VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

WEB PROGRAMMING (CO3050)

Design and Implementation of a Chocolate E-Commerce Platform

Lecturer: Nguyen Duc Thai, *HCMUT*

Students: Tran Trung Vinh - 2252914

HO CHI MINH CITY, APRIL 2025

Contents

List of Figures

Source code

The entire source code for this assignment can be found on this github repository or go to the next url: https://github.com/Vinzecta/E-commerce

1 Requirement Analysis & Planning

1.1 Requirement Document

1.1.1 Domain Context

In this individual web assignment, I developed a web-based application called "Chocoley", designed for selling a variety of chocolates such as dark chocolate, milk chocolate, truffle chocolate, and more. Chocoley caters to chocolate enthusiasts, offering them a curated selection of high-quality products.

Guests and customers can browse the product catalog and view detailed information about any item they're interested in. However, only registered users are allowed to make purchases and receive their orders via our cash-on-delivery (COD) shipping service.

On the other hand, the seller (also referred to as the admin) has full control over product management — including adding, editing, and deleting chocolate items available for users.

1.1.2 Stakeholders and Needs

- Guests: They can access the list of chocolate products and view the details of each item, but they cannot proceed with a purchase.
- Users: Possesses the same access as guests, with the added ability to complete purchases of chocolate products.
- Seller (Admin): Can add, edit, or delete posted products.

1.1.3 Functional Requirement

Guest

- View the list of products available in the shop.
- View the detail of the product they clicked.
- Cannot make a purchase without sign up or sign in.

User

- Sign up to create an account.
- Sign in if the user has an account.
- Edit account profile.
- View the list of products available in the shop.
- View the detail of the product they clicked.
- Make a purchase for a product.

Seller (Admin)

- Sign in using an account created by the Database Administrator.
- Edit account profile.
- Create a new product.
- Edit products.
- Delete products.
- Create a new category based on a form located in create products section.

1.1.4 Non-Functional Requirement

- The website should be compatible with major browsers (e.g., Chrome, Firefox, Safari) and responsive on various devices, including desktops, tablets, and mobile phones.
- Passwords are securely stored using hash function before importing into the database
- The picture in edit profile section must be JPEG, PNG, GIF formats with the max size of 2 MB.
- The picture in add product section must be JPEG, PNG, GIF formats with the max size of 2 MB.
- The picture in edit product section must be JPEG, PNG, GIF formats with the max size of 2 MB.

1.2 ERD (Entity-Relationship Diagram) for database design

Based on the ERD provided in Figure 1, the users table stores basic information, including email, password, username, phone number, gender, birthdate, profile image, and role. When users access the sign up page to create an account, they are required to fill in the mandatory fields, which include email, username, and password. The remaining user attributes can be filled in later by accessing the edit profile section.

The **categories** table contains only two attributes: category_id and name (for the category name). This table is responsible for storing the categories of chocolate products, enabling the classification of different chocolate types.

The **products** table is responsible for storing basic product information, including name, description, price, category name, and product image. Additionally, the category_id in this table references the category_id in the categories table with a cascade delete. This means that if a category is deleted, any products associated with that category will also be deleted automatically.

The **seller** table contains the seller (admin) ID and the product ID corresponds to the product that they have posted on the website. These two IDs reference other tables: the seller_id connects to the **users** table, and the product_id connects to the **products** table. Both of these relationships have cascade delete functionality, meaning that if an entry in either the **users** or **products** table is deleted, the corresponding entry in the **seller** table will also be deleted.

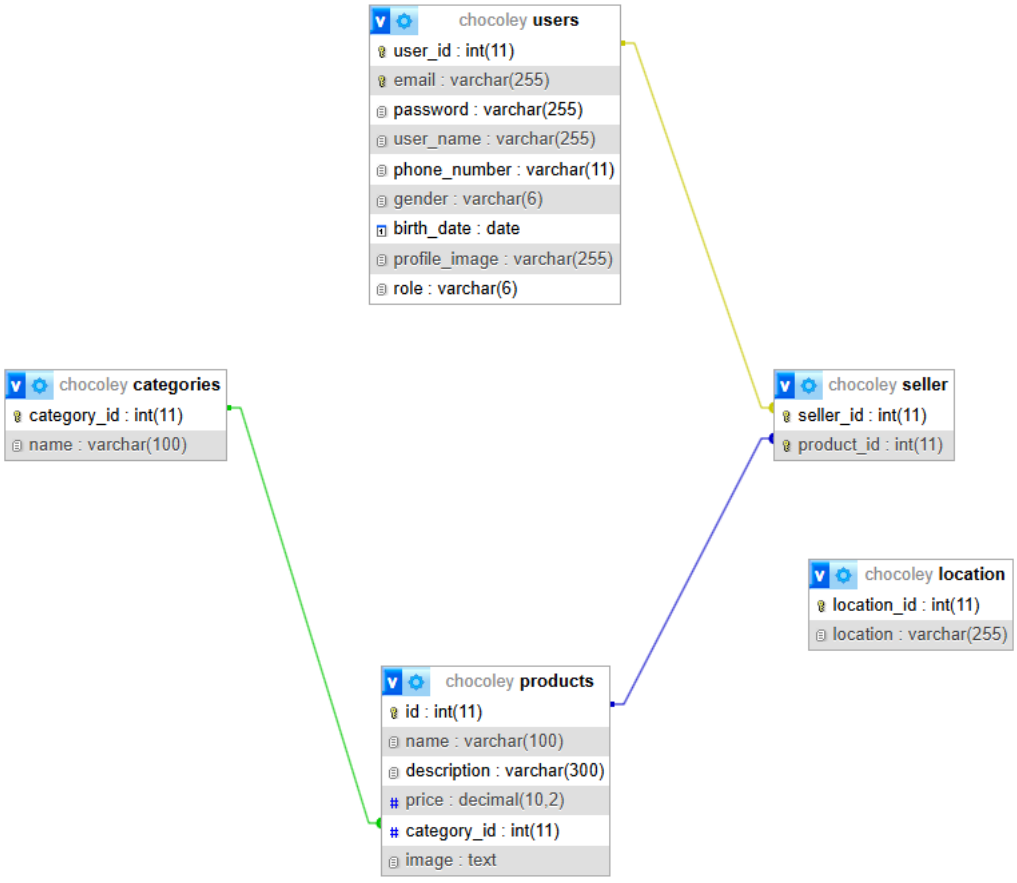

Figure 1: ERD for Chocoley website

2 Chocoley Web Features

2.1 Authentication

2.1.1 Sign Up

To access additional services, users must first create an account—making a sign-up page essential for any e-commerce website. On Chocoley, the sign-up page collects basic user information, including a username, email address, and password. Once the form is completed successfully, the website redirects users to the sign-in page, where they can log in using their registered credentials.

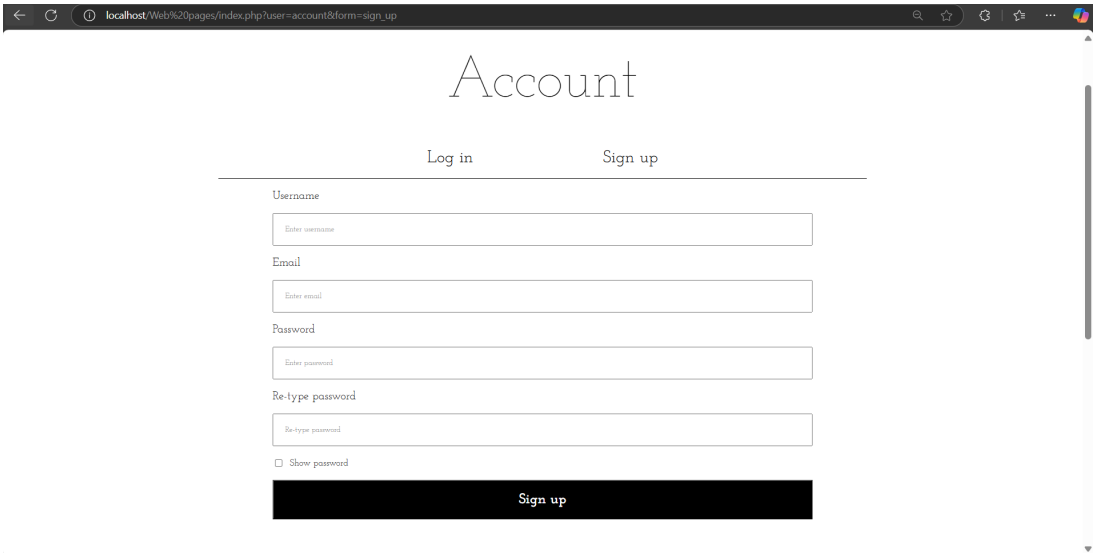

Figure 2: Chocoley sign up page

To prevent unauthorized access, both front-end and back-end validations are implemented on the sign-up page. Specifically, JavaScript is used on the front end to block form submission if required fields are left empty or if the entered information does not meet the website's validation rules.

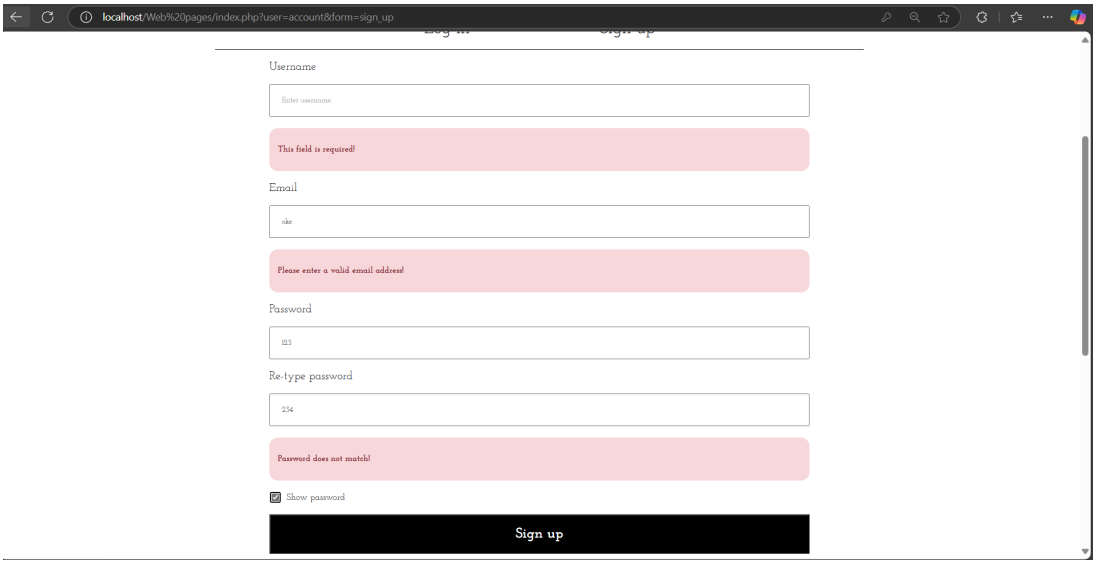

Figure 3: Chocoley sign up validations

On the other hand, when all inputs meet the website's validation rules, the back end will verify whether the email address already exists in the database. If it does, an alert will be displayed prompting the user to enter a different email address.

Figure 4: Chocoley email address shown in **users** table

For example, if a user fills in all fields correctly according to the website's validation rules, but the email address entered already exists (e.g. vinh.trantrung@hcmut.edu.vn) in the system—as shown in Figure 4—an alert will appear, as illustrated in Figure 5, notifying the user that the email is already in use.

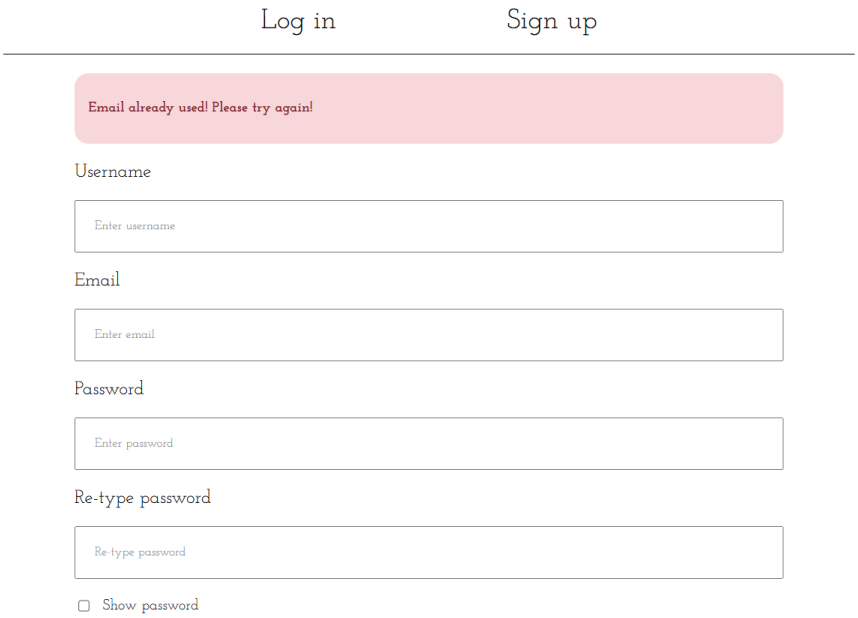

Figure 5: Email already in used alert

When a user successfully completes the sign-up form, the backend inserts all the provided information into the users table, automatically assigning the role as "user". However, for sellers (admins), since the sign-up process is intended for user accounts only, they can access the seller page only by logging in with an account that has the seller role, or access the user page if logged in with a user role.

For example, if a user enters "abc" as the username, "user@gmail.com" as the email, and "123" for both the password and re-typed password, all inputs except for the re-typed password will be inserted into the **users** table.

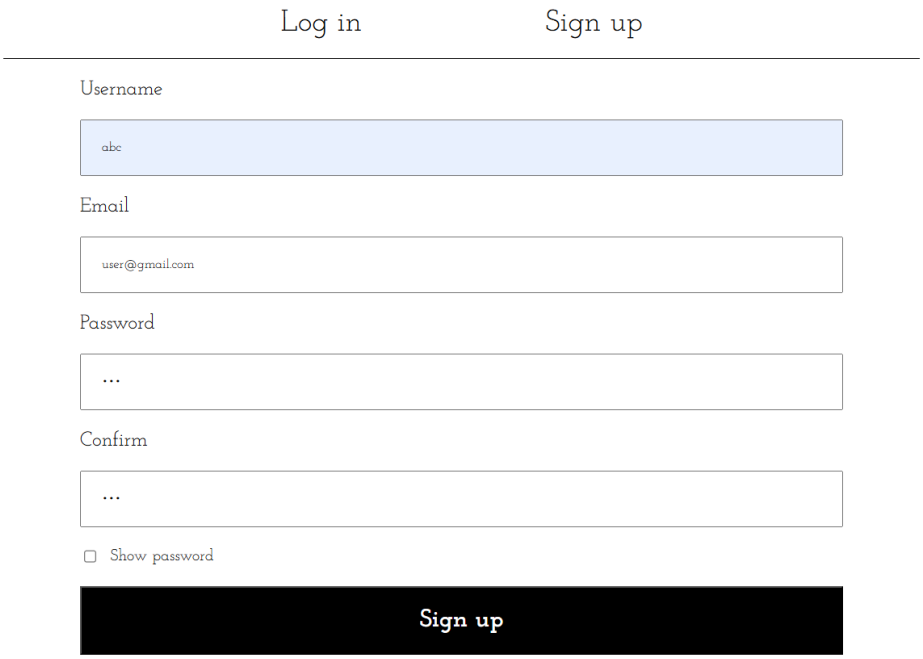

Figure 6: Sign up credential example

However, before being inserted into the database, the password input is processed using **PASSWORD_DEFAULT in PHP**, which applies the strongest available hashing algorithm based on the PHP version. For instance, in PHP 8.0 and above, **PASSWORD_DEFAULT** uses **bcrypt** for hashing. Hashing the password before storing it in the database significantly enhances security, making it much more difficult for attackers to retrieve the original password through techniques such as brute force attacks, compared to storing plain text passwords.

Listing 2.1: PHP code snippet of hashing passwords to store to the database

```
1 $password = password_hash($password, PASSWORD_DEFAULT);
```


Figure 7: **users** table after sign up

2.1.2 Sign In

Once a user either already has an account or completes the sign-up process, they must enter their credentials in the sign-in section to access the website's services.

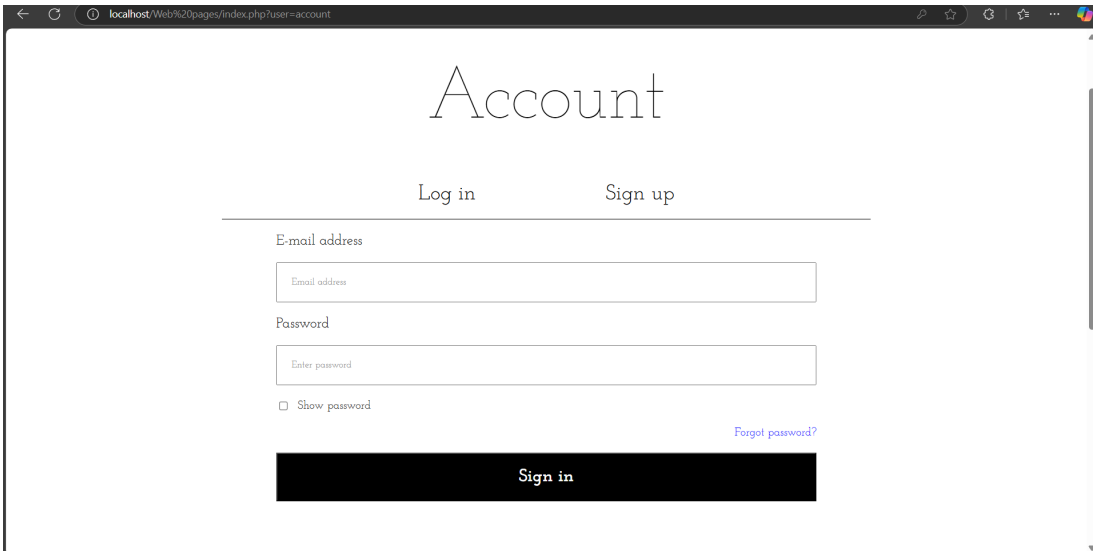

Figure 8: Chocoley sign in section

JavaScript is used to handle client-side input validation. Similar to the sign-up section, the sign-in form prevents submission if any fields are left empty or if any validation alerts are triggered.

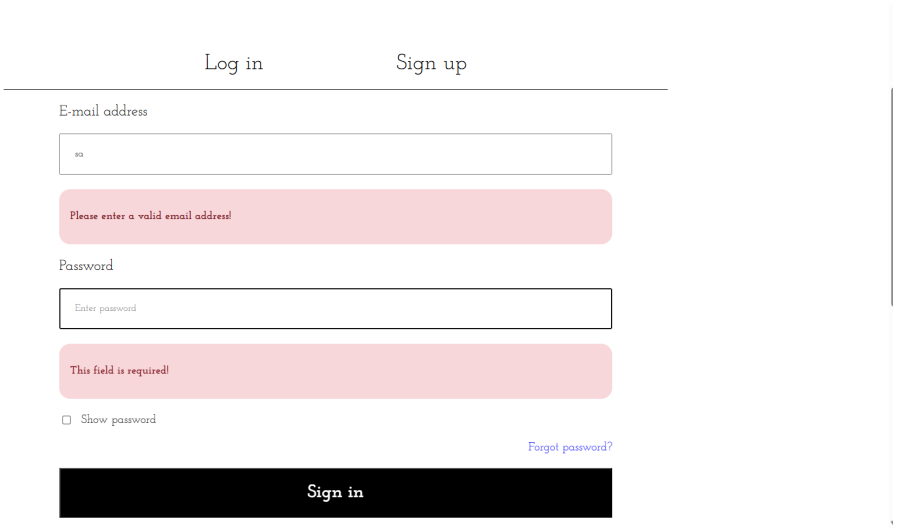

Figure 9: Sign in section client-side validation

When a user successfully enters their credentials and passes the form validation, the backend sign-in process is triggered to compare the provided email and password with the data stored in the database. However, the server cannot directly compare the plain text password with the stored one, as the original password was hashed during the account creation process. Instead of decrypting the stored hash (which is not possible), the server uses a password verification function to check whether the input password matches the hashed version in the database.

If the server successfully matches the input credentials with the data in the database, it creates a session to store the user's basic information, which can be used for dynamic content display on various pages of the website. If the credentials do not match, an error alert is displayed through an error session, indicating an invalid username or password.

Listing 2.2: PHP code snippet of sign in process

```php
    $user_name = "";
    $password = "";

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

        $user_name_sql = "SELECT␣user_id,␣user_name,␣email,␣password,␣role␣,␣profile_image
            ␣FROM␣users␣WHERE␣email␣=␣'$email'";
        $result = mysqli_query($conn, $user_name_sql);

        if (mysqli_num_rows($result) > 0) {
            $account = mysqli_fetch_assoc($result);
            if (password_verify($password, $account['password'])) {
                $_SESSION['username'] = $account['user_name'];
                $_SESSION['email'] = $account['email'];
                $_SESSION['user_id'] = $account['user_id'];
                $_SESSION['image'] = $account['profile_image'];
                $_SESSION['role'] = $account['role'];

                if ($account['role'] == 'seller') {
                    header("Location:␣../Web␣pages/index.php?seller=home");
                    exit();
                } else {
                    header('location:␣../Web␣pages/index.php?user=home');
                    exit();
                }
            } else {
                $_SESSION['error_login'] = '<div␣class="alert">
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<p>Username␣or␣password␣are␣incorrect!␣Please␣try␣again!</p>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</div>';
                header("Location:␣../Web␣pages/index.php?user=account");
                exit();
            }
        } else {
            $_SESSION['error_login'] = '<div␣class="alert">
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<p>Username␣or␣password␣are␣incorrect!␣Please␣try␣
    again!</p>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</div>';
            header("Location:␣../Web␣pages/index.php?user=account");
            exit();
        }
    };
```

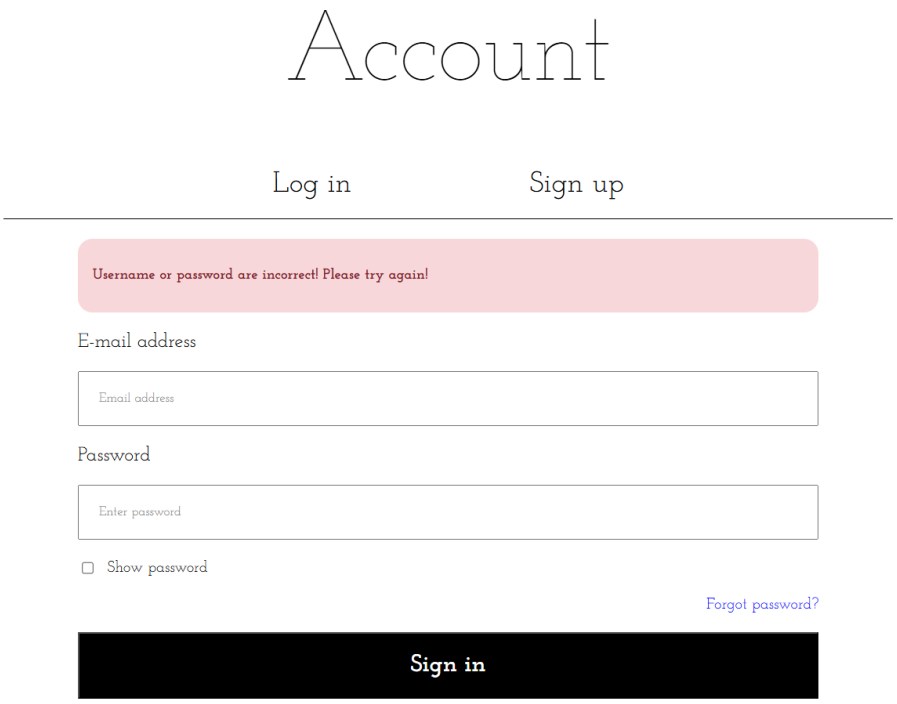

Figure 10: Invalid email or password example

2.1.3 Forgot Password

Registered users who forget their password can click the *Forgot password?* link located in the sign-in section, which redirects them to the **Forgot Password** page.

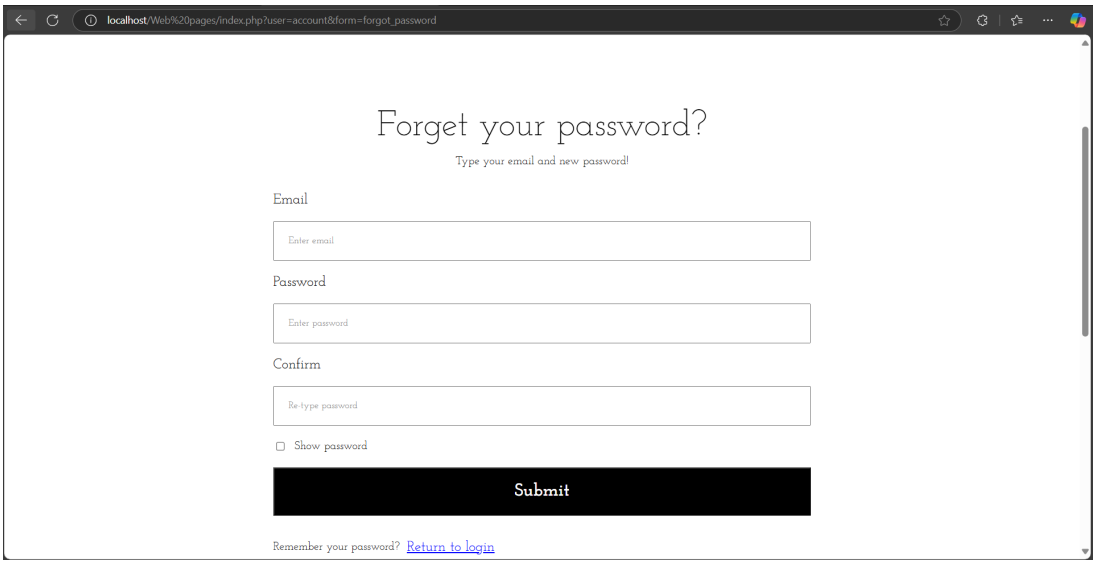

Figure 11: Chocoley forgot password section

Like the other forms mentioned above, the Forgot Password section uses both client-side and server-side validation. This not only guides the user on what input is required but also enhances the security of the website. When the user enters valid input according to client-side validation and submits the form, the server checks whether the provided email exists in the database. If the email is found, the backend updates the password. Otherwise, it stores an error message in the session and displays it to the user.

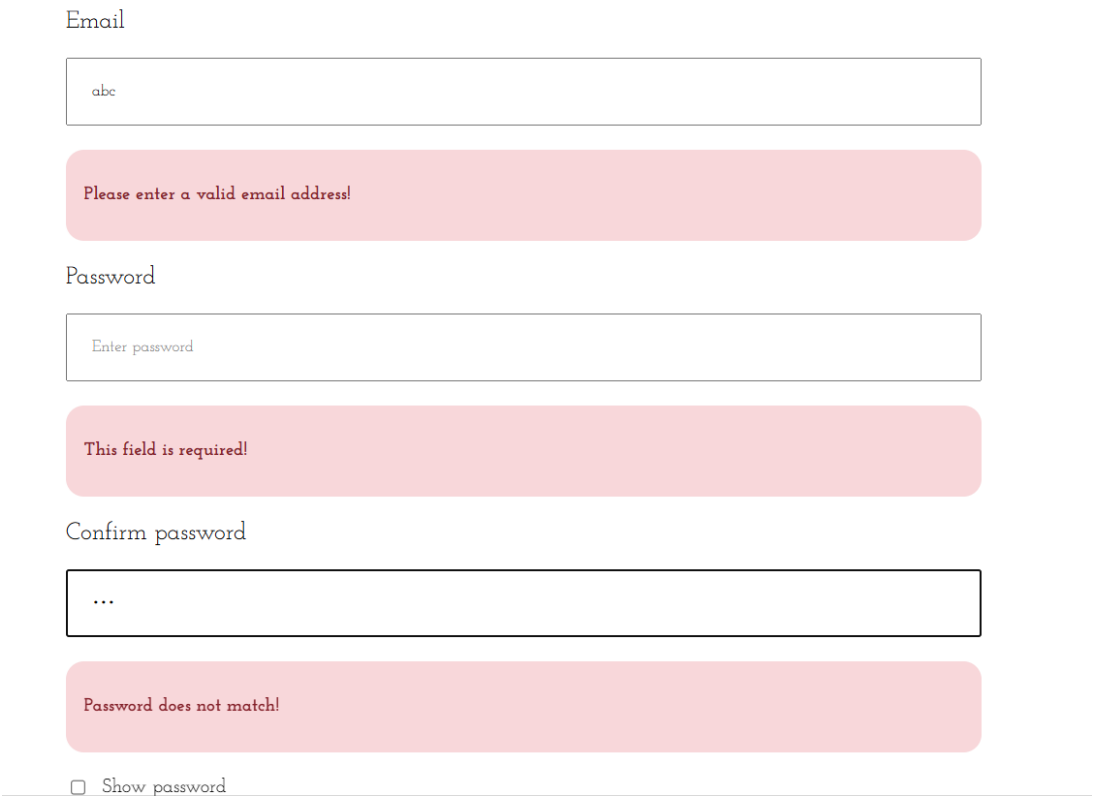

Figure 12: Client-side validation in **Forgot Password** section

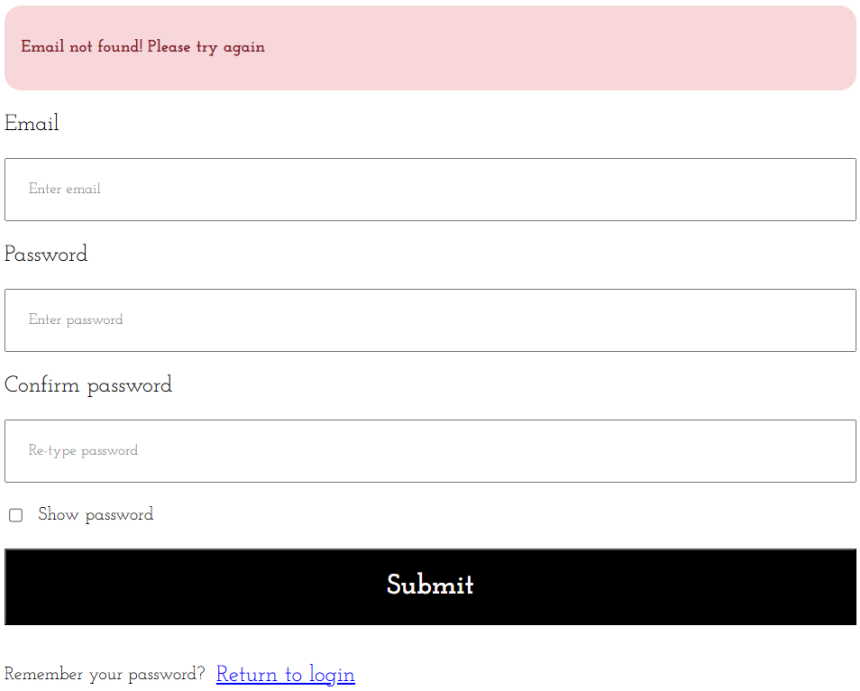

Figure 13: Server-side validation in **Forgot Password** section

When an email exists in the database, the server hashes the new password and replaces the old one associated with that email address. For example, using the registered data from Figure 6 with the email *user@gmail.com* and password *123*, if the user submits a new password *234*, the server will locate *user@gmail.com* in the database and update the

stored password with the hashed version of the new one.

Figure 14: User with old password

Figure 15: User with new password

2.2 Homepage

On e-commerce websites, the homepage is one of the most important pages, providing users with essential information about products, services, contact details, and more. On the Chocoley website, the homepage—like most other pages—follows a three-part layout: a header with navigation links, a main content body, and a footer.

Specifically, for both users and guests, the header includes four main navigation links: **Home**, **Product/Service** (for browsing products), **Contact** (for sending messages to administrators), and **Account** (for viewing profile information).

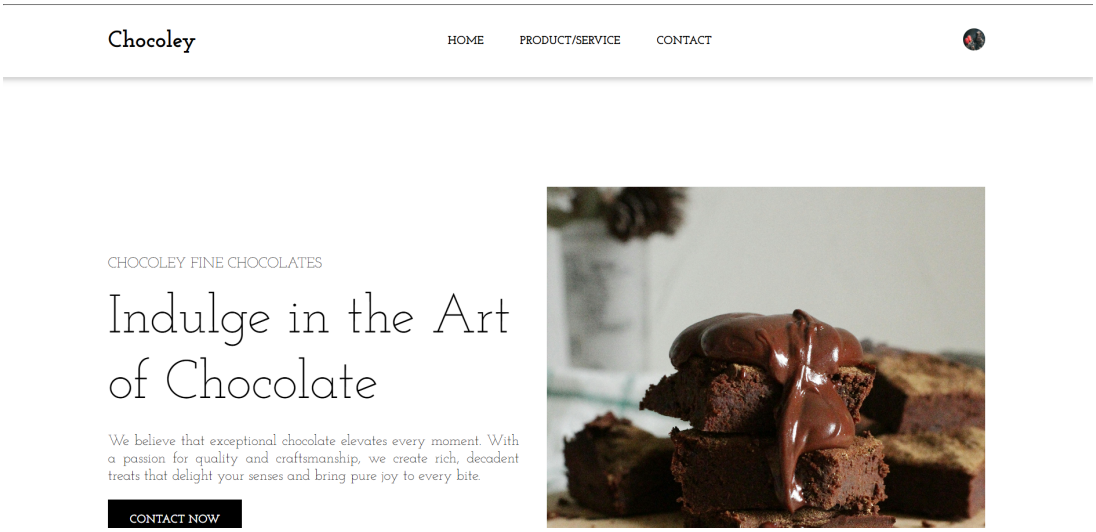

Figure 16: Chocoley homepage

2.3 Product/Service Page

The **Product/Service** pages provide a list of various chocolate products organized into different categories, allowing users to browse a wide selection based on their interests. These pages feature pagination to display a limited number of products per page, sorting options by name and price, and a live search function to help users quickly find their desired products.

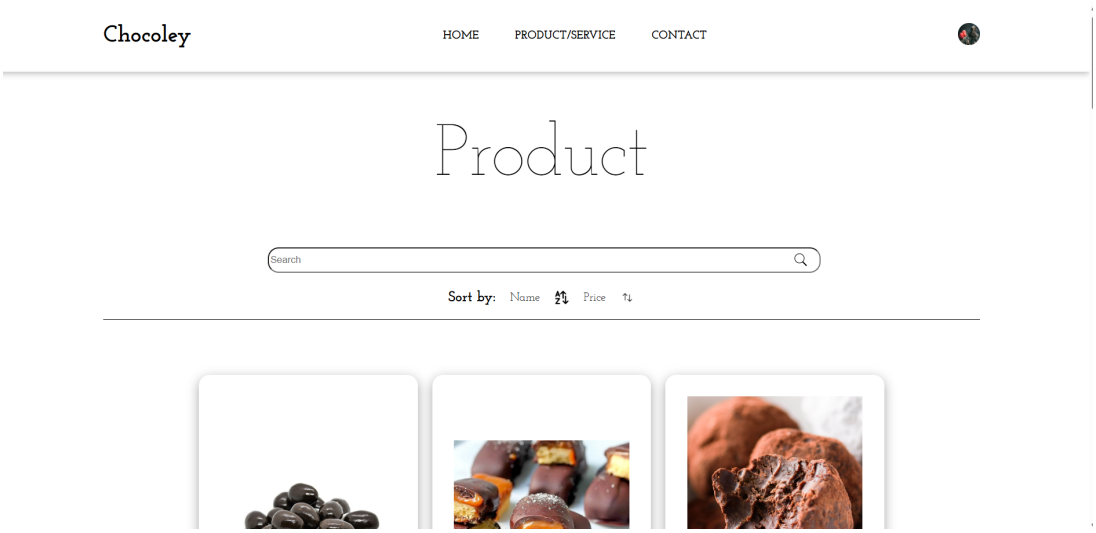

Figure 17: Chocoley Product/Service page

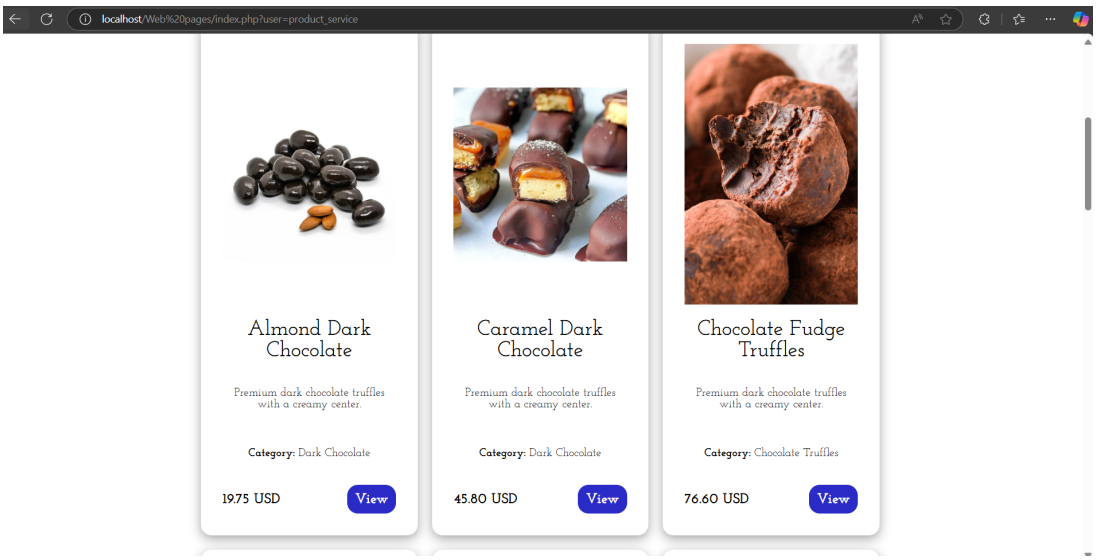

Figure 18: Chocoley Product/Service page 1 (shown in url)

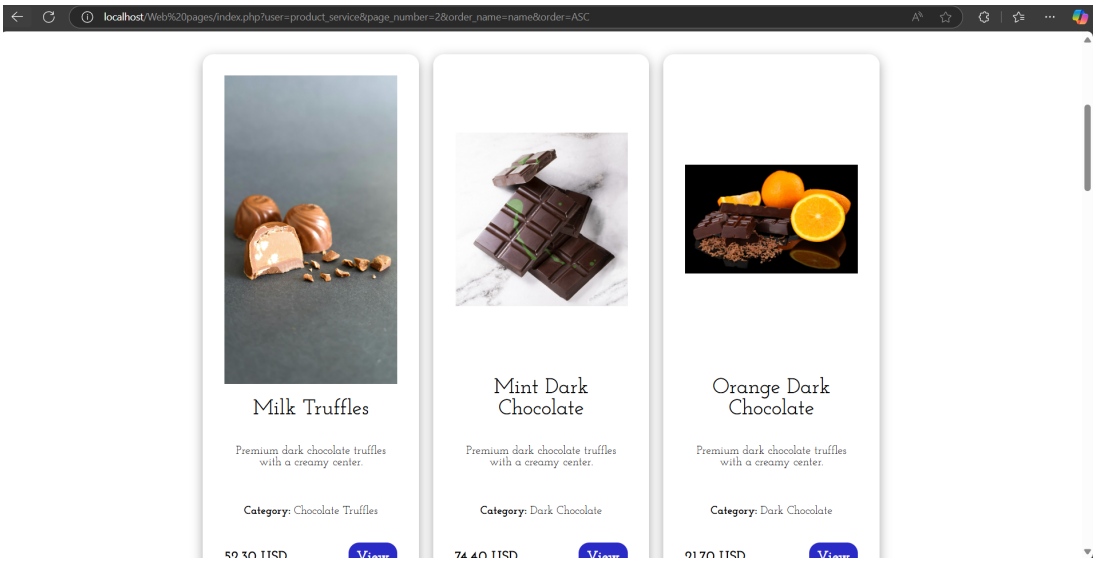

Figure 19: Chocoley Product/Service page 2 (shown in url)

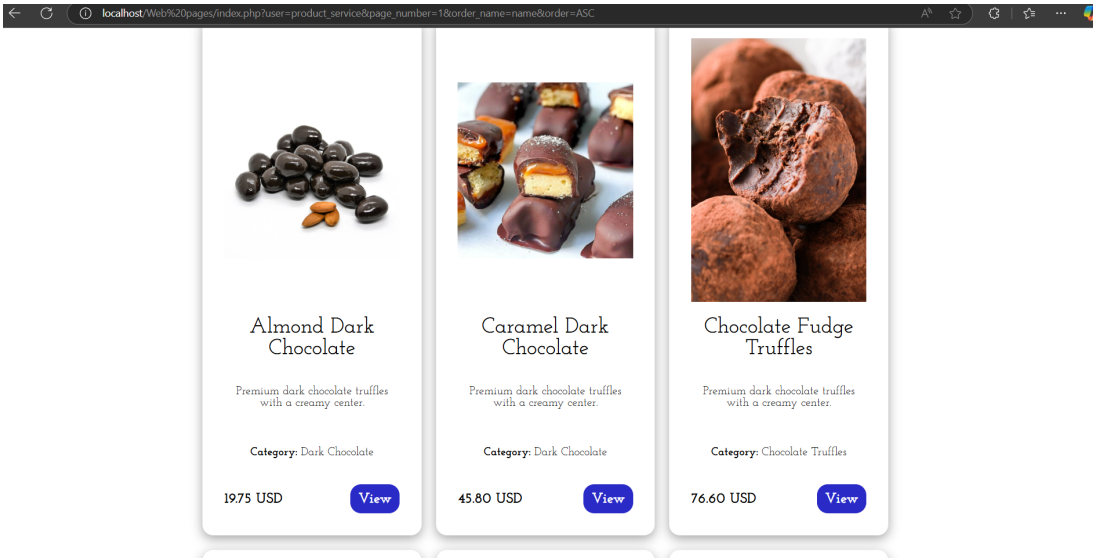

Figure 20: Chocoley Product/Service after sorting name (ascending)

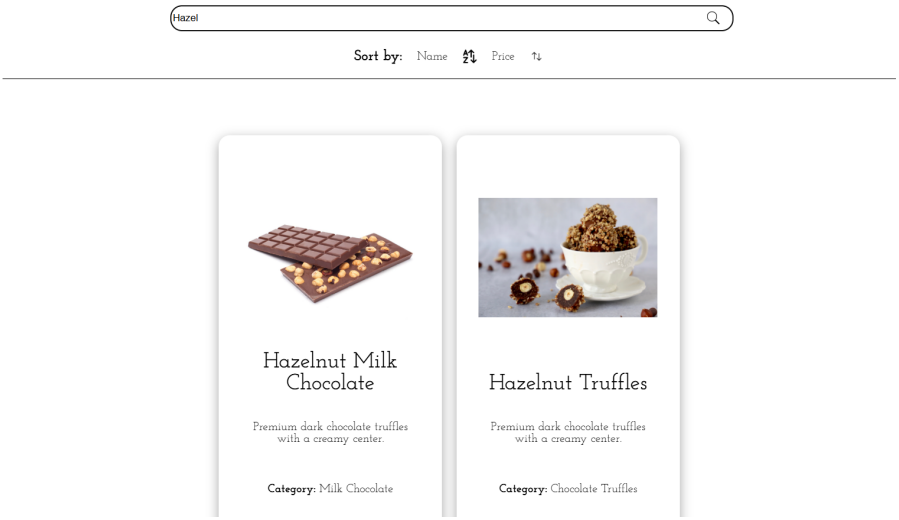

Figure 21: Chocoley Product/Service after searching

2.4 Contact Page

The Contact Us page allows users and visitors to send messages or inquiries directly to the website administrators. It typically includes a form where users can enter their name, email address, subject, and message content. This page ensures smooth communication between the website and its users, whether for customer support, product inquiries, or general feedback.

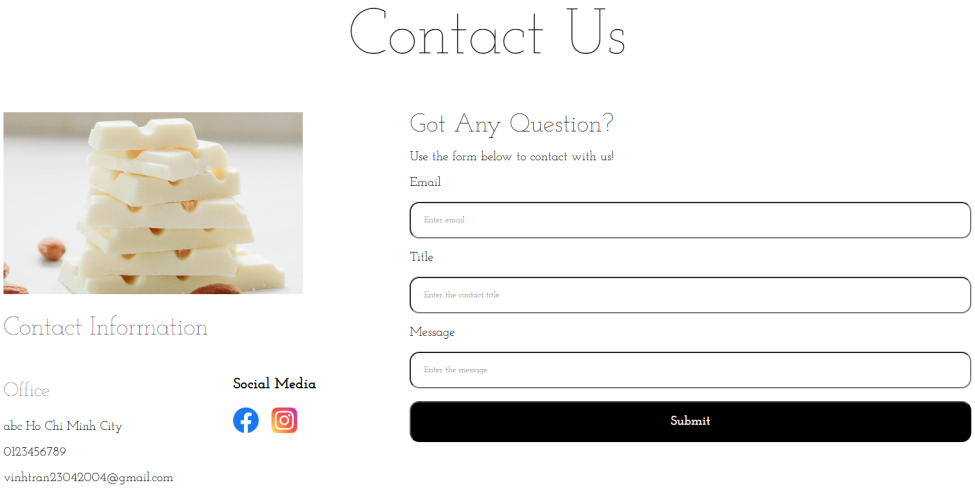

Figure 22: Chocoley Contact page

2.5 Account Page

For users and sellers with their information stored in session data, the Account page displays the account profile section if a valid session exists. If no session is found, the page instead shows the authentication section, prompting the user to log in or register.

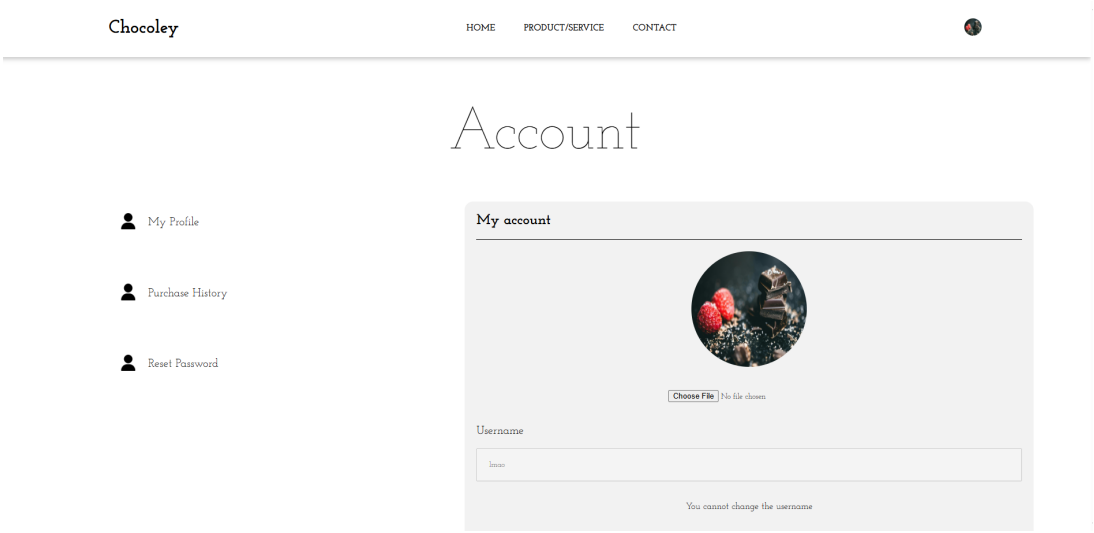

Figure 23: Chocoley Account page

2.5.1 My Profile

The **My Profile** section allows users to view and update their profile information that was not included during the **Sign Up** process. Additional attributes available for editing include phone number, gender, and date of birth. However, username and email are not editable and cannot be changed by the user.

After the user successfully edits their profile, the additional information will be updated—either replacing any existing values or filling in previously null fields. Additionally, if the user uploads a new profile image, the session will store the link to the newly uploaded image.

Listing 2.3: PHP code snippet of edit profile process

```
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        $email = $_SESSION['email'];
        $phone_number = $_POST['phone_number'];
        $gender = $_POST['gender'];
        $birth_date = $_POST['birthdate'];

        $search_info = "SELECT user_id FROM users WHERE email = '$email'";
        $query = mysqli_query($conn, $search_info);
        $result = mysqli_fetch_assoc($query);
        echo $_FILES['image']['error'];
        if (isset($_FILES['image']) && $_FILES['image']['error'] == 0) {
            $image = $_FILES['image']['name'];
            $tmp_image = $_FILES['image']['tmp_name'];
            // $new_name = uniqid() . "_" . basename($image);
            $target_dir = '../Images/User/' .$result["user_id"];
            $target_file = $target_dir . '/' . $image;

            if (!is_dir($target_dir)) {
                mkdir($target_dir,0777, true);
            }

            if (move_uploaded_file($tmp_image, $target_file)) {
                $update = "UPDATE users
                                SET phone_number = '$phone_number',
                                gender = '$gender',
                                birth_date = '$birth_date',
                                profile_image = '$image'
                                WHERE email = '$email'";
                mysqli_query($conn, $update);
                $_SESSION['image'] = $image;
                header("location: ../Web pages/index.php?user=account");
                exit;
```

```
            }
        } else {
            $update = "UPDATE users
                SET phone_number = '$phone_number',
                    gender = '$gender',
                    birth_date = '$birth_date'
                    WHERE email = '$email'";
            mysqli_query($conn, $update);
            header("location: ../Web pages/index.php?user=account");
            exit;
        }
    }
```

When an image is uploaded, the server will create a new **User** folder inside the **Images** repository if it does not already exist. Then, it will create a subfolder named after the user's ID and move the uploaded image into that destination.

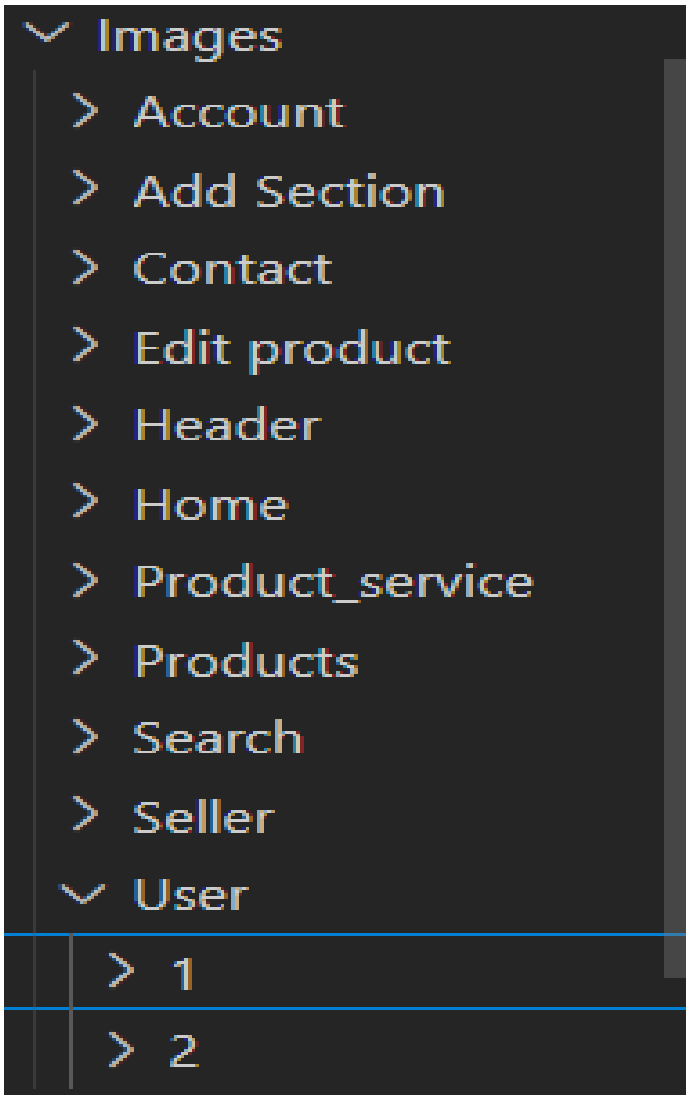

Figure 24: Image folder structure for users

After that, the server will update the user's information based on the email stored in the session. If no image is uploaded, the server will simply skip the image handling process and proceed to update the remaining information in the database.

2.5.2 Reset Password

The Reset Password section is a form that contains two fields: *New Password* and *Confirm Password*.

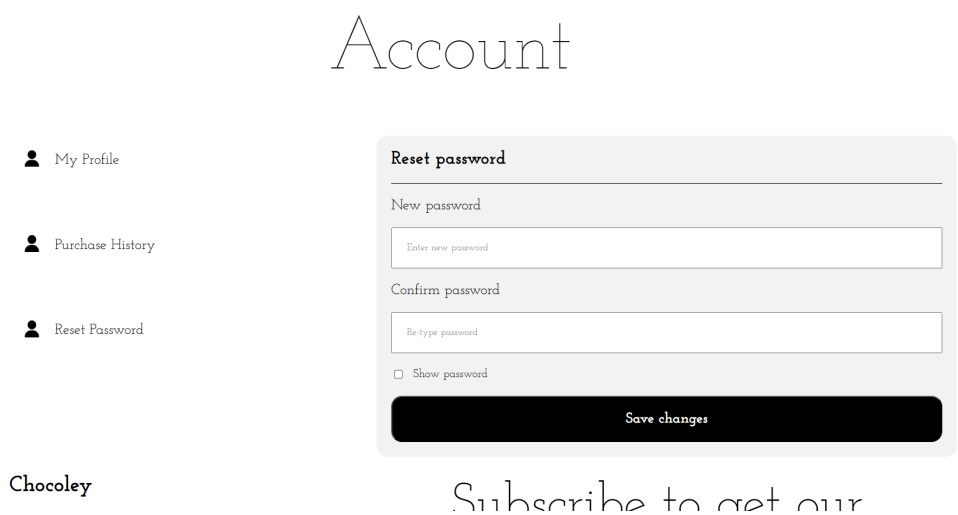

Figure 25: Reset Password section

This section functions similarly to the Forgot Password section described in 2.1.3. However, unlike the Forgot Password process, the server does not need to verify the availability of an email address in the database. Instead, it retrieves the user information using the email stored in the session and updates the corresponding password directly.

Listing 2.4: PHP code snippet of reset password process

```
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        $password = password_hash($_POST['password'], PASSWORD_DEFAULT);
        $email = $_SESSION['email'];

        $update = "UPDATE users
                   SET password = '$password'
                   WHERE email = '$email'";
        mysqli_query($conn, $update);
        header("location: ../Web pages/index.php?user=account");
        exit();
    }
```

2.6 Seller Features

2.6.1 Header

Unlike users and guests, the seller (admin) header includes only three navigation options: **Home**, **Your Products** (which displays the products posted by the seller), and **Account**. However, the Home and Account pages retain the same layout and functionality as those for users.

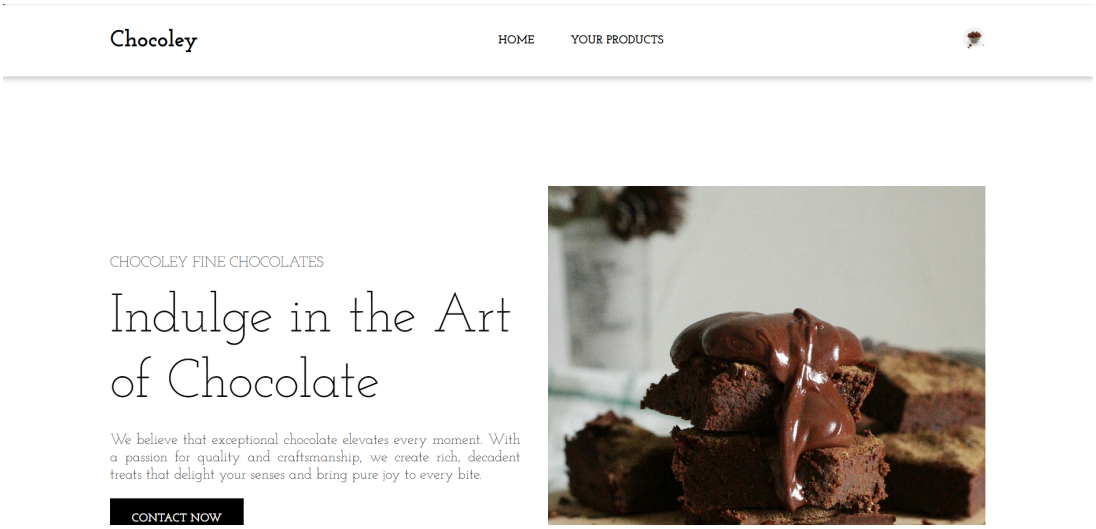

Figure 26: Seller homepage

2.6.2 Seller Product page

The **Your Products** page displays a list of products posted by the currently logged-in seller. This page is personalized, showing different types and quantities of products depending on the seller. The pagination, sorting, and search functionalities are identical to those on the **Product/Services** page for user roles. Additionally, a plus icon is located at the bottom-left corner of the page, which links to the **Add Products** section.

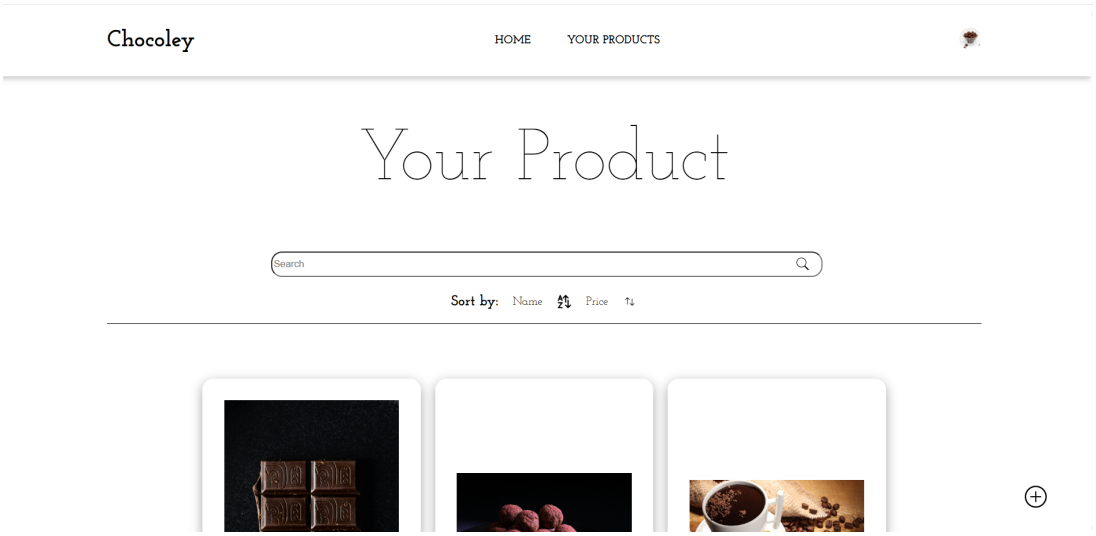

Figure 27: Seller product page

2.6.3 Edit Product

To edit a product, the seller must click the Edit button located on each product card in the Your Products page. The Edit Product page displays a form pre-filled with information about the selected product, retrieved using its product ID.

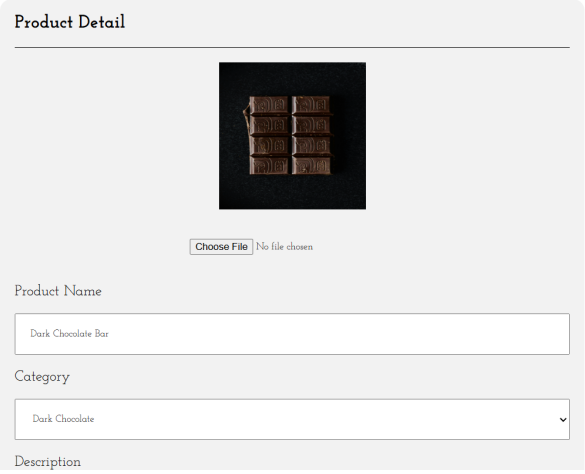

Figure 28: Seller Edit Product page

When the user successfully edits the product information, the server updates the corresponding entry in the products table based on the product ID.

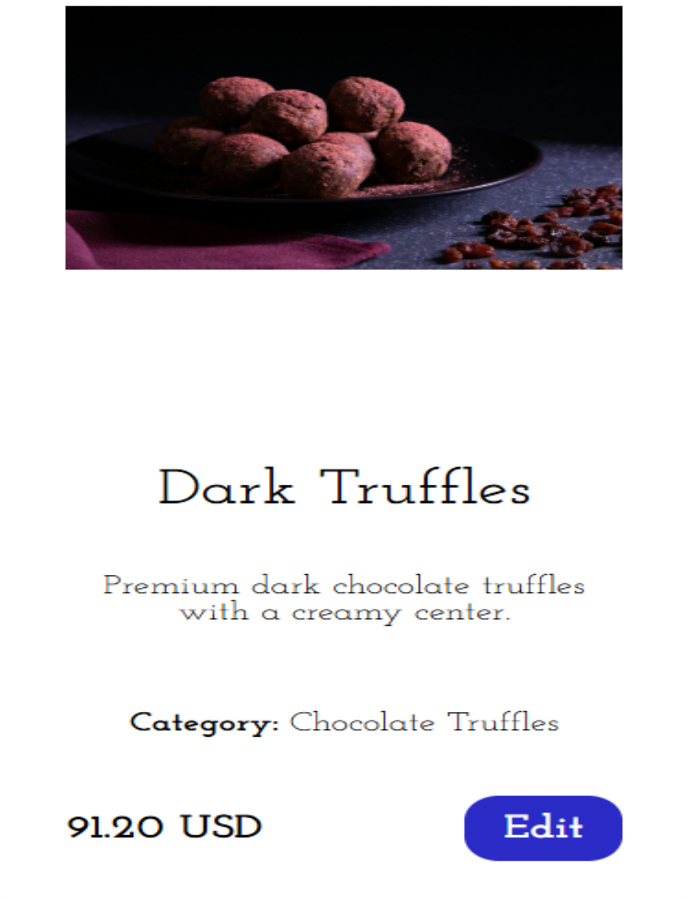

Figure 29: Dark Truffles before edit

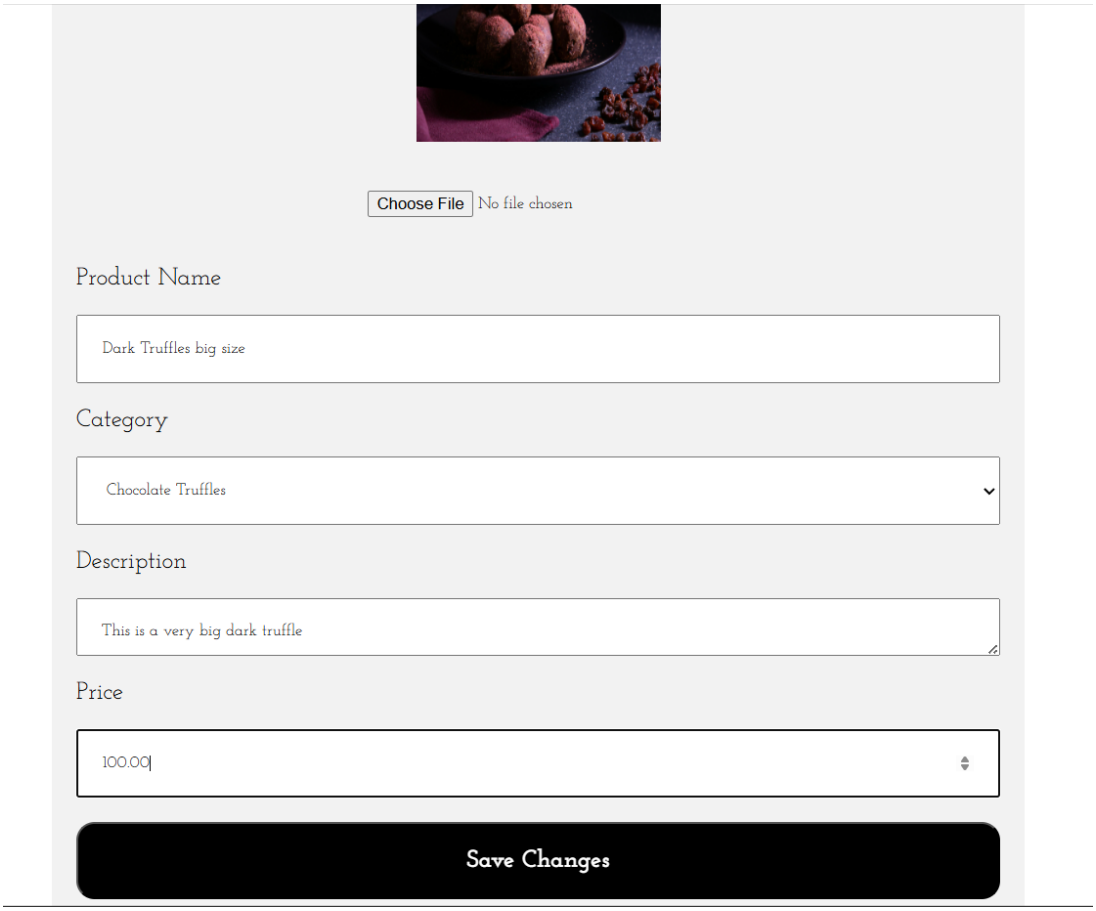

Figure 30: Changing information of Dark Truffle

Figure 31: Dark Truffle after edit

If a seller wants to delete a product, they can simply click the **Delete** button located below the **Save Changes** button on the Edit Product page. Upon clicking, the server locates the product in the database using its product ID and deletes it. Due to the cascade delete mechanism described in Section 1.2, the associated category ID linked to the seller ID in the **seller** table will also be removed. As a result, the deleted product will no longer appear on the **Your Products** page.

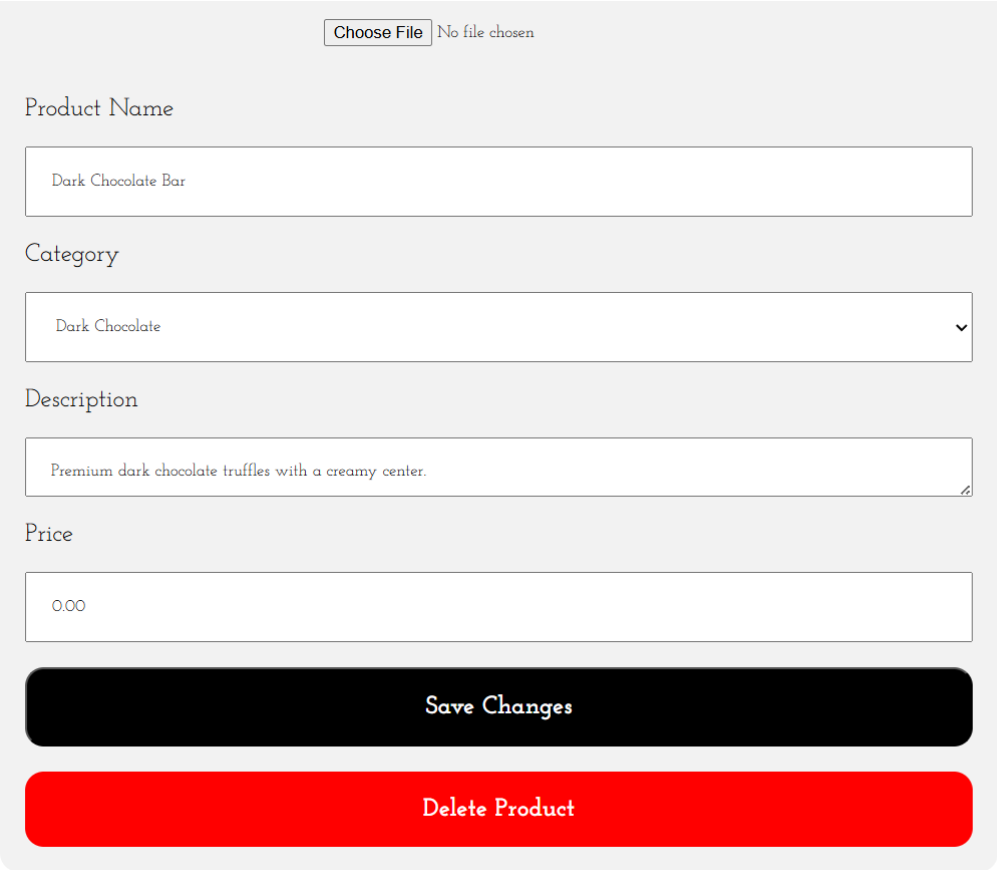

Figure 32: Delete Dark Chocolate product

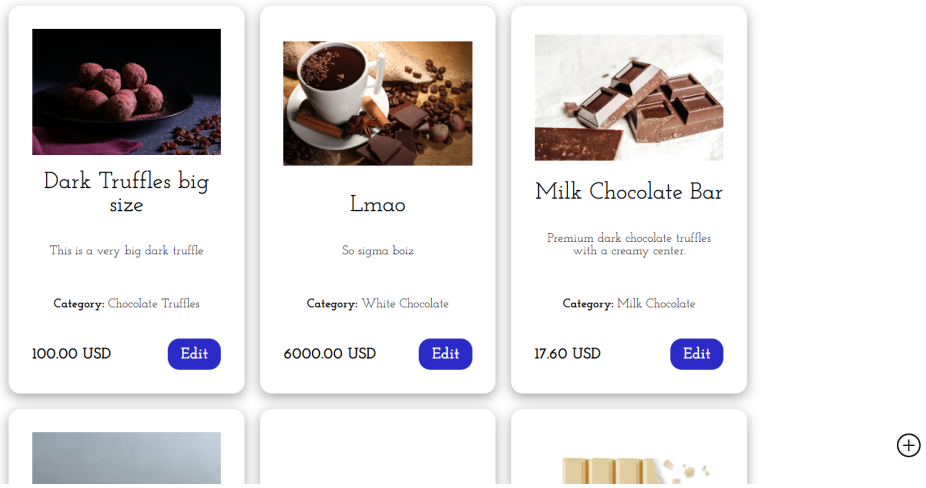

Figure 33: Your Product page after delete Dark Chocolate

2.6.4 Add Product

The **Add Product** page allows sellers to add new products to the database. To access this page, the seller can click the plus icon located at the bottom-left corner of the **Your Products** page. The **Add Product** page contains a form with fields for entering product information. A product cannot be added unless all required fields are filled out by the seller.

Figure 34: Add Product page

Once the seller has filled out all the required information, the server inserts the new product into the **products table**. Additionally, a new entry is added to the **seller table**, linking the newly generated product ID with the seller's ID to indicate which seller posted the product.

If a seller wants to add a new category, they can do so through the Add Product form by clicking the Click here link. When the seller submits a new category, the server first checks if it already exists in the categories table. If the category is found, the process continues as usual. If it does not exist, the server inserts the new category into the categories table before proceeding with the product insertion.

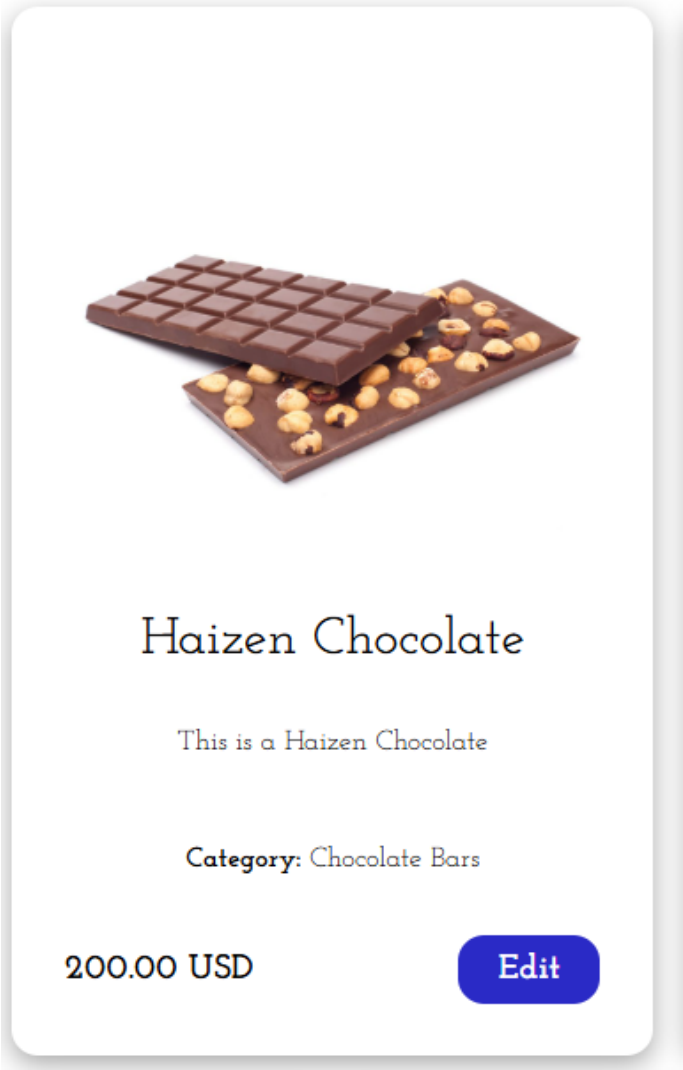

Figure 35: New inserted chocolate product

3 How to set up the website

1. Install and setup XAMPP through this links or or go to the next url: https://drive.google.com/file/d/1ttqLfJJMmGcsZQckl9YW4lUSS3gX2scE/view?usp=sharing
2. Dowload the zip and extract it in htdocs folder or fork and clone the project through the github link to the htdocs folder
3. Open http://localhost/phpmyadmin/ and create a database name chocoley
4. Navigate to the sql folder in the project, locate the chocoley.sql file
5. In http://localhost/phpmyadmin/, click in the chocoley database and click import tab to import the chocoley.sql file
6. Open "http://localhost/Web pages/" to get into index.php
7. Use the following account with email and password to use the website:
    - Seller role: Email: seller@example.com, password: 234
    - User role: Email: vinh.trantrung@hcmut.edu.vn, password: 123

4 Conclusion

Completing the individual web has been an enriching experience, allowing myself to deepen my understanding of web development. I successfully implemented beginner-level CRUD features, which have strengthened my practical knowledge and skills in creating, reading, updating, and deleting data within web applications. Some of the features includes:

1. Sign in / Sign up
2. Sessions
3. Hashing password
4. Sanitize input
5. Dynamic display products
6. AJAX based search
7. Sorting
8. Add, edit, delete products
9. Edit profile information

Due to time constraints, there were certain tasks that our team was unable to complete before the deadline. Some of these include:

1. Final working web application deployed on a server.
2. Product Detail and Purchasing
3. View history of user and seller role
4. Google map API because I cannot register to Google Cloud website

5 References

1. W3 Schools PHP tutorial. https://www.w3schools.com/php/.
2. Nixon, R. (2014). Learning PHP, mysql, JavaScript, CSS & HTML5: A step-by-step guide to creating dynamic websites. O'Reilly.
3. Elmasri, R., & Navathe, S. (2020). Fundamentals of Database Systems. Pearson.