

Assignment 2: Full-stack E-commerce Website

Author 1 SID: 520526392

Author 2 SID: 520401963

Author 3 SID: 510512333

ISYS2160



THE UNIVERSITY OF
SYDNEY

October 22, 2023

Contents

1	Overview	2
1.1	Tech Stack and setup	2
2	User Interface	2
2.1	Top bar	2
2.2	Checkout	3
2.3	Login Module	3
2.3.1	Account Manager	3
2.4	Cart	4
2.5	Admin Dashboard	4
2.6	Order Management	4
2.7	Login	4
2.8	Registration	5
2.9	Password Reset	5
2.10	OTP verification	5
3	Features	6
3.1	CSV Writer	6
3.2	CSV Parser	6
3.3	Mailer	7
3.4	Admin Manager	7
3.4.1	Product Management	7
3.5	User Experience	8
3.6	Authentication	9
3.6.1	Login	9
3.6.2	Logout	10
3.7	Forgot Password	10
3.7.1	Initial Form	10
3.8	Registration	11
3.9	User Data Management	12

1 Overview

The aim of this project is to design a user-friendly electronic commerce website. Our website design in particular focuses on selling technological goods such as smartphones, gaming consoles and smart accessories. The website has features such as email verification and notification, user order tracking, administrative order tracking and purchasing data tracking which can be extracted in a .csv format.

1.1 Tech Stack and setup

Our group employs the WAMP web development environment. WAMP stands for Windows, Apache, MySQL, PHP which defines our tech stack for operating system, local hosting, database and primary coding language. In order to initialize our local hosting server, we used the WampServer application which can be accessed via [this link](#).

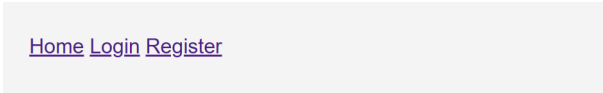
More detailed instructions on the process of setting up and initializing the server can be found on our [Github Repository](#).

2 User Interface

The user interface module consists of 11 different pages interlinking with one another. It is primarily coded in PHP and HTML/CSS with a little bit of Javascript.

2.1 Top bar

The topbar has a couple of different arrangements depending on whether a user has logged in or not and also the admin rights of that user.



[Home](#) [Login](#) [Register](#)

Figure 1: No user logged in



[Home](#) [Go to Cart](#) [Manage Account](#) [Logout](#)

Figure 2: Customer logged in



[Home](#) [Go to Cart](#) [Manage Account](#) [Admin Dashboard](#) [Logout](#)

Figure 3: Admin logged in

2.2 Checkout

The checkout page has a collection of prompts for the user to fill in which are relevant to payment processing.

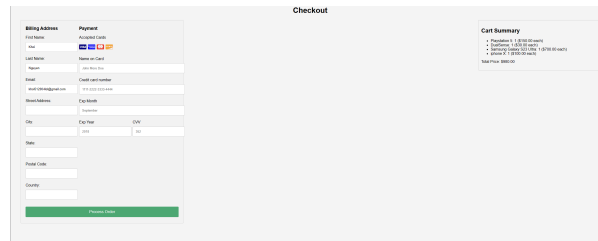


Figure 4: Checkout Page

2.3 Login Module

The home page has two modes depending whether the user has logged in or not. In both modes it will render the available products in the database with it's associated properties. If the user is logged in it will also render an add to cart button.



Figure 5: When logged in

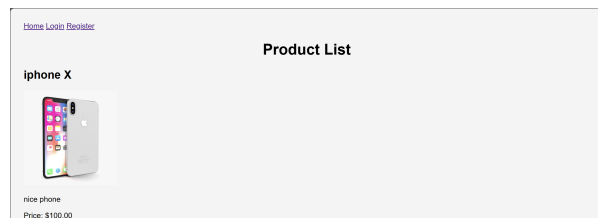


Figure 6: When logged out

2.3.1 Account Manager

The account manager page is accessible regardless of admin rights and allows users to update/change information.

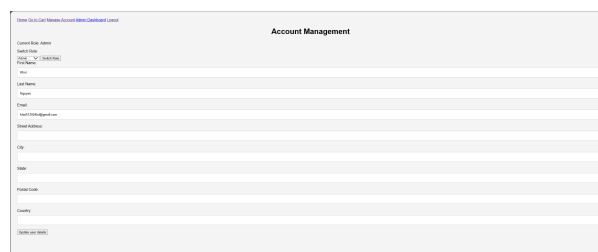


Figure 7: Account management

2.4 Cart

The cart renders items that has been added to cart and their quantities and your total checkout price. It also lets you change your quantities based on input.

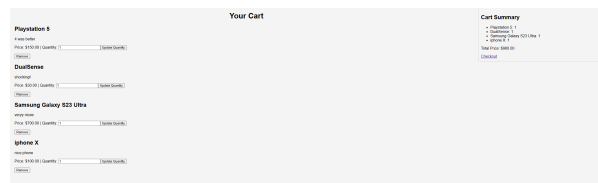


Figure 8: Cart

2.5 Admin Dashboard

Displays all pending orders and products and allow for the admins to edit their current state or delete them.

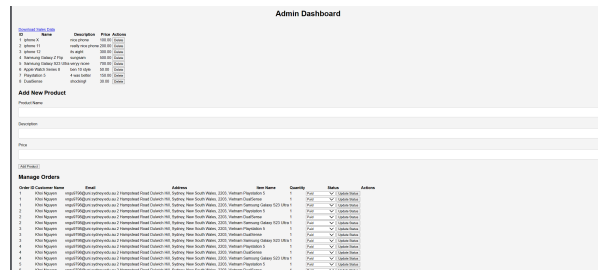


Figure 9: Admin Dashboard

2.6 Order Management

Shows the customers ordered items, quantities and their current state.



Figure 10: Order management

2.7 Login

Allows for the user to enter their login credential to authenticate. Has a username and password prompt as well as some helpers that redirects the users to create an account or to generate a new password.

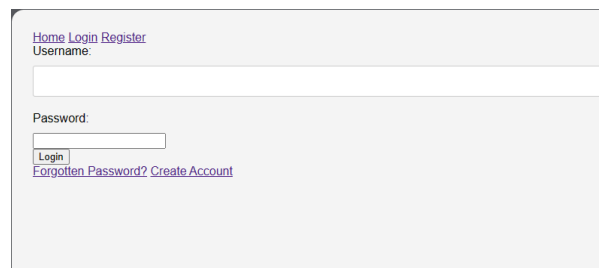
A login form with a light gray background. At the top left, there are three links: 'Home', 'Login', and 'Register'. Below them is a 'Username:' label followed by a text input field. Further down is a 'Password:' label followed by another text input field. At the bottom left, there is a 'Login' button. To the right of the button are two links: 'Forgotten Password?' and 'Create Account'.

Figure 11: Login Form

2.8 Registration

Allows the users to create a new account with prompts like username, password, first name, last name and email.

A registration form with a light gray background. At the top left, there are three links: 'Home', 'Login', and 'Register'. Below them is a 'Username:' label followed by a text input field. Further down is a 'Password:' label followed by a text input field, and below that is a 'Confirm Password:' label followed by another text input field. Below these are 'First Name:' and 'Last Name:' labels, each followed by a text input field. At the bottom is an 'Email:' label followed by a text input field. At the very bottom left is a 'Register' button.

Figure 12: Registration Form

2.9 Password Reset

Allows the user to change their password if they have forgotten or wish to have a new one. Prompts the user to enter their unique username.

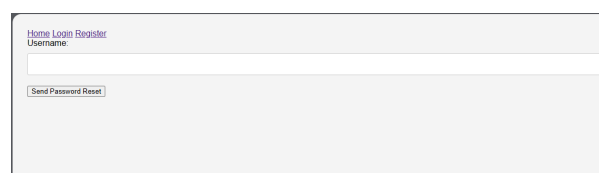
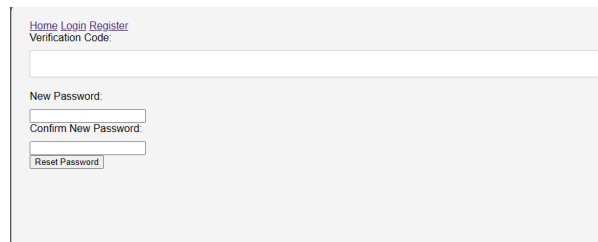
A form for password reset with a light gray background. At the top left, there are three links: 'Home', 'Login', and 'Register'. Below them is a 'Username:' label followed by a text input field. At the bottom left is a 'Send Password Reset' button.

Figure 13: Username prompt

2.10 OTP verification

Once a user has sent a request to change their password, an OTP will be sent to their email and the user interface will redirect to the OTP verification and new password selection page.



Home Login Register

Verification Code:

New Password:

Confirm New Password:

Figure 14: Enter Caption

3 Features

The website has a couple of unique features that allows it to function flexibly. This section will cover key/important features and how they were implemented.

3.1 CSV Writer

The CSV Writer is a simple tool that allows the website to collect data and put them into an accessible format which can be accessed by the administrators for purposes like data logging and data analytics. The CSV Writer module consists of one function *addSale()* that takes in an array of strings and writes them to a pre-specified CSV file.

```

3
4
5 function addSale($data) {
6     global $csv_file;
7     $csv_file = fopen($csv_file, 'w');
8     #write data into csv
9     fputcsv($csv_file, $data);
10 }
11 ?>

```

Figure 15: AddSale() function

3.2 CSV Parser

The CSV Parser tool is a tool that allows adding products into the website in bulk, using a specified format. The Parser module reads a pre-specified CSV file line by line and adds it's contents into the products database.

```

1 <?php
2
3 $csvFile = file_get_contents('products/products.csv');
4 $data = [];
5
6 for ($i = 1; $i < count($csvFile); $i++) {
7     $data[] = str_getcsv($csvFile[$i]);
8 }
9
10
11 foreach ($data as $product) {
12     $url = $product[1];
13     $image_dir = 'products/images/' . $product[4] . '.jpg';
14     file_put_contents($image_dir, file_get_contents($url));
15
16     $check = "SELECT * FROM products WHERE id='{$product[4]}';";
17     $result = $conn->query($check);
18
19     if ($result->num_rows > 0) {
20         continue;
21     }
22
23     $sql = "INSERT INTO products (id, name, description, price, image, category, quantity, sale)
24     values ('{$product[4]}', '{$product[0]}', '{$product[1]}', '{$product[2]}', '{$image_dir}', '{$product[5]}', '{$product[6]}', '{$product[7]}');";
25     $conn->query($sql);
26
27     // print_r($conn->error);
28 }
29
30 ?>

```

Figure 16: Parser Module

The Mailer module is the module that handles communications with the customers via mailing platforms. It utilizes Outlook SMTP via an addresses that we've created and consists of many functions depending on the use case of the mail. The PHPMailer class was obtained via an open source library obtained [here](#). Only the Exception, SMTP and PHPMailer files were used in this project.



This module allows the admin to either add or remove items from the products pool. This is done via SQL insertions or deletions.




```

1 <?php
2 include 'db_config.php';
3 session_start();
4
5 if (!isset($_SESSION['user_id']) || $_SESSION['role'] != 'admin') {
6     header('Location: login.php');
7     exit;
8 }
9
10 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
11     $name = $_POST['name'];
12     $description = $_POST['description'];
13     $price = $_POST['price'];
14
15     if (empty($name) || empty($description) || !is_numeric($price)) {
16         header('Location: admin_dashboard.php?error=Invalid input');
17         exit;
18     }
19
20     $sql = "INSERT INTO products (name, description, price) VALUES (?, ?, ?)";
21     $stmt = $conn->prepare($sql);
22     $stmt->bind_param("ssd", $name, $description, $price);
23
24     if ($stmt->execute()) {
25         header('Location: ../admin_dashboard.php?message=Product added successfully');
26     } else {
27         header('Location: ../admin_dashboard.php?error=Failed to add product');
28     }
29
30     $stmt->close();
31 }
32
33 $conn->close();
34 }

```

Figure 20: Product Adding

3.5 User Experience

Allows the users to perform tasks like adding to cart and checking out.

```

1 <?php
2 session_start();
3 include 'db_config.php';
4
5 $data = json_decode(file_get_contents('php://input'), true);
6 $product_id = $data['product_id'];
7 $user_id = $_SESSION['user_id'];
8
9 $checkSql = "SELECT quantity FROM cart WHERE user_id = $user_id AND product_id = $product_id";
10 $checkResult = $conn->query($checkSql);
11
12 if ($checkResult->num_rows > 0) {
13     $row = $checkResult->fetch_assoc();
14     $newQuantity = $row['quantity'] + 1;
15     $updateSql = "UPDATE cart SET quantity = $newQuantity WHERE user_id = $user_id AND product_id = $product_id";
16     if ($conn->query($updateSql) === TRUE) {
17         echo json_encode(['message' => 'Product quantity updated successfully.']);
18     } else {
19         echo json_encode(['message' => 'Error: ' . $updateSql . '<br>' . $conn->error]);
20     }
21 } else {
22     $insertSql = "INSERT INTO cart (user_id, product_id, quantity) VALUES ($user_id, $product_id, 1)";
23     if ($conn->query($insertSql) === TRUE) {
24         echo json_encode(['message' => 'Product added to cart successfully.']);
25     } else {
26         echo json_encode(['message' => 'Error: ' . $insertSql . '<br>' . $conn->error]);
27     }
28 }
29
30 $conn->close();
31 }

```

Figure 21: Add to cart

```

1 <?php
2 session_start();
3 include 'db_config.php';
4
5 if (!isset($_SESSION['user_id'])) {
6     echo json_encode(['message' => 'User not logged in']);
7     exit;
8 }
9
10 $data = json_decode(file_get_contents('php://input'), true);
11 $product_id = $data['product_id'];
12 $user_id = $_SESSION['user_id'];
13
14 $sql = "DELETE FROM cart WHERE user_id = $user_id AND product_id = $product_id";
15 if ($conn->query($sql) === TRUE) {
16     echo json_encode(['message' => 'Product removed from cart successfully.']);
17 } else {
18     echo json_encode(['message' => 'Error: ' . $sql . '<br>' . $conn->error]);
19 }
20
21 $conn->close();
22 }

```

Figure 22: Removing From Cart

Figure 23: Fetch Cart

Figure 24: Order Processing

Allows a user or admin to create an account, login or to change their password depending on their choices.

The login module gets a post request from the front-end and verifies it's information by querying for relevant user information in the database. If it is deemed valid then it will redirect to the home page and begin a session.

```

1 <?php
2 include 'db_config.php';
3
4 $username = $_POST['username'];
5 $password = $_POST['password'];
6
7 $sql = "SELECT * FROM users WHERE username='$username'";
8 $result = $conn->query($sql);
9
10 if ($result->num_rows > 0) {
11     $user = $result->fetch_assoc();
12     if (password_verify($password, $user['password'])) {
13         session_start();
14         $_SESSION['user_id'] = $user['id'];
15         $_SESSION['email'] = $user['email'];
16         $_SESSION['role'] = $user['role'];
17         header('Location: ../index.php');
18     } else {
19         echo "Invalid password.";
20     }
21 } else {
22     echo "Invalid username.";
23 }
24
25 $conn->close();
26 ?>

```

Figure 25: Login

3.6.2 Logout

Terminates the current session and redirects the user back to the homepage.

```

1 <?php
2 session_start();
3 session_destroy();
4 header('Location: ../login.php');
5 ?>

```

Figure 26: Logout

3.7 Forgot Password

A collection of smaller modules that verifies if a user genuinely wants to get a new password and allow them to set a new one.

3.7.1 Initial Form

The initial form prompts the user to fill in their username in order for the program to determine their email to send. It does some basic input checking to ensure the input matches with the constraints set by registration. If deemed valid it generates a random OTP and uses the mailing module to send the user a confirmation email. Then appends the OTP code to the *verification_code* column in the user database for that user and redirects the header to the OTP form.

```

<?php
include 'db_config.php';
include 'mailer.php';
$username = $_POST['username'];

if(empty($username)) {
    echo "Username cannot be empty.";
    exit;
}

if (!preg_match("/[a-zA-Z0-9]*/", $username)) {
    echo "Only letters and numbers allowed.";
    exit;
}

$sql = "SELECT * FROM users WHERE username = '$username'";
$result = $conn->query($sql);

// foreach($result as $row) {
//     //echo $row['column_name']; // Print a single column data
//     echo print_r($row); // Print the entire row data
// }

if($result->num_rows > 0) {
    $data = $result->fetch_all();
    $name = $data[0][4];
    $email = $data[0][6];
    $verification_code = random_int(100000, 999999);

    $send_otp = "UPDATE users SET verification_code = '$verification_code' WHERE username = '$username'";
    $result = $conn->query($send_otp);

    sendPasswordResetEmail($email, $name, $verification_code);
    header("location: ../verification_form.php");
}
else {
    echo "Username does not exist.";
    exit;
}
}

```

Figure 27: Initial Form

OTP Form The OTP form verifies that the input OTP is correct and does some input checking to ensure that the password is secure. If so then it will set a new password for that user in the database.

```

<?php
include 'db_config.php';
include 'mailer.php';
$username = $_POST['username'];

if(empty($username)) {
    echo "Username cannot be empty.";
    exit;
}

if (!preg_match("/[a-zA-Z0-9]*/", $username)) {
    echo "Only letters and numbers allowed.";
    exit;
}

$sql = "SELECT * FROM users WHERE username = '$username'";
$result = $conn->query($sql);

// foreach($result as $row) {
//     //echo $row['column_name']; // Print a single column data
//     echo print_r($row); // Print the entire row data
// }

if($result->num_rows > 0) {
    $data = $result->fetch_all();
    $name = $data[0][4];
    $email = $data[0][6];
    $verification_code = random_int(100000, 999999);

    $send_otp = "UPDATE users SET verification_code = '$verification_code' WHERE username = '$username'";
    $result = $conn->query($send_otp);

    sendPasswordResetEmail($email, $name, $verification_code);
    header("location: ../verification_form.php");
}
else {
    echo "Username does not exist.";
    exit;
}
}

```

Figure 28: OTP Form

3.8 Registration

The registration module inserts into the 'users' table within the database information that the user has inputted onto the front-end obtained via a post request.

```

$username = $_POST['username'];
$password = $_POST['password'];
$password2 = $_POST['password2'];
$email = $_POST['email'];
$firstname = $_POST['firstname'];
$lastname = $_POST['lastname'];

if (empty($username) || empty($password)) {
    echo "Username and password cannot be empty.";
    exit;
}

if ($password != $password2) {
    echo "Passwords do not match.";
    exit;
}

if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Invalid email format.";
    exit;
}

if (preg_match("/^[a-zA-Z]*$/", $firstname) || preg_match("/^[a-zA-Z]*$/", $lastname)) {
    echo "Only letters and white space allowed.";
    exit;
}

if (preg_match("/^[a-zA-Z0-9]*$/", $username)) {
    echo "Only letters and numbers allowed.";
    exit;
}

if (strlen($password) < 8) {
    echo "Password must be at least 8 characters.";
    exit;
}

if (preg_match("/[A-Z]/", $password)) {
    echo "Password must contain at least one uppercase character.";
    exit;
}

if (preg_match("/[a-z]/", $password)) {
    echo "Password must contain at least one lowercase character.";
    exit;
}

if (preg_match("/[0-9]/", $password)) {
    echo "Password must contain at least one number.";
    exit;
}

$sql = "SELECT * FROM users WHERE username='$username'";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    echo "Username already exists.";
    $conn->close();
    exit;
}

$password_hashed = password_hash($password, PASSWORD_DEFAULT);

$sql = "INSERT INTO users (username, password, last_name, first_name, email) VALUES ('$username', '$password_hashed', '$lastname', '$firstname', '$email')";
if ($conn->query($sql) === TRUE) {
    header("Location: ../login.php");
    exit;
} else {
    echo "Error: " . $sql . " , " . $err . " , $conn->error";
}

$conn->close();

```

Figure 29: Registration Form

3.9 User Data Management

Allows the users to change their information based on front end inputs obtained via post requests. This information is then updated onto the SQL database.

```

1
2 <?php
3 session_start();
4 include 'backend/db_config.php';
5
6 if (!isset($_SESSION['user_id'])) {
7     header("Location: login.php");
8     exit;
9 }
10
11 $user_id = $_SESSION['user_id'];
12 $sql = "SELECT * FROM users WHERE id = $user_id";
13 $result = $conn->query($sql);
14 $row = $result->fetch_assoc();
15 $current_role = $row['role'];
16 $first_name = $row['first_name'];
17 $last_name = $row['last_name'];
18 $email = $row['email'];
19
20 $sql = "SELECT * FROM addresses WHERE user_id = $user_id";
21 $address_result = $conn->query($sql);
22 $address = $address_result->fetch_assoc();
23 $conn->close();
24
25 ?>

```

Figure 30: Fetching Details

```

end > update_user_detail.php

<?php
session_start();
include 'db_config.php';

header('Content-Type: application/json');

if (!isset($_SESSION['user_id'])) {
    echo json_encode(['error' => 'User not logged in']);
    exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $user_id = $_SESSION['user_id'];

    // User details
    $first_name = $_POST['first_name'];
    $last_name = $_POST['last_name'];
    $email = $_POST['email'];

    // Address details
    $street_address = $_POST['street_address'];
    $city = $_POST['city'];
    $state = $_POST['state'];
    $postal_code = $_POST['postal_code'];
    $country = $_POST['country'];

    // Update user details
    $user_sql = "UPDATE users SET first_name=?, last_name=?, email=? WHERE id=?";
    $user_stmt = $conn->prepare($user_sql);
    $user_stmt->bind_param("sssi", $first_name, $last_name, $email, $user_id);
    $user_stmt->execute();

    // Update address details
    $address_sql = "INSERT INTO addresses (user_id, street_address, city, state, postal_code, country)
VALUES (?, ?, ?, ?, ?, ?)
ON DUPLICATE KEY UPDATE street_address = VALUES(street_address), city = VALUES(city),
state = VALUES(state), postal_code = VALUES(postal_code),
country = VALUES(country)";
    $address_stmt = $conn->prepare($address_sql);
    $address_stmt->bind_param("isssss", $user_id, $street_address, $city, $state, $postal_code, $country);
    $address_stmt->execute();

    if ($user_stmt->affected_rows > 0 || $address_stmt->affected_rows > 0) {
        echo json_encode(['message' => 'Profile and address updated successfully.']);
    } else {
        echo json_encode(['error' => 'Unable to update profile or address, or no changes made.']);
    }
}
$conn->close();
}

```

Figure 31: Update Details