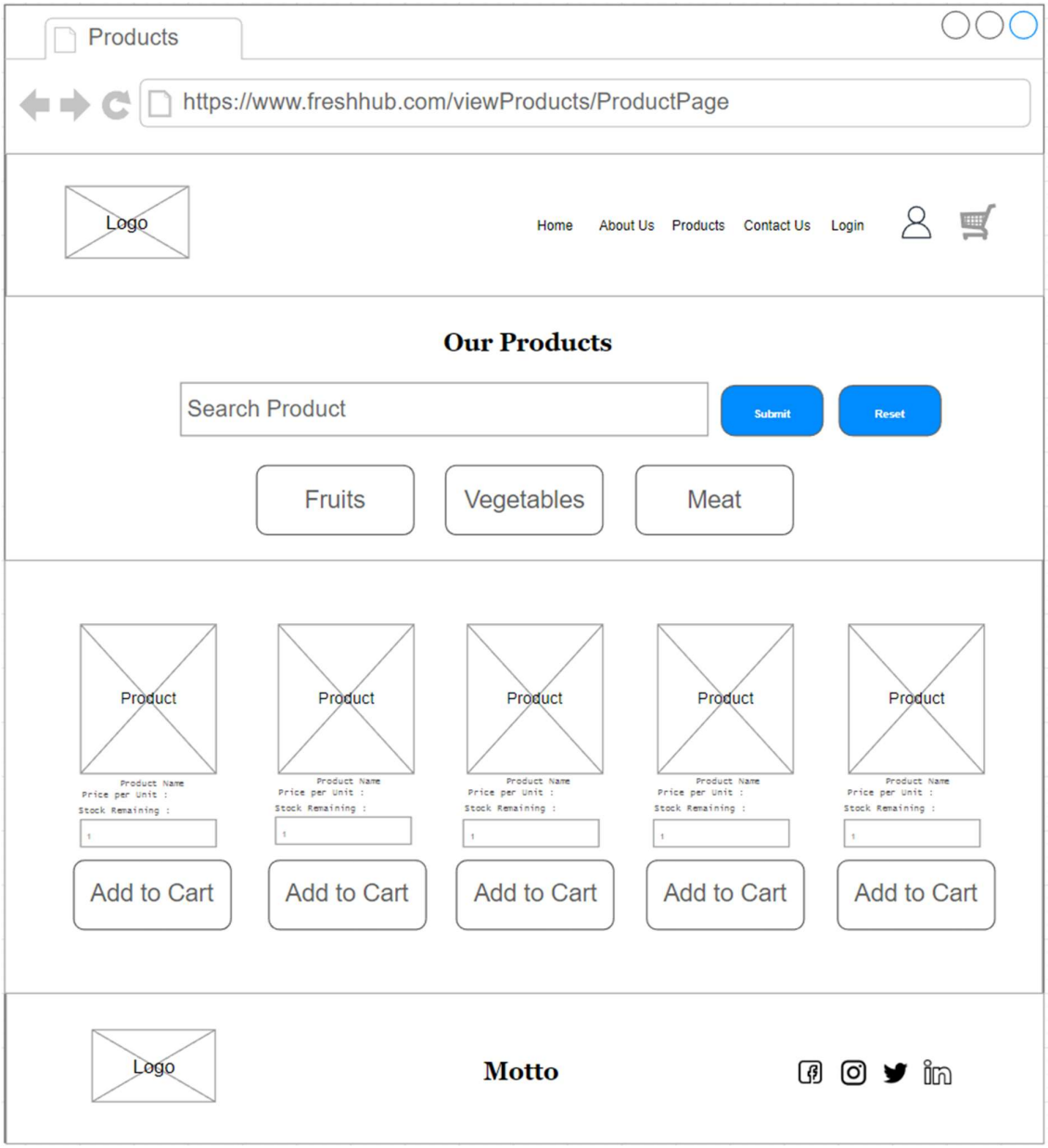


Figure 3.10 – *Products Page wireframe, draw.io*

The main products page displays all products that are currently being sold. Users can also choose to view by category by clicking on the available “Fruits”, “Vegetables” or “Meat” buttons. Clicking on any of these will bring them to a page consisting of products only within that category. Users can also view the Price per Unit and Stock Remaining of products that are being displayed. Users can also search for a specific product through the search bar above. Finally, there is an “Add to Cart” button for every individual item being displayed.

3.1.8 Product Page (Category Selected)

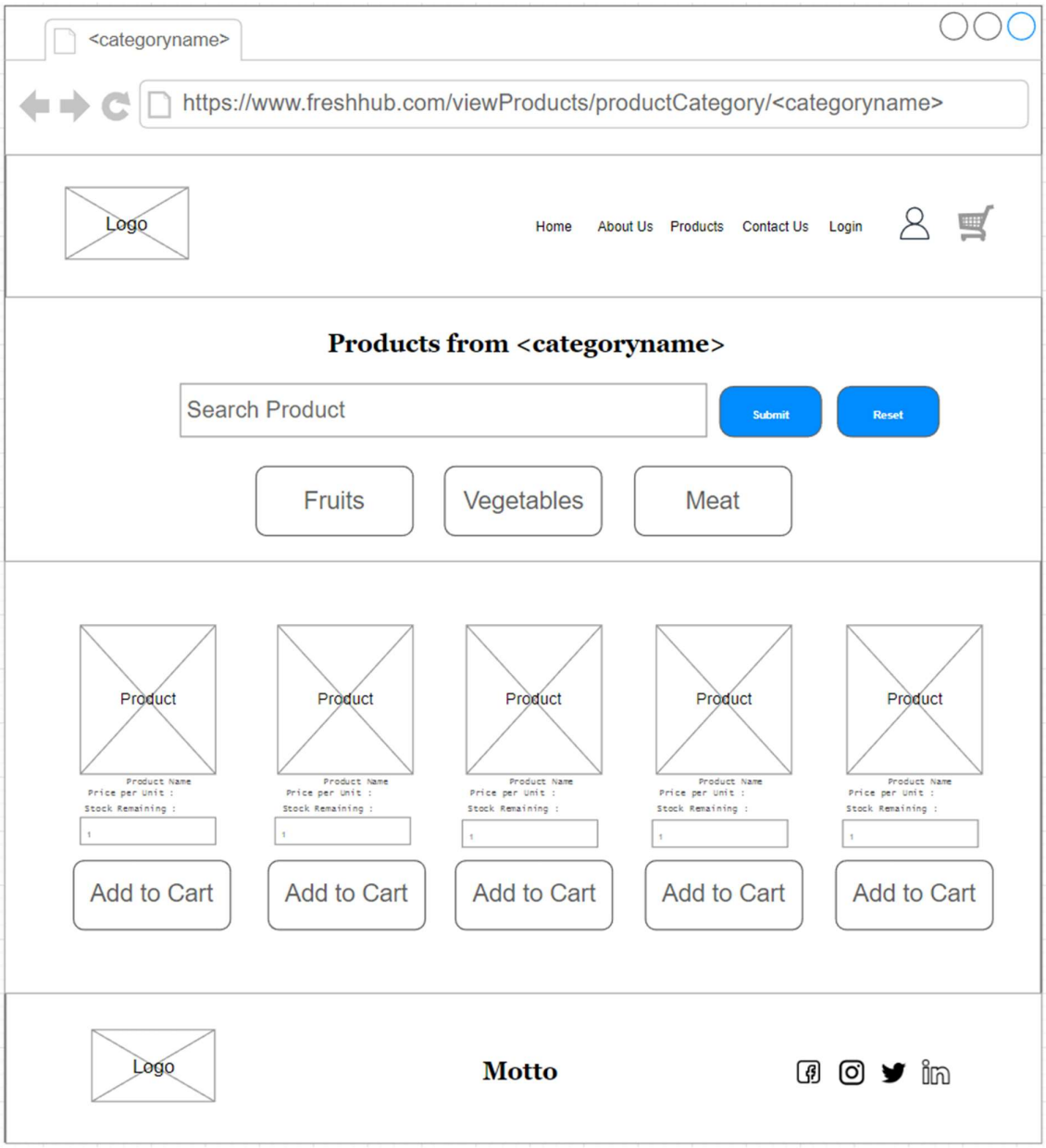


Figure 3.11 – *Products Page (Category Selected) wireframe, draw.io*

This page is highly similar to that of **3.1.7 Product Page**, incorporating all of the same main features. The main difference is in the Title of the Page (*tab on top of the browser*), where it displays the name of the category selected. The header of the page will also display "Products from", followed by the category name. *E.G. "Products from Vegetables"*

3.1.9 Login

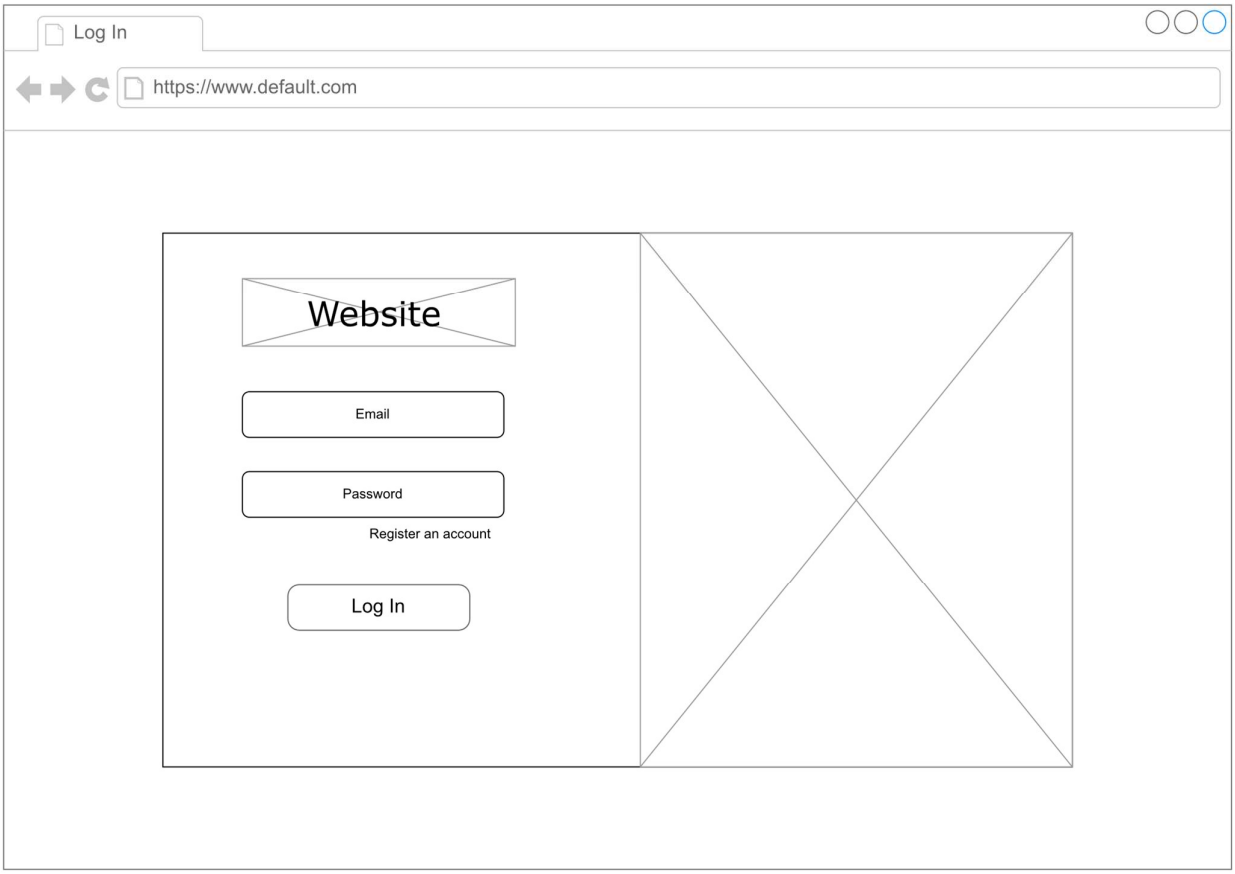


Figure 3.12 – *Login wireframe*

There are two input boxes that prompts the user for their email and password. A grocery image is displayed at the right side of the login form. Users can click on the “Register an account” to be redirected to the sign-up page.

3.1.10 Register

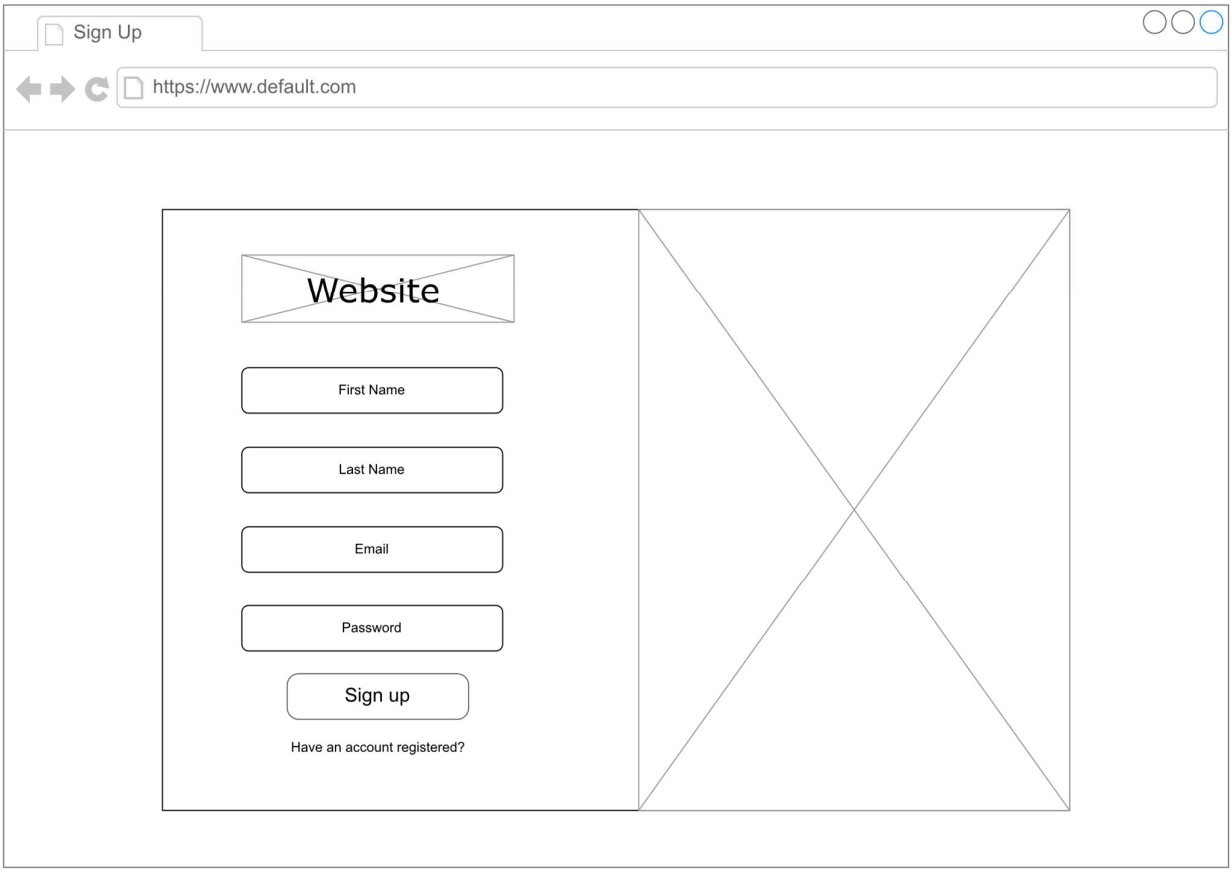


Figure 3.13 – *Sign Up wireframe*

The four required inputs are first name, last name, email and password. A grocery image is displayed at the right side of the login form. Users can click on the “Have an account registered?” to be redirected to the login page.

3.1.11 Forgot Password

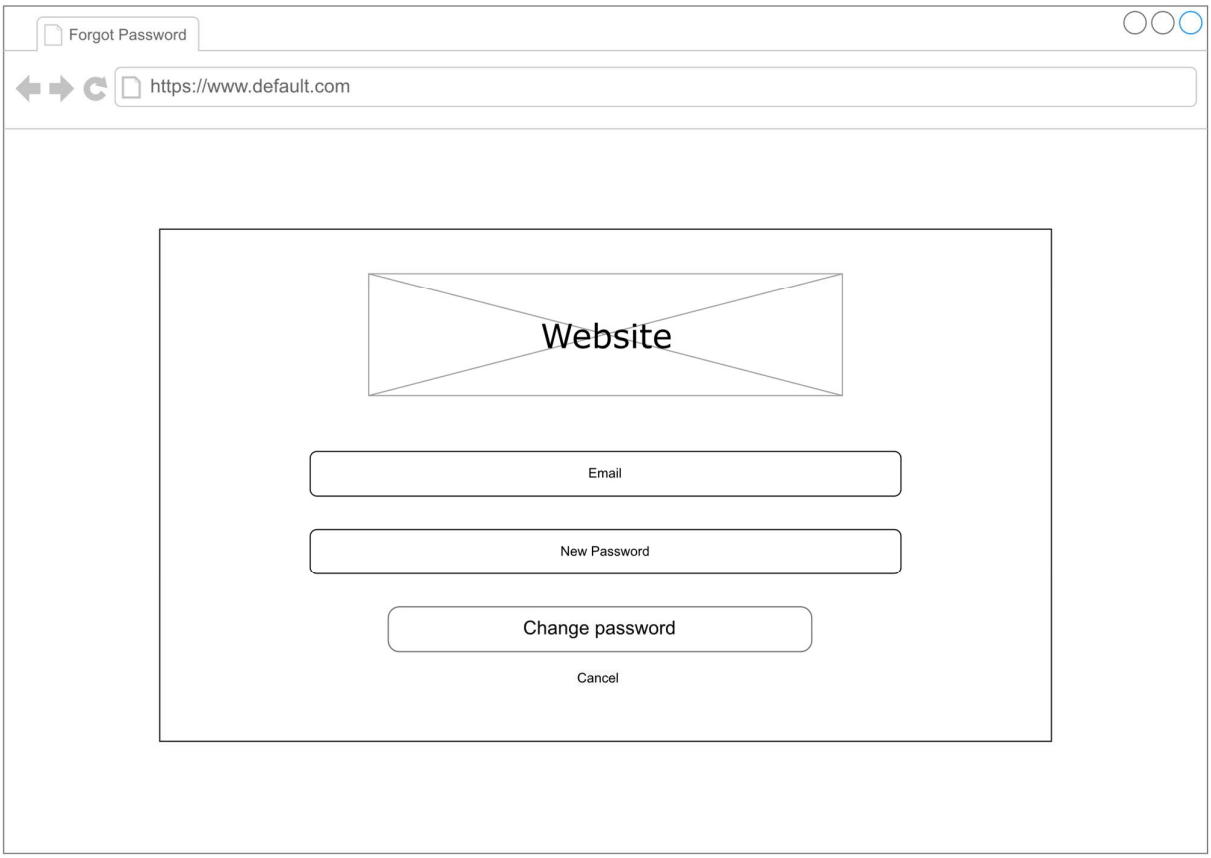


Figure 3.14 – *Forgot Password wireframe*

The forgot password page will have two required inputs which are email and user's new password. Users can click on "Cancel" to return to the login page.

3.1.12 Shopping Cart

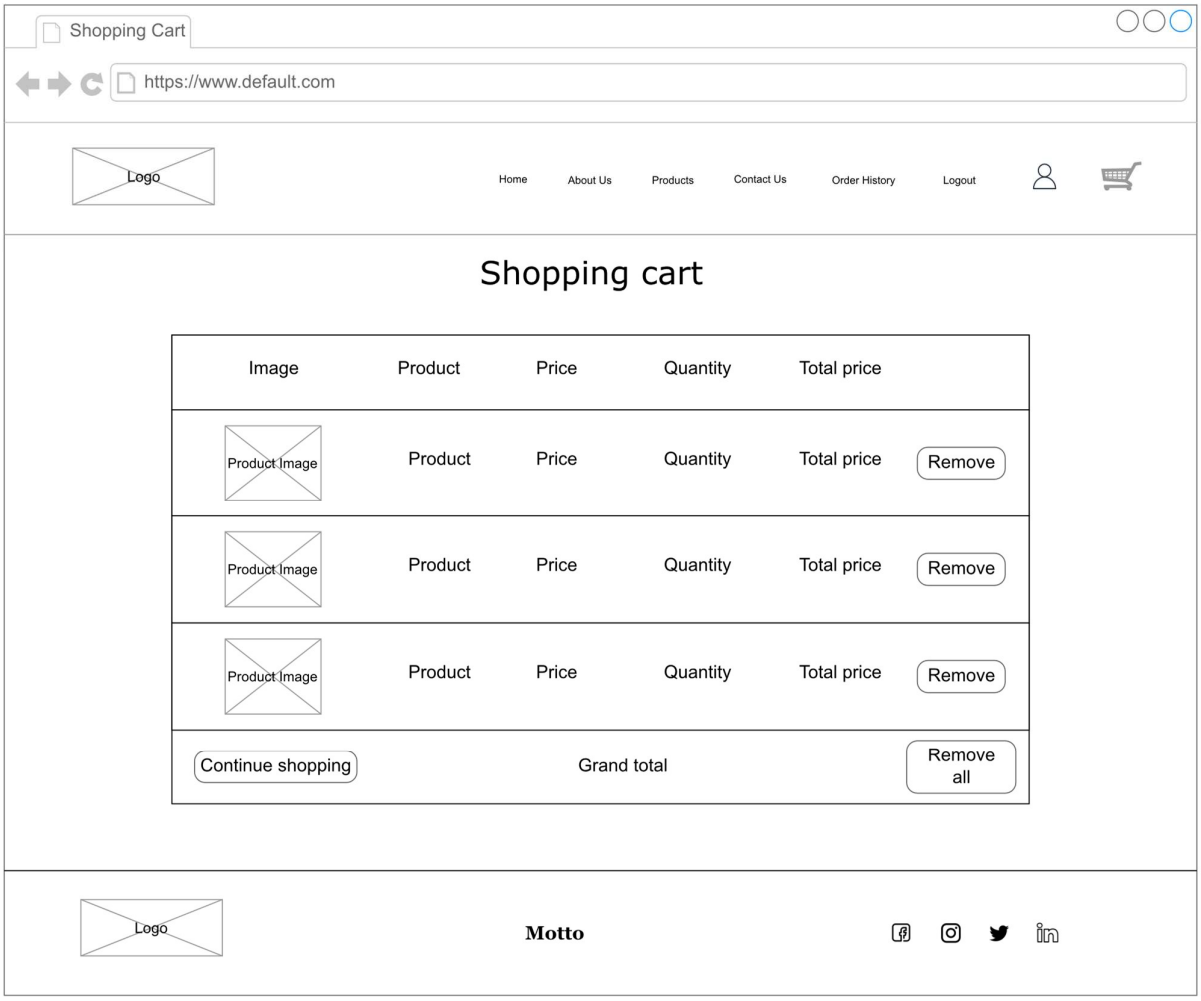


Figure 3.15 – Shopping Cart wireframe

Shopping cart will display a table summary of the items added to the user's cart. Every row will contain the product's image, name, price, quantity, total price and a remove button. At the final row of the table, users can click on the "Remove all" button to remove all products or click on the "Continue shopping" button to return to the products page.

3.1.13 Checkout

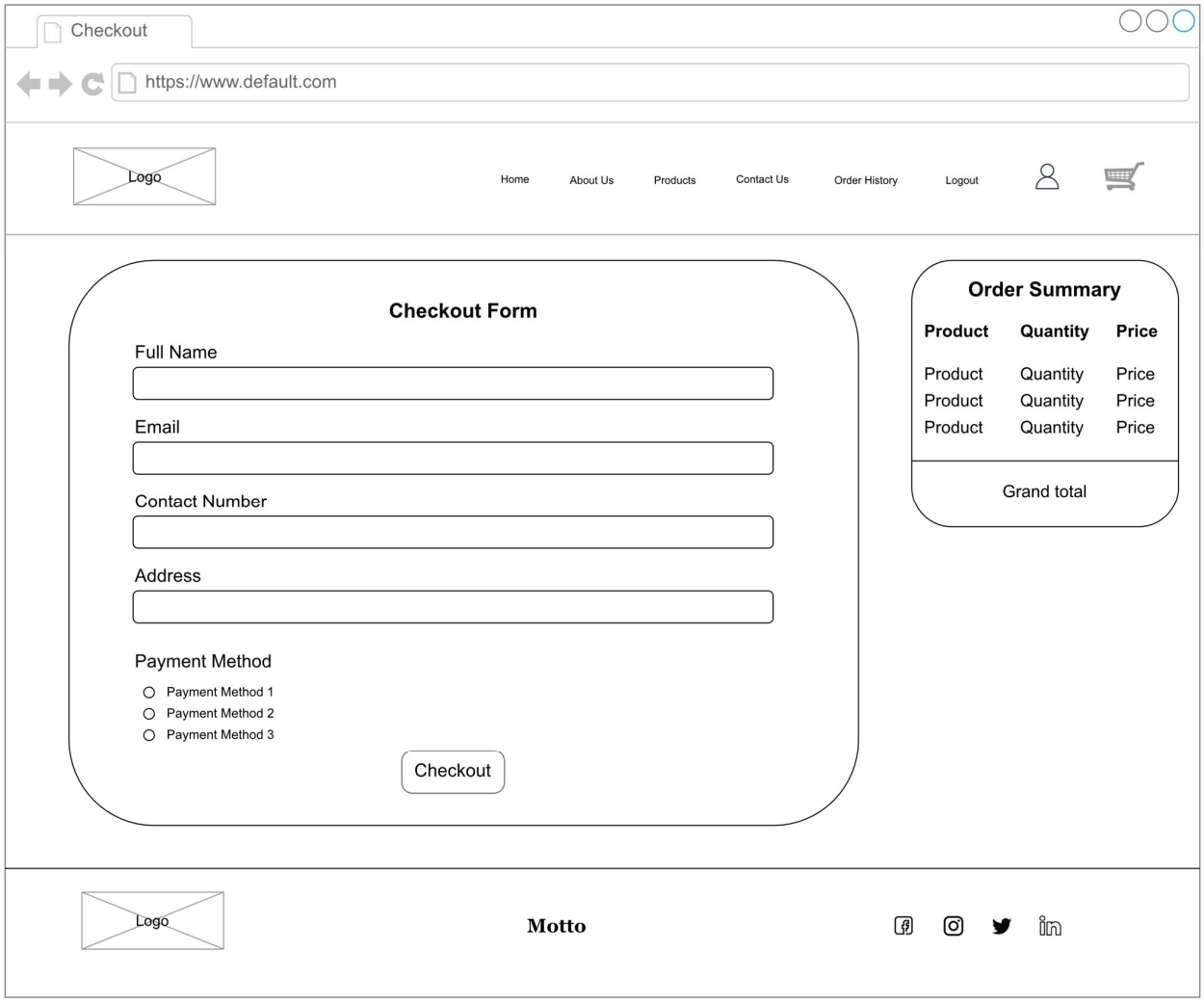


Figure 3.16 – *Checkout wireframe*

Before checking out, users must fill in their information and select their preferred payment method. A summary of the user's order is displayed at the right side of the screen to inform the users of their order before checking out.

3.1.14 Thank you

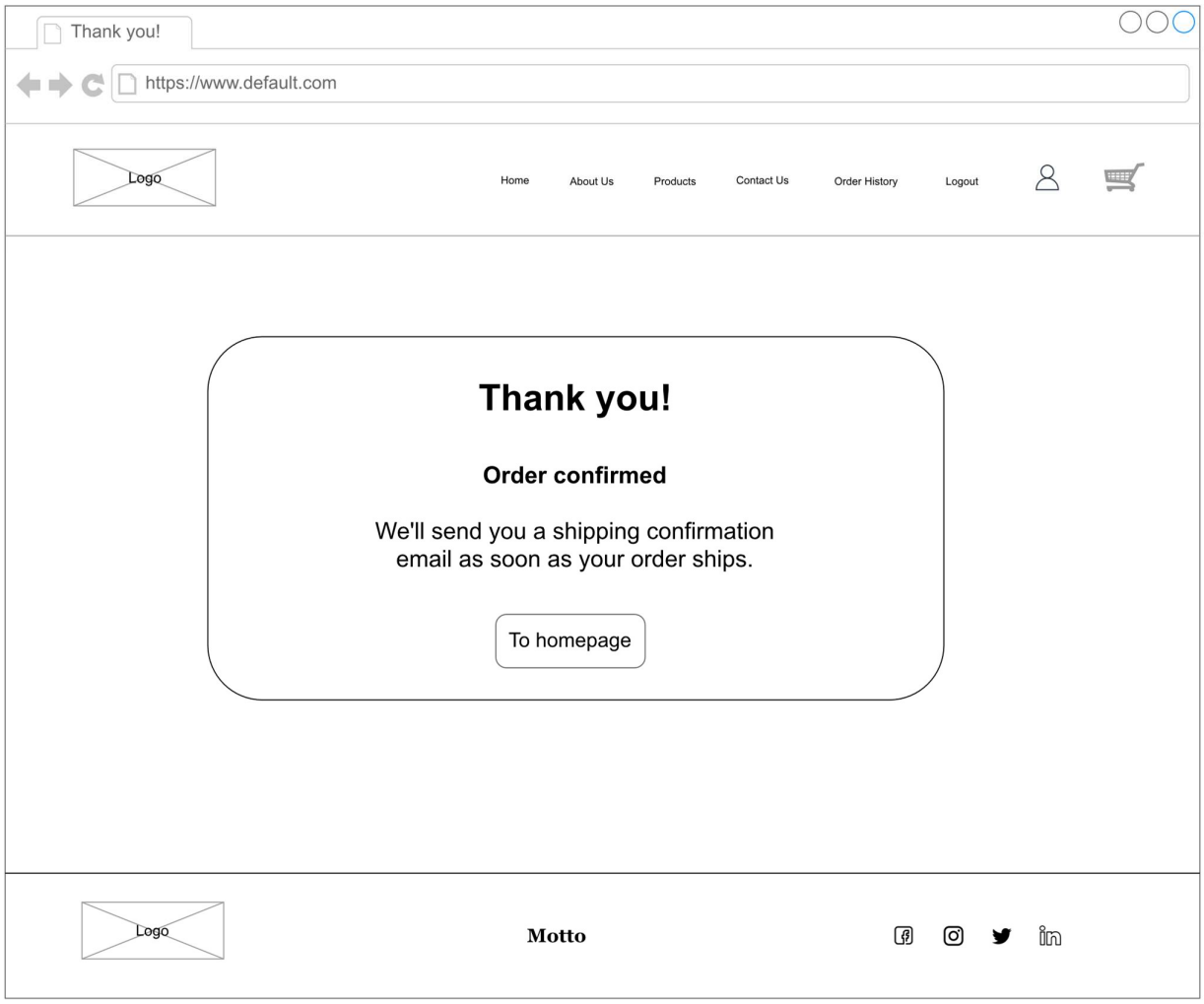


Figure 3.17 *–Thank You wireframe*

Users will be redirected to the thank you page after checking out. A notice will be displayed to inform the user that a confirmation email will be sent to them. Users can optionally click on the “To homepage” button to return to the homepage.

3.1.15 Edit Profile

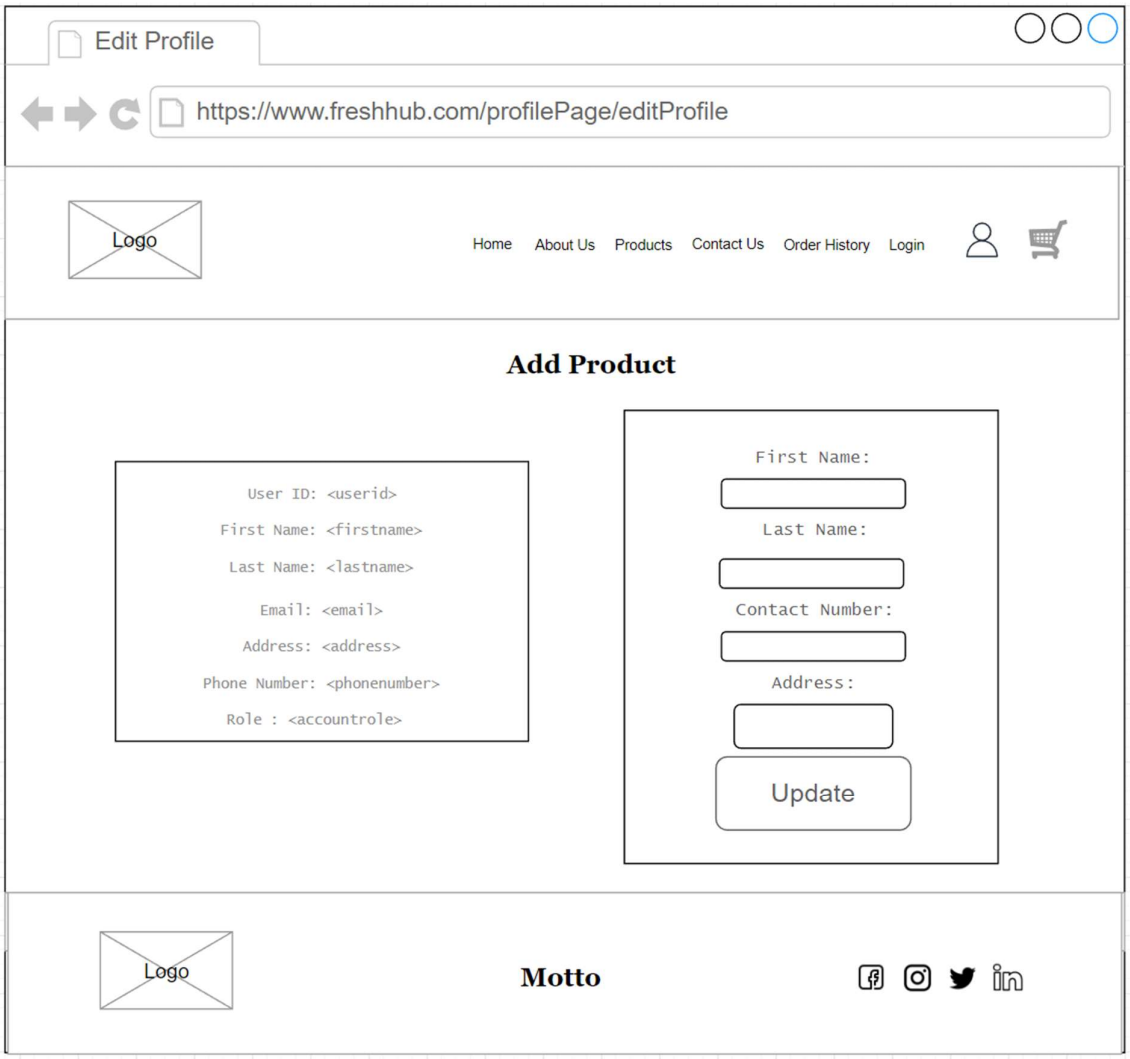


Figure 3.18 – *Edit Profile Wireframe, draw.io*

The edit profile page displays the user's current account information on the left-hand side. On the right, users can edit their First Name, Last Name, Contact Number, and Address. The Update Button can be clicked and will update the system with user inputted information.

3.1.16 Add Product

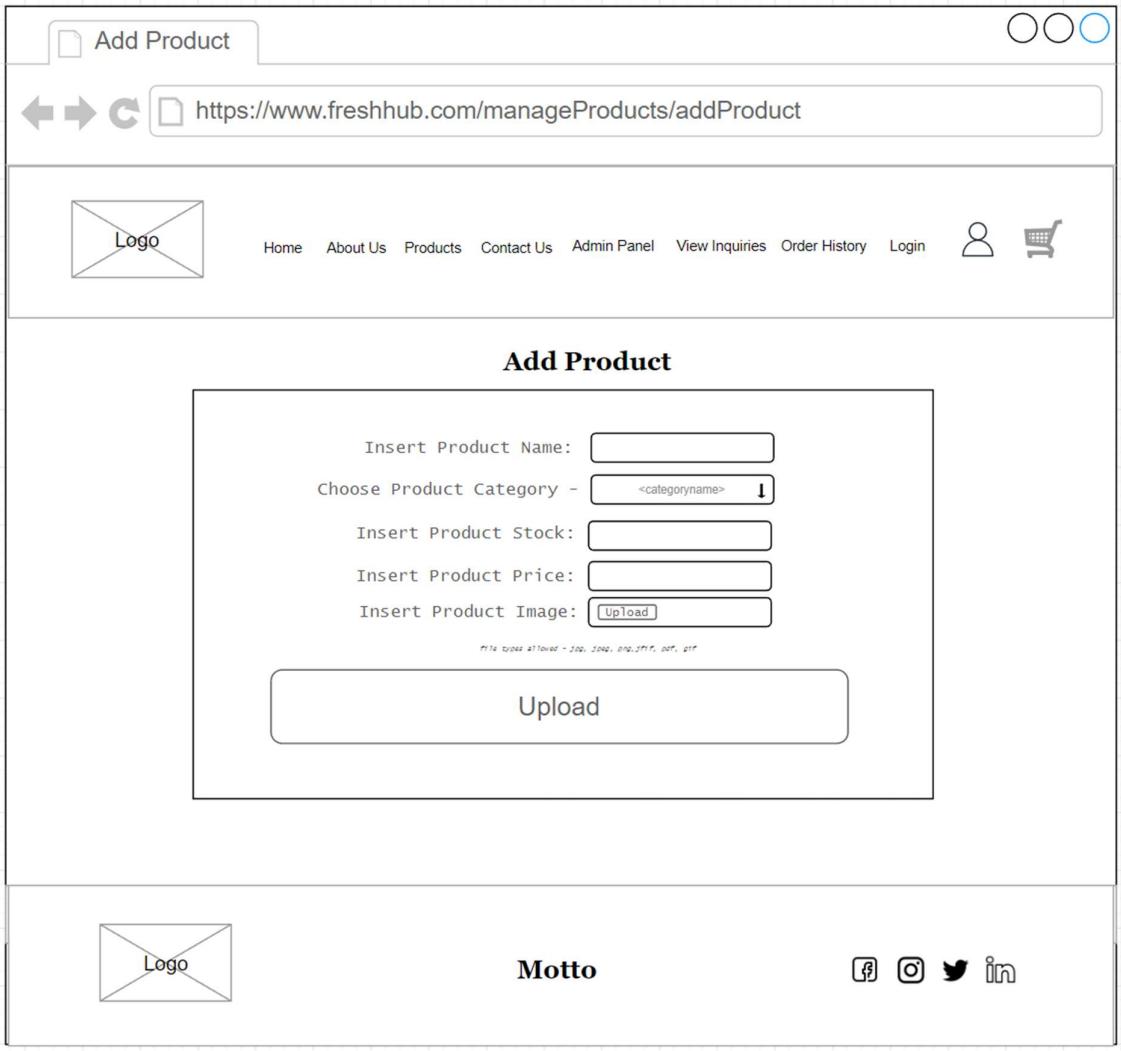


Figure 3.19 – *Add Product Wireframe, draw.io*

The add product wireframe displays a box where product information can be inputted by an admin. The fields include a Product Name, Product Stock and Product Price. For Categories, it is a drop-down box where admins can select between “Fruits”, “Vegetables”, and “Meat”. For uploading images, there is an Upload field. For an admin to submit the product information, they can click on the Upload button at the bottom of the form.

3.1.17 Modify Product (Main Page)

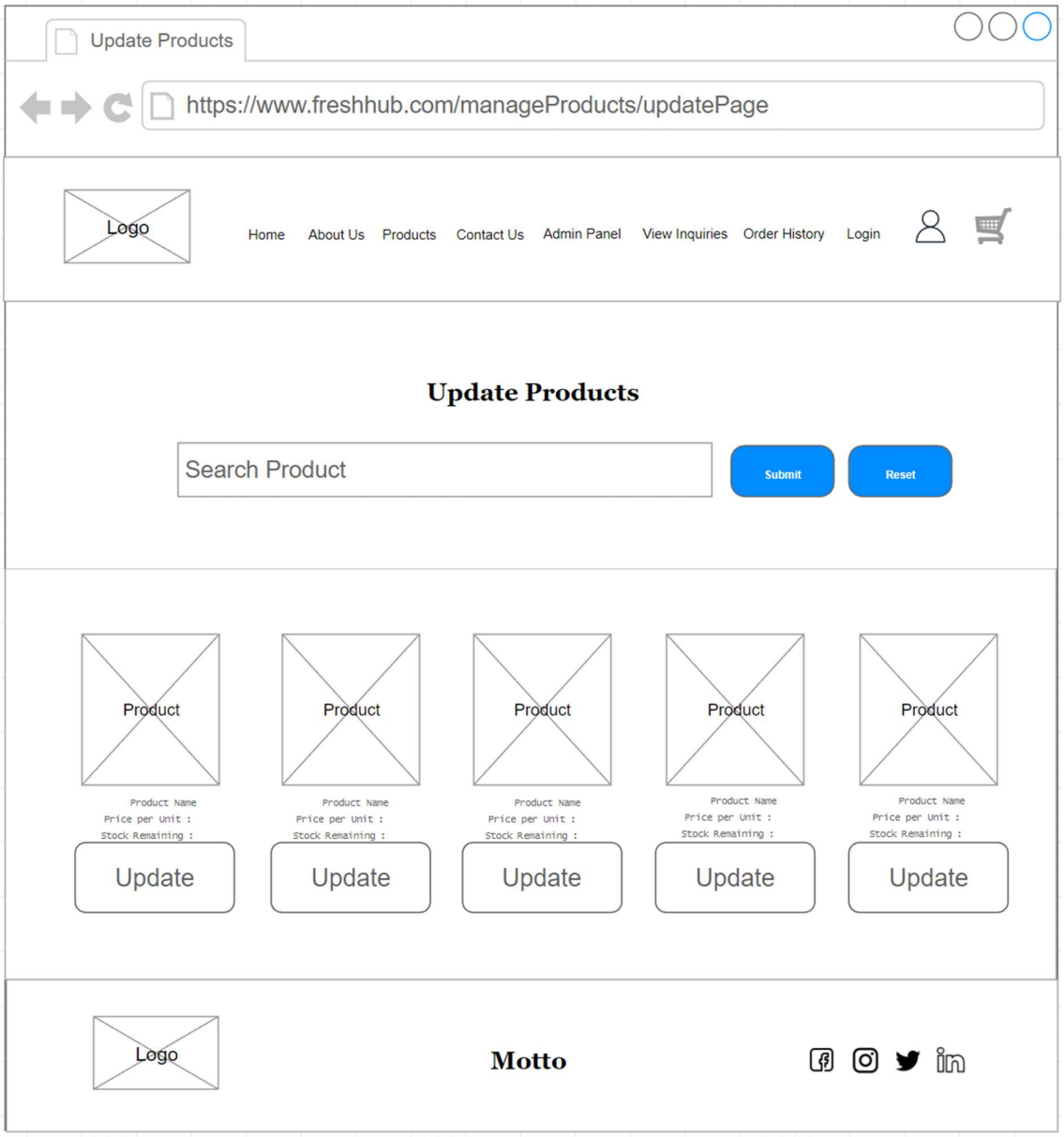


Figure 3.20 – *Modify Product Main Page Wireframe, draw.io*

The structure of Modify Products' main page is very similar to **3.1.7 Product Page.** Incorporating the search bar for easy finding of a product. Price and Stock of a Product are still displayed for the Admin to view. An admin can update a product by clicking on the update button, bringing them to the product's specific update page.

3.1.18 Modify Product (Update Product Page)

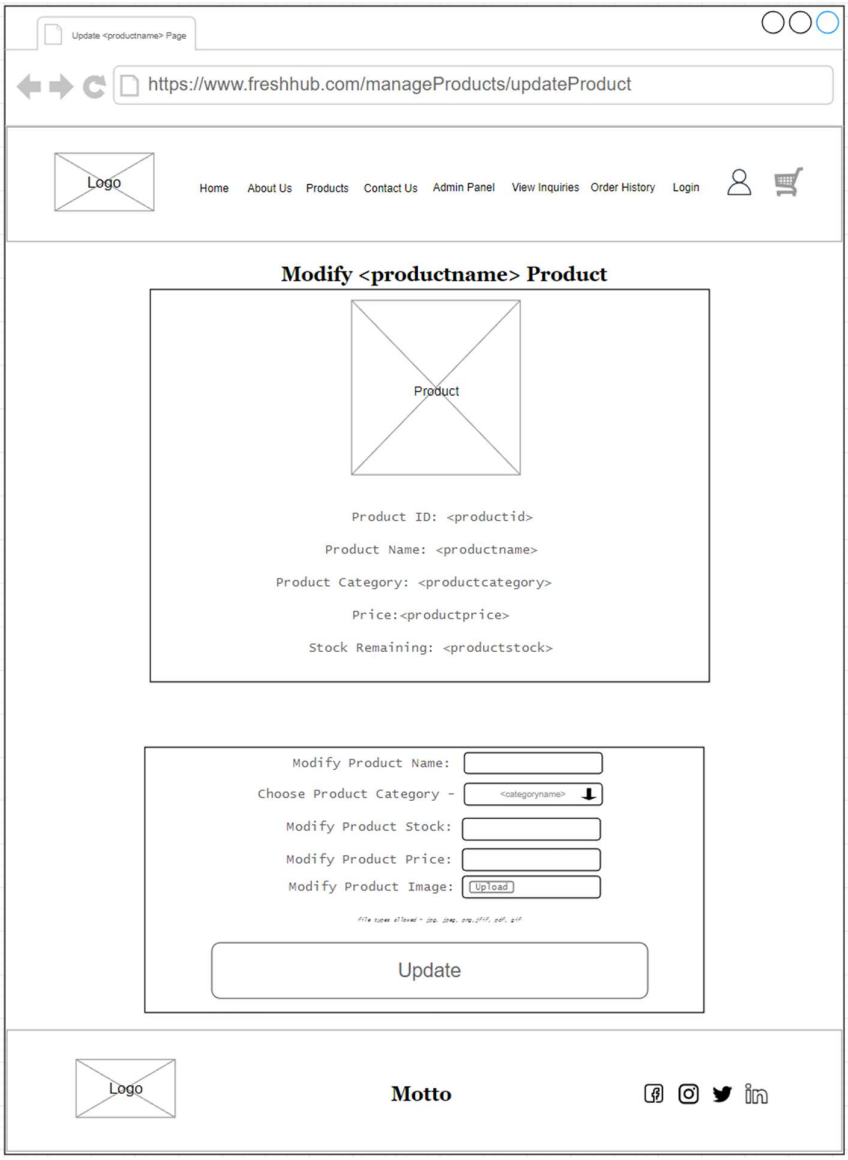


Figure 3.21 – *Modify Product Update Product Page Wireframe, draw.io*

This page shows detailed information of the product requested to be updated. Including information of the current Product ID, Name, Category, Price, and Stock Remaining. Similarly to Adding a Product, Admin's can change the Product Name, Category, Stock, Price, and Image through the given fields. Finally, an Update button is available at the bottom once an Admin is satisfied with the inputted details, ready for submission to the system.

3.1.19 Delete Product

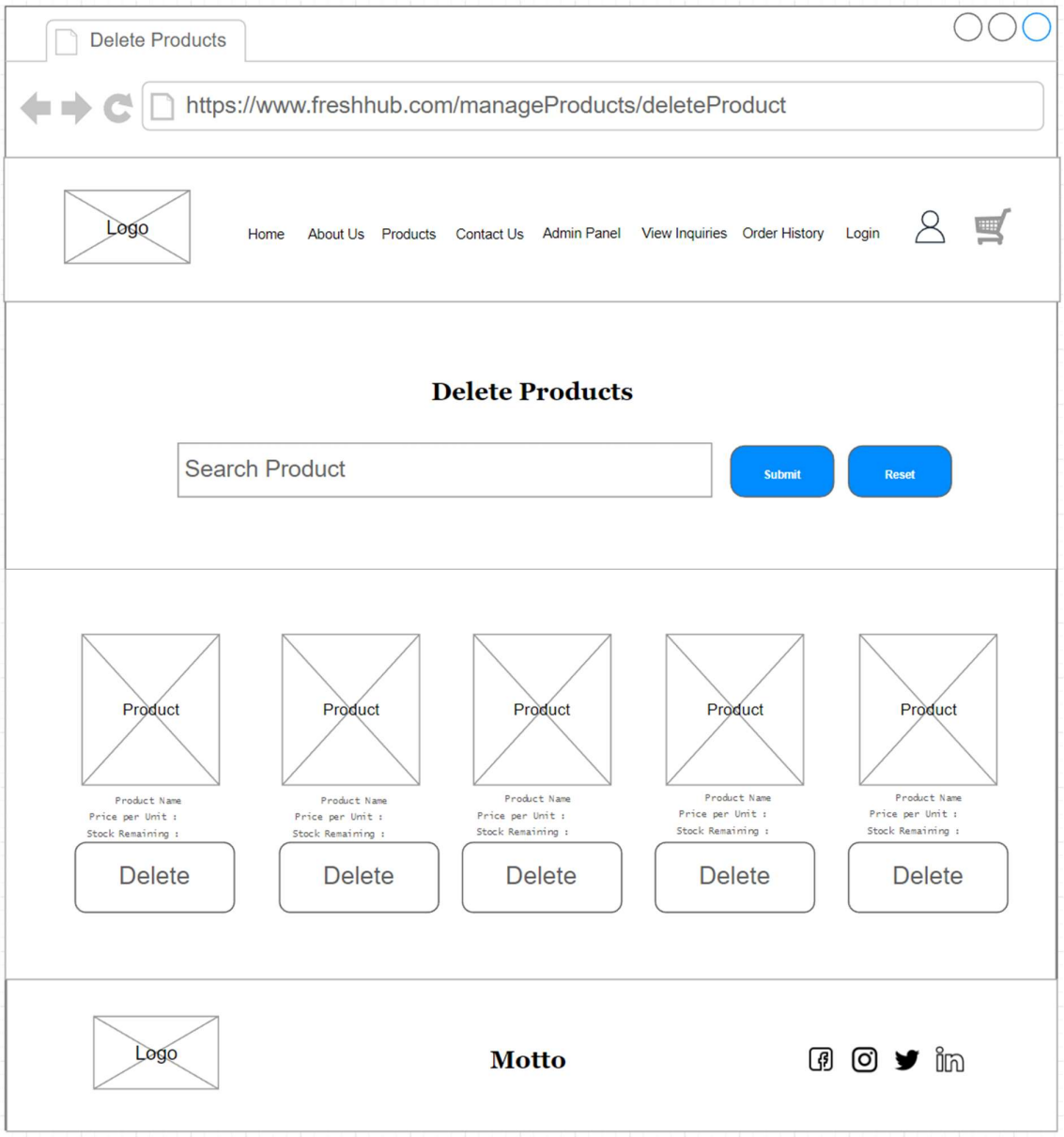


Figure 3.22 – *Delete Product Wireframe, draw.io*

The structure of this page's wireframe is similar to that of **3.1.17 Modify Product,** displaying the product's name, price per unit, and stock remaining. The admin can delete the product by simply clicking on the delete button.

3.2 Navigational structure

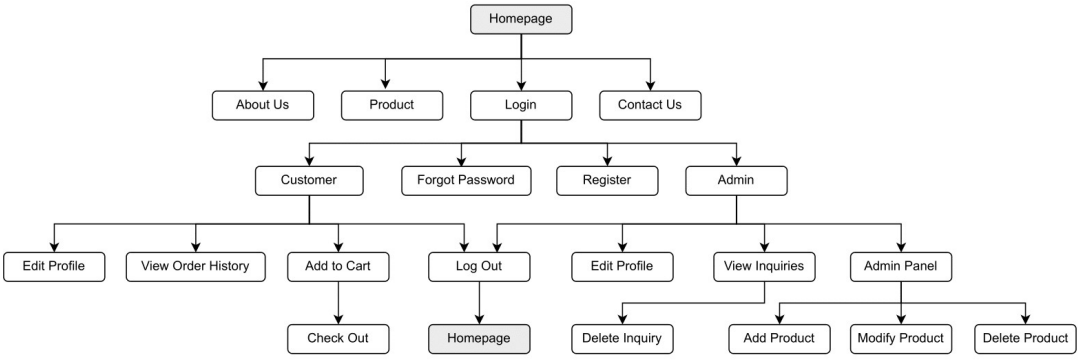


Figure 3.23 – *Navigational Structure, draw.io*

3.3 Entity Relationship Diagram

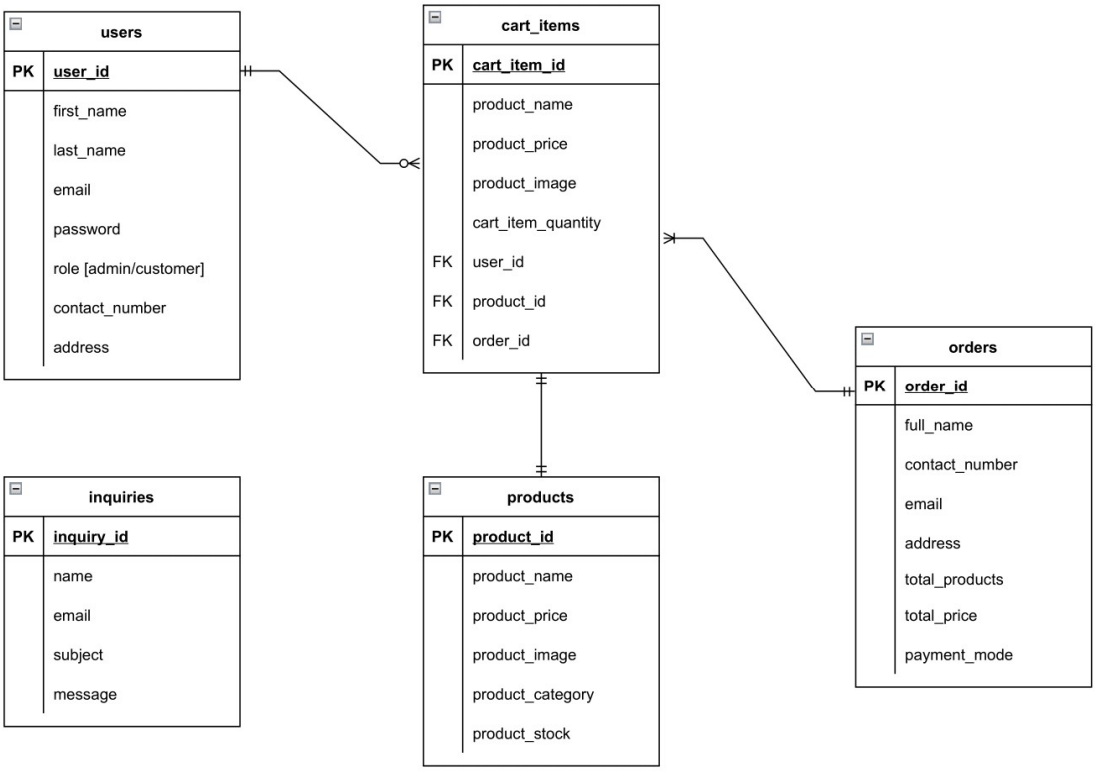


Figure 3.24 – *Entity Relationship Diagram, draw.io*

3.4 Activity Diagram

3.4.01 Login

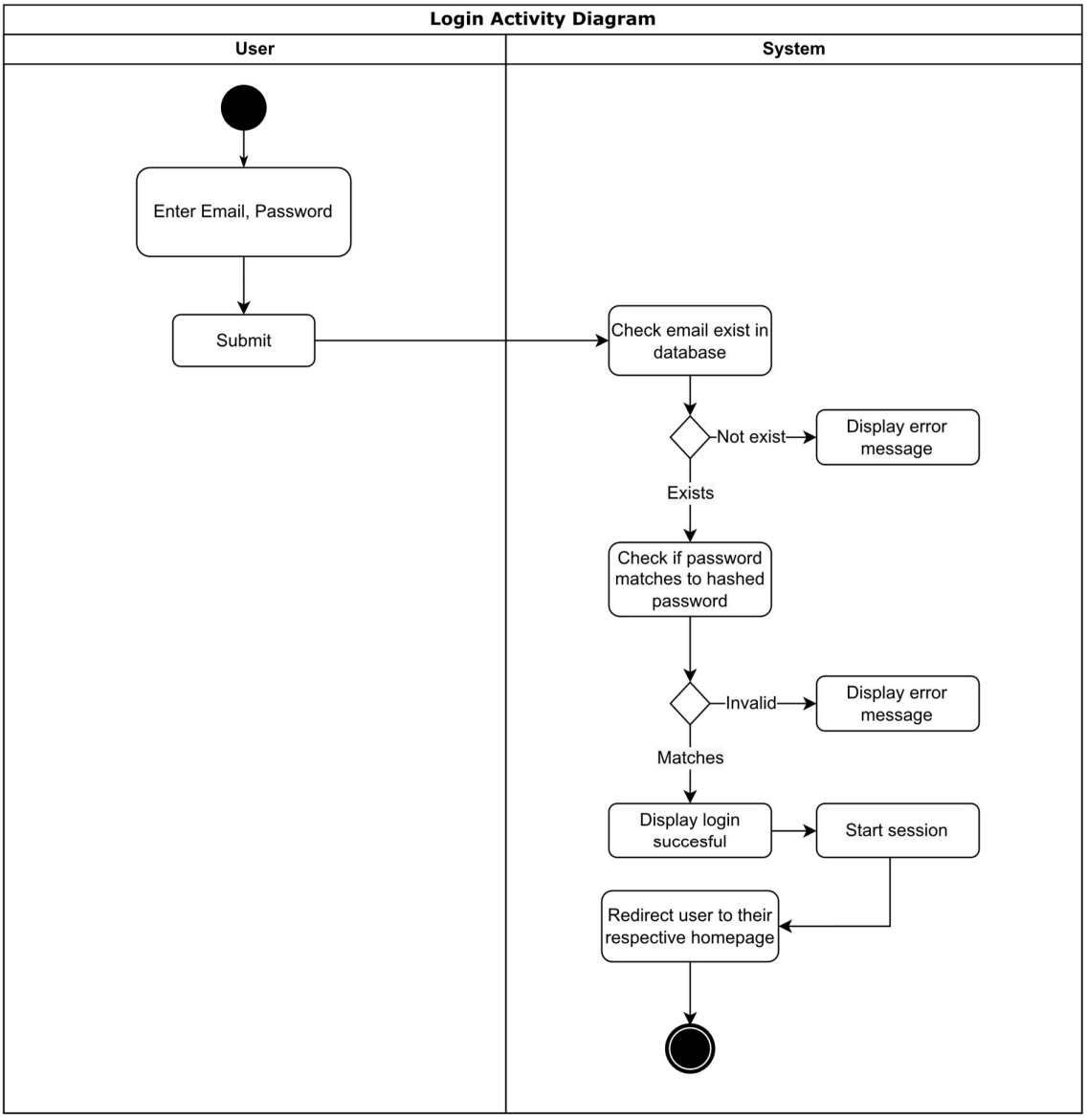


Figure 3.25 – *Login activity diagram*

Users are required to enter their registered email and password to log in. Upon submitting the form, the system will check if the email exists in the database. The password is then verified to see if it matches what is stored in the system. Then the system will display login successful and redirect the user to the homepage of FreshHub.

3.4.02 Register

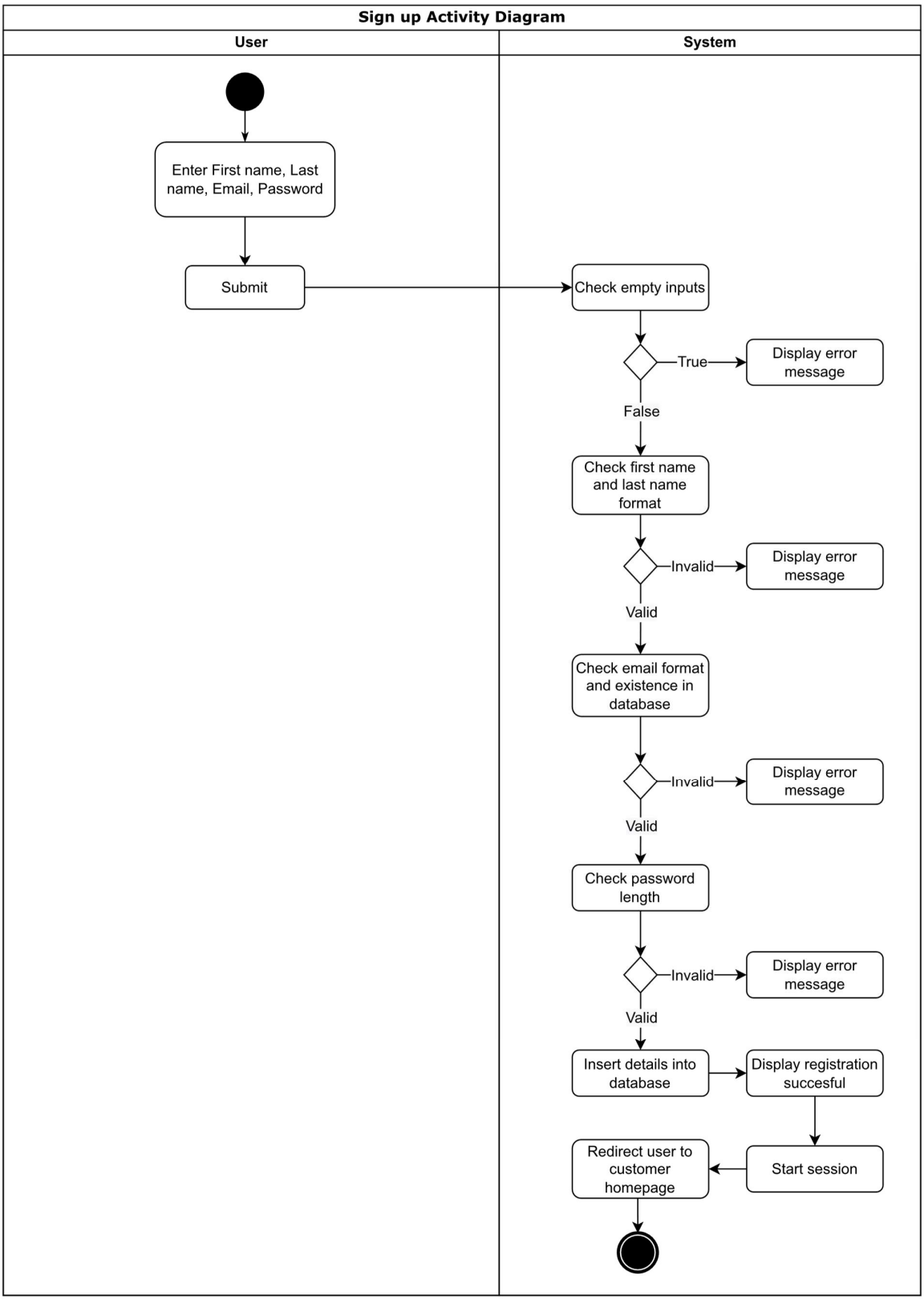


Figure 3.26 – *Sign Up activity diagram*

Users are required to enter their first name, last name, email, and password to sign up for an account. The system performs multiple validations on the inputs before inserting the data into the database. First name and last name is checked if it only has characters in the string. Then, the email format is checked and to verify if it already exists in the database. Then, the password length is checked to determine if it contains more than 8 characters. If the inputs have succeeded all validations, the system will display a registration successful message and redirect the user to homepage.

3.4.03 Forgot Password

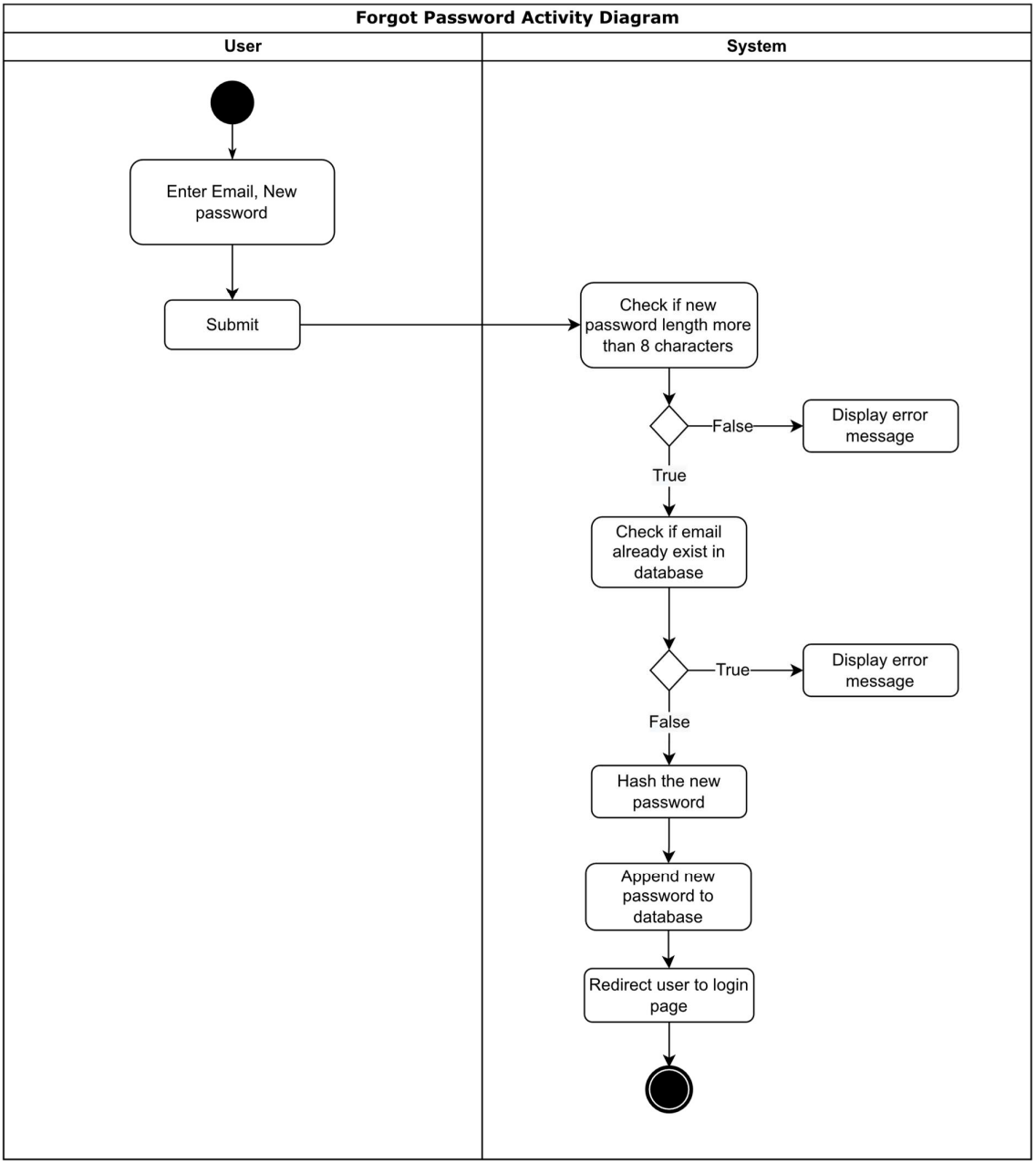


Figure 3.27 – *Forgot Password activity diagram*

Users are required to enter their email and new password to replace their existing password that is stored in the database. Validations are made to determine if the email exists in the database and the new password is more than 8 characters. The new password is hashed and updated in the database.

3.4.04 Remove cart item

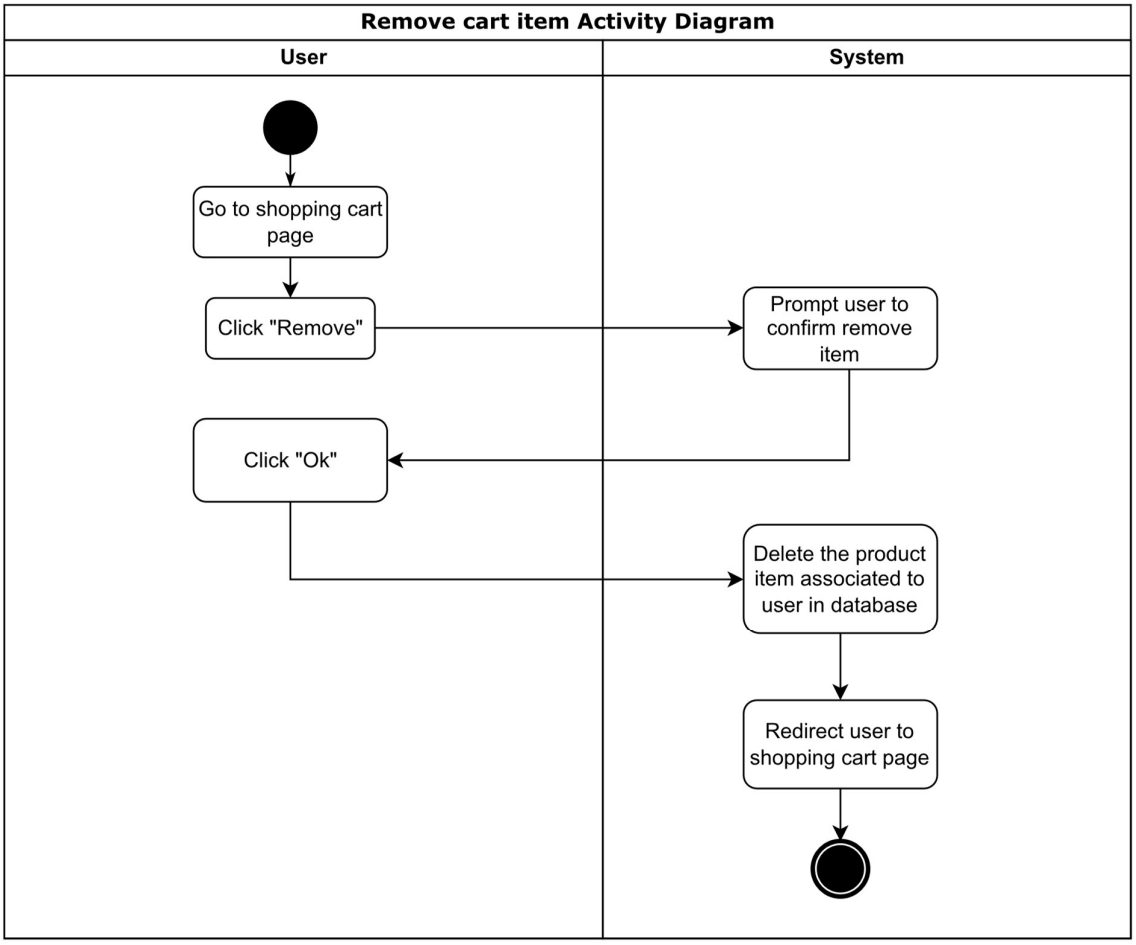


Figure 3.28 – *Remove cart item activity diagram*

To remove a cart item, users are required to enter the shopping cart page. Clicking on the “Remove” button will remove the item from the user’s cart and delete the related row from the database.

3.4.05 Checkout

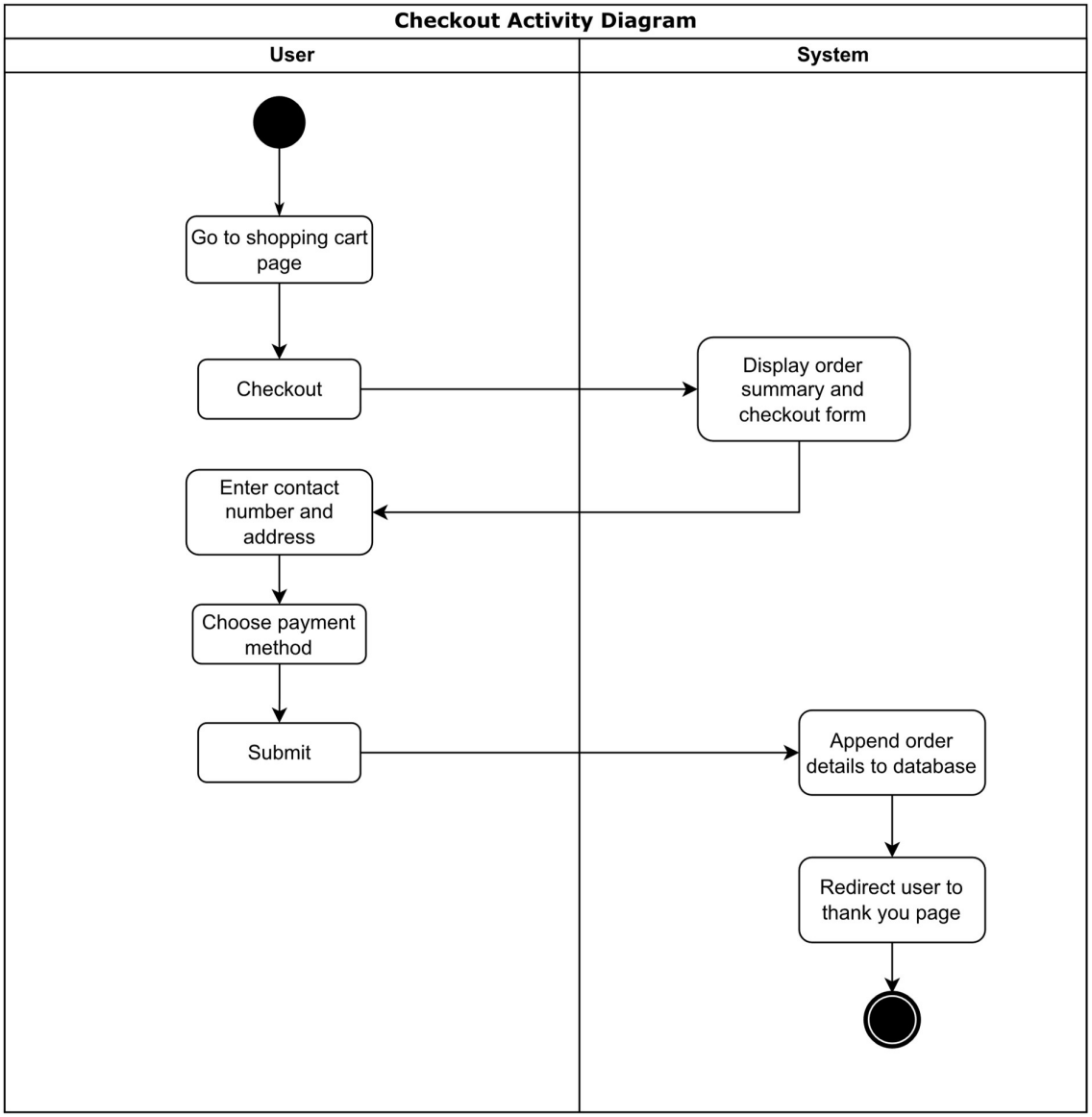


Figure 3.29 – *Checkout activity diagram*

Users can begin the checkout process by clicking on the “Proceed to checkout” button in the shopping cart page. Then, users are required to enter their contact number, address, and choose a payment method. Upon submitting the form, the order will be appended into the database and user will be redirected to the thank you page.

3.4.06 Admin Panel & Admin Functions (Ian)

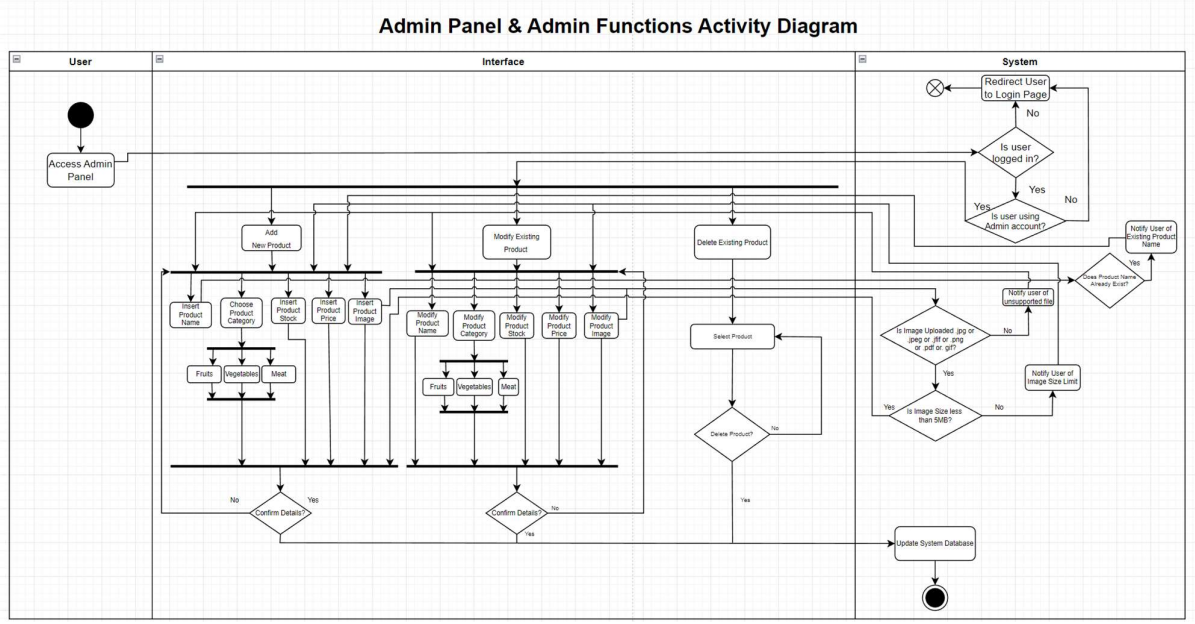


Figure 3.30 – *Admin Panel & Admin Functions Activity Diagram, draw.io*

In this diagram, it is focused on the Admin Panel Page, as well as the functions available**.** First, the system checks if the user is currently logged in, if they are not, the system redirects them to the login page. If they are logged in, the system then checks if the user is using an Admin account. If they are not, they are prohibited access and redirected to the login page.

After the account checking process, the Admin can choose between **Add New Product, Modify Existing Product,** and **Delete Existing Product.** If Add New Product is chosen, the Admin Insert the Product Name, Choose the Product Category, Insert Product Stock, Product Price, and Upload an image for the product. If the Product Name already exists, the system will reject the upload request and notify the admin to re-enter all product details. Following up, if the Admin uploads an image not supported by the system, the system also rejects the request. Another point of validation is the image size, if the image is over 5 Megabytes, the request to add a product too is rejected. Once all of these details are confirmed, the system accepts the request and updates the system's database. All processes and validations when Adding a Product are the same when Modifying a Product.

When deleting a product, the admin clicks on the delete button and are prompted if they want to delete the product. If so, the system accepts the request and removes the product from the database. Otherwise, the product stays and the Admin stays on the page.

3.4.07 Products Page (Ian)

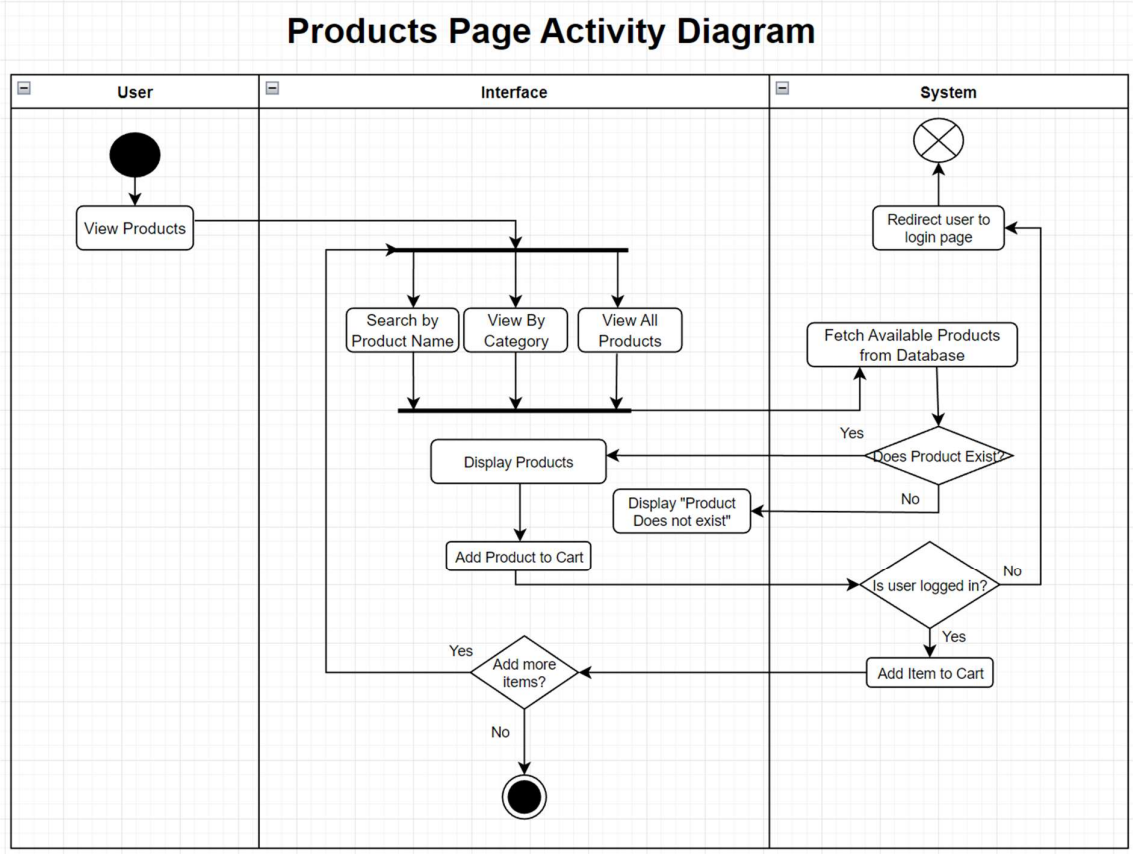


Figure 3.31 – *Products Page Activity Diagram, draw.io*

When users view the product page, the user can choose options between **viewing all products**, viewing products by **category**, or **searching** for a product by its name. Once a user selects any of the three options, the system fetches available products based on the user's request, and displays the products appropriately. If no product matches the user's query, the system displays a message "Product does not exist".

After this, a user may choose to add a product to the cart. The system then first checks if the user is logged in. If they are not, they will be re-directed to the login page, as it is a requirement for users to log in before purchasing anything. If they are logged in, the product is then added into their cart. Users can then choose to further add other products into their cart.

3.4.08 Edit Profile (Ian)

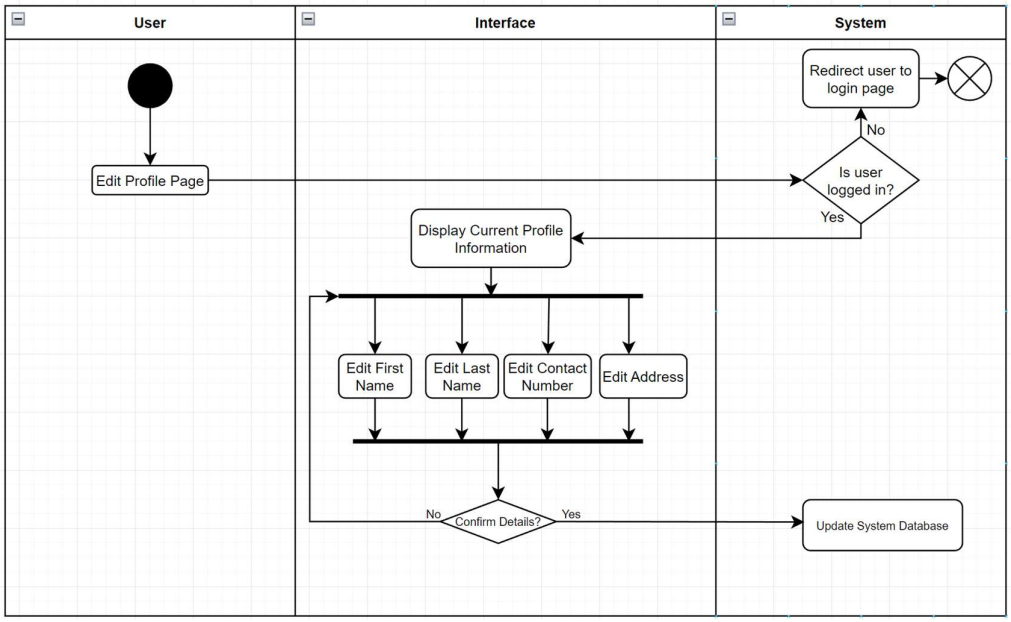


Figure 3.32 – *Edit Profile Activity Diagram, draw.io*

When clicking on the Edit Profile Icon, it brings the user to the Edit Profile Page. The system first checks if the user is logged in. If they are not, they will be redirected to the login page. Otherwise, they are granted access to the Edit Profile Page. Current Account Information will be displayed upon viewing of the page. If a user wishes to edit details of their account, a form is displayed, giving the user options to change their First Name, Last Name, Contact Number, and Address. Once the user is satisfied, they can click on the Update button and the system will accept this request and update their details accordingly.

3.4.09 Submit an Inquiry (Contact Us)

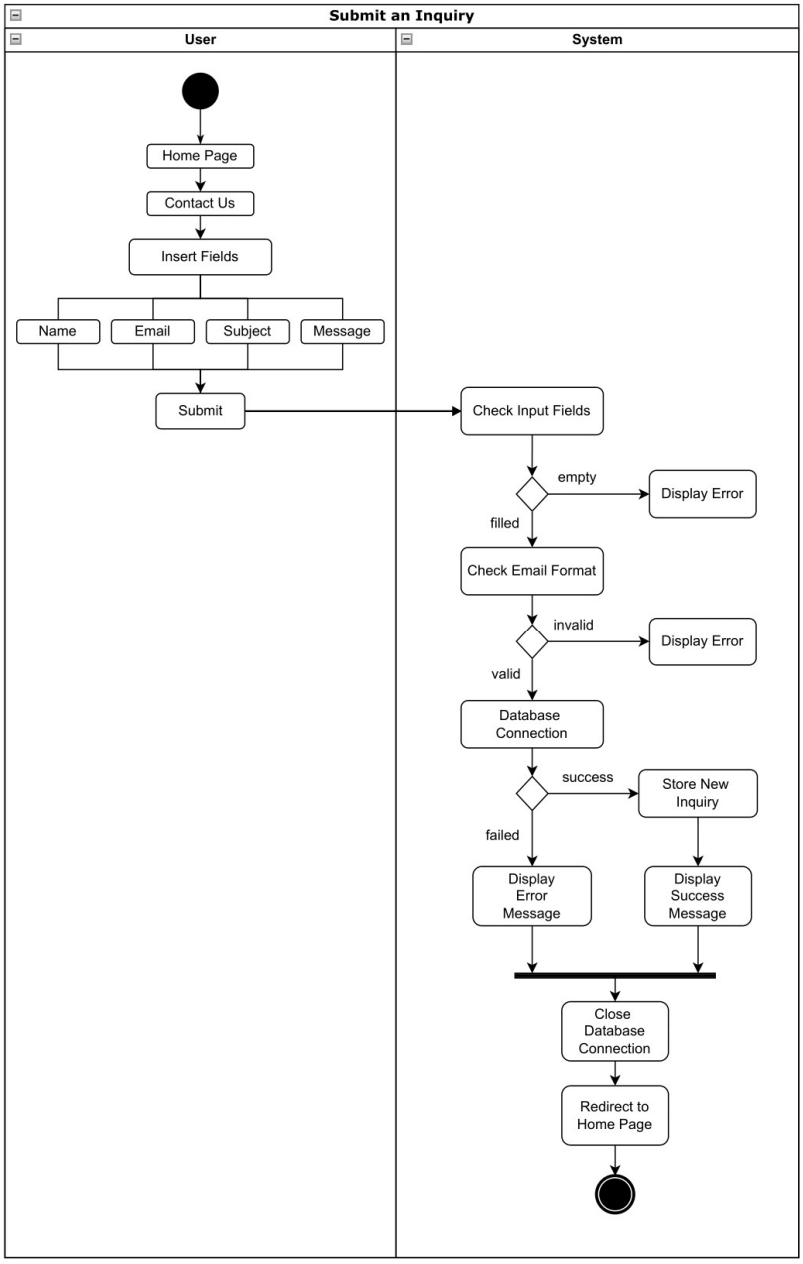


Figure 3.33 – *Contact Us / Submit an Inquiry Activity Diagram, draw.io*

To submit an inquiry, the user will need to navigate to the contact us page and insert the specific fields that are requested by the system. Once the submit button is clicked, the system checks for any empty fields and/or invalid email format. If the validation passes, the inquiry will be stored into the database and the user will be redirected to the homepage after a success message.

3.4.10 View Inquiries

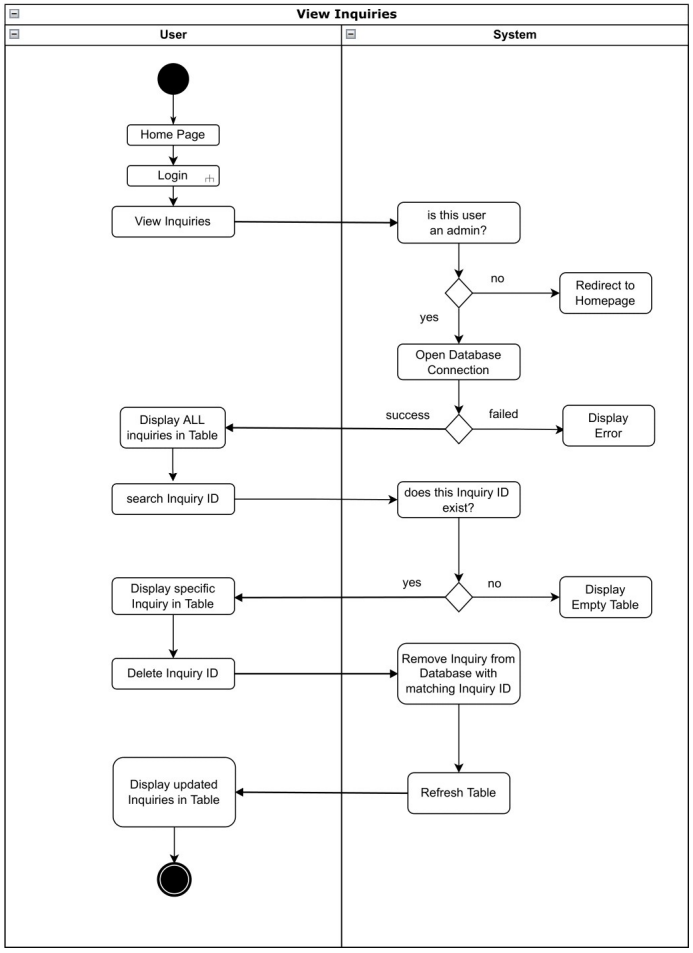


Figure 3.34 – *View Inquiries Activity Diagram, draw.io*

To view an inquiry, user must be logged in with an Admin account. The system will check for the user roles and redirect customers back to homepage if an attempt was made to bypass the permission. Once validated, the system will display ALL inquiries for admin in a table format. Admin can then optionally choose to search or delete inquiries based on the inquiry ID inserted.

3.4.11 View Order History

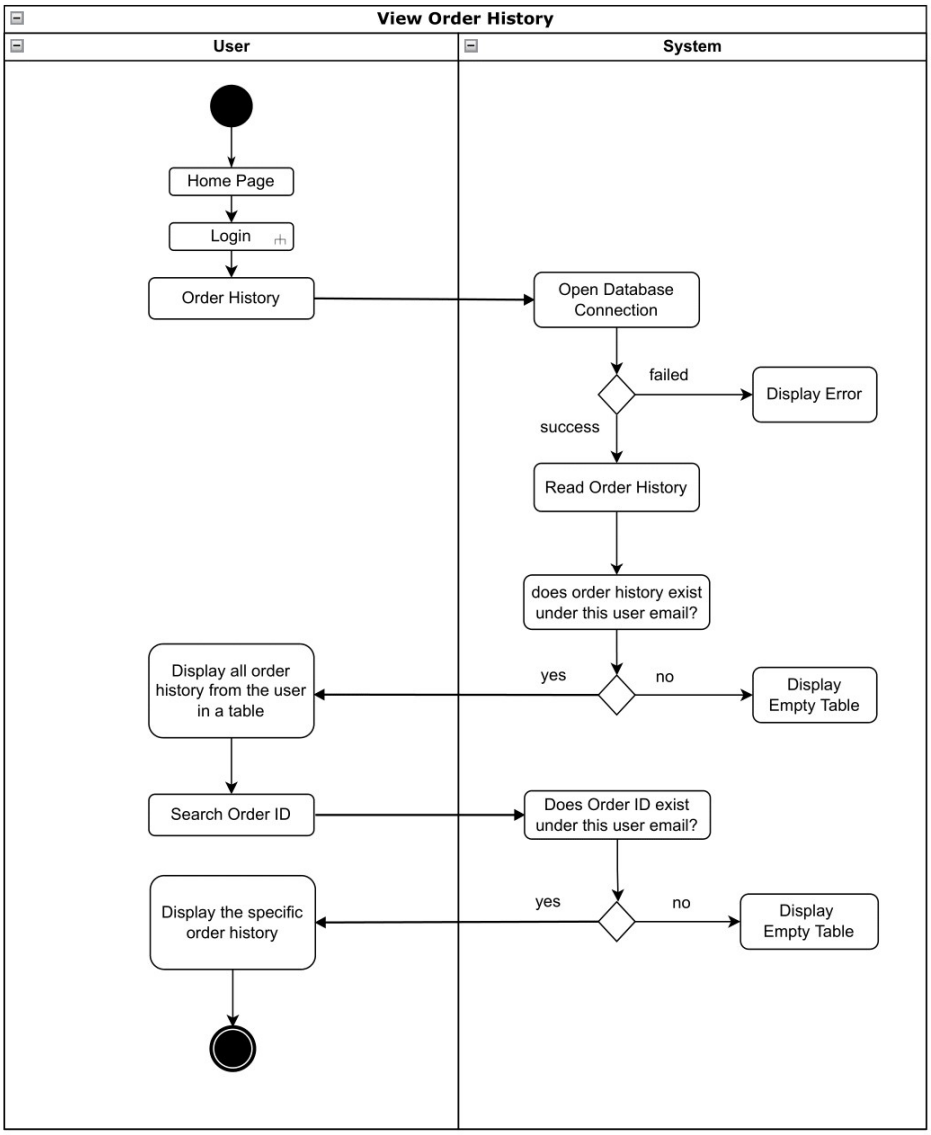


Figure 3.35 – *View Order History Activity Diagram, draw.io*

To view order history, user must be logged in to an existing customer or admin account. Upon requesting this protocol, the system will fetch all order history based on the users' email as its reference. User will NOT be able to view orders that DOES NOT belong to them. User can also optionally search for a specific order by inserted Order ID into the search tab.

3.4.12 Logout

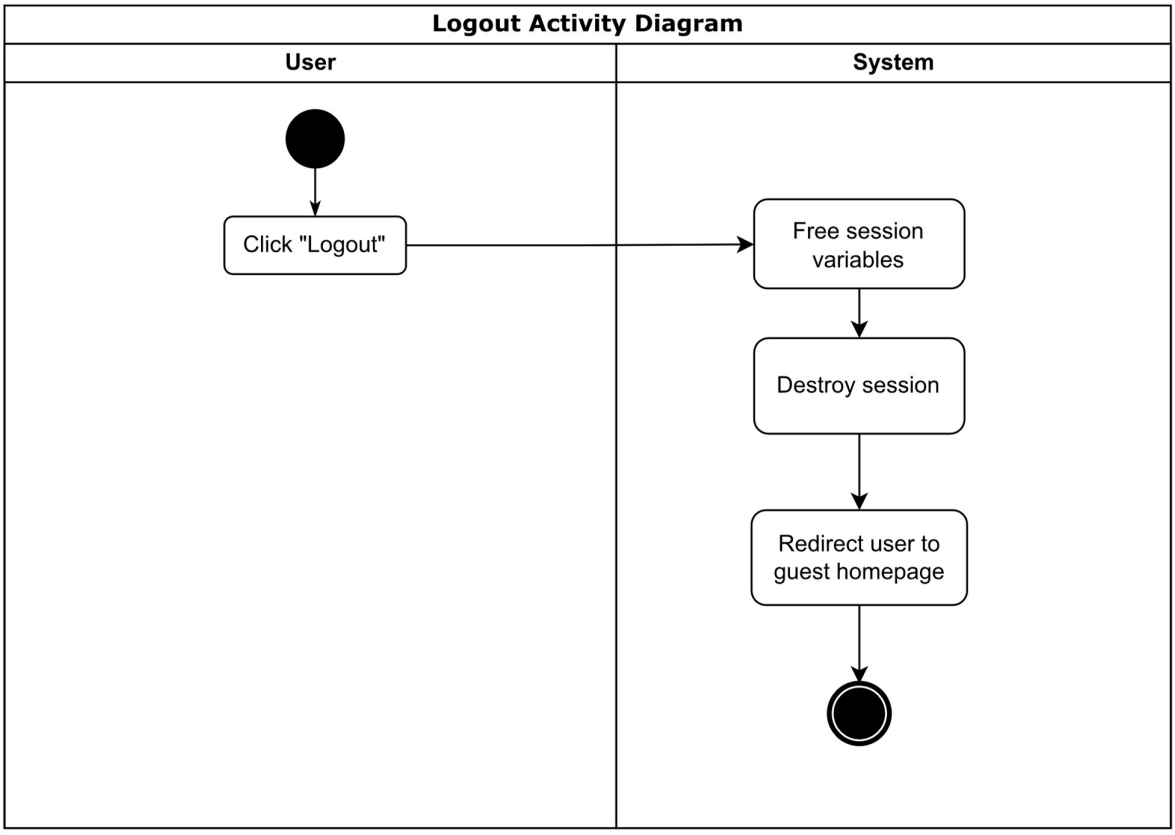


Figure 3.36 – *Logout activity diagram*

Upon clicking the logout button, all current session variables will be freed and destroyed. User is then redirected to the guest homepage.

4.0 IMPLEMENTATION

4.1 Create

4.1.1 Register user account

```php
function appendUser($conn, $firstName, $lastName, $email, $password, $role)
{
    $hashPass = password_hash($password, PASSWORD_DEFAULT);

    $sql = "INSERT INTO `users` (`first_name`, `last_name`, `email`, `password`, `role`) VALUES (?, ?, ?, ?, ?)";
    $stmt = mysqli_prepare($conn, $sql);
    mysqli_stmt_bind_param($stmt, "sssss", $firstName, $lastName, $email, $hashPass, $role);
    mysqli_stmt_execute($stmt);

    // Retrieve user ID
    $sql = "SELECT * FROM `users` WHERE email = ? ";
    $stmt = mysqli_prepare($conn, $sql);
    mysqli_stmt_bind_param($stmt, "s", $email);
    mysqli_stmt_execute($stmt);
    $result = mysqli_stmt_get_result($stmt);
    $row = mysqli_fetch_assoc($result);

    // Assign session user id
    session_start();
    $_SESSION['userID'] = $row['user_id'];
    $_SESSION['userRole'] = $row['role'];
    $_SESSION['userEmail'] = $row['email'];
}
```

Figure 4.00 – *Append user PHP code*

If the user inputs have succeeded all validations in the registration page, the account will then be created and user details are inserted into the database using the INSERT statement. Prepared statements are used to help prevent possible SQL injection attacks when inserting the values into the database. The user's ID, user's role, and user's email is then retrieved to be assigned into the $_SESSION variable.

4.1.2 Contact us

```
26  <!-- contact page section -->
27  <div class="contact-section">
28      <h1>Contact Us</h1>
29      <div class="contact-border"></div>
30      <form action="#" class="contact-form" method="get">
31          <input type="text" class="contact-form-text" placeholder="Your Name" name="txtName" required>
32          <input type="email" class="contact-form-text" placeholder="Your Email" name="txtEmail" required>
33          <input type="text" class="contact-form-text" placeholder="Your Subject" name="txtSubject" required>
34          <textarea class="contact-form-text" placeholder="Your Message" name="txtMessage" required></textarea>
35          <input type="submit" class="contact-form-btn contact-form-btn-animation" value="Submit" name="btnSubmit">
36
37  <!-- display status message after submission -->
38          <?php
39          include('../homepage/dbConnection.php');
40          if(isset($_GET['btnSubmit'])) {
41              $name = $_GET['txtName'];
42              $email = $_GET['txtEmail'];
43              $subject = $_GET['txtSubject'];
44              $message = $_GET['txtMessage'];
45              $query = "INSERT INTO `inquiries` (`name`, `email`, `subject`, `message`) VALUES ('$name','$email','$subject','$message')";
46              if (mysqli_query($connection, $query)) {
47                  mysqli_close($connection);
48                  echo "<p class='btn-response-message'>Thank you for submitting your query! We will get back to you soon!</p>"; ?>
49                  <script type="text/JavaScript">
50                      setTimeout("location.href = '../homepage/homepage.php';",2000);
51                  </script>
52
53              <?php } else {
54                  mysqli_close($connection);
55                  echo "<p class='btn-response-message'>Failed to send feedback"; ?>
56                  <script type="text/JavaScript">
57                      setTimeout("location.href = '../homepage/homepage.php';",2000);
58                  </script>
59              <?php }
60          }
61          ?>
62      </form>
63  </div>
64
```

Figure 4.01 – *contactpage.php, VS Code*

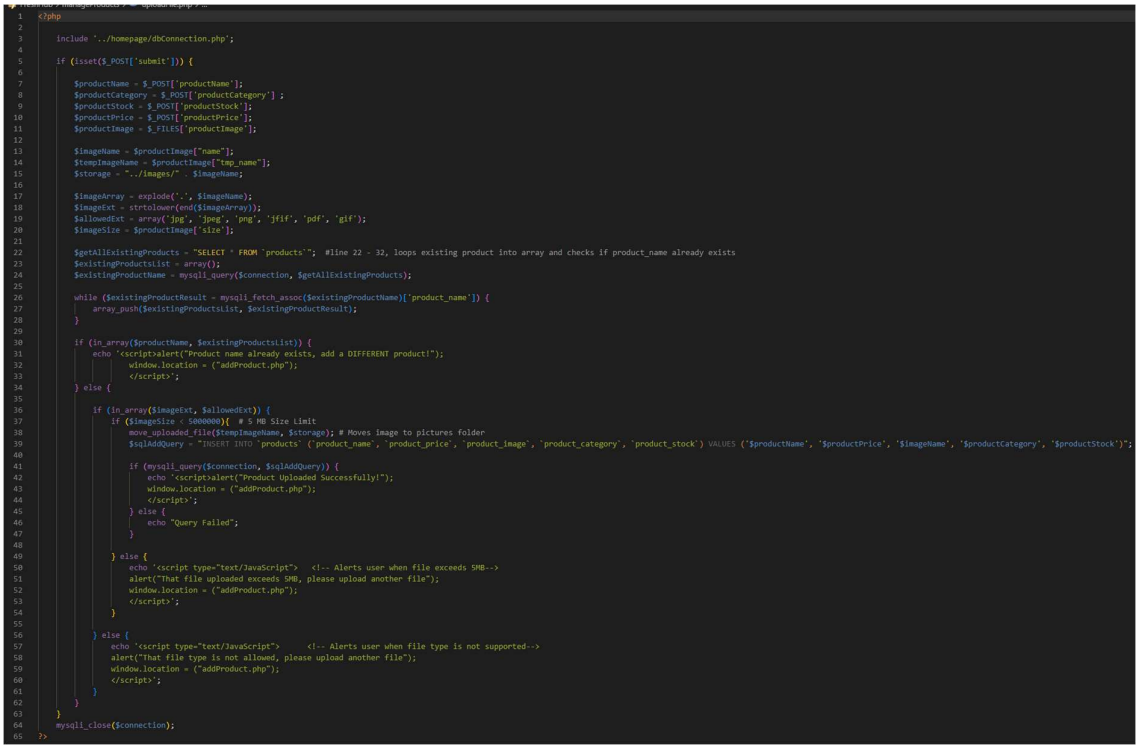
To **create** a new inquiry in the database, a user must be begin by filling in the HTML form created at line-30 of *contactpage.php*. Inside the HTML form tag, we specified the form action to stay on the same page after sending the information using the # attribute. The method we use is 'GET' as the information is not sensitive, and that method 'GET' runs better on browser compared to 'POST' due to the simplistic nature of it just appending to the URL (Williams, 2022). The 4 input fields are: 'Name', 'Email', 'Subject', and 'Message'. These fields contain the 'required' attribute to prevent empty forms from being submitted. A simple 'type' attribute is applied for email to validate its format as well. Finally, once the user presses the submit button, the PHP back-end takes over and appends the data from each field into the respective table columns in the SQL table named `inquiries`. Once the data is stored successfully, a success message is echoed using PHP, and the user is redirect to the Homepage using JavaScript at line-50. The 'setTimeout' function simply redirects the user to a given URL after 2 milliseconds. However, if the data fails to connect, an error message is echoed using PHP, and the page is refreshed for retry.

4.1.3 Add product

```
30  <!DOCTYPE html>
31  <html lang="en">
32  <head>
33      <meta charset="UTF-8">
34      <meta http-equiv="X-UA-Compatible" content="IE=edge">
35      <meta name="viewport" content="width=device-width, initial-scale=1.0">
36      <title>Add Products</title>
37      <link href = "../homepage/homepage.css" rel = "stylesheet" type = "text/css">
38      <link href = "productsAndProfiles.css" rel ="stylesheet" type = "text/css">
39  </head>
40  <body>
41      <h1 class = "product_title">Add Product</h1>
42      <div>
43          <form action = "uploadFile.php" method = "post" enctype = "multipart/form-data" class = "submissionForm">
44              Insert Product Name: <input type = "text" name = "productName" required> <br>
45              <section>
46                  Choose Product Category -
47                  <select name = "productCategory" required>
48                      <option value = "Fruits">Fruits</option>
49                      <option value = "Vegetables">Vegetables</option>
50                      <option value = "Meat">Meat</option>
51                  </select>
52              </section>
53              Insert Product Stock: <input type = "number" name = "productStock"  min = '0' required> <br>
54              Insert Product Price: <input type = "number" name = "productPrice" min = '0' step = 0.01 required> <br>
55              Insert Product Image: <input type = "file" name = "productImage" required>
56              <p style = "font-style: italic; font-size: 12px;">file types allowed - jpg, jpeg, png, jfif, pdf, gif</p>
57              <button type = "submit" name = "submit" class = "btn">Upload</button>
58          </form>
59      </div>
60  </body>
61  </html>
62
63  <?php
64      include '../homepage/footer.php';
65      mysqli_close($connection);
66  ?>
```

Figure 4.02 - *Add Products HTML Code*

The Add Products HTML code mainly consists of a form (**line 44 to line 58).** Users can enter information through the input fields, varying from text to numbers. A select element is used for Categories, allowing the user to select from Fruits, Vegetables, and Meat in a drop-down box (**line 45 to line 51)**. A file type input is also used for inserting images (**line 55).** Submitting this form will send this information to uploadFile.php for processing, through the method "POST".



```php
<?php

    include '../homepage/dbConnection.php';

    if (isset($_POST['submit'])) {

        $productName = $_POST['productName'];
        $productCategory = $_POST['productCategory'] ;
        $productStock = $_POST['productStock'];
        $productPrice = $_POST['productPrice'];
        $productImage = $_FILES['productImage'];

        $imageName = $productImage["name"];
        $tempImageName = $productImage["tmp_name"];
        $storage = "../images/" . $imageName;

        $imageArray = explode('.', $imageName);
        $imageExt = strtolower(end($imageArray));
        $allowedExt = array('jpg', 'jpeg', 'png', 'jfif', 'pdf', 'gif');
        $imageSize = $productImage['size'];

        $getAllExistingProducts = "SELECT * FROM `products`";  #line 22 - 32, loops existing product into array and checks if product_name already exists
        $existingProductsList = array();
        $existingProductName = mysqli_query($connection, $getAllExistingProducts);

        while ($existingProductResult = mysqli_fetch_assoc($existingProductName)['product_name']) {
            array_push($existingProductsList, $existingProductResult);
        }

        if (in_array($productName, $existingProductsList)) {
            echo '<script>alert("Product name already exists, add a DIFFERENT product!");
                    window.location = ("addProduct.php");
                    </script>';
        } else {

            if (in_array($imageExt, $allowedExt)) {
                if ($imageSize < 5000000){  # 5 MB Size Limit
                    move_uploaded_file($tempImageName, $storage); # Moves image to pictures folder
                    $sqlAddQuery = "INSERT INTO `products` (`product_name`, `product_price`, `product_image`, `product_category`, `product_stock`) VALUES ('$productName', '$productPrice', '$imageName', '$productCategory', '$productStock')";

                    if (mysqli_query($connection, $sqlAddQuery)) {
                        echo '<script>alert("Product Uploaded Successfully!");
                        window.location = ("addProduct.php");
                        </script>';
                    } else {
                        echo "Query Failed";
                    }

                } else {
                    echo '<script type="text/JavaScript">   <!-- Alerts user when file exceeds 5MB-->
                    alert("That file uploaded exceeds 5MB, please upload another file");
                    window.location = ("addProduct.php");
                    </script>';
                }

            } else {
                echo '<script type="text/JavaScript">      <!-- Alerts user when file type is not supported-->
                alert("That file type is not allowed, please upload another file");
                window.location = ("addProduct.php");
                </script>';
            }
        }
    }
    mysqli_close($connection);
?>
```

Figure 4.03 – *Add Products PHP Code (uploadFile.php)*

Information is gathered here through method POST, after a user submits information through addProduct.php. Information is stored into the declared variables from **line 7 to line 11.** Other variables declared are things such as the image name, file extension, and size, which will be used for validation later.

The query on **line 22** queries the database for all existing product names, then loops those values and compares them with the variable *$productName*, which contains the value of the name of the new product being added. If the names match, JavaScript code is executed, notifying the user that the Product Name already exists, then redirecting them to the same page, the new product will not be added (**line 30)**.

Other checks of validation include restrictions on Image Extensions and Image Size using IF statements at **line 30 and line 31**. JavaScript code also executes, notifying the user of any unsupported file types or file size limits when uploading an image (**line 50 to line 60).**

4.1.4 Create order

```php
function appendOrder($conn, $fullName, $contactNumber, $email, $address, $totalProducts, $totalPrice, $pMethod)
{
    $query = "INSERT INTO `orders` (`full_name`, `contact_number`, `email`, `address`,
    `total_products`, `total_price`, `payment_mode`) VALUES (?,?,?,?,?,?,?)";

    $stmt = mysqli_prepare($conn, $query);

    mysqli_stmt_bind_param($stmt, "sssssss", $fullName, $contactNumber, $email, $address,
    $totalProducts, $totalPrice, $pMethod);

    mysqli_stmt_execute($stmt);
}
```

Figure 4.04 – *Append order PHP code*

Once the user has clicked on the checkout button, the order will be created and inserted into the database. User's full name, contact number, email, address, summary of products, total price and payment method is inserted into the database using the INSERT statement. Prepared statements are used to avoid possible SQL injections.

4.2 Read

4.2.1 View inquiries

```php
<!-- search / delete Inquiry -->
<div class="modify-inquiry">
    <form class="inquiry-box" action="#" method="get">
        <input type="text" class="inquiry-form" placeholder="Search Inquiries ID" name="txtInquiryID" required>
        <input type="submit" value="Search" name="btnSubmit" class="inquiry-submit">
        <input type="submit" value="Delete" name="btnDelete" class="inquiry-delete">
    </form>
    <a href="../contactUs/viewInquiries.php"><button class="inquiry-reset">Reset</button></a>
</div>

<!-- result table -->
<table>
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Email</th>
        <th>Subject</th>
        <th>Message</th>
    </tr>
    <?php
        $conn = mysqli_connect("localhost", "root", "", "freshhub");
        if ($conn-> connect_error) {
            die("Connection failed:". $conn-> connect_error);
        }

        $sql = "SELECT * FROM inquiries ORDER BY inquiry_id DESC";
        $result = $conn-> query($sql);

        if (isset($_GET['btnSubmit'])) {
            $inquiry_id = $_GET['txtInquiryID'];
            $sqlQuery = "SELECT * FROM inquiries WHERE inquiry_id = '$inquiry_id'";
            $result = mysqli_query($conn, $sqlQuery);
            while ($row = mysqli_fetch_assoc($result)) {
                echo
                "<tr><td class='inquiry-id'>" . $row["inquiry_id"] .
                "</td><td class='inquiry-name'>" . $row["name"] .
                "</td><td class='inquiry-email'>" . $row["email"] .
                "</td><td class='inquiry-subject'>" . $row["subject"] .
                "</td><td class='class='inquiry-textarea'>" . $row["message"] .
                "</td></td>";
            }
            echo "</table>";

        } else if (isset($_GET['btnDelete'])) {
            $inquiry_id = $_GET['txtInquiryID'];
            $sqlQuery = "DELETE FROM `inquiries` WHERE inquiry_id= '$inquiry_id'";
            $result = mysqli_query($conn, $sqlQuery);?>
            <script type="text/JavaScript">
                setTimeout("location.href = '../contactUs/viewInquiries.php';",0);
            </script>
        <?php }
        else {
            $sql = "SELECT inquiry_id, name, email, subject, message FROM inquiries";
            $result = $conn-> query($sql);
            while ($row = $result -> fetch_assoc()) {
                echo
                "<tr><td class='inquiry-id'>" . $row["inquiry_id"] .
                "</td><td class='inquiry-name'>" . $row["name"] .
                "</td><td class='inquiry-email'>" . $row["email"] .
                "</td><td class='inquiry-subject'>" . $row["subject"] .
                "</td><td class='class='inquiry-textarea'>" . $row["message"] .
                "</td></td>";
            }
            echo "</table>";
        }

        $conn-> close();
    ?>
</table>
</div>
```

Figure 4.05 – *viewInquiries.php, VS Code*

To READ from the SQL database, a database connection must first be established. This is done at line-48 of *viewInquiries.php* using the mysqli_connect() function. After that, we use the $sql variable at line-53 to retrieve all data from the inquiry table using the SELECT command with the asterisk * symbol. Additionally, the sequence is set to be in a descending order using the ORDER BY keyword paired with DESC which denotes a descending format. Besides that, Admin can search for specific Inquiries by inserting the Inquiries ID into the HTML form at line-30. Upon clicking submit, an algorithm at line-56 will execute, which will save the user input as $inquiry_id and retrieve matching inquiry_id into the $row variable from the SQL inquiries tables. These results are then echoed into the HTML table using PHP for admin reference.

4.2.2 View order history

```php
<div class="database-order">
    <h1 class="order-header">Order History</h1>
    <div class="order-border"></div>

    <!-- search / delete Order -->
    <div class="search-order">
        <form class="order-box" action="#" method="get">
            <input type="text" class="order-form" placeholder="Search Order ID" name="txtOrderID" required>
            <input type="submit" value="Search" name="btnSubmit" class="order-submit">
        </form>
        <a href="../viewOrderHistory/orderHistory.php"><button class="order-reset">Reset</button></a>
    </div>

    <!-- result table -->
    <table>
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Contact Number</th>
            <th>Email</th>
            <th>Address</th>
            <th>Total Products</th>
            <th>Total Price</th>
            <th>Payment Mode</th>
        </tr>
        <?php
            $conn = mysqli_connect("localhost", "root", "", "freshhub");
            if ($conn-> connect_error) {
                die("Connection failed:". $conn-> connect_error);
            }

            $sql = "SELECT inquiry_id, name, email, subject, message FROM inquiries ORDER BY inquiry_id DESC";
            $result = $conn-> query($sql);

            if (isset($_GET['btnSubmit'])) {
                $order_id = $_GET['txtOrderID'];
                $sqlQuery = "SELECT * FROM orders WHERE order_id = '$order_id' AND email = '$userEmail'";
                $result = mysqli_query($conn, $sqlQuery);
                while ($row = mysqli_fetch_assoc($result)) {
                    echo
                    "<tr><td>" . $row["order_id"] .
                    "</td><td>" . $row["full_name"] .
                    "</td><td>" . $row["contact_number"] .
                    "</td><td>" . $row["email"] .
                    "</td><td>" . $row["address"] .
                    "</td><td>" . $row["total_products"] .
                    "</td><td>" . $row["total_price"] .
                    "</td><td>" . $row["payment_mode"] .
                    "</td></td>";
                }
                echo "</table>";

            } else {
                // $sql = "SELECT inquiry_id, name, email, subject, message FROM inquiries";
                $sql = "SELECT * FROM `orders` WHERE `email` = '$userEmail' ORDER BY order_id DESC";
                $result = $conn-> query($sql);
                while ($row = $result -> fetch_assoc()) {
                    echo
                    "<tr><td>" . $row["order_id"] .
                    "</td><td>" . $row["full_name"] .
                    "</td><td>" . $row["contact_number"] .
                    "</td><td>" . $row["email"] .
                    "</td><td>" . $row["address"] .
                    "</td><td>" . $row["total_products"] .
                    "</td><td>" . $row["total_price"] .
                    "</td><td>" . $row["payment_mode"] .
                    "</td></td>";
                }
                echo "</table>";
            }

            $conn-> close();
        ?>
    </table>
</div>
```

Figure 4.06 – *orderHistory.php, VS Code*

The *orderHistory.php* file function similarly to the previous *viewInquiries.php* file. After establishing a database connection at line-60, it will retrieve all order history from the `orders` table that contains the **same email** found in the SESSION created by the system through user login. This validation as detailed in line-88 ensures that users can only view their **own** orders, and not that of others. Users may optionally insert an order id into the HTML form located at line-40, which will be used for reference when fetching the specific order history later on at line-70. By utilizing SESSIONS, we have improved the READ functionality of SQL with added security measures and privacy.

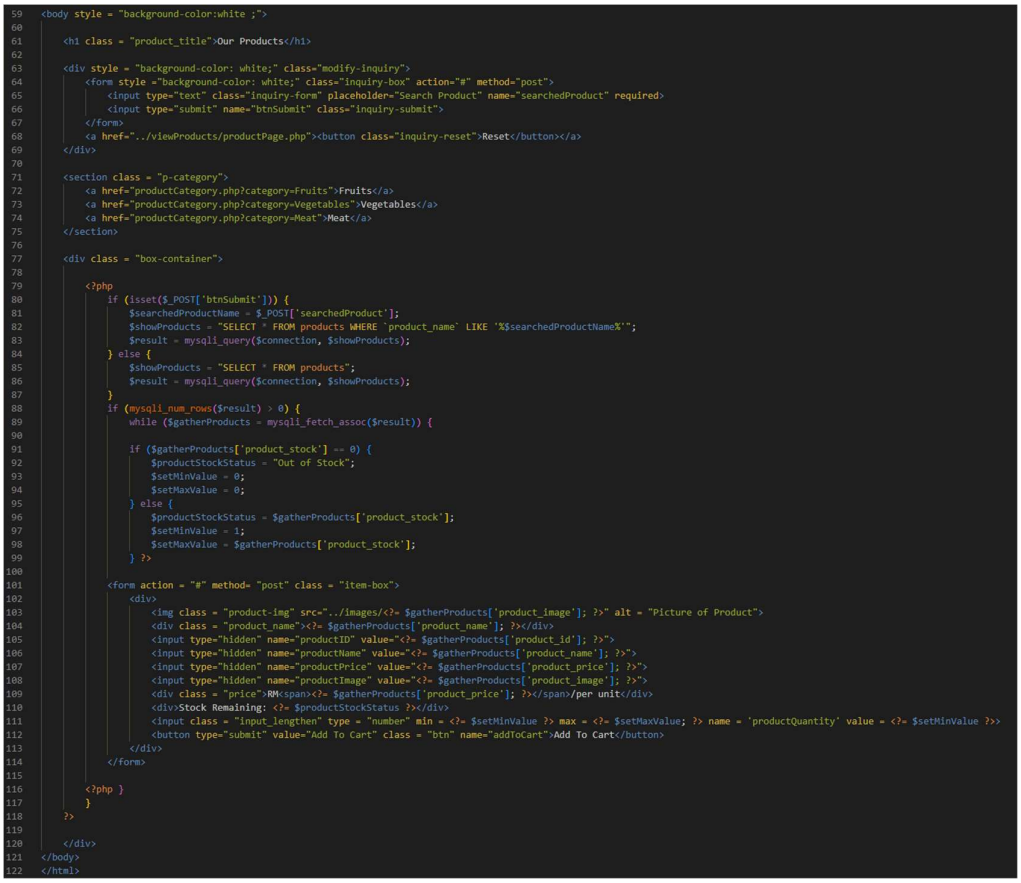
4.2.3 View shopping cart

```php
<?php
$select_cart_query = "SELECT * FROM cart_items WHERE `user_id` = '$userID'";
$grand_total = 0;
$results = mysqli_query($conn, $select_cart_query);
$products = mysqli_fetch_all($results, MYSQLI_ASSOC);
if ($products) {
    foreach ($products as $product) {

?>
        <tr class="products">
            <td><img src="../images/<?php echo $product['product_image']; ?>" alt="<?= $product['product_name']; ?> image"></td>
            <td><?php echo $product['product_name']; ?></td>
            <td>RM <?php echo number_format($product['product_price'], 2, '.', ''); ?></td>
            <td>
                <!-- Send form to same page -->
                <form action="" method="post">
                    <input type="hidden" name="update_product_id" value="<?php echo $product['product_id']; ?>">
                    <input type="number" name="update_quantity" min="1" value="<?php echo $product['cart_item_quantity']; ?>">
                    <input type="submit" value="Update" name="update_quantity_btn">
                </form>
            </td>
            <td>RM <?php echo $sub_total = number_format($product['product_price'] * $product['cart_item_quantity'], 2, '.', ''); ?></td>
            <td><a href="cart.php?remove=<?php echo $product['product_id']; ?>" onclick="return confirm('Remove item from cart?')"
            class="delete-btn"> <i class="fas fa-trash"></i> Remove</a></td>
        </tr>

<?php
        $grand_total += $sub_total;
        $grand_total = number_format($grand_total, 2, ".", ",");
    }
}
?>
```

Figure 4.07 – *Display cart items PHP code*

A SELECT query is created to locate all cart items that are associated to the user from the database table. A loop is used to display all the product details in multiple table rows. Each table row contains product image, product name, product price, product quantity and the total price. The grand total is then computed and displayed at the final row of the table.

4.2.4 View Products (Main Page)

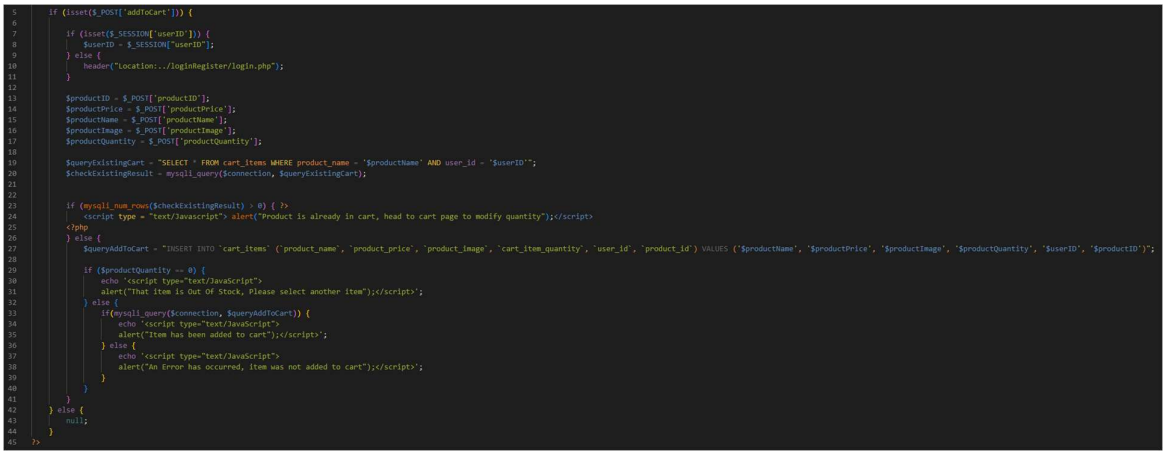


```
59  <body style = "background-color:white ;">
60
61      <h1 class = "product_title">Our Products</h1>
62
63      <div style = "background-color: white;" class="modify-inquiry">
64          <form style ="background-color: white;" class="inquiry-box" action="#" method="post">
65              <input type="text" class="inquiry-form" placeholder="Search Product" name="searchedProduct" required>
66              <input type="submit" name="btnSubmit" class="inquiry-submit">
67          </form>
68          <a href="../viewProducts/productPage.php"><button class="inquiry-reset">Reset</button></a>
69      </div>
70
71      <section class = "p-category">
72          <a href="productCategory.php?category=Fruits">Fruits</a>
73          <a href="productCategory.php?category=Vegetables">Vegetables</a>
74          <a href="productCategory.php?category=Meat">Meat</a>
75      </section>
76
77      <div class = "box-container">
78
79          <?php
80              if (isset($_POST['btnSubmit'])) {
81                  $searchedProductName = $_POST['searchedProduct'];
82                  $showProducts = "SELECT * FROM products WHERE `product_name` LIKE '%$searchedProductName%'";
83                  $result = mysqli_query($connection, $showProducts);
84              } else {
85                  $showProducts = "SELECT * FROM products";
86                  $result = mysqli_query($connection, $showProducts);
87              }
88              if (mysqli_num_rows($result) > 0) {
89                  while ($gatherProducts = mysqli_fetch_assoc($result)) {
90
91                  if ($gatherProducts['product_stock'] == 0) {
92                      $productStockStatus = "Out of Stock";
93                      $setMinValue = 0;
94                      $setMaxValue = 0;
95                  } else {
96                      $productStockStatus = $gatherProducts['product_stock'];
97                      $setMinValue = 1;
98                      $setMaxValue = $gatherProducts['product_stock'];
99                  } ?>
100
101             <form action = "#" method= "post" class = "item-box">
102                 <div>
103                     <img class = "product-img" src="../images/<?= $gatherProducts['product_image']; ?>" alt = "Picture of Product">
104                     <div class = "product_name"><?= $gatherProducts['product_name']; ?></div>
105                     <input type="hidden" name="productID" value="<?= $gatherProducts['product_id']; ?>">
106                     <input type="hidden" name="productName" value="<?= $gatherProducts['product_name']; ?>">
107                     <input type="hidden" name="productPrice" value="<?= $gatherProducts['product_price']; ?>">
108                     <input type="hidden" name="productImage" value="<?= $gatherProducts['product_image']; ?>">
109                     <div class = "price">RM<span><?= $gatherProducts['product_price']; ?></span>/per unit</div>
110                     <div>Stock Remaining: <?= $productStockStatus ?></div>
111                     <input class = "input_lengthen" type = "number" min = <?= $setMinValue ?> max = <?= $setMaxValue; ?> name = 'productQuantity' value = <?= $setMinValue ?>>
112                     <button type="submit" value="Add To Cart" class = "btn" name="addToCart">Add To Cart</button>
113                 </div>
114             </form>
115
116         <?php }
117         }
118     ?>
119
120     </div>
121 </body>
122 </html>
```

Figure 4.08 – *View Products HTML (Main Page)*

At the top of the page displays a section of hyperlinks to “productCategory.php” (**line 71 – line 75)**, sending along with it the string of the category. This is so that productCategory.php is sent correct info to display the products associated with that category appropriately.

PHP code is then used from **line 79 – line 99** to query the database for products available. If the user chooses to use the search bar at (**line 63 – line 68),** data is entered through a form element and submitted to the same page. PHP code from **line 80** to **line 87** then stores the string entered in the form and queries the database through a while loop (**line 89)** to find a product containing that string. If one or more products are found, they are all displayed in the form element from **line 101 to line 114.** If the search bar is not used, all products will be displayed. The queries for each searching and displaying all products are on **line 82 and line 85.** If users click on the “Add to Cart” button on the website, information from this form containing information on the product is sent to the same .php file through method “POST”.



```
5   if (isset($_POST['addToCart'])) {
6
7       if (isset($_SESSION['userID'])) {
8           $userID = $_SESSION["userID"];
9       } else {
10          header("Location:../loginRegister/login.php");
11      }
12
13      $productID = $_POST['productID'];
14      $productPrice = $_POST['productPrice'];
15      $productName = $_POST['productName'];
16      $productImage = $_POST['productImage'];
17      $productQuantity = $_POST['productQuantity'];
18
19      $queryExistingCart = "SELECT * FROM cart_items WHERE product_name = '$productName' AND user_id = '$userID'";
20      $checkExistingResult = mysqli_query($connection, $queryExistingCart);
21
22
23      if (mysqli_num_rows($checkExistingResult) > 0) { ?>
24          <script type = "text/Javascript"> alert("Product is already in cart, head to cart page to modify quantity");</script>
25      <?php
26      } else {
27          $queryAddToCart = "INSERT INTO `cart_items` (`product_name`, `product_price`, `product_image`, `cart_item_quantity`, `user_id`, `product_id`) VALUES ('$productName', '$productPrice', '$productImage', '$productQuantity', '$userID', '$productID')";
28
29          if ($productQuantity == 0) {
30              echo '<script type="text/JavaScript">
31              alert("That item is Out Of Stock, Please select another item");</script>';
32          } else {
33              if(mysqli_query($connection, $queryAddToCart)) {
34                  echo '<script type="text/JavaScript">
35                  alert("Item has been added to cart");</script>';
36              } else {
37                  echo '<script type="text/JavaScript">
38                  alert("An Error has occurred, item was not added to cart");</script>';
39              }
40          }
41      }
42  } else {
43      null;
44  }
45  ?>
```

Figure 4.09 – *View Products PHP Code*

Continuing from the HTML Code, all product information that is to be added into a cart is sent to this PHP code. Upon clicking the "Add to Cart" button, triggering the if(isset) code on **line 5**¸ this checks if a session already exists, and stores that session into *$userID.* If a session has not been set, indicating that the user has not logged in yet, they will be redirected to the login page (**line 10).** This reinforces that the user cannot add items to a cart without logging in.

Otherwise, information on a product is stored into declared variables from **line 13 to line 17**. Code from **line 19** then queries cart_items in the database to see if the item that the user had wished to add already existed in the cart. If so, a JavaScript alert code is executed **(line 24),** notifying the user that the item already exists in the cart, redirecting them back to the View Products Page.

If an item is not in the user's cart yet, a query at **line 27** is executed to insert the variables declared from **line 13 to line 17** into the appropriate fields within the database. However, if the product added had a stock of 0, **line 29 to line 31** executes using an if statement, triggering JavaScript code to alert the user that the Item is out of stock. The code then also redirects them back to the View Products Page.

Else, the query is executed, adding the desired product to the user's cart, then notifying the user through JavaScript, alerting them if the Item has been successfully added, or had failed (**line 33-38).**

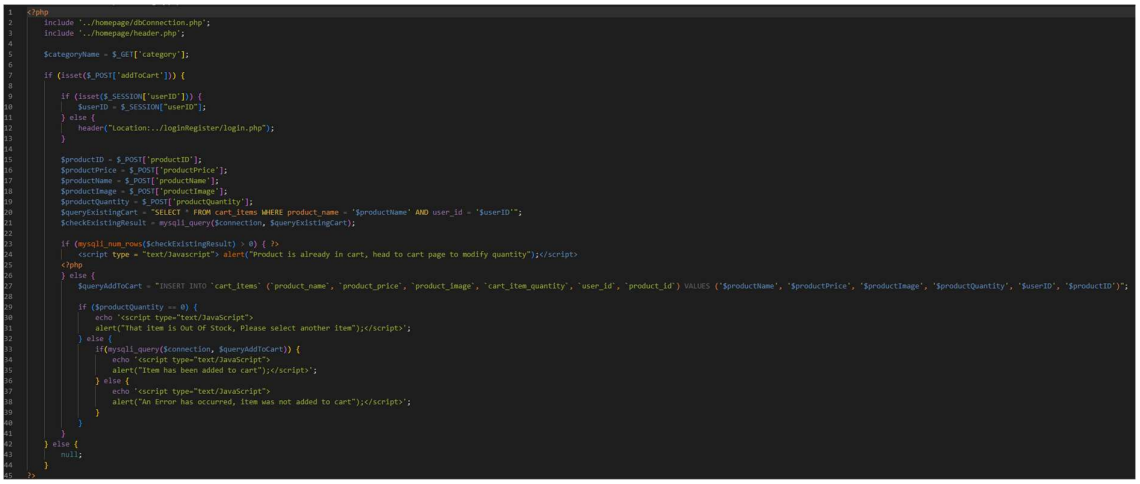
4.2.5 View Products (Category Page)

```
<body style = "background-color:white ;">
    <h1 class = "product_title">Products From <?=$categoryName?></h1>

    <div style = "background-color: white;" class="modify-inquiry">
        <form style ="background-color: white;" class="inquiry-box" action="#" method="post">
            <input type="text" class="inquiry-form" placeholder="Search Product" name="searchedProduct" required>
            <input type="submit" name="btnSubmit" class="inquiry-submit">
        </form>
        <a href="../viewProducts/productCategory.php?category=<?=$categoryName?>"><button class="inquiry-reset">Reset</button></a>
    </div>

    <section class = "p-category">
        <a href="productCategory.php?category=Fruits">Fruits</a>
        <a href="productCategory.php?category=Vegetables">Vegetables</a>
        <a href="productCategory.php?category=Meat">Meat</a>
    </section>
    <div class = "box-container">
    <?php

        if (isset($_POST['btnSubmit'])) {
            $searchedProductName = $_POST['searchedProduct'];;
            $productQuery = "SELECT * FROM products WHERE `product_category` = '$categoryName' AND  `product_name` LIKE '%$searchedProductName%'";
            $result = mysqli_query($connection, $productQuery);
        } else {
            $productQuery = "SELECT * FROM products WHERE `product_category` = '$categoryName'";
            $result = mysqli_query($connection, $productQuery);
        }

        $rowCount = mysqli_num_rows($result);

        if($rowCount > 0) {
            while ($gatherProducts = mysqli_fetch_assoc($result)) {

                if ($gatherProducts['product_stock'] == 0) {
                    $productStockStatus = "Out of Stock";
                    $setMinValue = 0;
                    $setMaxValue = 0;
                } else {
                    $productStockStatus = $gatherProducts['product_stock'];
                    $setMinValue = 1;
                    $setMaxValue = $gatherProducts['product_stock'];
                } ?>

            <form action = "#" method= "post" class = "item-box">
                <img class ="product-img" src="../images/<?= $gatherProducts['product_image']; ?>" alt = "Picture of Product">
                <div class = "product_name"><?= $gatherProducts['product_name']; ?></div>
                <input type="hidden" name="productID" value="<?= $gatherProducts['product_id']; ?>">
                <input type="hidden" name="productName" value="<?= $gatherProducts['product_name']; ?>">
                <input type="hidden" name="productPrice" value="<?= $gatherProducts['product_price']; ?>">
                <input type="hidden" name="productImage" value="<?= $gatherProducts['product_image']; ?>">
                <div class = "price">RM<span><?= $gatherProducts['product_price']; ?></span>/per unit</div>
                <div>Stock Remaining: <?= $productStockStatus?></div>
                <input class = "input_lengthen" type = "number" min = <?= $setMinValue ?> max = <?= $setMaxValue; ?> name = 'productQuantity' value = <?= $setMinValue ?>>
                <button type="submit" value="Add To Cart" class = "btn" name="addToCart">Add To Cart</button>
            </form>
        <?php } 

    } else {
        echo 'No Products from this Category Available';
    }
        ?>
    </div>
```

Figure 4.10 – *View Products (Category Page) HTML Code*

The View Products Category Page works very similarly to **4.2.5 View Products Main Page.** The main difference now are in the queries at **line 79** and **line 82**, where there queries are now both displayed based on the category instead. The variable $categoryName stores the category name after clicking the hyperlink on the **View Products Main Page.**

Functions of searching for a product are also still present here, as can be seen with the form element from **line 61** to **line 67.** All other code is a repeat of the Main Product Page.



```php
<?php
    include '../homepage/dbConnection.php';
    include '../homepage/header.php';

    $categoryName = $_GET['category'];

    if (isset($_POST['addToCart'])) {

        if (isset($_SESSION['userID'])) {
            $userID = $_SESSION["userID"];
        } else {
            header("Location:../loginRegister/login.php");
        }

        $productID = $_POST['productID'];
        $productPrice = $_POST['productPrice'];
        $productName = $_POST['productName'];
        $productImage = $_POST['productImage'];
        $productQuantity = $_POST['productQuantity'];
        $queryExistingCart = "SELECT * FROM cart_items WHERE product_name = '$productName' AND user_id = '$userID'";
        $checkExistingResult = mysqli_query($connection, $queryExistingCart);

        if (mysqli_num_rows($checkExistingResult) > 0) { ?>
            <script type = "text/Javascript"> alert("Product is already in cart, head to cart page to modify quantity");</script>
        <?php
        } else {
            $queryAddToCart = "INSERT INTO `cart_items` (`product_name`, `product_price`, `product_image`, `cart_item_quantity`, `user_id`, `product_id`) VALUES ('$productName', '$productPrice', '$productImage', '$productQuantity', '$userID', '$productID')";

            if ($productQuantity == 0) {
                echo '<script type="text/JavaScript">
                alert("That item is Out Of Stock, Please select another item");</script>';
            } else {
                if(mysqli_query($connection, $queryAddToCart)) {
                    echo '<script type="text/JavaScript">
                    alert("Item has been added to cart");</script>';
                } else {
                    echo '<script type="text/JavaScript">
                    alert("An Error has occurred, item was not added to cart");</script>';
                }
            }
        }
    } else {
        null;
    }
?>
```

Figure 4.11 – *View Products (Category Page) PHP Code*

Similar to the Main Products Page, a user cannot add items to their cart if they are not logged in, as they will be re-directed back to the login page, through the use of an IF statement and session variables (**line 7 to line 13)**. Information on a product is stored into the declared variables from **line 15 to line 19**. A query on **line 20** checks if the item being added into the cart already exists in the user's cart. If it is, a JavaScript prompt will alert the user that the item already exists in their cart (**line 24).**

If the item does not exist in the cart, the query $queryAddToCart then executes and inserts the declared variables into the database according to the appropriate fields. On **line 29,** an IF statement is used to validate if an item is in stock. If an item is out of stock, JavaScript code on **line 30** will execute, alerting the user that the item is out of stock. Other forms of validation through JavaScript can be seen on **line 34 and line 38**, notifying the user of the success or failure of an item being added to a cart.

4.3 Update

4.3.1 Modify product

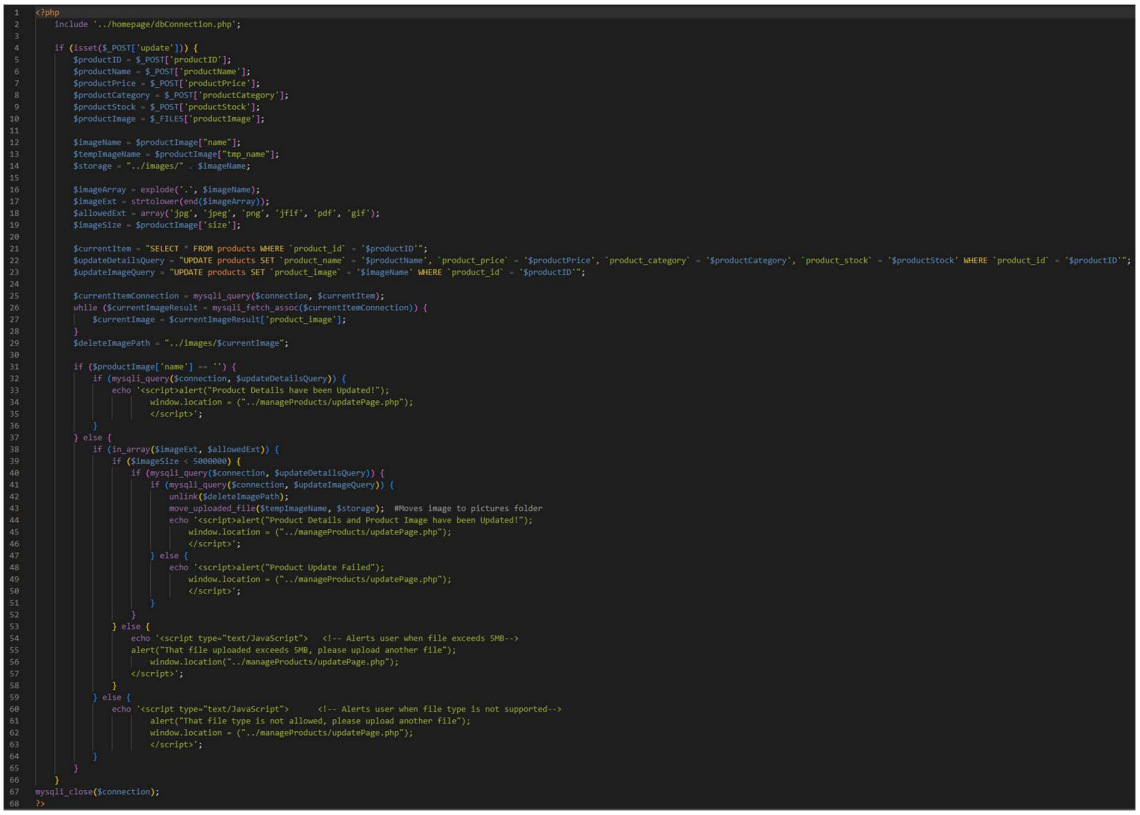
```
28      $productID = $_POST['productID'];
29      $sqlFetchDetails = "SELECT * FROM `products` WHERE `product_id` = '$productID'";
30      $executeQuery = mysqli_query($connection, $sqlFetchDetails);
31  ?>
32
33  <!DOCTYPE html>
34  <html lang="en">
35  <head>
36      <meta charset="UTF-8">
37      <meta http-equiv="X-UA-Compatible" content="IE=edge">
38      <meta name="viewport" content="width=device-width, initial-scale=1.0">
39      <title>Update <?=$_POST['productName']?> Page</title>
40      <link href = "../homepage/homepage.css" rel = "stylesheet" type = "text/css">
41      <link href = "productsAndProfiles.css" rel = "stylesheet">
42  </head>
43  <body>
44          <?php while ($queryResult = (mysqli_fetch_assoc($executeQuery))) { ?>
45          <h1 class = "product_title">Update <?=$queryResult['product_name'] ?> Page</h1>
46          <div class = "box">
47                  <img class ="product-img" src="../images/<?= $queryResult['product_image']; ?>" alt = "Picture of Product">
48                  <div>Product ID : <?= $queryResult['product_id']; ?></div>
49                  <div>Product Name : <?= $queryResult['product_name']; ?></div>
50                  <div>Product Category : <?= $queryResult['product_category']; ?></div>
51                  <div>Price : RM<span><?= $queryResult['product_price']; ?></span>/per unit</div>
52                  <div>Stock Remaining : <?= $queryResult['product_stock']; ?></div>
53          </div>
54
55          <form action = "updateProcess.php" method = "post" class = "submissionForm" enctype = "multipart/form-data">
56              <input type = "hidden" name = "productID" value = "<?= $queryResult['product_id']; ?>"> <!-- Hidden Product ID-->
57              Modify Product Name: <input type = "text" name = "productName" required value = "<?= $queryResult['product_name']; ?>"> <br>
58              <section>
59                  Choose Product Category -
60                  <select name = "productCategory" required >
61                      <option <?php if ($queryResult['product_category'] == "Fruits") {echo ("selected");}?>>Fruits</option>
62                      <option <?php if ($queryResult['product_category'] == "Vegetables") {echo ("selected");}?>>Vegetables</option>
63                      <option <?php if ($queryResult['product_category'] == "Meat") {echo ("selected");}?>>Meat</option>
64                  </select>
65              </section>
66              Modify Product Price: <input type = "number" name = "productPrice" min = '0' step = 0.01 required value = "<?= $queryResult['product_price']; ?>"> <br>
67              Modify Product Stock: <input type = "number" name = "productStock"  min = '0' required value = "<?= $queryResult['product_stock']; ?>"> <br>
68              Modify Product Image: <input type = "file" name = "productImage">
69              <p style = "font-style: italic; font-size: 12px;">file types allowed - jpg, jpeg, png, jfif, pdf, gif</p>
70              <button type = "submit" name = "update" class = "btn" style = "background-color: #E48929; width: 65%;">Update</button>
71          </form>
72  </body>
73  </html>
74
75  <?php }
76      mysqli_close($connection);
77      include '../homepage/footer.php';
78  ?>
```

Figure 4.12 – *Update Product Main HTML*

Main HTML code for Updating Products first query for the product in the database based on its Product ID (**line 28 – line 30)**. Then, results are displayed through a while loop, displaying the product to be updated (**line 44)**. Product information is displayed in div elements within a div (**line 46 – line 53).**

Looking lower, a form from **line 55** to **line 71** displays a form for the user to input the details required to update a product, incorporating inputs of different types, such as text and number. The <select tag> from **line 60** to **line 64** is also used when choosing a product category, displaying it in a drop-down form, only allowing the user to choose one of three categories available.

Minimum and Maximum values were applied appropriately to input fields for product stock and product price. Values from this form are submitted towards updateProcess.php.



```php
<?php
    include '../homepage/dbConnection.php';

    if (isset($_POST['update'])) {
        $productID = $_POST['productID'];
        $productName = $_POST['productName'];
        $productPrice = $_POST['productPrice'];
        $productCategory = $_POST['productCategory'];
        $productStock = $_POST['productStock'];
        $productImage = $_FILES['productImage'];

        $imageName = $productImage["name"];
        $tempImageName = $productImage["tmp_name"];
        $storage = "../images/" . $imageName;

        $imageArray = explode('.', $imageName);
        $imageExt = strtolower(end($imageArray));
        $allowedExt = array('jpg', 'jpeg', 'png', 'jfif', 'pdf', 'gif');
        $imageSize = $productImage['size'];

        $currentItem = "SELECT * FROM products WHERE `product_id` = '$productID'";
        $updateDetailsQuery = "UPDATE products SET `product_name` = '$productName', `product_price` = '$productPrice', `product_category` = '$productCategory', `product_stock` = '$productStock' WHERE `product_id` = '$productID'";
        $updateImageQuery = "UPDATE products SET `product_image` = '$imageName' WHERE `product_id` = '$productID'";

        $currentItemConnection = mysqli_query($connection, $currentItem);
        while ($currentImageResult = mysqli_fetch_assoc($currentItemConnection)) {
            $currentImage = $currentImageResult['product_image'];
        }
        $deleteImagePath = "../images/$currentImage";

        if ($productImage['name'] == '') {
            if (mysqli_query($connection, $updateDetailsQuery)) {
                echo '<script>alert("Product Details have been Updated!");
                        window.location = ("../manageProducts/updatePage.php");
                        </script>';
            }
        } else {
            if (in_array($imageExt, $allowedExt)) {
                if ($imageSize < 5000000) {
                    if (mysqli_query($connection, $updateDetailsQuery)) {
                        if (mysqli_query($connection, $updateImageQuery)) {
                            unlink($deleteImagePath);
                            move_uploaded_file($tempImageName, $storage);  #Moves image to pictures folder
                            echo '<script>alert("Product Details and Product Image have been Updated!");
                                window.location = ("../manageProducts/updatePage.php");
                                </script>';
                        } else {
                            echo '<script>alert("Product Update Failed");
                                window.location = ("../manageProducts/updatePage.php");
                                </script>';
                        }
                    }
                } else {
                    echo '<script type="text/JavaScript">   <!-- Alerts user when file exceeds 5MB-->
                    alert("That file uploaded exceeds 5MB, please upload another file");
                        window.location("../manageProducts/updatePage.php");
                    </script>';
                }
            } else {
                echo '<script type="text/JavaScript">      <!-- Alerts user when file type is not supported-->
                        alert("That file type is not allowed, please upload another file");
                        window.location = ("../manageProducts/updatePage.php");
                        </script>';
            }
        }
    }
mysqli_close($connection);
?>
```

Figure 4.13 – *Update Process PHP Code*

Information from the previous HTML form is gathered in this PHP code through the “POST” method, including all variables related to a product (**line 5 – line 10). Line 12 – line 19** include details of the image uploaded, such as its name, file extension, and size. This is used for validating if the image is suitable to be uploaded, complying with the constraints set from **line 31 – line 39.** Separate update queries were made for updating a product’s details and product image (**line 22 and line 23).** This is practical because when updating a product, it isn’t convenient for a user to constantly upload the image every time during updating.

Remaining JavaScript code from **line 44** to **line 64** alerts the user of successful Updates to a product, as well as failures, including those of where the Image Size is over 5 Megabytes, or where the file type has not followed the system’s constraints. The image size constraint can be seen on **line 39** through an if statement. The system checks the variable *$imageExt* on **line 17,** checking if it is inside of array *$allowedExt* **,** declared on **line 18.** This checking process can be seen on **line 38**, using an if statement and the in_array() PHP function.

4.3.2 Modify shopping cart

```php
if (isset($_POST['update_quantity_btn'])) {
    $updateValue = $_POST['update_quantity'];
    $updateID = $_POST['update_product_id'];

    $update_quantity_query = "UPDATE `cart_items` SET `cart_item_quantity` = ?
    WHERE `product_id` = ? and `user_id` = ?";
    $stmt = mysqli_prepare($conn, $update_quantity_query);
    mysqli_stmt_bind_param($stmt, "iii", $updateValue, $updateID, $userID);
    mysqli_stmt_execute($stmt);
}
```

Figure 4.14 – *Update item quantity*

Upon submitting the form in **figure 4.07**, the values are retrieved by using the $_POST variable. The cart item associated to the user id will have its cart item quantity updated in the database with the new quantity entered by the user by using the UPDATE statement.

4.3.3 Edit profile

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Profile</title>
    <link href = "../homepage/homepage.css" rel = "stylesheet" type = "text/css">
    <link href = '../manageProducts/productsAndProfiles.css' rel = 'stylesheet'>
</head>
<body>
    <h1 class = "product_title">Edit Profile
        <img width="30px" src = "../images/edit.png" style = "margin-left:10px;">
    </h1>
    <?php
        $showProfile = "SELECT * FROM users WHERE user_id = '$userID'";
        $result = mysqli_query($connection, $showProfile);
        if (mysqli_num_rows($result) > 0) {
            while ($showProfile = mysqli_fetch_assoc($result)) { ?>

            <section class = "box-container" style = "grid-template-columns:1fr 1fr;">
                <div class = "item-box" style = "width:100%;" >
                    <div class = "box" style = "padding-bottom:10px; margin-right:100px; width: 50%;">
                        <div>User ID : <?= $showProfile['user_id']; ?></div>
                        <div>First Name : <?= $showProfile['first_name']; ?></div>
                        <div>Last Name : <?= $showProfile['last_name']; ?></div>
                        <div>Email : <?= $showProfile['email']; ?></div>
                        <div>Address : <?= $showProfile['address']; ?></div>
                        <div>Phone Number : <?= $showProfile['contact_number']; ?></div>
                        <div style = "text-transform:capitalize;">Role: <?= $showProfile['role']; ?></div>
                    </div>
                </div>

                <form action = "#" method = "post" class = "submissionForm" style = "margin-left: 100px; width:40%;">
                    First Name: <br> <input class = "input_lengthen" type = "text" name = "firstName" value = "<?= $showProfile['first_name']; ?>" required> <br>
                    Last Name: <br> <input class = "input_lengthen" type = "text" name = "lastName" value = "<?= $showProfile['last_name']; ?>" required> <br>
                    Contact Number: <br> <input class = "input_lengthen" type = "text" min = "0" name = "contactNumber" required value = "<?= $showProfile['contact_number'];?>" > <br>
                    Address: <br> <input style = "height: 50px; length: 150px; margin-top:15px;" type = "text" name = "address" required value = "<?= $showProfile['address'];?>"> <br>
                    <button type = "submit" name = "update" class = "btn" style = "background-color: #E48929; width: 70%; font-size:24px;">Update</button>
                </form>
            </section>

        <?php }
    } ?>
</body>
</html>

<?php
    mysqli_close($connection);
    include ("../homepage/footer.php");
?>
```

Figure 4.15 – *Edit Profile Main HTML Code*

The user's profile information is first fetched through the query on **line 47**, querying info from the users database based on the *$userID* attained through a SESSION variable. User information is then displayed in a div within a div, containing individual elements such as User ID, First Name, Last Name, and more (**line 53 – line 62).**

After this , a form is displayed containing inputs, allowing the user to edit their First Name, Last Name, Contact Number, and Address (**line 65 – line 71)**. Information from this form is submitted within the same .php file.

```
1  <?php
2      include ("../homepage/dbConnection.php");
3      include ("../homepage/header.php");
4
5      if (isset($_SESSION['userID'])) {
6          $userID = $_SESSION["userID"];
7      } else {
8          header("Location:../loginRegister/login.php");
9      }
10
11     if (isset($_POST['update'])) {
12         $firstName = $_POST['firstName'];
13         $lastName = $_POST['lastName'];
14         $address = $_POST['address'];
15         $contactNumber = $_POST['contactNumber'];
16
17         $editProfileQuery = "UPDATE users SET `first_name` = '$firstName', `last_name` = '$lastName', `address` = '$address', `contact_number` = '$contactNumber' WHERE `user_id` = '$userID'";
18         $executeQuery = mysqli_query($connection, $editProfileQuery);
19
20         if ($executeQuery) {
21             echo '<script>alert("Profile has been Updated!");
22                         window.location = ("../profilePage/editProfile.php");
23                         </script>';
24         } else {
25             '<script>alert("Profile Update Failed!");
26                         window.location = ("../profilePage/editProfile.php");
27                         </script>';
28         }
29     }
30 ?>
```

Figure 4.16 – *Edit Profile Main PHP Code*

As mentioned earlier, *$userID* is attained through the current SESSION (**line 6)**. If there is no current session, the user will be redirected to the login page, where a SESSION will be declared after logging in (**line 8)**.

Information from the form from (**line 65 – line 71)** is sent to this PHP code through method "POST". Variables *$firstName, $lastName, $address,* and *$contactNumber* are declared (**line 12 – line 15)**, and information from the form is stored in these variables. **Line 17** displays the update query where variables are set to the correct fields based on the SQL database. The query is then executed on **line 20.**

Remaining JavaScript code (**line 21 – line 27)** alerts the user on the success or a failure of updating the Profile, then redirecting them back to the same Edit Profile page.

4.3.4 Forgot password

```
1  function changePass($conn, $email, $newPass)
2  {
3      if ($row = emailExist($conn, $email)) {
4          $hashPass = password_hash($newPass, PASSWORD_DEFAULT);
5          $id = $row['user_id'];
6          $sql = "UPDATE `users` SET `password` = ? WHERE `user_id` = ?";
7          $stmt = mysqli_prepare($conn, $sql);
8          mysqli_stmt_bind_param($stmt, "ss", $hashPass, $id);
9          mysqli_stmt_execute($stmt);
10     } else {
11         return false;
12     }
13 }
```

Figure 4.17 – *Change password*

A function changePass is defined to update the existing user's password. The user's email is first validated to check if it exists in the database. Then the password entered is hashed and the existing password is replaced by using the UPDATE statement defined in line 6. If the email does not exist, the function will return a false value.

4.4 Delete

4.4.1 Delete product

```
<!DOCTYPE html>
<html lang="en" style = "background-color:white ;">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Delete Products</title>
    <link href = "../homepage/homepage.css" rel = "stylesheet" type = "text/css">
    <link href = "productsAndProfiles.css" rel ="stylesheet" type = "text/css">
    <link rel="stylesheet" href="../contactUs/inquiriesTable.css">
</head>
<body style = "background-color:white ;">
    <h1 class = "product_title">Delete Products</h1>

    <div style = "background-color: white;" class="modify-inquiry">
        <form style ="background-color: white;" class="inquiry-box" action="#" method="post">
            <input type="text" class="inquiry-form" placeholder="Search Product" name="searchedProduct" required>
            <input type="submit" name="btnSubmit" class="inquiry-submit">
        </form>
        <a href="../manageProducts/deleteProduct.php"><button class="inquiry-reset">Reset</button></a>
    </div>

    <div class = "box-container">

        <?php
            if (isset($_POST['btnSubmit'])) {
                $searchedProductName = $_POST['searchedProduct'];
                $showProducts = "SELECT * FROM products WHERE `product_name` LIKE '%$searchedProductName%'";
                $result = mysqli_query($connection, $showProducts);
            } else {
                $showProducts = "SELECT * FROM products";
                $result = mysqli_query($connection, $showProducts);
            }

            if (mysqli_num_rows($result) > 0) {
                while ($gatherProducts = mysqli_fetch_assoc($result)) { ?>

            <form action = "#" method= "post" class = "item-box">
                <div>
                    <img class ="product-img" src="../images/<?= $gatherProducts['product_image']; ?>" alt = "Picture of Product">
                    <div class = "product_name"><?= $gatherProducts['product_name']; ?></div>
                    <input type="hidden" name="productID" value="<?= $gatherProducts['product_id']; ?>">
                    <input type="hidden" name="productName" value="<?= $gatherProducts['product_name']; ?>">
                    <input type="hidden" name="productPrice" value="<?= $gatherProducts['product_price']; ?>">
                    <input type="hidden" name="productImage" value="<?= $gatherProducts['product_image']; ?>">
                    <div class = "price">RM<span><?= $gatherProducts['product_price']; ?></span>/per unit</div>
                    <div>Stock Remaining: <?= $gatherProducts['product_stock']; ?></div>
                    <button type="submit" value="Delete Product" class = "btn" name="deleteProduct">Delete</button>
                </div>
            </form>

            <?php }
            } else {
                echo "<h1 style = 'text-align: center;'>No Products To Delete</h1>";
            }
        ?>
    </div>
</body>
</html>

<?php
    mysqli_close($connection);
    include '../homepage/footer.php';
?>
```

Figure 4.18 – *Delete Product Main HTML*

Main HTML Code from **line 83** to **line 102** displays products through a form, using PHP code from **line 70** to **line 81**. The PHP code allows users to enter a product name of their choice, then queries the database for all products including those characters, as can be seen in **line 73** with the use of *"LIKE %$searchedProductName%"*. Otherwise, all products are queried from the database and displays them on a page through a while loop on **line 80.** If no products are found through the query, the system echoes the line "No Products to Delete", as can be seen in **line 99**.

```php
    if (isset($_POST['deleteProduct'])) {

        $productID = $_POST['productID'];
        $productImage = $_POST['productImage'];
        $productName = $_POST['productName'];
        $deleteQuery = "DELETE FROM products WHERE product_id = '$productID'";
        $imagePath = "../images/$productImage";

        if (mysqli_query($connection, $deleteQuery)) {
            if (unlink ($imagePath));
            echo '<script>alert("Product has been removed");
                window.location("deleteProduct.php")
                </script>';
        } else {
            echo "Product Deletion Failed";
        };
    }
?>
```

Figure 4.19 – *Delete Product Main PHP, Cont'*

As seen from the Delete Product Main HTML earlier, information from that form is posted towards this PHP Code. Details such as Product ID, Product Image and more are stored into variables. The PHP Code then runs a query to delete the product based on its Product ID (**line 32)**. The image path is also mentioned on **line 33**, so that the image can also be deleted from the file consequently.

If the query runs and a product is deleted, unlink is used, followed by the path of the image to remove the image from the local folders (**line 36)**. Remaining PHP and JavaScript code is used to alert the user if Product Deletion was successful or a failure. Both outcomes redirect the user back to the Delete Product Page.

4.4.2 Delete inquiries

```
28    <!-- search / delete Inquiry -->
29    <div class="modify-inquiry">
30        <form class="inquiry-box" action="#" method="get">
31            <input type="text" class="inquiry-form" placeholder="Search Inquiries ID" name="txtInquiryID" required>
32            <input type="submit" value="Search" name="btnSubmit" class="inquiry-submit">
33            <input type="submit" value="Delete" name="btnDelete" class="inquiry-delete">
34        </form>
35        <a href="../contactUs/viewInquiries.php"><button class="inquiry-reset">Reset</button></a>
36    </div>
37
38    <!-- result table -->
39    <table>
40        <tr>
41            <th>ID</th>
42            <th>Name</th>
43            <th>Email</th>
44            <th>Subject</th>
45            <th>Message</th>
46        </tr>
47        <?php
48            $conn = mysqli_connect("localhost", "root", "", "freshhub");
49            if ($conn-> connect_error) {
50                die("Connection failed:". $conn-> connect_error);
51            }
52
53            $sql = "SELECT * FROM inquiries ORDER BY inquiry_id DESC";
54            $result = $conn-> query($sql);
55
56            if (isset($_GET['btnSubmit'])) {
57                $inquiry_id = $_GET['txtInquiryID'];
58                $sqlQuery = "SELECT * FROM inquiries WHERE inquiry_id = '$inquiry_id'";
59                $result = mysqli_query($conn, $sqlQuery);
60                while ($row = mysqli_fetch_assoc($result)) {
61                    echo
62                    "<tr><td class='inquiry-id'>" . $row["inquiry_id"] .
63                    "</td><td class='inquiry-name'>" . $row["name"] .
64                    "</td><td class='inquiry-email'>" . $row["email"] .
65                    "</td><td class='inquiry-subject'>" . $row["subject"] .
66                    "</td><td class='class='inquiry-textarea'>" . $row["message"] .
67                    "</td></td>";
68                }
69                echo "</table>";
70
71            } else if (isset($_GET['btnDelete'])) {
72                $inquiry_id = $_GET['txtInquiryID'];
73                $sqlQuery = "DELETE FROM `inquiries` WHERE inquiry_id= '$inquiry_id'";
74                $result = mysqli_query($conn, $sqlQuery);?>
75                <script type="text/JavaScript">
76                    setTimeout("location.href = '../contactUs/viewInquiries.php';",0);
77                </script>
```

Figure 4.20 – *viewInquiries.php delete function, VS Code*

After addressing the inquiries, admin may want to delete the records to make space for new tickets. As such, a delete feature is introduced, Admin can enter an inquiry ID into the HTML form at line-30 and click on the ‘delete’ button on the webpage. The ‘btnDelete’ input sent by the delete button will trigger line-71 to execute, which includes a DELETE query at line-73. This SQL query will only delete the record that matches the input given by the Admin which is saved under $inquiry_id. Once the data has been deleted, the webpage is refreshed using JavaScript at line-76 to display the updated records.

4.4.3 Checkout

```php
function deleteCartItems($conn, $userID)
{
    $query = "DELETE FROM `cart_items` WHERE `user_id` = ?";
    $stmt = mysqli_prepare($conn, $query);
    mysqli_stmt_bind_param($stmt, "s", $userID);
    mysqli_stmt_execute($stmt);
}
```

Figure 4.21 – *deleteCartItems function PHP code*

A function deleteCartItems is defined to clear the user's cart after successfully checking out. A DELETE statement is used to remove all the cart items that are associated with the user's ID.

4.5 Login

```php
function verifyLogin($conn, $email, $password)
{
    if ($row = emailExist($conn, $email)) {
        // Verify user input password and db password
        $verify = password_verify($password, $row['password']);
        if ($verify) {
            // Assign session variables
            session_start();
            $_SESSION['userID'] = $row['user_id'];
            $_SESSION['userRole'] = $row['role'];
            $_SESSION['userEmail'] = $row['email']; //added email into session for view order history
            return true;
        } else {
            return "badPassword";
        }
    } else {
        return "badEmail";
    }
}
```

Figure 4.22 – *Verify login PHP code*

A verifyLogin function is defined to check if the email exists and if the password matches to what is stored in the system. Values are returned depending on the results of the validations.

```php
function emailExist($conn, $email)
{
    $sql = "SELECT * FROM `users` WHERE `email` = ?";

    $stmt = mysqli_prepare($conn, $sql);
    mysqli_stmt_bind_param($stmt, "s", $email);
    mysqli_stmt_execute($stmt);
    $result = mysqli_stmt_get_result($stmt);
    if ($row = mysqli_fetch_assoc($result)) {
        return $row;
    } else {
        return false;
    }
}
```

Figure 4.23 – *Verify email PHP code*

The emailExist function checks if the email entered by the user exists in the database. If it exists, the user's detail is returned, otherwise a false value is returned.

```php
<?php
if (isset($_GET['error'])) {
    if ($_GET['error'] == "badEmail") {
        echo "<p class='error'>Email does not exist!</p>";
    }
    if ($_GET['error'] == "badPassword") {
        echo "<p class='error'>Wrong password!</p>";
    }
    // To homepage
    if ($_GET['error'] == "noError") {
        echo "<p class='noError'>Login Success!</p>";
        header("Refresh:1, URL=../homepage/homepage.php");
    }
}
?>
```

Figure 4.24 – *Login PHP code*

Depending on the value returned from the functions defined, the system will display an error message if the credentials entered by the user does not match the system. If the credentials are valid, the user will be redirected to the homepage of FreshHub.

4.6 Signup

Register (validation)

```php
//Validations for signup inputs
if (emptyInput($firstName, $lastName, $email, $password) !== false) {
    header("location: ../../loginRegister/signup.php?error=emptyInputs");
    die();
}

if (invalidName($firstName, $lastName) !== false) {
    header("location: ../../loginRegister/signup.php?error=invalidName");
    die();
}

if (invalidEmail($email) !== false) {
    header("location: ../../loginRegister/signup.php?error=invalidEmail");
    die();
}

if (emailExist($conn, $email) !== false) {
    header("Location:../../loginRegister/signup.php?error=emailExist");
    die();
}

if (invalidPass($password) !== false) {
    header("location: ../../loginRegister/signup.php?error=invalidPassword");
    die();
}
```

Figure 4.25 – *Validate sign up inputs PHP code*

Multiple validations are made upon submitting user credentials into the sign-up form. Multiple functions are defined to check if the inputs submitted have:

- Empty inputs.
- Digits in first name or last name.
- Email that does not have the correct format.
- Email that already exists in database.
- Password that is not more than or equal to 8 characters.

```php
// Check for number in string
function numPresent($string)
{
    for ($i = 0; $i < strlen($string); $i++) {
        if (ctype_digit($string[$i])) {
            return true;
        }
    }
    return false;
}


// Check length or has num
function invalidName($firstName, $lastName)
{

    if (strlen($firstName) > 25 or strlen($firstName) < 3) {
        return true;
    } else if (strlen($lastName) > 25 or strlen($lastName) < 3) {
        return true;
    } else if (numPresent($firstName)) {
        return true;
    } else if (numPresent($lastName)) {
        return true;
    } else {
        return false;
    }
}
```

Figure 4.26 – *Validate name PHP code*

Figure 4.26 displays two functions defined to validate user's first name and last name. In line 17 and 19 onwards, user's first name and last name cannot be more than 25 characters or less than 3 characters. The strings are then checked to see if there are any numbers present. If the first name and last names are valid, a false value is returned to indicate that the inputs are valid.

```php
function appendUser($conn, $firstName, $lastName, $email, $password, $role)
{
    $hashPass = password_hash($password, PASSWORD_DEFAULT);

    $sql = "INSERT INTO `users` (`first_name`, `last_name`, `email`, `password`, `role`) VALUES (?, ?, ?, ?, ?)";
    $stmt = mysqli_prepare($conn, $sql);
    mysqli_stmt_bind_param($stmt, "sssss", $firstName, $lastName, $email, $hashPass, $role);
    mysqli_stmt_execute($stmt);

    // Retrieve user ID
    $sql = "SELECT * FROM `users` WHERE email = ? ";
    $stmt = mysqli_prepare($conn, $sql);
    mysqli_stmt_bind_param($stmt, "s", $email);
    mysqli_stmt_execute($stmt);
    $result = mysqli_stmt_get_result($stmt);
    $row = mysqli_fetch_assoc($result);

    // Assign session user id
    session_start();
    $_SESSION['userID'] = $row['user_id'];
    $_SESSION['userRole'] = $row['role'];
    $_SESSION['userEmail'] = $row['email'];
}
```

Figure 4.27 – *Append user PHP code*

If the user inputs have succeeded all validations, it will then be inserted into the database using the INSERT statement. Prepared statements are used to help prevent attacks like SQL injection when inserting the values into the database. The user's ID, user's role, and user's email is then retrieved to be assigned into the $_SESSION variable.

```php
<?php
if (isset($_GET['error'])) {
    if ($_GET['error'] == "emptyInputs") {
        echo "<p class='error'>Please fill in all inputs</p>";
    }
    if ($_GET['error'] == "invalidName") {
        echo "<p class='error'>Invalid name</p>";
    }
    if ($_GET['error'] == "invalidEmail") {
        echo "<p class='error'>Invalid email</p>";
    }
    if ($_GET['error'] == "emailExist") {
        echo "<p class='error'>Email exists, please enter another email</p>";
    }
    if ($_GET['error'] == "invalidPassword") {
        echo "<p class='error'>Password should be more than or equal to 8 characters</p>";
    }
    // To homepage
    if ($_GET['error'] == "noError") {
        echo "<p class='noError'>Successfully registered!</p>";
        header("Refresh:1, URL=../homepage/homepage.php");
    }
}
?>
```

Figure 4.28 – *Sign up PHP code*

If the user inputs did not pass the validations, an appropriate error message will be echoed to the user. For example, if the user's email already exists, the system will display "Email exists, please enter another email". Otherwise, if the user inputs have passed all validations, the user will be redirected to the homepage of FreshHub with the account automatically logged in.

4.7 Self-created CSS

4.7.1 Declaring CSS Variables

The styling begins by declaring the CSS variables for the main colours using the :root selector. This allows us to call this variable later in the sheet by using the var() function, saving us time, and reducing potential errors or discrepancy overall.

```
:root {
    --forest: #163A24;
    --sunset: #F6BB27;
    --sahara: #E48929;
    --oak: #E48929;
}
```

Figure 4.29 – *CSS Variables*

4.7.2 Define the universal selector

The Universal Selector is also defined using the asterisk (*) symbol, with margin and padding set to 0, and the box-sizing to border-box. This ensure there's no gap between the webpages and the browsers, and that it fully utilize the space it is given.

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

Figure 4.30 – *CSS Universal Selectors*

4.7.3 Applying Flexbox

Flexbox is a useful organizing tool that helps to lay a collection of items in one direction or another. By using flex-direction column, we are essential allowing the website to stack the contents vertical downwards, which makes for a smooth scrolling experience. The Didact Gothic font is also applied to the body tag to set the default typeface for improved consistency of our webpage.

```
body {
    font-family: 'Didact Gothic', sans-serif;
    display: flex;
    flex-direction: column;
    min-height: 100vh;
}
```

Figure 4.31 – *CSS Flexbox & Fonts*

4.7.4 Animating Buttons

The beauty of CSS is that you can add interactive animations to entity such as a button or an input box. In this example, we have given the offer button (btnOffer) a colour of #DD5353 (red), a border-radius of 30px to make its corner round, and a transition of background with a 0.5 second duration. This compliments with the :hover selector so when user hover the button, the button will turn from Red to Orange smoothly.

```
.btnOffer {
    display: inline-block;
    background: #DD5353;
    color: #FFF;
    padding: 10px 35px;
    margin: 30px 0;
    border-radius: 30px;
    transition: background 0.5s;
}


.btnOffer:hover {
    background: #F1A661;
}
```

Figure 4.32 – *CSS Animations*

4.7.5 Creating Effects

Apart from animation, we can also add effects such as box-shadow to certain content. By using a rgba value of (0, 0, 0, 0.2), we have set the colour to black with an opacity (alpha) value of 0.2 . This effect is applied behind the box we've created in the About Us page, and a transition of 'transform' is applied so it glides upwards when you hover over it.

```
.testimonial .col-3 {
    text-align: center;
    padding: 40px 20px;
    box-shadow: 0 0 20px 0 rgba(0,0,0,0.2);
    cursor: pointer;
    transition: transform 0.5s;
    background: #FAF7F0;
    height: 450px;
}
```

Figure 4.33 – *CSS Effects*

4.7.6 Using Filters

We utilized the CSS 'filter' property in our 'Partners Page' to create a glowing logo that illuminate colours when you hover over them. We achieve this effect by using the 'grayscale' attribute and setting the intensity to from 100% to 0% when a user hover over the brands.

```
.col-5 img {
    width: 90%;
    cursor: pointer;
    filter: grayscale(100%);
}


.col-5 img:hover {
    filter: grayscale(0%);
}
```

Figure 4.34 – *CSS Filters*

4.7.7 Sticking the Header

The header of our webpage contains lots of tabs that are useful for navigating around the page. As such, we have used CSS to stick the header to the top of the webpage for enhanced convenience when scrolling. This is done by using **position: sticky**, which will lock the header, at **top: 0,** which is the top of the browser

```
.header {
    background: #163A24;
    position: sticky;
    top: 0;
}
```

Figure 4.35 – *CSS sticky headers*

4.7.8 Querying Media

We applied the @media rule to apply a tailored stylesheet to certain objects such as those found in the Shopping Cart Page. By applying this technique, we basically programmed the CSS to respond to the aspect ratio of the browser and change accordingly.

```
/* Media queries */

@media (max-width:1312px) {
    .shopping-cart table tbody td {
        padding: 0.6rem;
    }

    .shopping-cart table input[type="number"] {
        padding: 0.4rem;
    }

    .shopping-cart table input[type="submit"] {
        padding: 2% 4%;
        font-size: 1rem;
    }

    .delete-btn,
    .continue-btn {
        padding: 0.7rem;
    }

}
```

Figure 4.36 – *CSS Media*

By applying CSS in a creative yet functional way, we are able to create an aesthetic user interface and create a stunning website with just a few lines of codes.

4.8 Self-created JavaScript

```
19
20          if ($executeQuery) {
21              echo '<script>alert("Profile has been Updated!");
22                          window.location = ("../profilePage/editProfile.php");
23                          </script>';
24          } else {
25              '<script>alert("Profile Update Failed!");
26                          window.location = ("../profilePage/editProfile.php");
27                          </script>';
28          }
29      }
```

Figure 4.37 – *JavaScript Code (Edit Profile)*

JavaScript code was mainly implemented to alert and notify a user about changes. In the image above, the user is alerted upon updating their profile. If the profile was updated successfully, the code executes and a prompt “Profile has been Updated!” will appear on their browser, notifying them the update was a success. Otherwise, a prompt “Profile Update Failed!” appears, notifying the user the process has failed. Regardless of success or failure, the system will redirect them back to the edit profile page, as can be seen in line 22 and line 26, using window.location.

```
38          if (in_array($imageExt, $allowedExt)) {
39              if ($imageSize < 5000000) {
40                  if (mysqli_query($connection, $updateDetailsQuery)) {
41                      if (mysqli_query($connection, $updateImageQuery)) {
42                          unlink($deleteImagePath);
43                          move_uploaded_file($tempImageName, $storage);  #Moves image to pictures folder
44                          echo '<script>alert("Product Details and Product Image have been Updated!");
45                              window.location = ("../manageProducts/updatePage.php");
46                              </script>';
47                      } else {
48                          echo '<script>alert("Product Update Failed");
49                              window.location = ("../manageProducts/updatePage.php");
50                              </script>';
51                      }
52                  }
53              } else {
54                  echo '<script type="text/JavaScript">   <!-- Alerts user when file exceeds 5MB-->
55                  alert("That file uploaded exceeds 5MB, please upload another file");
56                      window.location("../manageProducts/updatePage.php");
57                  </script>';
58              }
59          } else {
60              echo '<script type="text/JavaScript">      <!-- Alerts user when file type is not supported-->
61                      alert("That file type is not allowed, please upload another file");
62                      window.location = ("../manageProducts/updatePage.php");
63                      </script>';
64          }
```

Figure 4.38 – *JavaScript Code (Updating Products)*

Similarly, the same application and purpose is used when updating a product. If the product has been successfully updated, the user is alerted with a prompt “Product Details and Product Image have been Updated!” (line 44). The same concept applies in line 49, alerting the user that the update of a product had failed. Line 54 and Line 60 show even more purposeful uses of JavaScript, alerting the user if the image they had uploaded was of a file type not supported, or an image that was exceeding the size limit of 5 Megabytes. All JavaScript code in this instance redirect’s the user back to the Update Page using window.location.

5.0 MAIN SECTION

(Screenshots of webpages)

5.1 Sign Up page

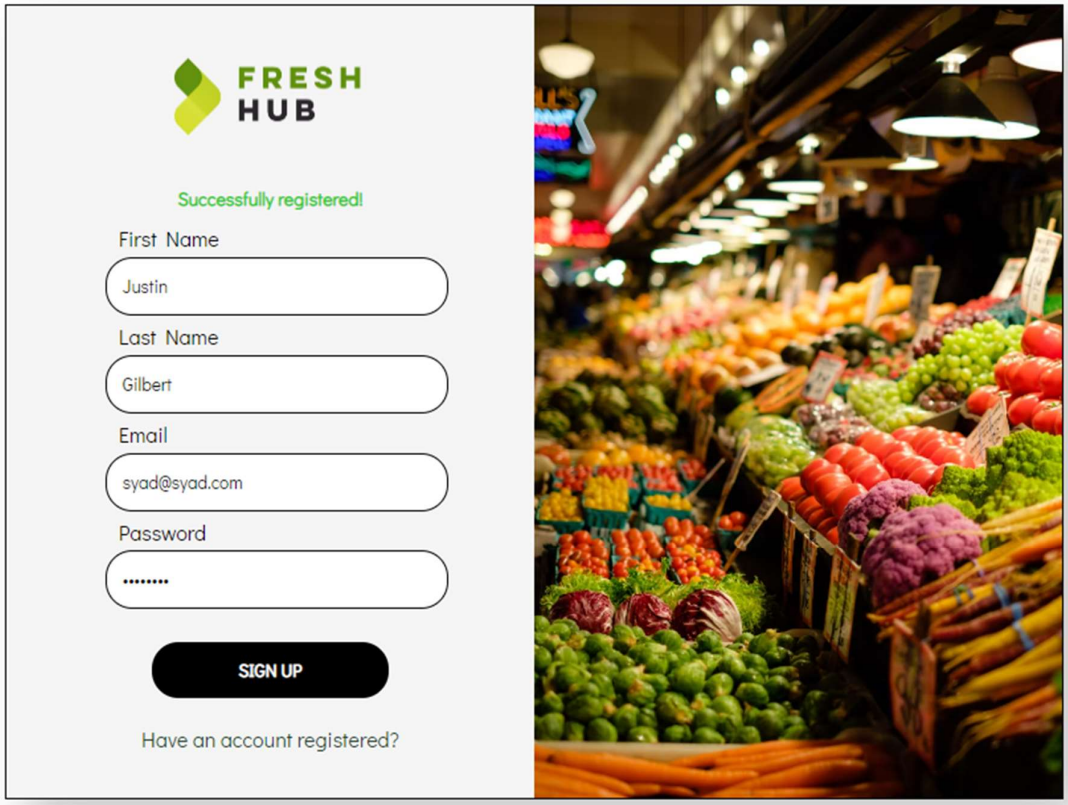


Figure 5.00 – *Sign Up page*

Users are required to register a customer account before logging into the website. Four required credentials are First name, Last name, Email and Password. Upon clicking on signup, multiple validations will be performed, such as determining if the user's email is already in use. If there are no errors, a green "Successfully registered" message will be displayed upon clicking the sign-up button and the user will shortly be redirected to the homepage with their brand-new account logged in.

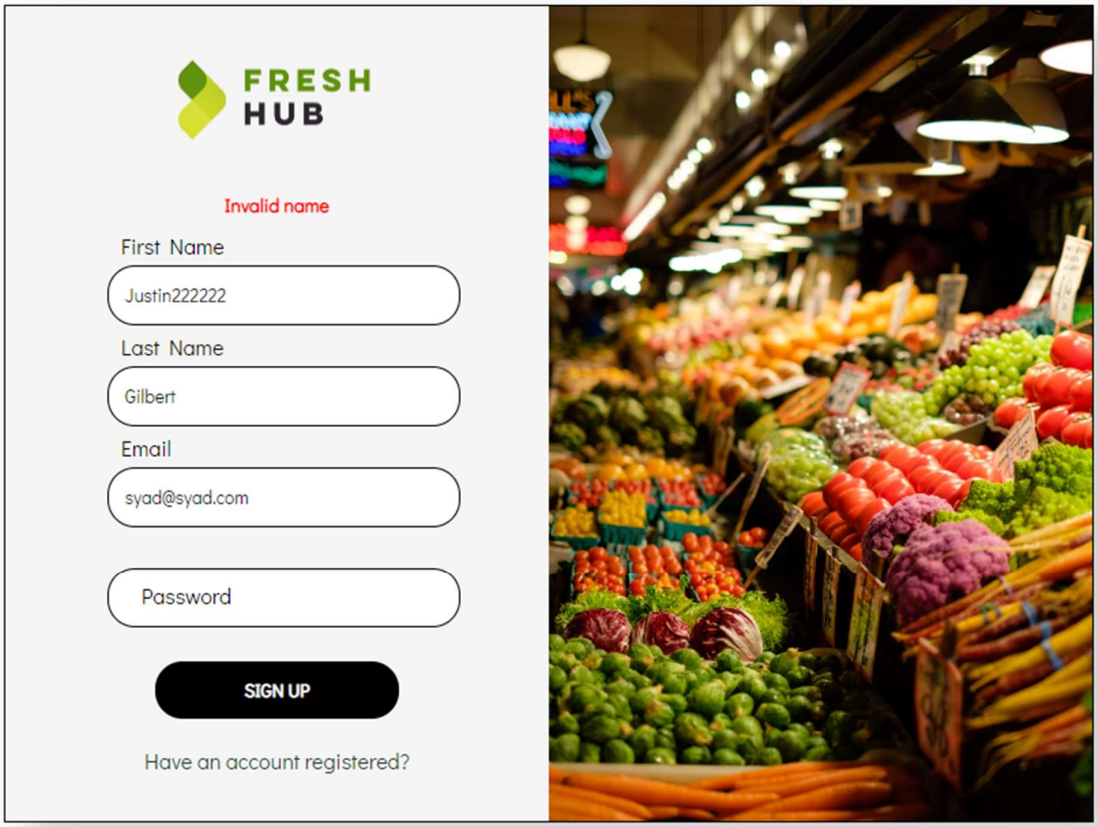


Figure 5.01 – *Sign up error*

Figure 5.01 shows the "Invalid name" error message if the system detects an error with the first name input given by the user.

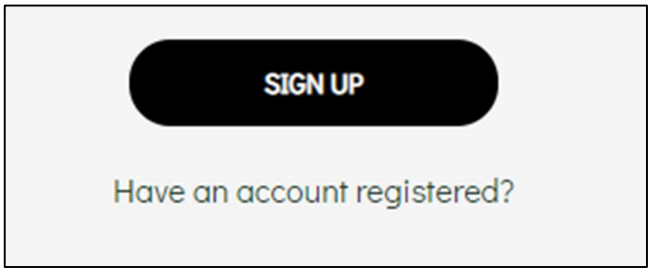


Figure 5.02 – *Sign up button*

If the user already has an account registered, they can optionally click on the "Have an account registered?" link below to be redirected to the login page.

5.2 Login page

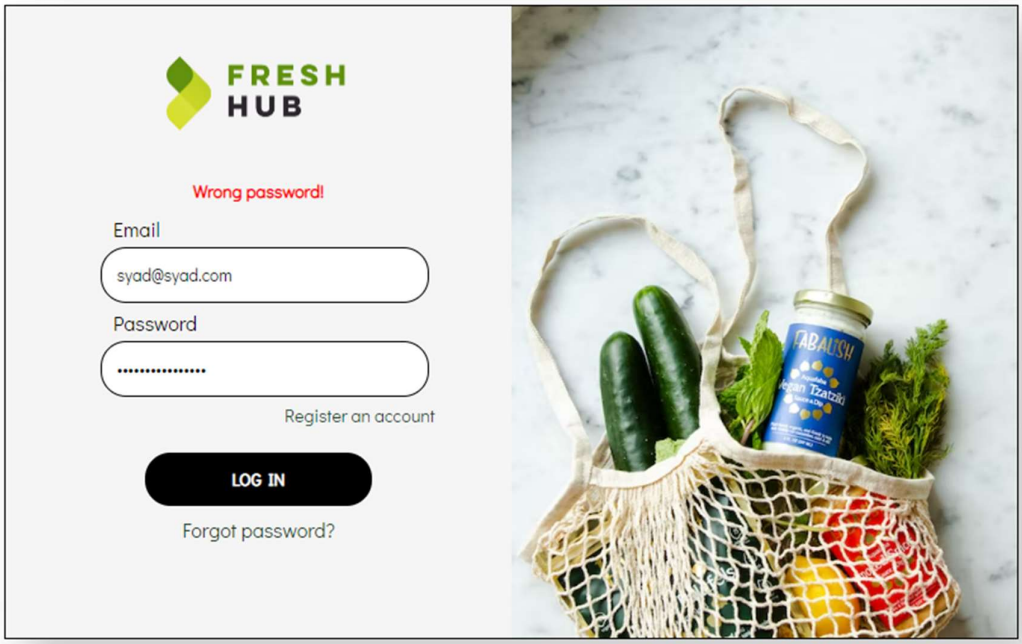


Figure 5.03 – *Login error*

Users are required to log in to access their customer or admin accounts. The two required credentials are email and password. If the credentials do not match to what is stored in the database, it will display an error message. **Figure 5.03** shows the error message if the password is incorrectly entered.

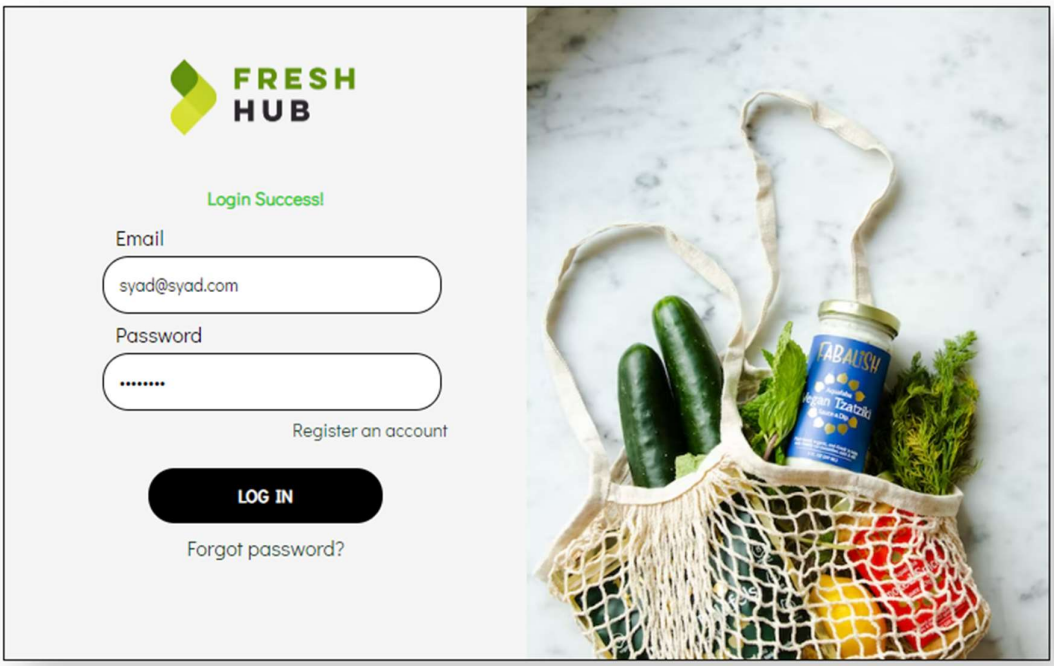


Figure 5.04 – *Login success*

If the credentials entered matches to what is stored in the database, a green “Login Success!” message will be displayed upon clicking the log in button and will shortly redirect the user to the homepage.

Figure 5.05 – *Login button / Forgot Password*

In the case that the user has forgotten their password, they can click the 'forgot password' link below, and they will be redirected to register for a new password.

5.3 Forgot password page

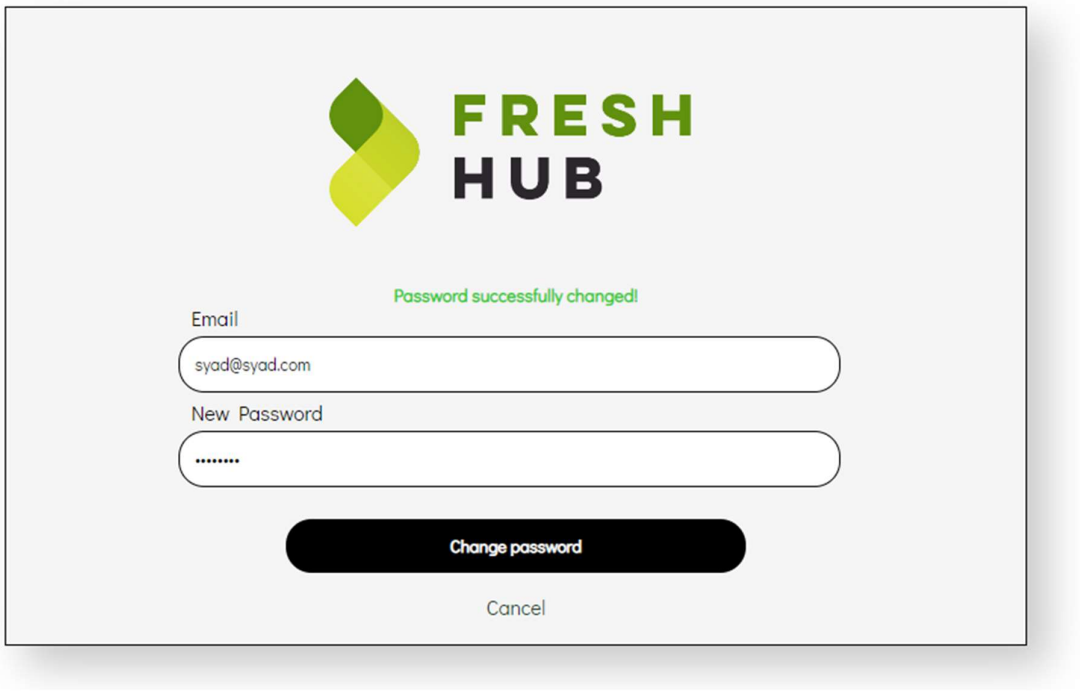
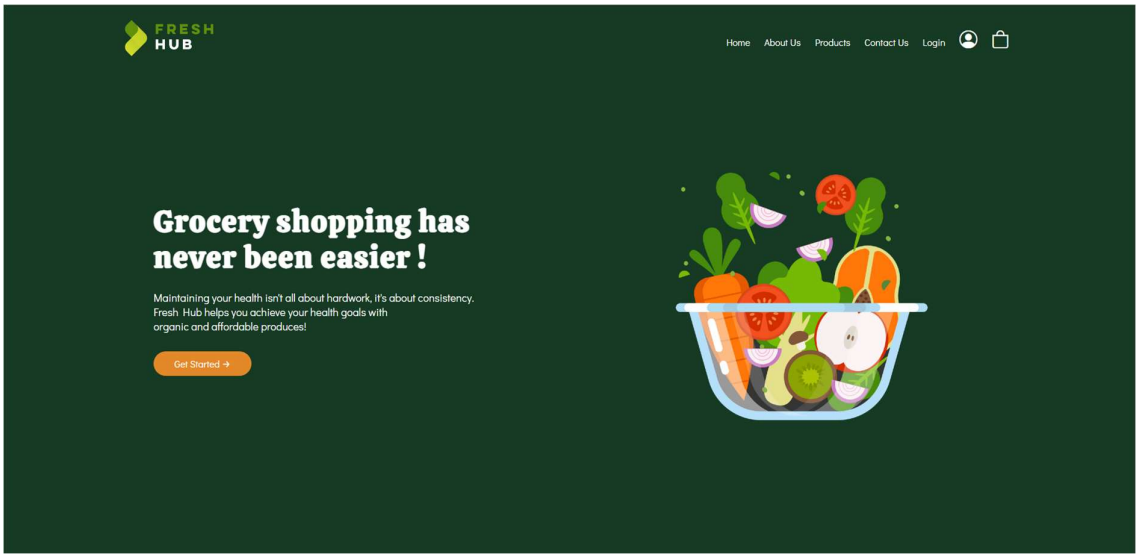
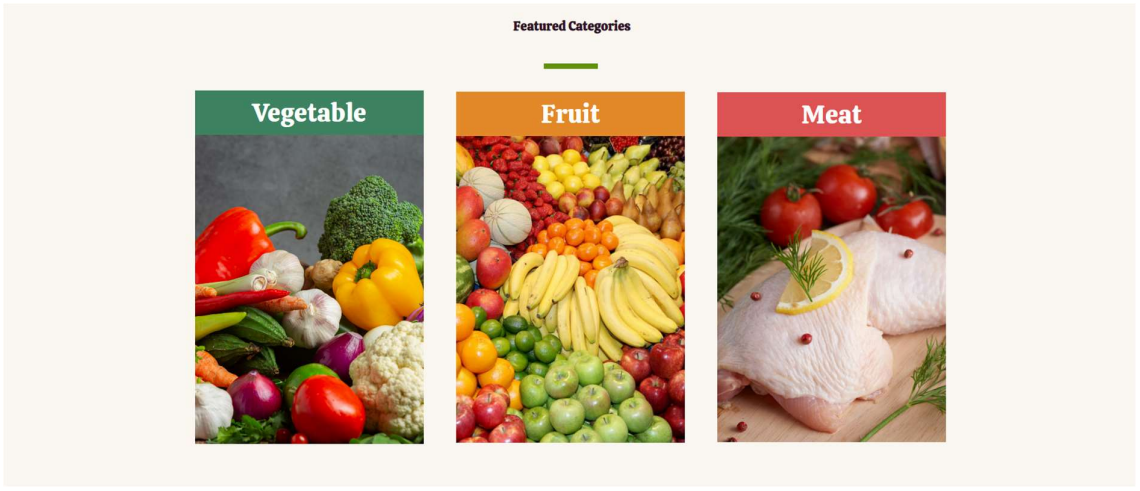


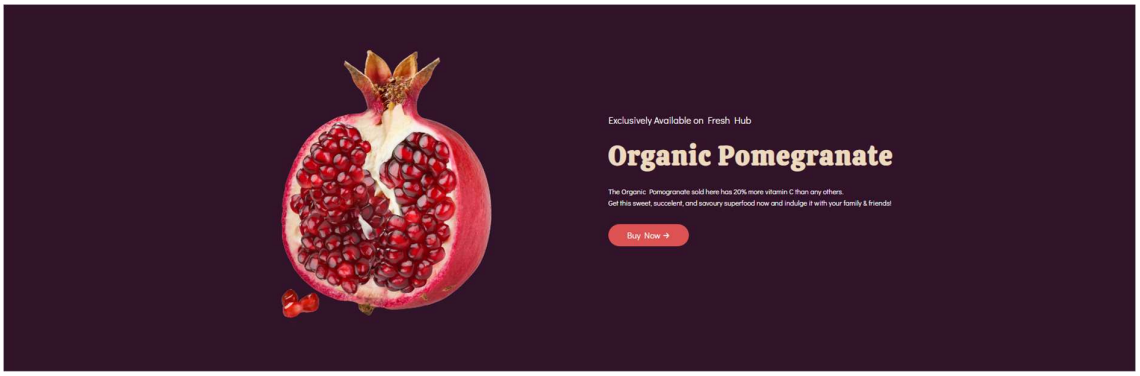
Figure 5.06 – *Forgot Password page*

Users are required to enter their registered email and their preferred new password. If the email exists and the new password entered is more than or equal to 8 characters, a green “Password successfully changed” message will be displayed upon clicking the “Change password” button and the user will be redirected to the login page shortly thereafter.

5.4 Homepage


FRESH
HUB
Home
About Us
Products
Contact Us
Login
Grocery shopping has never been easier !
Maintaining your health isn't all about hardwork, it's about consistency. Fresh Hub helps you achieve your health goals with organic and affordable produces!
Get Started →


Featured Categories
Vegetable
Fruit
Meat


Exclusively Available on Fresh Hub
Organic Pomegranate
The Organic Pomegranate sold here has 20% more vitamin C than any others.
Get this sweet, succelent, and savoury superfood now and indulge it with your family & friends!
Buy Now →

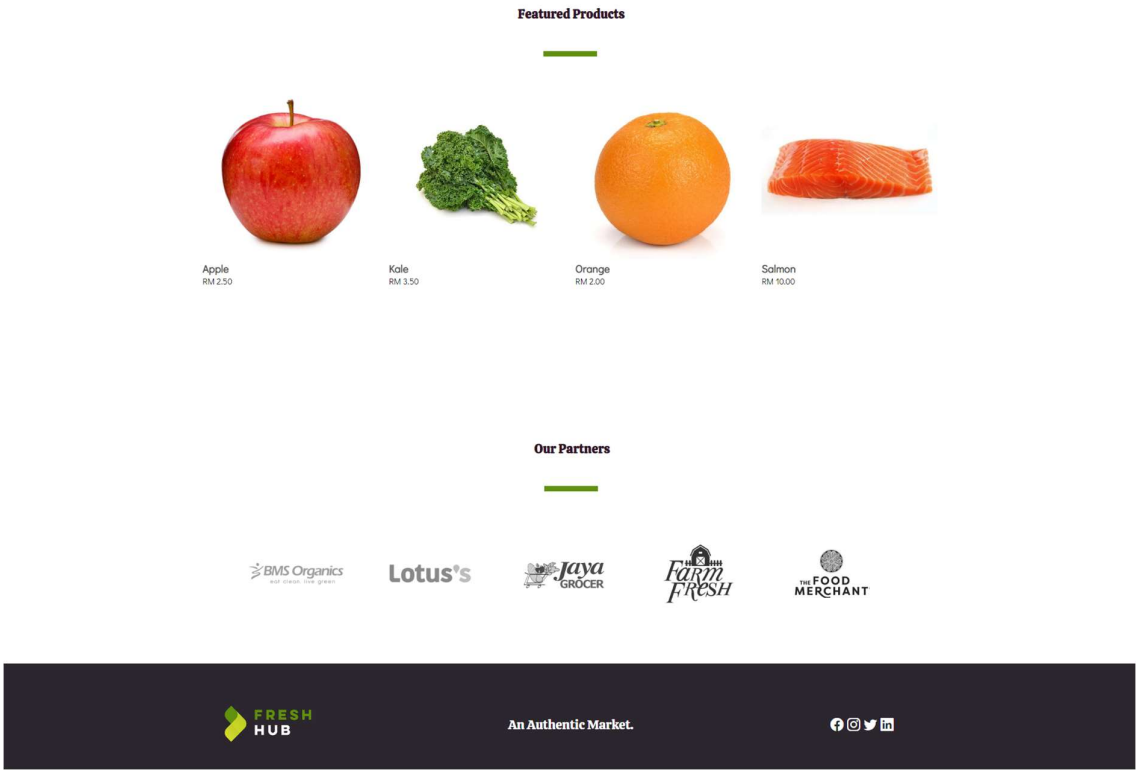


Figure 5.07 – *Homepage*

The Homepage features a header with our company Logo, and a couple navigation bars on the right side. There is a giant thumbnail of a fresh salad bowl with a Slogan attached to the left. A tiny description can be seen and a call-to-action button is also positioned to facilitate interaction. After scrolling, the category page appears, which upon clicking the image, brings users to the specific product page of said category. The offers page that follows shows a Organic Pomegranate that is exclusively sold at Fresh Hub. The 'Buy Now' button is coloured in Red to stimulate psychology temptation. The 'Featured Products' page highlights some of our best-selling products, with their respective names and price listed below. Finally, the partner page shows the groceries chains we work with, and the footer shows our motto of being 'An Authentic Market' with our social links and logo at the side.

5.5 About Us

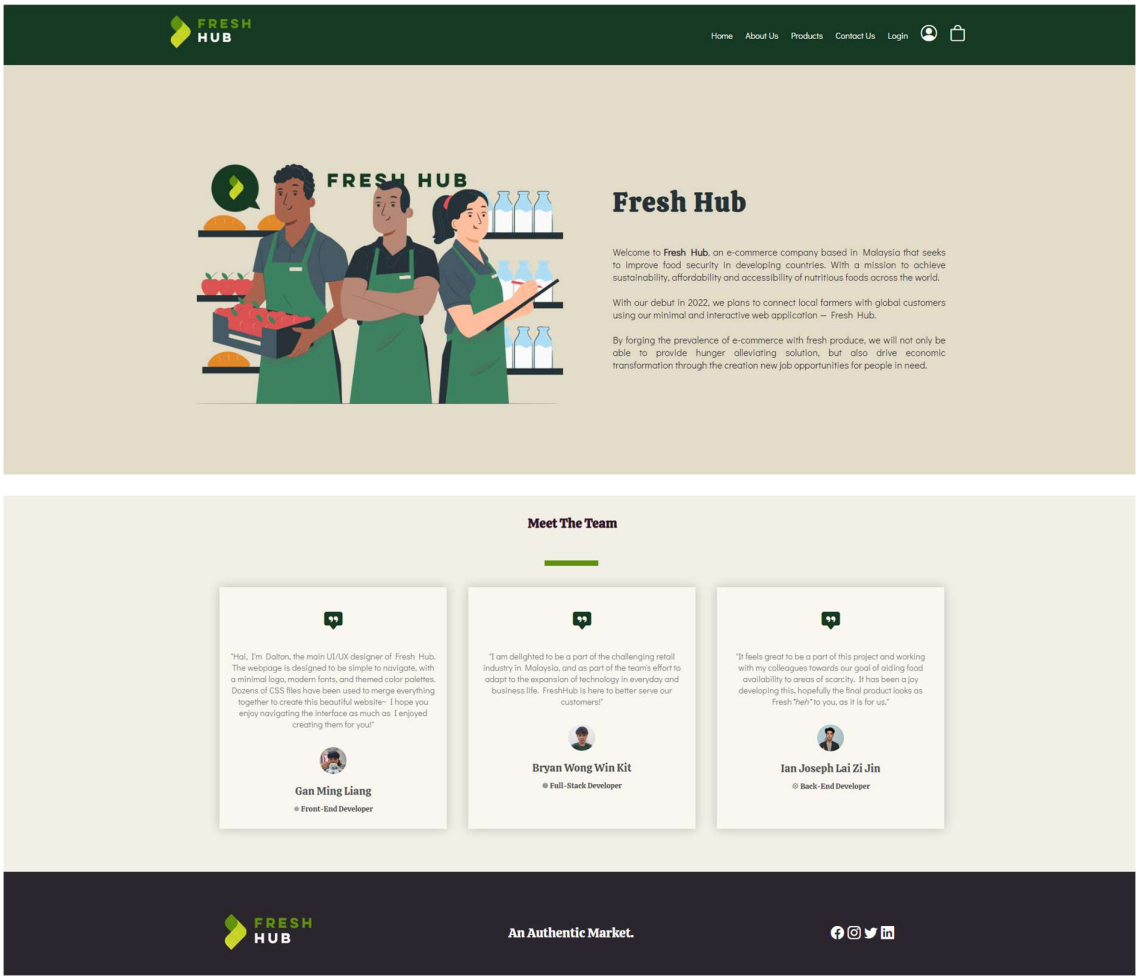


Figure 5.08 – *About Us*

The About Us Page features a company introduction with a beautiful illustration of our grocery store, as well as a short message detailing the history and objective of Fresh Hub. The 'Meet The Team' page also features the 3 dedicated web-developers that have spend countless nights coding the website, with their own profile picture, testimonials, and job title.

5.6 Contact Us

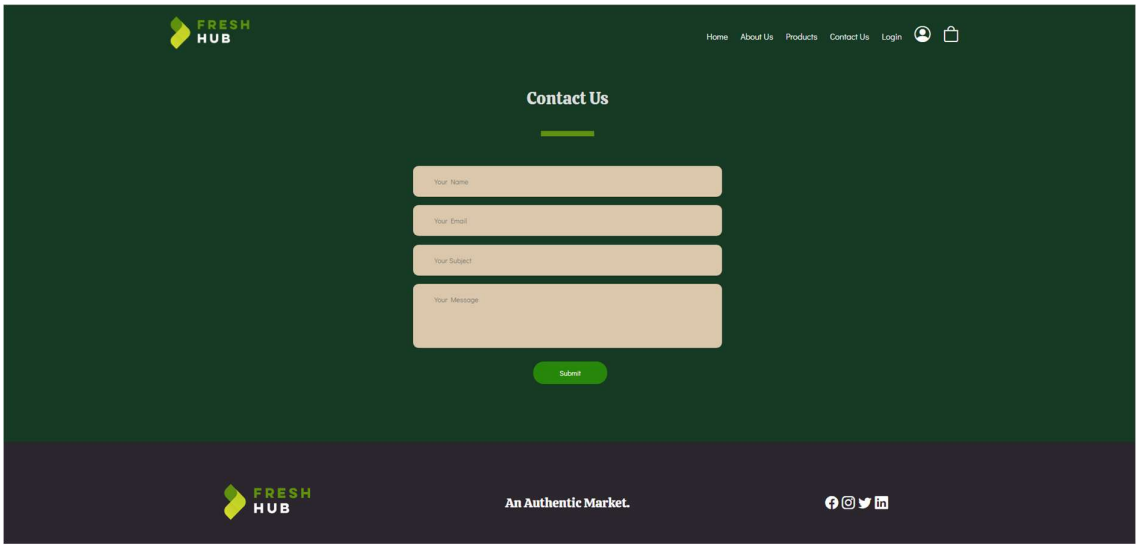


Figure 5.09 – *Contact Us*

The Contact Us page features a minimal HTML form that have been styled using CSS to fit our overall theme. Upon clicking the submit button, a series of validation will be performed to check the fields as well as email format. A success message will be printed at the bottom when it goes through, and user will be redirected back to the homepage.

5.7 Edit Profile Page

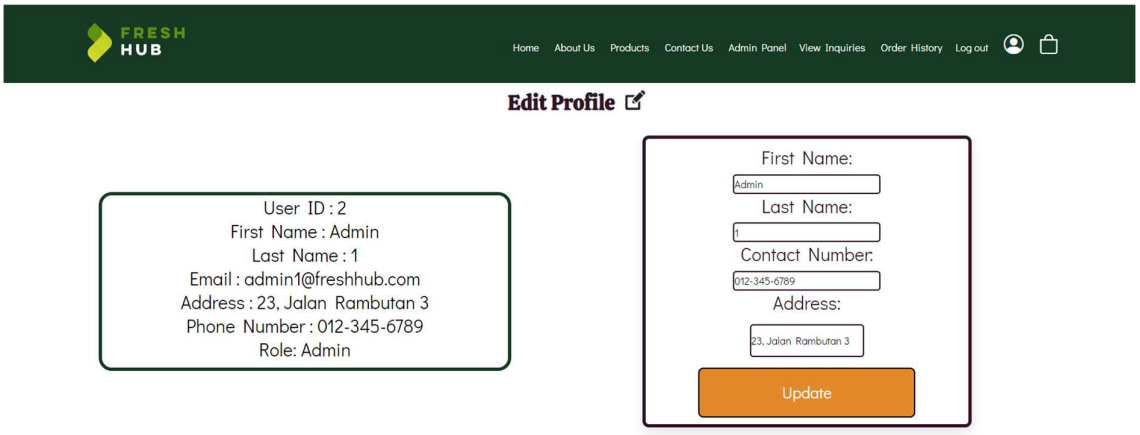


Figure 5.10 – *Edit Profile Page*

In this page, an interface displaying the account's current profile settings are displayed on the right. The box displays information containing

- User ID
- First Name
- Last Name
- Email
- Address
- Phone Number
- Role

Users can then choose to update their profile / account information by inputting information into the form on the right. Current profile information is already pre-loaded into the form for user convenience, the user only needs to change information they wish to change. A User is only allowed to change their First Name, Last Name, Contact Number, and default Address.

5.8 View Products Page

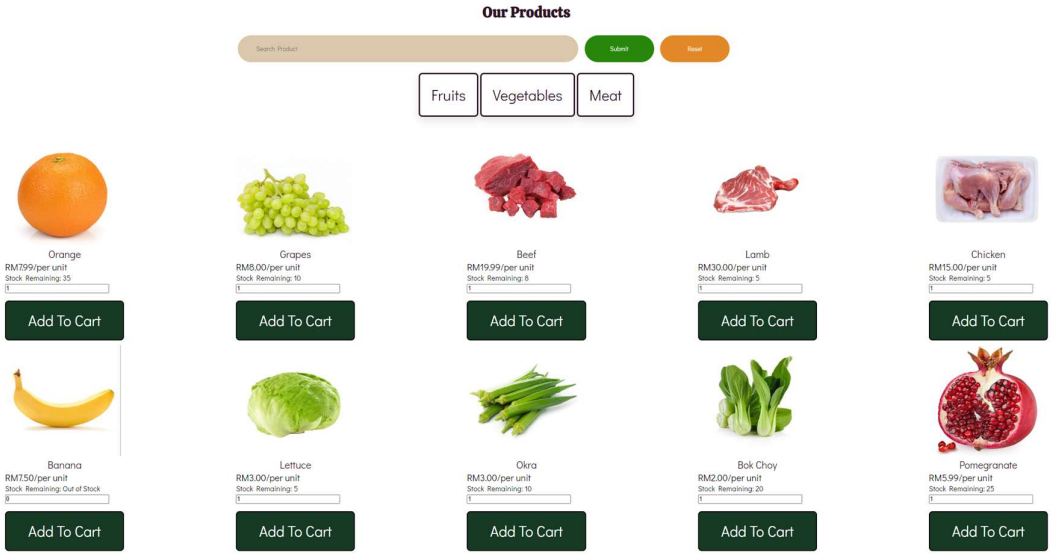


Figure 5.11 – *Main Products Page*

The Main Products Page displays all products available in the system, and displays them rows of five. A search bar is present at the top of the page, allowing users to search for a product containing the character they have entered. A reset button is also present if the user wishes to reset the page back as usual to view all products again.

Below the search bar, the users can click on the available buttons, which are hyperlinks to the Product Category Page, displaying only products that are part of that Category.

As for the products themselves, the name of the product is displayed, along with the price per unit and the stock remaining for each product. Finally, a user can simply add the product by clicking on the Add to Cart button below each product.

5.9 View Products (Category) Page

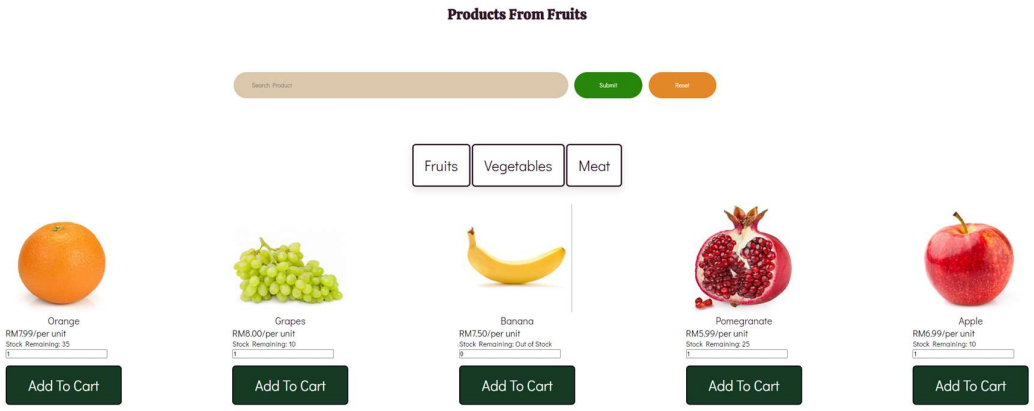


Figure 5.12 – *Product Category Page*

Similarly to the Main Product Page, A search bar and category buttons are present for easy finding of products, allowing users to conveniently navigate the website. The main difference in the Products Category page is of course, only displaying products from a specific category, as well as the main Title, where it displays “Products From”, followed by the Category itself, such as “Fruits”, “Vegetables”, or “Meat”.

All other information regarding the product is still shown, such as the Name, Price per Unit, and Stock remaining. Users can add these products to their cart by clicking on the Add to Cart button below each product.

5.10 Admin Panel Page

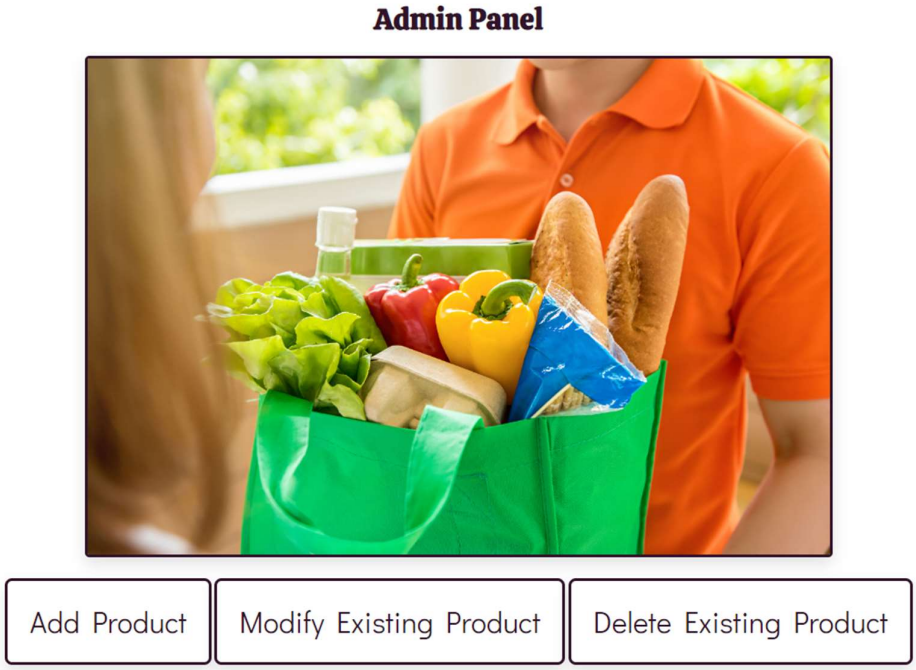


Figure 5.13 – *Admin Panel Page*

In the Admin Panel Page, an image of a man holding groceries is displayed (for decoration purposes). Below the image are buttons of hyperlinks to their respective pages, allowing the user to access the Add Product Page, Modify Existing Product Page, and Delete Existing Product Page.

5.11 Add Product Page

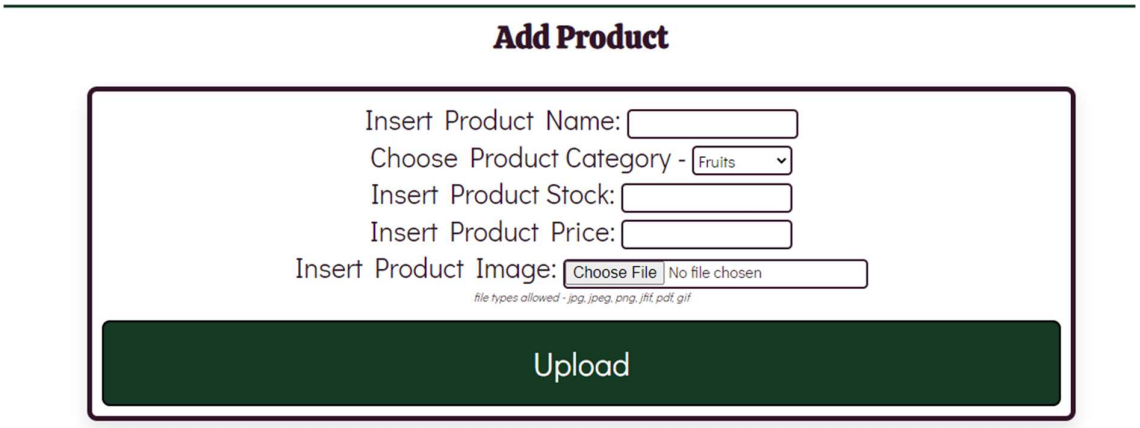


Figure 5.14 – *Add Products Page*

The Add Products Page contains a form where a user can insert information on the product they wish to add to the system. This information includes –

- Product Name
- Product Category
- Product Stock
- Product Price
- Product Image

A drop-down list is shown upon clicking “Fruits”, allowing the user to select between Fruits, Vegetables, and Beef for the product category. The “Choose File” button allows users to upload an image from their computer. Once all information has been inserted, the user can click on the Upload button at the bottom to submit the information, adding the product to the system.

5.12 Modify Product Page

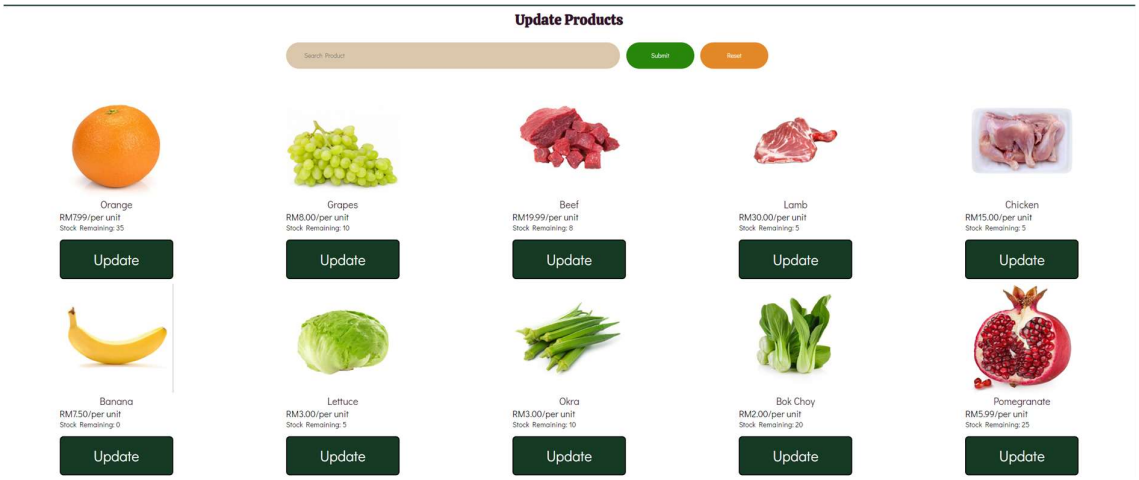


Figure 5.15 – *Main Update Products Page*

The Main Update Products page as a similar interface to the Main Products Page, displaying all products in rows of five, along with the search bar on top to search for a specific product.

Users can modify a product by clicking on the Update button, which will re-direct them to a page specially for updating product details, containing information on the product.

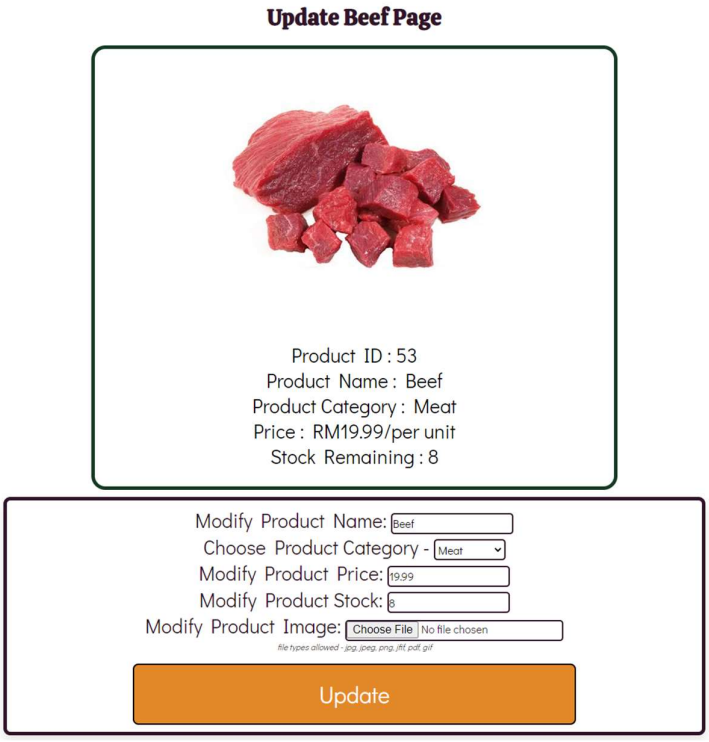


Figure 5.16 – *Update Products Page (Selected Product)*

This page was specially made to display individual product details, which include the

- Product ID
- Product Name
- Product Category
- Price
- Stock Remaining
- Image

of each product. The user can then modify product details in the form below the product details box, allowing users to change the Product Name, Product Price, and Product Stock. The product category can be changed by selecting either Fruits, Vegetables, or Meat through a drop-down menu, similar to the process when adding a product. Changing the image of a file is also the same, where users can click on “Choose File”, and they will be prompted to upload an image from their computer.

Once the desired information is entered, users can click on the Update button present at the bottom of this form.

5.13 Delete Product Page

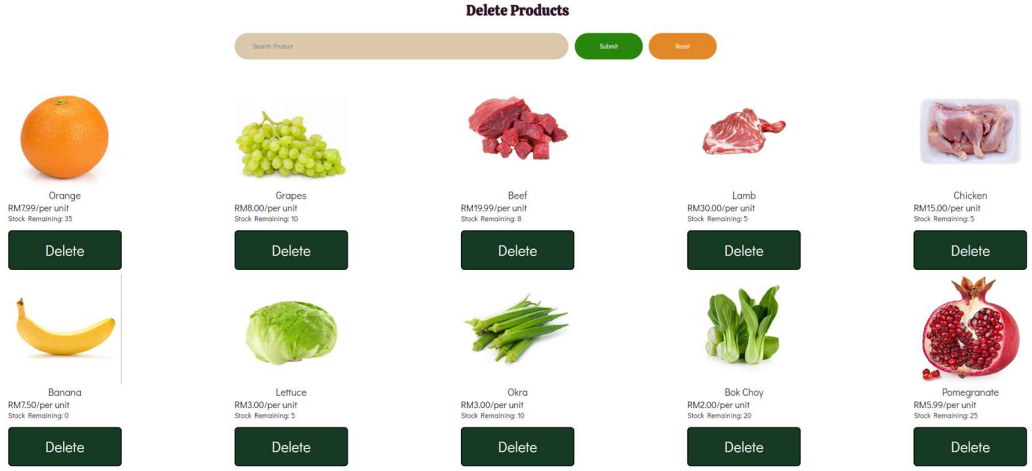


Figure 5.17 – *Delete Product Page*

The Delete Product Page is practically a carbon-copy of the Update Product page, except now the button at the bottom of each product displays “Delete”, instead of “Update”.

All products are still displayed in rows of five, and a search bar is still present at the top of the page to allow users to easily search for a product. Product information such as the image, product name, price per unit, and stock remaining are displayed as well.

The Delete button below each product can be clicked if a user wishes to remove a product from the system.

5.14 Shopping cart page

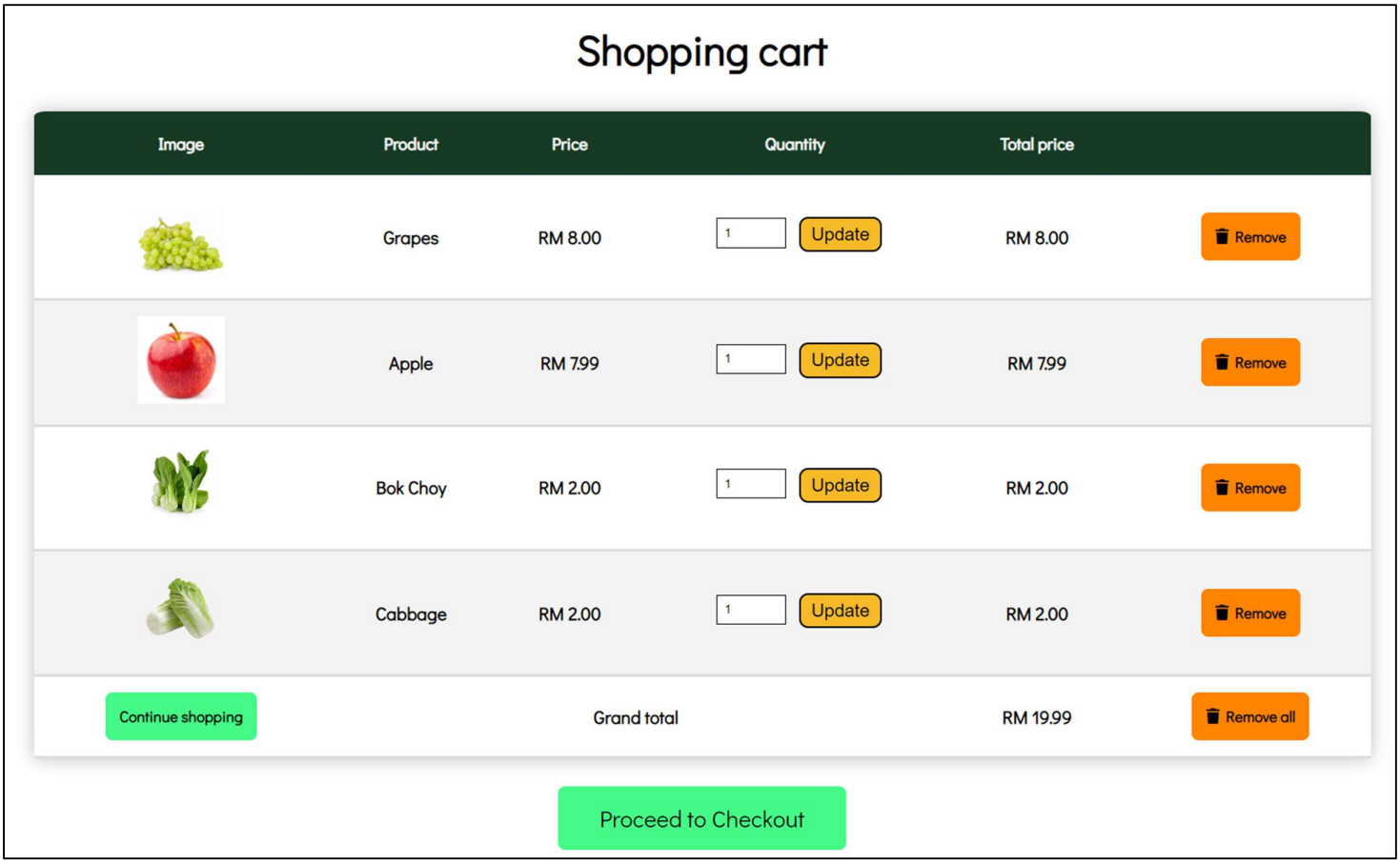


Figure 5.18 – *Shopping Cart page*

Users can access the shopping cart page to view the products that has been successfully added to their cart. Users can change the quantity of the product by entering their preferred quantity and clicking the "Update" button. Once ready, customers can click on the "Proceed to Checkout" button to be redirected to the checkout page. Otherwise, customers can continue browsing the products page by clicking on the "Continue shopping" button.

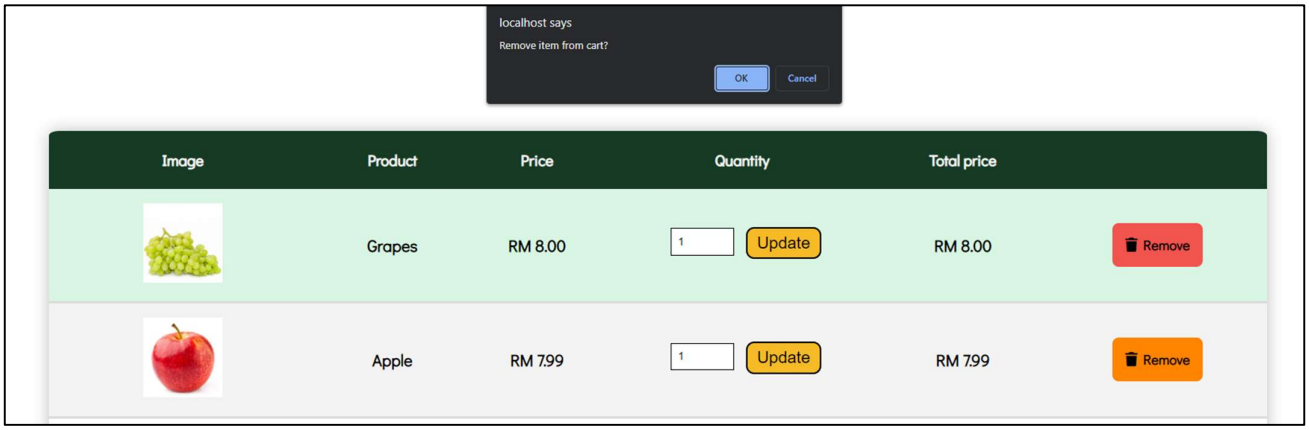


Figure 5.19 – *Remove item*

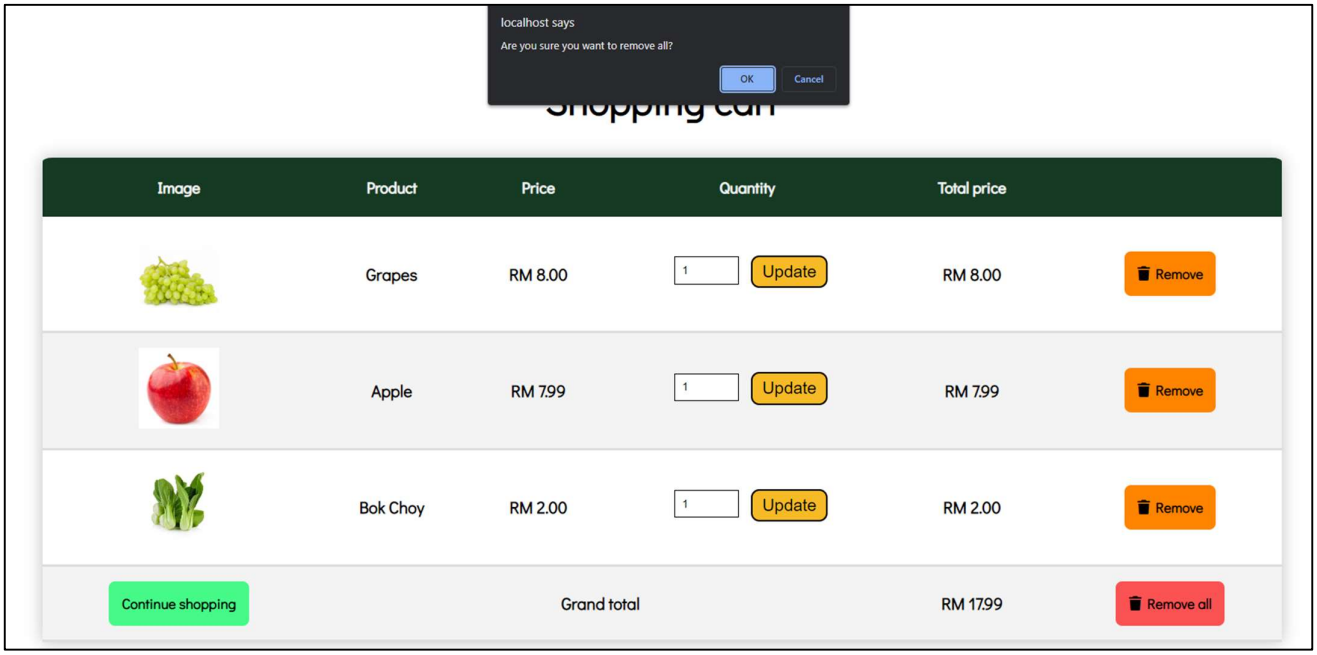


Figure 5.20 – *Remove all items*

Users can also remove products from their cart by clicking on the “Remove” button that has a bin icon beside it. Similarly in **figure 5.20** (second pic), they have the option to remove all items in the cart immediately by clicking on the “Remove all” button. A pop-up alert will appear and prompts the user for confirmation before removing the product from the cart.

5.15 Checkout page

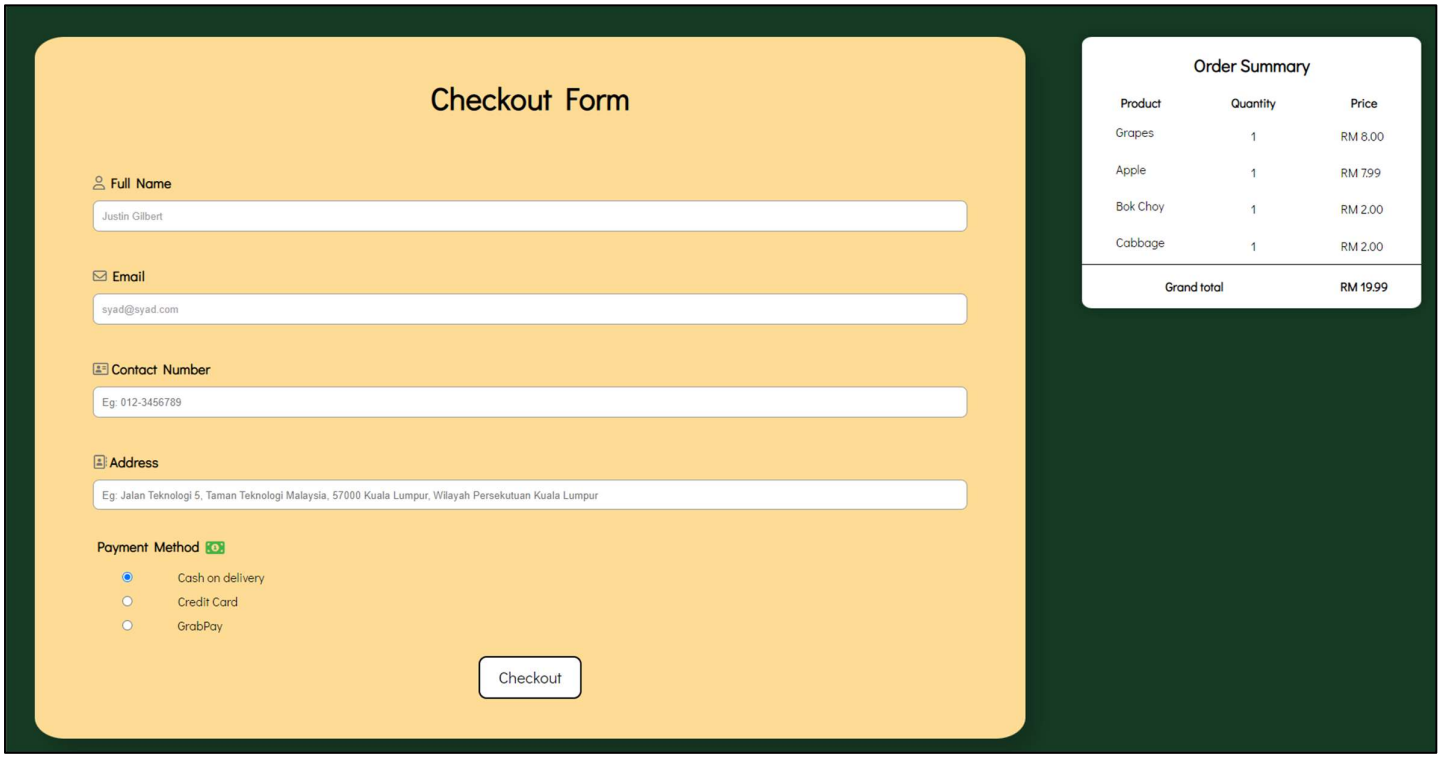


Figure 5.21 – *Checkout page*

The checkout page comprises of an order summary and a checkout form. Users are required to provide their contact number, address, and preferred payment method to checkout. The user's full name and email are filled in automatically and cannot be edited. Upon clicking the "Checkout" button, the user is then redirected to a thank you page.

Figure 5.22 – *Thank you page*

Upon reaching the thank you page, users will be thanked and notified that a confirmation email will be sent as soon as the order ships. Users can optionally click on the "To homepage" button to be redirected back to the homepage of FreshHub.

5.17 View Order History

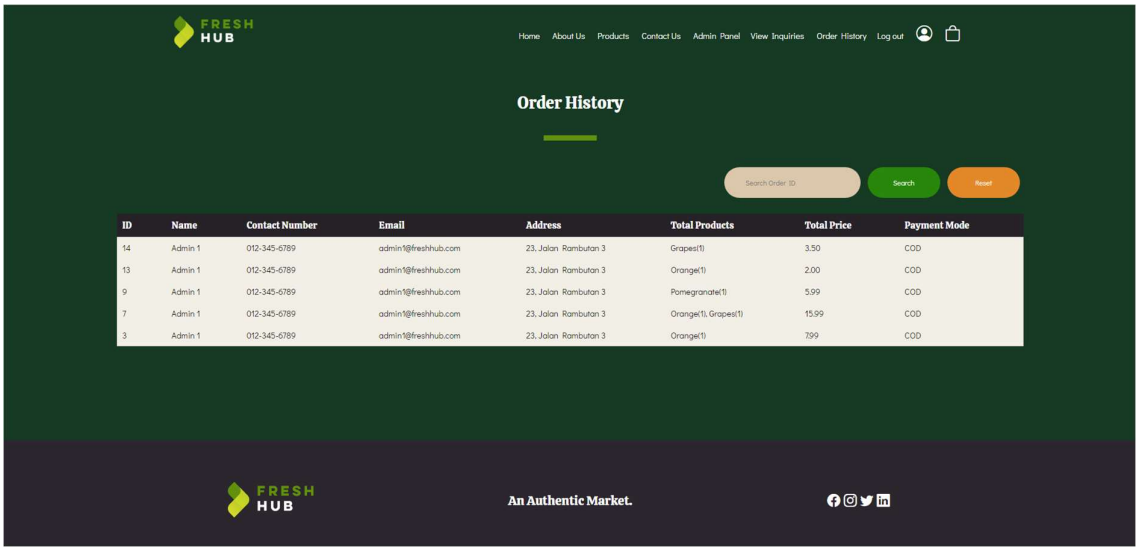


Figure 5.23 – *View Order History*

The view order history page is designed such that it contains a table and a search bar to allow users to see all their past purchase with ease. Users may only search for their own orders in the search tab as searching someone else's' order history will return you 0 results. This is because the SQL query have been set to protect the privacy of users and block unauthorized access to sensitive information.

5.18 View Inquiries

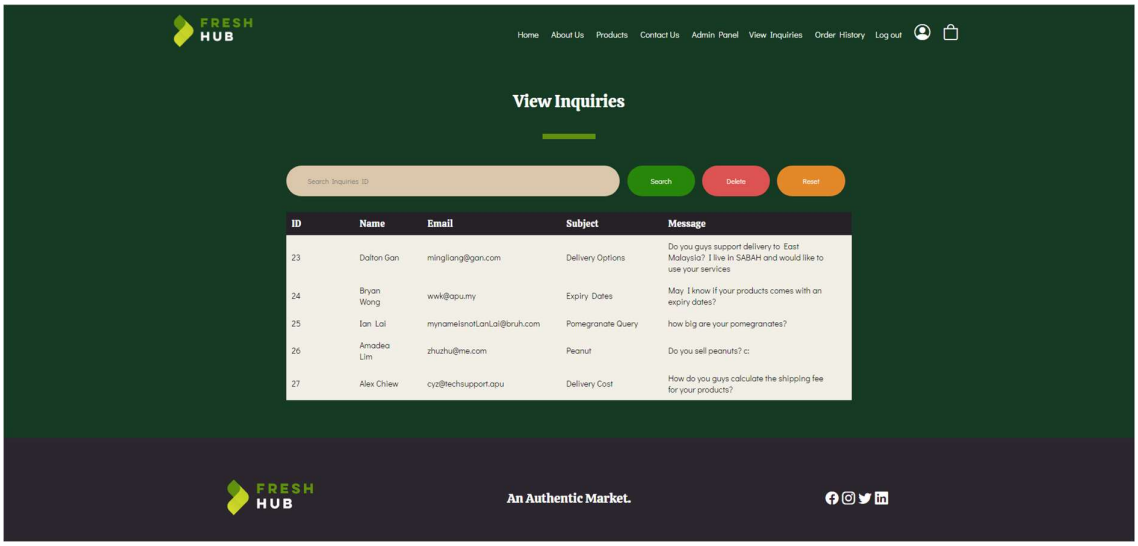


Figure 5.24 – *View Inquiries*

The view inquiries page is exclusive to users with the 'ADMIN' roles only. This page features a GUI that is cute and colourful, yet functionally useful as well. By searching the inquiries ID, Admin can display specific Inquiries based on the inputted results. The inquiries here are sorted in ascending order unlike order history which is sorted in descending order. This is because admin is encouraged to answer old queries as they are on queue for the longest period of time. Once the ticket has been solved, admin can enter the Inquiry ID and click on the DELETE button to delete the ticket.

6.0 CONCLUSION

6.1 Reflection: Dalton

As this was my first time creating a website, I naturally struggled a lot especially at the start. Simplest things such as moving elements on a HTML webpage using CSS was tough as it all had to be written in codes. Instead of finding solutions on Google or using template such as Bootstrap, I forced myself to take baby steps and learn the CSS Box Model from scratch. Understanding the logic behind how margin/padding works and how to apply effects and animation has rekindled my passion in design. I plan to take my newfound interest/hobby of Front-End developing to the next level by learning Angular and React.js after this assignment so I can become a full-time UI/UX designer in my future endeavours.

6.2 Reflection: Bryan

Throughout the course of this project, I have encountered numerous challenges particularly when working with backend code. I have learned that it is essential to have a good sense of logical thinking when approaching a problem. The ability to identify problems, breaking things down and reassembling the pieces together with a solution is necessary to troubleshoot problems and to approach the situation from a different angle.

In addition, I have realized the significance of implementing functions in code. Functions promotes reusability since it can be invoked as many times as necessary in the system. The same function can also be shared and reused by my teammates to avoid code redundancy. I am also able to decompose any problems into smaller chunks by having each function execute a specific task, as opposed to constructing the entire program in one big block of statements.

6.3 Reflection: Ian

During the process of working on this project, I have had many struggles, but a lot of fun too as well. I have learned how to work in a new syntax, bringing in my previously acquired knowledge from Python and working with SQL Databases. However, I have also learned new things as well, such as working with PHP code to insert the data into the database. Working with CSS was also very interesting, changing the code and trying out all sorts of different styles to see what looks best for our website.

After looking back at my current code, I have realised that I could've implemented functions in my code, as there are many repeats of the same code being used in multiple PHP files. The code could also be neater and declared more clearly through better naming conventions. These were realisations I have made upon looking at my groupmate's code and will be something I learn and get used to in future projects.

Speaking of my groupmates, I am extremely grateful to be working alongside Dalton and Bryan, as we have worked extremely hard to finish this project together. It was not as smooth sailing as we thought it would've been, but no great team is without its arguments, debates, and disconnections. I believe that we as a collective, are extremely proud of this project we have made, and will look forward to many more great things we can achieve together.

7.0 REFERENCES

Birch, N. (02 September, 2020). *Typography: Anatomy of a Letterform*. Retrieved from designmodo: https://designmodo.com/letterform/#:~:text=Each%20one%20can%20be%20classified,script%2C%20monospaced%2C%20and%20display.

Chapman, C. (20 May, 2021). *Color Theory for Designers, Part 1: The Meaning of Color*. Retrieved from Smashing Magazine: https://www.smashingmagazine.com/2010/01/color-theory-for-designers-part-1-the-meaning-of-color/#:~:text=Red%3A%20Passion%2C%20Love%2C%20Anger,%3A%20New%20Beginnings%2C%20Abundance%2C%20Nature

Fitzgerald, A. (31 August, 2022). *A Basic Walkthrough of the CSS Box Model*. Retrieved from hubspot: https://blog.hubspot.com/website/css-box-model

Williams, L. (10 September, 2022). *GET Vs. POST: Key Difference Between HTTP Methods*. Retrieved from Guru99: https://www.guru99.com/difference-get-post-http.html