BACHELOR OF SCIENCE WITH HONOURS IN CYBERSECURITY

Final Year Project Report

"Bookworm" A Secure Bookstore Web Application

Reported by: Kyaw Zin Latt

Supervisor: Zulkifli Bin Jalil

Due Date: 13 June 2024

DECLARATION STATEMENT

I certify that all the information given in this curriculum vitae is in accordance with my knowledge and experience. I take full responsibility for the truthfulness of the provided information below. Moreover, I understand that if I have deliberately given any false information, I am liable for prosecution for fraud and/or perjury.

Student Full Name: Kyaw Zin Latt

Student Registration Number: 14096552

Recoverable Signature

X Latt

Kyaw Zin Latt
Student
Signed by: db976590-df14-4d25-9504-e1d955f8962a

Signed:

ABSTRACT

In this paper, I am going to focus on the development of my prototype which is a secure bookstore web application. This website is a user-friendly platform with robust security features which protect user data and transactions. This report consists of detailed analysis of the system's architecture, the system's design, and the implementation. The implementation section can be comprised of building, security testing, and the deployment plan.

ACKNOWLEDGEMENTS

In this section, I want to express my sincere gratitude towards my professor for helping me throughout this project.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Since online shopping has rigorously increased, it is necessary to develop secure and user-friendly e-commerce platforms. My developed web application which name is bookstore, it is primarily aim at customers' data security, and usability. In this web application, a wide range of cybersecurity measures have been utilized. The main objective is to protect user data, provide an incredible shopping experience, and secure users' transactions.

There are mainly four chapters in this report. The analysis section outlines the functional and non-functional requirements of Bookworm website. The design chapter explained the project's architecture. The building section is composed of building, functionality testing, and penetration testing of this web application. A short summary of this report will be placed in the last part.

2. ANALYSIS

2.1 Functional Requirements

Function	**Description**
Registration	Users and admin can be able to create an account by providing their email and password. The prototype validates the input and store users' details in the database securely.
Password Hashing	Passwords are hashed before storing in the database by utilizing PHP's default password hashing function.
Login	After registration, admins and users can login with their email and passwords stored in the database.
OTP Verification	For additional security during the login process, I have implemented one-time-password (OTP) system. They will receive 6-digits OTP via their registered email.
Forgot Password	In case of forgetting password, Bookworm have also included the feature of resetting password. On successful OTP verification, users can set a new password.
SSL Certificate	The website utilizes SSL certificate to ensure data transmission between the user and the server.
Message Encryption	Users input messages are encrypted in the database with AES symmetric encryption system. The encryption key is generated with openssl and stored securely using environmental variables.
User Features	• About: Provide information about the Bookworm. • Order: Display order details. • Contact: Able to communicate with admin. • Shop: Display products to purchase. • Cart: Add purchased products in the cart and checkout.
Admin Panel	• Dashboard: Provide overview statistics • Product: Can add, update and delete products. • Order: Can manage users' orders. • Users: Can manage users' details. • Messages: View users' messages and respond directly to users' emails.
Security Functions	• Used code sanitization to handle cross site scripting attacks. • Define CSRF tokens generations in login and registration form to prevent CSRF attacks. • The login form detects and log SQL injection attacks.

2.2 Non-functional Requirements

Function	**Description**
Performance	During the designing of website's database, I have ensured that it can handle vast number of users, available products, orders, and users' messages.
Availability	Admins and users can access the website at any time by configuring ports in the apache server.
Data Integrity	Maintain users' data securely in the database by implementing robust security features.
Security Features	• Data Encryption: I have utilized SSL certificate in Xampp apache server to ensure data transmission securely. • Message Encryption: Use symmetric encryption algorithm to enhance security. • Password Security: Implement password hashing which is stored in the database. • SQL Injection Prevention: Use prepared statements to detect SQLi attacks. • XSS Prevention: Secure programming such as input sanitization and validations are used to prevent XSS. • CSRF Protection: Use CSRF tokens to protect against cross-site request forgery (CSRF) attacks. • Environmental variables: Database connections and encryption keys are store in environmental variable.

3. DESIGN

3.1 Model View Controller (MVC) Pattern

The following diagram shows the Model View Controller (MVC). I have created my project's directory structure according to MVC pattern.

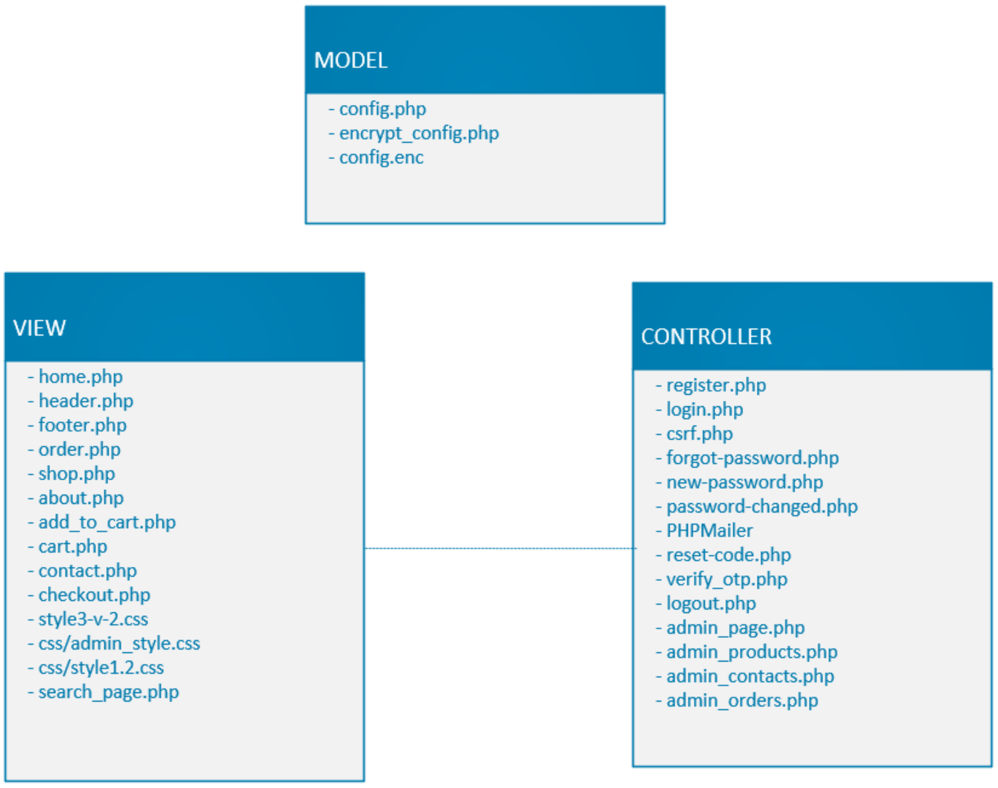


Figure 1 MVC Diagram

In Model, files serve as the medium between localhost and the database. I have added user facing pages and styling files in the view category. The controller has especially the managing and security feature files. The controllers cooperate with models to store data and render views to present data to the users. Files in the view display data from model as instructed by the controller' files.

3.2 Database Design

The structure of the database for this project can be understood by viewing the following diagram. The below image is the Entity Relationship Diagram (ERD) of the Bookworm web application. There are four entities which is made up of users, products, orders, and messages. Users represent both admin and the customers. Admin will update stocks in the products and users can orders from those updated stocks. In messages, users' communication with admins will be stored.

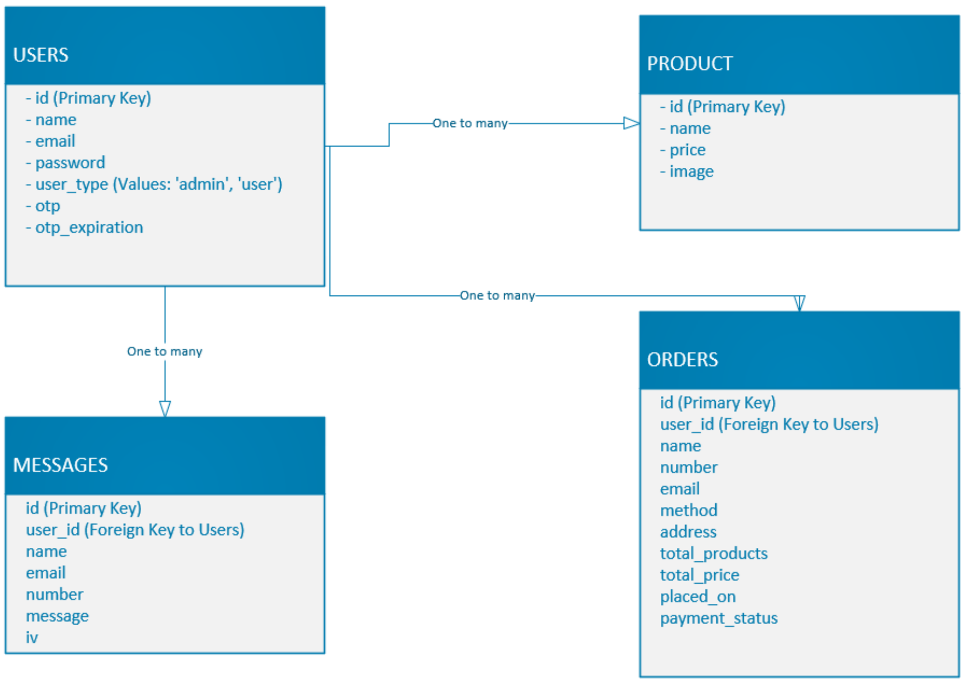


Figure 2 Entities Relationship Diagram

4. IMPLEMENTATION

In this implementation section, the security features and secure programming included in this project will be briefly explained.

4.1 Building

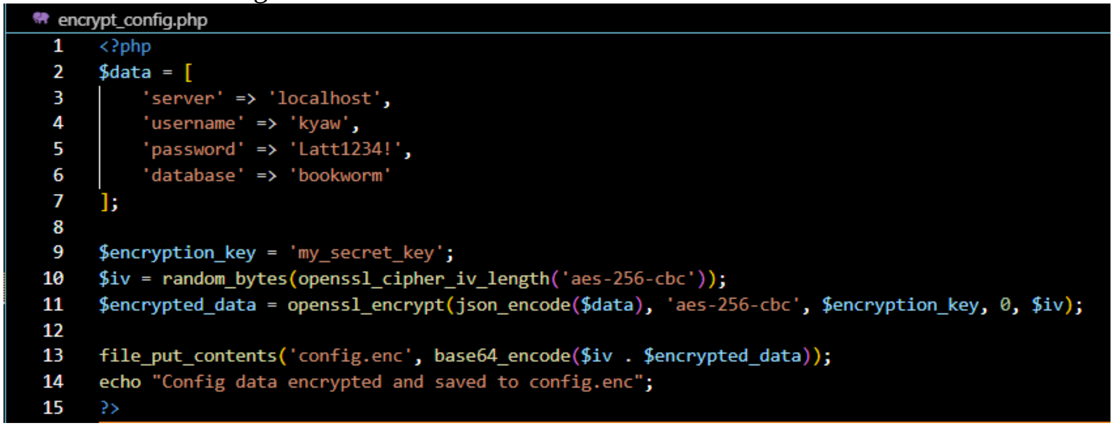
4.1.1 Server Setup

Since I am building a website by using local host, I have selected Xampp control panel to manage the localhost server and create the database.

- Install Xampp from the official website.
- Start Apache and MySQL services from Xampp control panel.
- Create a project file named Bookworm in htdocs.
- Open that file in Visual Studio Code for the coding.

4.1.2 Database Configuration

In design section, I have already explained the structure of database by illustrating Entities Relationship Diagram. Firstly, I have created config.php in my project directory which will connect with my database. After that, I have created database according to ERD. Since the file such as connection to database is vulnerable to some malicious attacks, so that I have encrypted the config.php by implementing the symmetric algorithms AES-256-CBC which uses base-64 encoding.

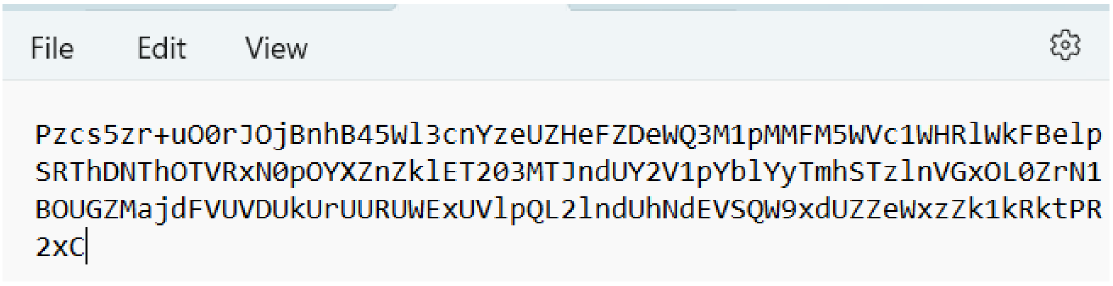


encrypt_config.php

```php
<?php
$data = [
    'server' => 'localhost',
    'username' => 'kyaw',
    'password' => 'Latt1234!',
    'database' => 'bookworm'
];

$encryption_key = 'my_secret_key';
$iv = random_bytes(openssl_cipher_iv_length('aes-256-cbc'));
$encrypted_data = openssl_encrypt(json_encode($data), 'aes-256-cbc', $encryption_key, 0, $iv);

file_put_contents('config.enc', base64_encode($iv . $encrypted_data));
echo "Config data encrypted and saved to config.enc";
?>
```

Figure 3 Encryption Algorithm for Database Configuration

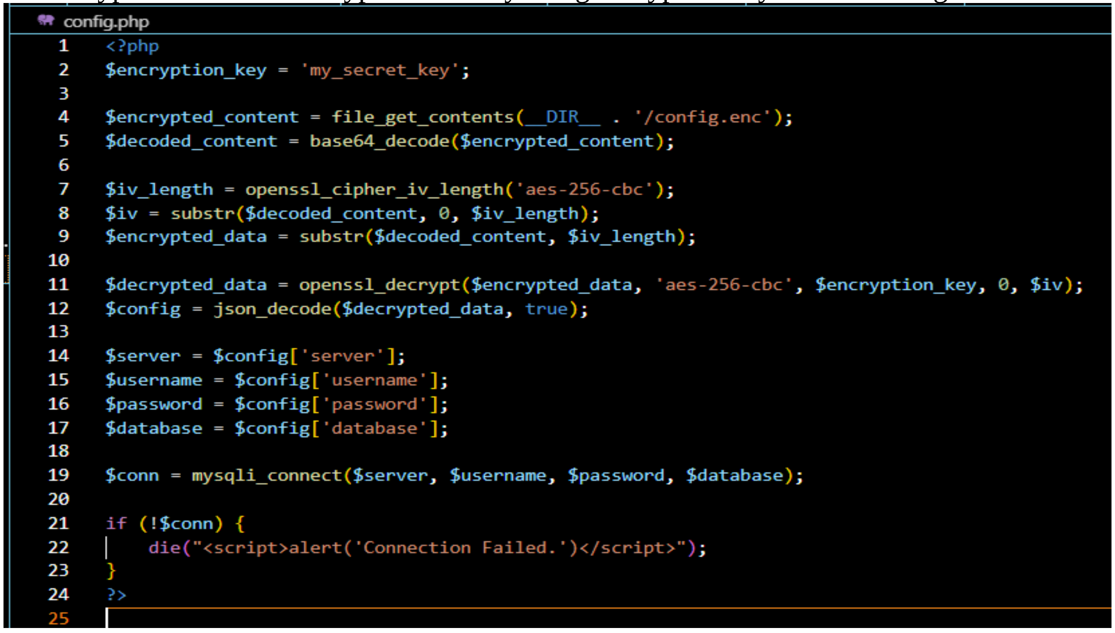
The above encryption algorithm outputs a config.enc file. I did not follow the secure coding method here. I have been getting errors in storing my encryption key in the environmental variables because I have already stored the message encryption key which I will explain later in this section.



```
Pzcs5zr+uO0rJOjBnhB45Wl3cnYzeUZHeFZDeWQ3M1pMMFM5WVc1WHRlWkFBelp
SRThDNThOTVRxN0pOYXZnZklET203MTJndUY2V1pYblYyTmhSTzlnVGxOL0ZrN1
BOUGZMajdFVUVDUkUrUURUWExUVlpQL2lndUhNdEVSQW9xdUZZeWxzZk1kRktPR
2xC
```

Figure 4.1 Encryption Output

In that way, I have created an encrypted database configuration file in which database details are encrypted. The code decrypts the file by using encryption key and the config.enc.



config.php

```php
<?php
$encryption_key = 'my_secret_key';

$encrypted_content = file_get_contents(__DIR__ . '/config.enc');
$decoded_content = base64_decode($encrypted_content);

$iv_length = openssl_cipher_iv_length('aes-256-cbc');
$iv = substr($decoded_content, 0, $iv_length);
$encrypted_data = substr($decoded_content, $iv_length);

$decrypted_data = openssl_decrypt($encrypted_data, 'aes-256-cbc', $encryption_key, 0, $iv);
$config = json_decode($decrypted_data, true);

$server = $config['server'];
$username = $config['username'];
$password = $config['password'];
$database = $config['database'];

$conn = mysqli_connect($server, $username, $password, $database);

if (!$conn) {
    die("<script>alert('Connection Failed.')</script>");
}
?>
```

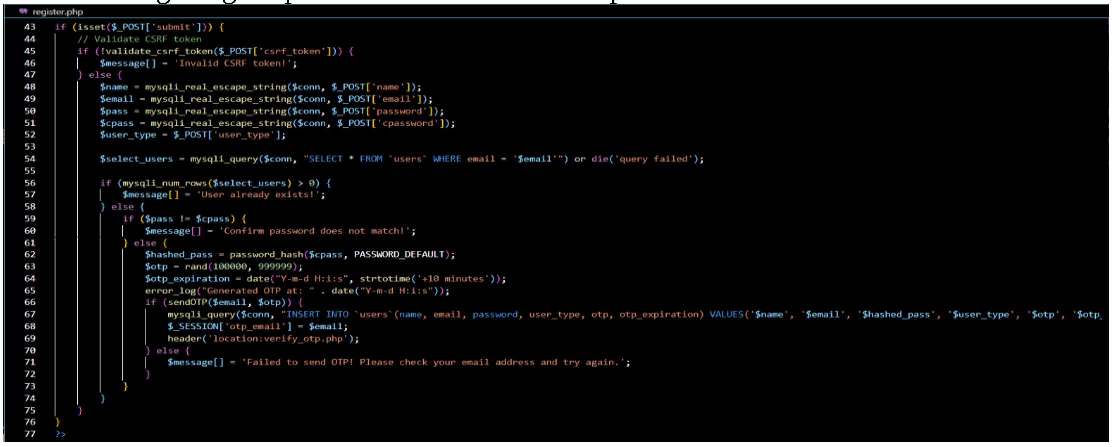
Figure 5 Database Connection

4.1.3 Password hashing in User Registration

Since I have been focusing on secure programming, I have implemented user registration with input validation and password hashing. I have made sure that the users' passwords will be hashed and stored in the database securely. For password hashing algorithms, PHP default built-in function has been used. It uses Bcrypt password hashing function which utilizes the Blowfish block cipher. The main advantages of this algorithm are:

- It uses one way hashing technique which is difficult to reverse.
- It cannot be penetrated by rainbow table attacks since it generates a cryptographic salt automatically.
- Using this function also protect against brute force attacks by its adaptive hashing function.

The following image depicts the utilization of PHP password hash function.



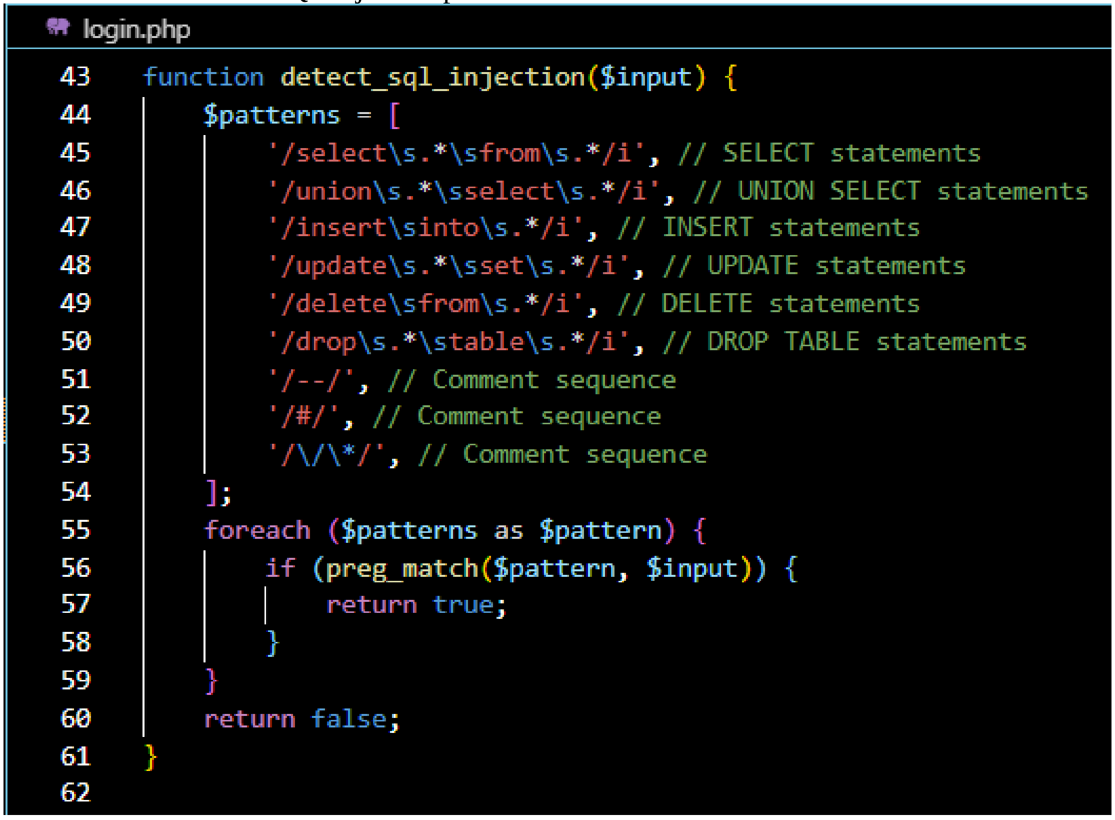
register.php

```php
if (isset($_POST['submit'])) {
    // Validate CSRF token
    if (!validate_csrf_token($_POST['csrf_token'])) {
        $message[] = 'Invalid CSRF token!';
    } else {
        $name = mysqli_real_escape_string($conn, $_POST['name']);
        $email = mysqli_real_escape_string($conn, $_POST['email']);
        $pass = mysqli_real_escape_string($conn, $_POST['password']);
        $cpass = mysqli_real_escape_string($conn, $_POST['cpassword']);
        $user_type = $_POST['user_type'];

        $select_users = mysqli_query($conn, "SELECT * FROM `users` WHERE email = '$email'") or die('query failed');

        if (mysqli_num_rows($select_users) > 0) {
            $message[] = 'User already exists!';
        } else {
            if ($pass != $cpass) {
                $message[] = 'Confirm password does not match!';
            } else {
                $hashed_pass = password_hash($cpass, PASSWORD_DEFAULT);
                $otp = rand(100000, 999999);
                $otp_expiration = date("Y-m-d H:i:s", strtotime('+10 minutes'));
                error_log("Generated OTP at: " . date("Y-m-d H:i:s"));
                if (sendOTP($email, $otp)) {
                    mysqli_query($conn, "INSERT INTO `users`(name, email, password, user_type, otp, otp_expiration) VALUES('$name', '$email', '$hashed_pass', '$user_type', '$otp', '$otp_
                    $_SESSION['otp_email'] = $email;
                    header('location:verify_otp.php');
                } else {
                    $message[] = 'Failed to send OTP! Please check your email address and try again.';
                }
            }
        }
    }
}
?>
```

Figure 6 Password Hashing Function

4.1.4 SQL Injection Detection in User Login Form

In login form, I have implemented a SQL injection detection system to safeguard against common attack vector in Cybersecurity which is SQL injection. I have created a function called detect_sql_injection which scans user inputs. In that function, I have defined an array that contains common SQL injection patterns.



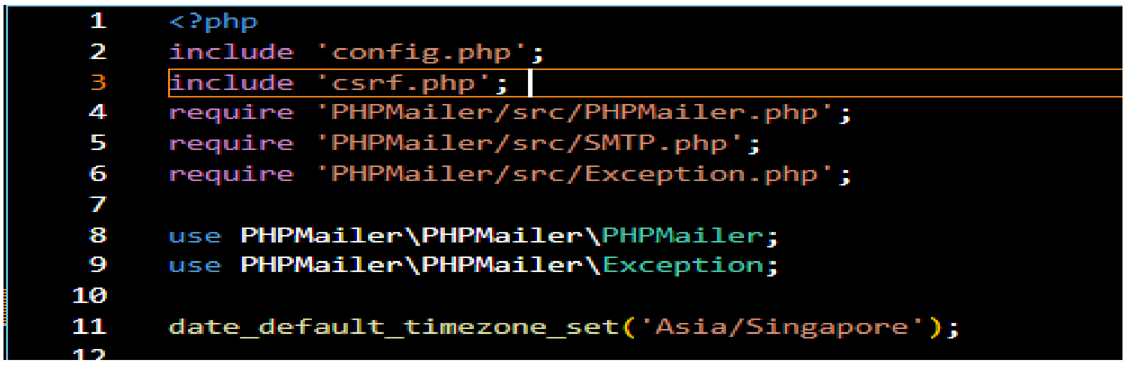
```php
login.php
function detect_sql_injection($input) {
    $patterns = [
        '/select\s.*\sfrom\s.*/i', // SELECT statements
        '/union\s.*\sselect\s.*/i', // UNION SELECT statements
        '/insert\sinto\s.*/i', // INSERT statements
        '/update\s.*\sset\s.*/i', // UPDATE statements
        '/delete\sfrom\s.*/i', // DELETE statements
        '/drop\s.*\stable\s.*/i', // DROP TABLE statements
        '/--/', // Comment sequence
        '/#/', // Comment sequence
        '/\/\*/', // Comment sequence
    ];
    foreach ($patterns as $pattern) {
        if (preg_match($pattern, $input)) {
            return true;
        }
    }
    return false;
}
```

Figure 7 SQL Injection Detection System

4.1.5 Two Factor Authentication (2FA)

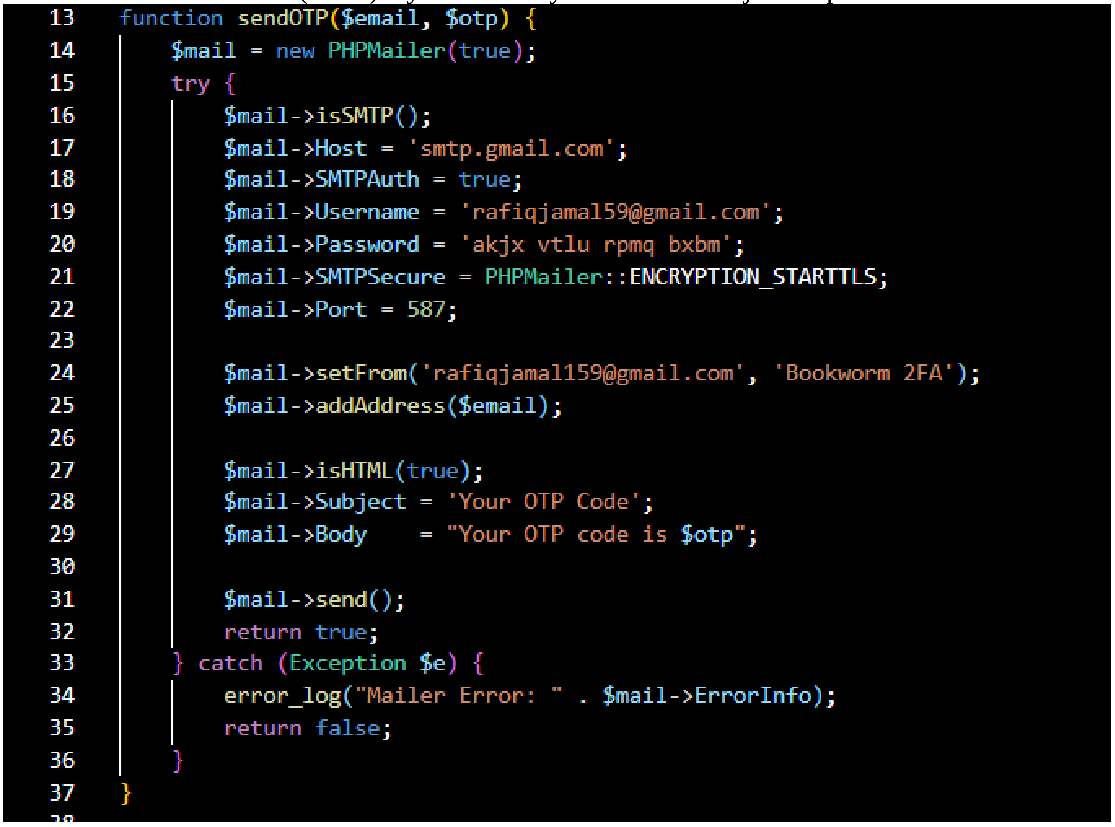
In both login and registration forms, verification with one time password (OTP) have been placed to enhance security. I have used PHPMailer library and google app password in building the OTP verification. Along with this, I have used Simple Mail Transfer Protocol (SMTP) port 587 which is the default port. This port is set with TLS encryption so that the email is submitted securely, and it follows the Internet Engineering Task Force (IETF) guidelines. Moreover, SMTP port 587 is not block in most upstream network. I avoid using the TLS port which is SMTP port 465 here since it is not compliance with the Request For Comments (RFC).

Firstly, I imported necessary libraries from PHPMailer in both login and registration forms and then define sentOTP function. After that I created an OTP verification file.



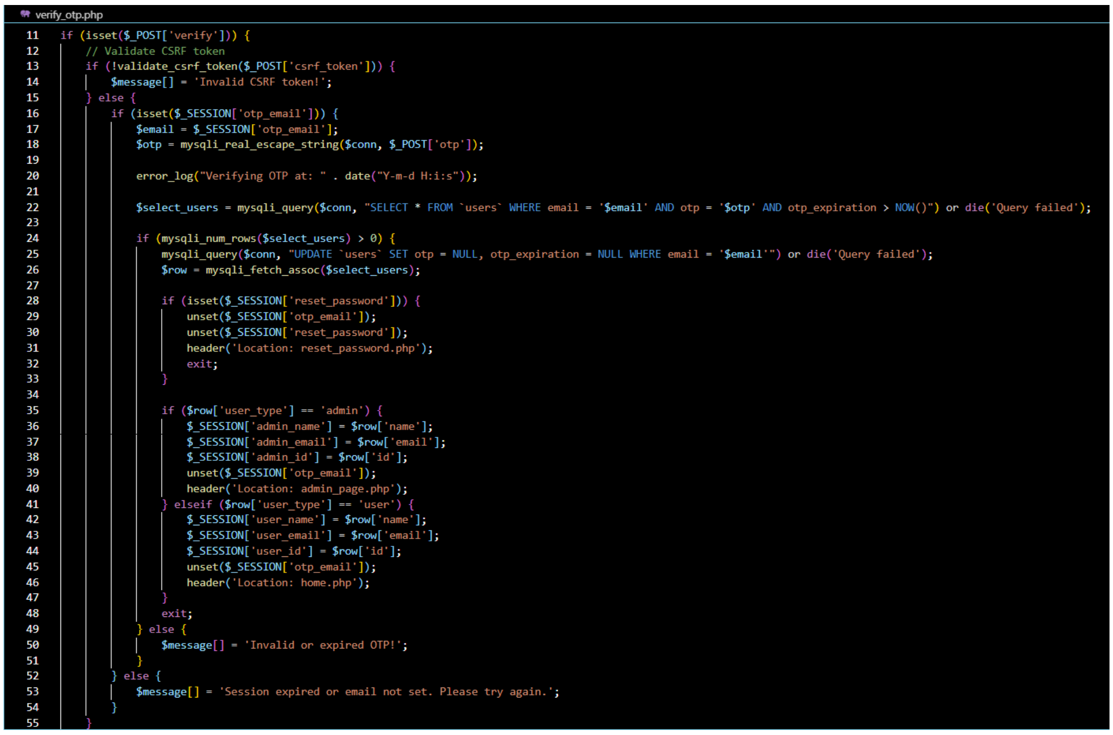
```php
<?php
include 'config.php';
include 'csrf.php';
require 'PHPMailer/src/PHPMailer.php';
require 'PHPMailer/src/SMTP.php';
require 'PHPMailer/src/Exception.php';

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

date_default_timezone_set('Asia/Singapore');
```

Figure 8 Imported Libraries from PHPMailer



```php
function sendOTP($email, $otp) {
    $mail = new PHPMailer(true);
    try {
        $mail->isSMTP();
        $mail->Host = 'smtp.gmail.com';
        $mail->SMTPAuth = true;
        $mail->Username = 'rafiqjamal59@gmail.com';
        $mail->Password = 'akjx vtlu rpmq bxbm';
        $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
        $mail->Port = 587;

        $mail->setFrom('rafiqjamal159@gmail.com', 'Bookworm 2FA');
        $mail->addAddress($email);

        $mail->isHTML(true);
        $mail->Subject = 'Your OTP Code';
        $mail->Body    = "Your OTP code is $otp";

        $mail->send();
        return true;
    } catch (Exception $e) {
        error_log("Mailer Error: " . $mail->ErrorInfo);
        return false;
    }
}
```

Figure 9 Sending OTP Function



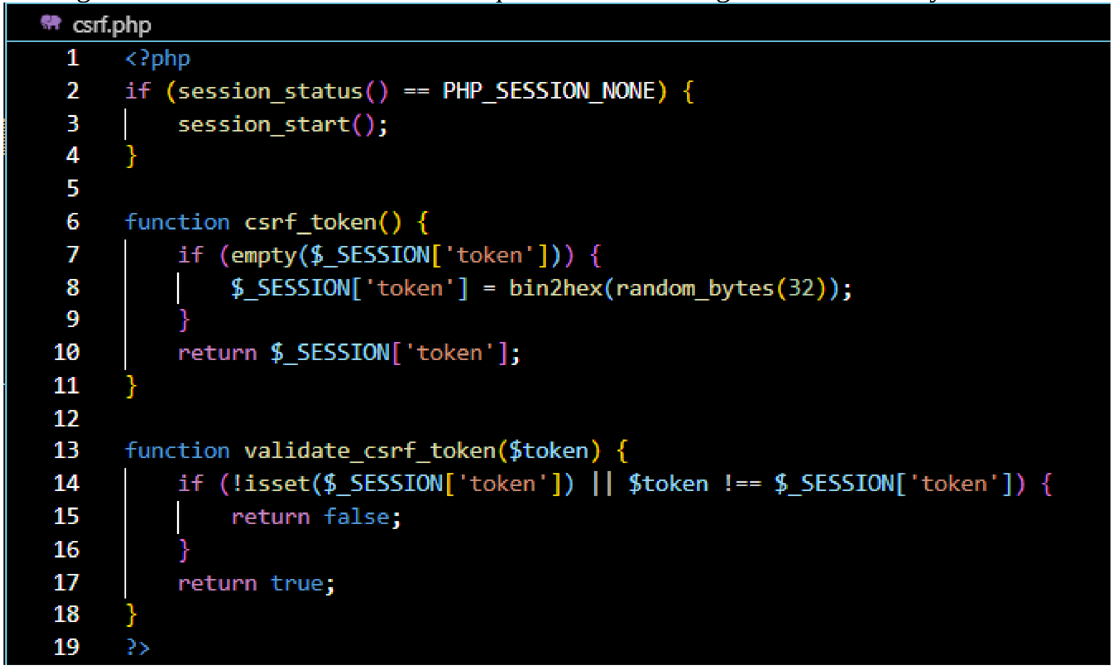
verify_otp.php

```php
if (isset($_POST['verify'])) {
    // Validate CSRF token
    if (!validate_csrf_token($_POST['csrf_token'])) {
        $message[] = 'Invalid CSRF token!';
    } else {
        if (isset($_SESSION['otp_email'])) {
            $email = $_SESSION['otp_email'];
            $otp = mysqli_real_escape_string($conn, $_POST['otp']);

            error_log("Verifying OTP at: " . date("Y-m-d H:i:s"));

            $select_users = mysqli_query($conn, "SELECT * FROM `users` WHERE email = '$email' AND otp = '$otp' AND otp_expiration > NOW()") or die('Query failed');

            if (mysqli_num_rows($select_users) > 0) {
                mysqli_query($conn, "UPDATE `users` SET otp = NULL, otp_expiration = NULL WHERE email = '$email'") or die('Query failed');
                $row = mysqli_fetch_assoc($select_users);

                if (isset($_SESSION['reset_password'])) {
                    unset($_SESSION['otp_email']);
                    unset($_SESSION['reset_password']);
                    header('Location: reset_password.php');
                    exit;
                }

                if ($row['user_type'] == 'admin') {
                    $_SESSION['admin_name'] = $row['name'];
                    $_SESSION['admin_email'] = $row['email'];
                    $_SESSION['admin_id'] = $row['id'];
                    unset($_SESSION['otp_email']);
                    header('Location: admin_page.php');
                } elseif ($row['user_type'] == 'user') {
                    $_SESSION['user_name'] = $row['name'];
                    $_SESSION['user_email'] = $row['email'];
                    $_SESSION['user_id'] = $row['id'];
                    unset($_SESSION['otp_email']);
                    header('Location: home.php');
                }
                exit;
            } else {
                $message[] = 'Invalid or expired OTP!';
            }
        } else {
            $message[] = 'Session expired or email not set. Please try again.';
        }
    }
```

Figure 10 OTP Verification Mechanisms

4.1.6 CSRF Tokens Generation

To handle against Cross-Site Request Forgery (CSRF) attacks, I have implemented the CSRF token generation. It makes sure that a unique CSRF token is generated for every user session.



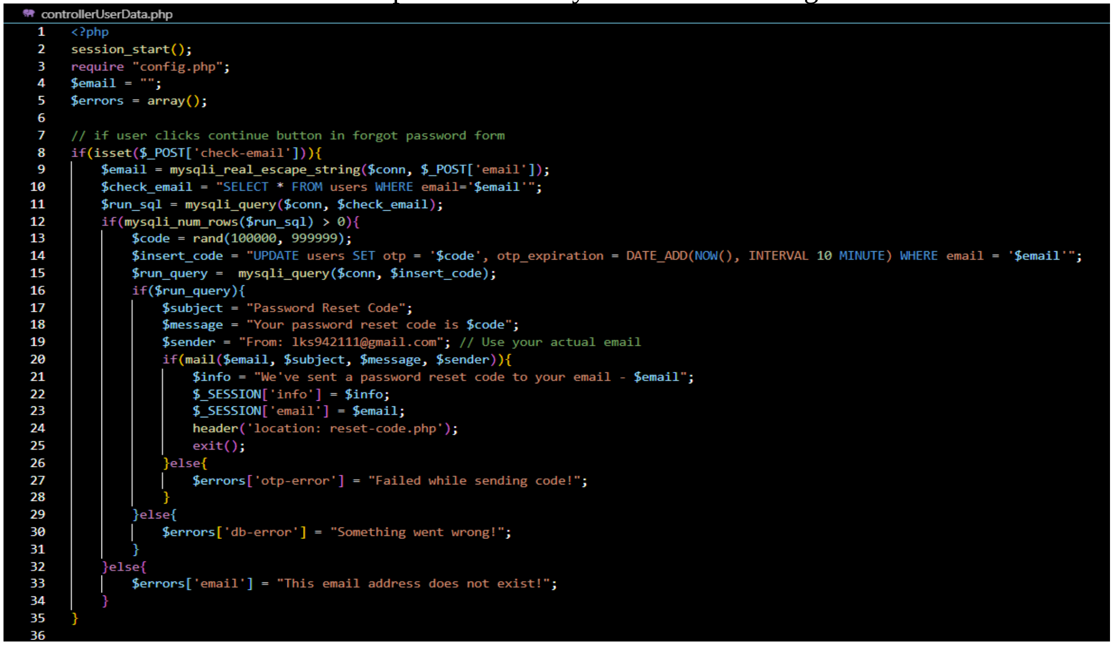
csrf.php

```php
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

function csrf_token() {
    if (empty($_SESSION['token'])) {
        $_SESSION['token'] = bin2hex(random_bytes(32));
    }
    return $_SESSION['token'];
}

function validate_csrf_token($token) {
    if (!isset($_SESSION['token']) || $token !== $_SESSION['token']) {
        return false;
    }
    return true;
}
?>
```

Figure 11 Generation of CSRF Token

4.1.7 Forgot Password and Reset Code Features

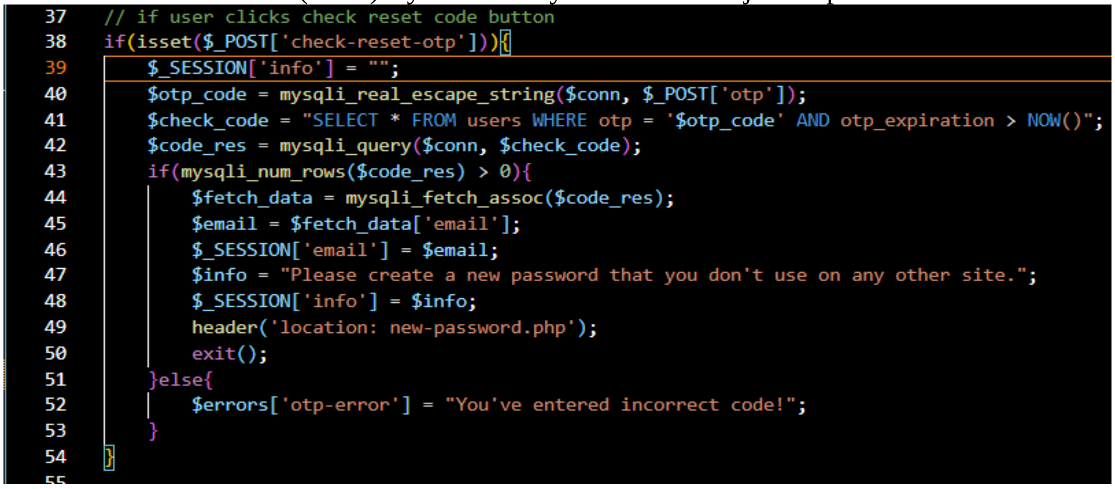
In case of user forgetting their password, I have defined the password reset function with OTP verification. Users can set a new password if they remember their registered email address.



controllerUserData.php

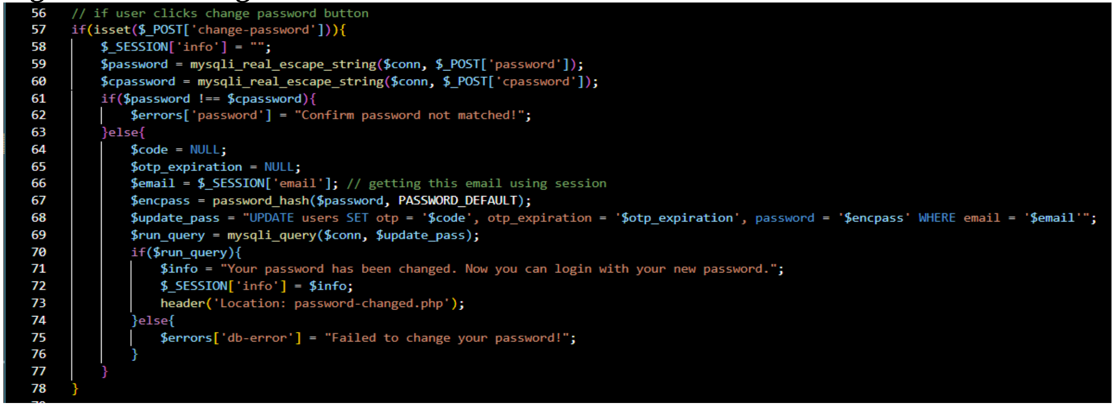
```php
<?php
session_start();
require "config.php";
$email = "";
$errors = array();

// if user clicks continue button in forgot password form
if(isset($_POST['check-email'])){
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $check_email = "SELECT * FROM users WHERE email='$email'";
    $run_sql = mysqli_query($conn, $check_email);
    if(mysqli_num_rows($run_sql) > 0){
        $code = rand(100000, 999999);
        $insert_code = "UPDATE users SET otp = '$code', otp_expiration = DATE_ADD(NOW(), INTERVAL 10 MINUTE) WHERE email = '$email'";
        $run_query =  mysqli_query($conn, $insert_code);
        if($run_query){
            $subject = "Password Reset Code";
            $message = "Your password reset code is $code";
            $sender = "From: lks942111@gmail.com"; // Use your actual email
            if(mail($email, $subject, $message, $sender)){
                $info = "We've sent a password reset code to your email - $email";
                $_SESSION['info'] = $info;
                $_SESSION['email'] = $email;
                header('location: reset-code.php');
                exit();
            }else{
                $errors['otp-error'] = "Failed while sending code!";
            }
        }else{
            $errors['db-error'] = "Something went wrong!";
        }
    }else{
        $errors['email'] = "This email address does not exist!";
    }
}

```

Figure 12 Checking User Email Prior to Password Reset



```php
// if user clicks check reset code button
if(isset($_POST['check-reset-otp'])){
    $_SESSION['info'] = "";
    $otp_code = mysqli_real_escape_string($conn, $_POST['otp']);
    $check_code = "SELECT * FROM users WHERE otp = '$otp_code' AND otp_expiration > NOW()";
    $code_res = mysqli_query($conn, $check_code);
    if(mysqli_num_rows($code_res) > 0){
        $fetch_data = mysqli_fetch_assoc($code_res);
        $email = $fetch_data['email'];
        $_SESSION['email'] = $email;
        $info = "Please create a new password that you don't use on any other site.";
        $_SESSION['info'] = $info;
        header('location: new-password.php');
        exit();
    }else{
        $errors['otp-error'] = "You've entered incorrect code!";
    }
}
```

Figure 13 Checking the Reset Code

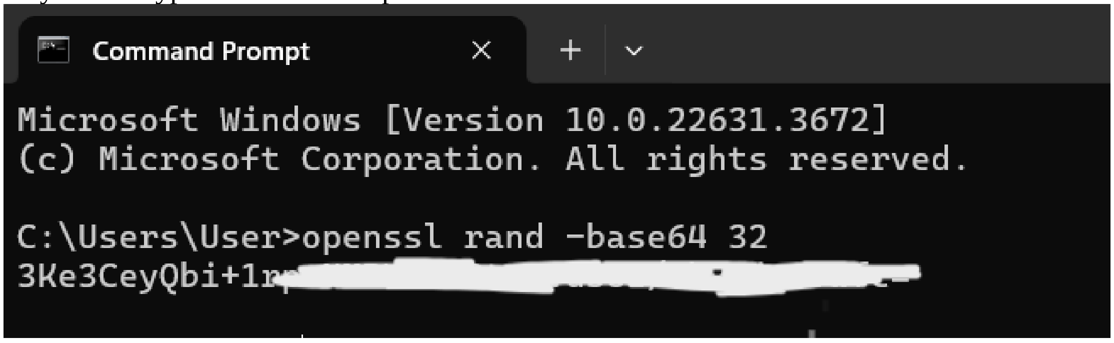


```php
// if user clicks change password button
if(isset($_POST['change-password'])){
    $_SESSION['info'] = "";
    $password = mysqli_real_escape_string($conn, $_POST['password']);
    $cpassword = mysqli_real_escape_string($conn, $_POST['cpassword']);
    if($password !== $cpassword){
        $errors['password'] = "Confirm password not matched!";
    }else{
        $code = NULL;
        $otp_expiration = NULL;
        $email = $_SESSION['email']; // getting this email using session
        $encpass = password_hash($password, PASSWORD_DEFAULT);
        $update_pass = "UPDATE users SET otp = '$code', otp_expiration = '$otp_expiration', password = '$encpass' WHERE email = '$email'";
        $run_query = mysqli_query($conn, $update_pass);
        if($run_query){
            $info = "Your password has been changed. Now you can login with your new password.";
            $_SESSION['info'] = $info;
            header('Location: password-changed.php');
        }else{
            $errors['db-error'] = "Failed to change your password!";
        }
    }
}
```

Figure 14 Update Password in Database

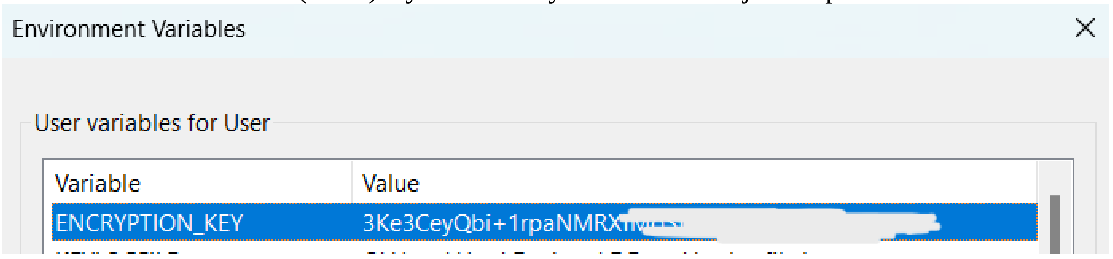
4.1.8 Message Encryption and Decryption

For confidentiality and security purposes, I have implemented massage encryption system using symmetric message encryption system. I have used AES-256-CBC encryption algorithm which utilizes 16 bytes (128bits) initialization vector (IV). The building of encryption system also stores IV in hexadecimal format in the database. For AES-256-CBC encryption, the key should be 32 bytes (256 bits) long, so I have generated base 64 encryption key by using Openssl in the command prompt. For the security enhancement, I have avoided hardcoding the encryption key in the php file and stored securely in the environmental variable. By implementing this system, users' messages are encrypted in the database, and they are decrypted in the admin panel.



```
Microsoft Windows [Version 10.0.22631.3672]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>openssl rand -base64 32
3Ke3CeyQbi+1r[redacted]
```

Figure 15 Generate Encryption Key with Openssl

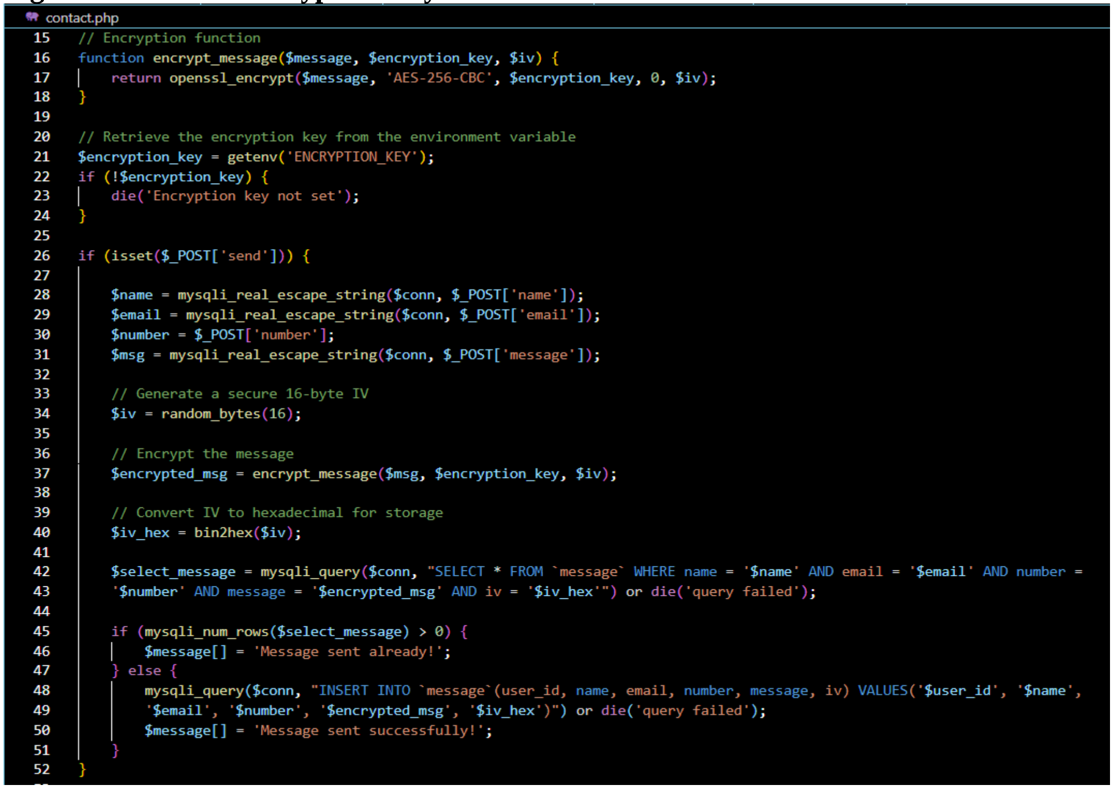


Environment Variables

User variables for User

Variable	Value
ENCRYPTION_KEY	3Ke3CeyQbi+1rpaNMRX[redacted]

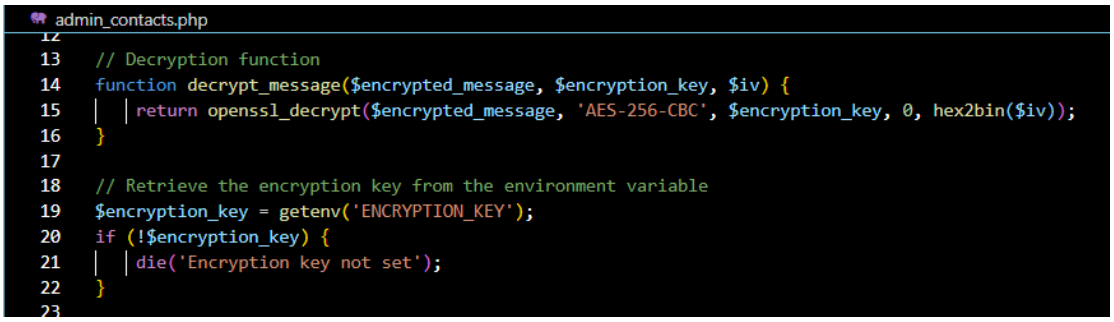
Figure 16.1 Stores Encryption Key in the Environmental Variable



contact.php

```php
// Encryption function
function encrypt_message($message, $encryption_key, $iv) {
    return openssl_encrypt($message, 'AES-256-CBC', $encryption_key, 0, $iv);
}

// Retrieve the encryption key from the environment variable
$encryption_key = getenv('ENCRYPTION_KEY');
if (!$encryption_key) {
    die('Encryption key not set');
}

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

    $name = mysqli_real_escape_string($conn, $_POST['name']);
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $number = $_POST['number'];
    $msg = mysqli_real_escape_string($conn, $_POST['message']);

    // Generate a secure 16-byte IV
    $iv = random_bytes(16);

    // Encrypt the message
    $encrypted_msg = encrypt_message($msg, $encryption_key, $iv);

    // Convert IV to hexadecimal for storage
    $iv_hex = bin2hex($iv);

    $select_message = mysqli_query($conn, "SELECT * FROM `message` WHERE name = '$name' AND email = '$email' AND number =
    '$number' AND message = '$encrypted_msg' AND iv = '$iv_hex'") or die('query failed');

    if (mysqli_num_rows($select_message) > 0) {
        $message[] = 'Message sent already!';
    } else {
        mysqli_query($conn, "INSERT INTO `message`(user_id, name, email, number, message, iv) VALUES('$user_id', '$name',
        '$email', '$number', '$encrypted_msg', '$iv_hex')") or die('query failed');
        $message[] = 'Message sent successfully!';
    }
}
```

Figure 17 Message Encryption Algorithm



admin_contacts.php

```php
// Decryption function
function decrypt_message($encrypted_message, $encryption_key, $iv) {
    return openssl_decrypt($encrypted_message, 'AES-256-CBC', $encryption_key, 0, hex2bin($iv));
}

// Retrieve the encryption key from the environment variable
$encryption_key = getenv('ENCRYPTION_KEY');
if (!$encryption_key) {
    die('Encryption key not set');
}
```

Figure 18 Message Decryption Algorithm

4.1.9 Using HTTPS with SSL Certificate

My website has become more secure with the free SSL certificate which I installed from ssl.indexnl.com. This essential improvement securely encrypts all data exchanges between our server and user browsers, thereby shielding private information from illegal access. The implementation of SSL certification enhances user data integrity and confidentiality and fosters user trust by indicating a secure connection, denoted by the 'https' protocol in the web address. This action reflects our dedication to upholding top-tier web security practices and ensuring the privacy of our users.

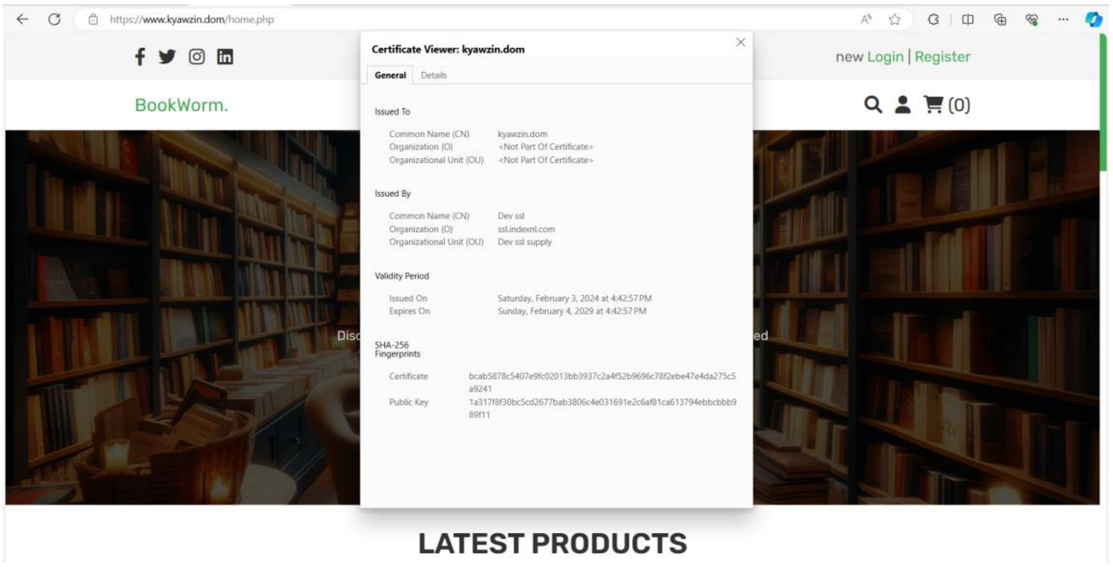
First, ssl.indexnl.com provides the certificate for me. Next, I set up the virtual host, windows host file, and SSL files in the Xampp apache extra folder. I then installed the certificate after that and added it into the Trusted Root Certificate Authorities folder.

```
<VirtualHost *:80>
    ServerAdmin webmaster@www.kyawzin.dom
    DocumentRoot "C:\xampp\htdocs\Bookworm"
    ServerName kyawzin.dom
    ServerAlias www.kyawzin.dom
    ErrorLog "logs/www.kyawzin.dom-error.log"
    CustomLog "logs/www.kyawzin.dom-access.log" common
</VirtualHost>
```

Figure 19 Configuration of Port 80

```
<VirtualHost *:443>
    ServerAdmin webmaster@www.kyawzin.dom
    DocumentRoot "C:\xampp\htdocs\Bookworm"
    ServerName kyawzin.dom
    ServerAlias www.kyawzin.dom
        SSLEngine on
         SSLCertificateFile "C:\xampp\apache\conf\ssl.crt\kyawzin.dom.crt"
         SSLCertificateKeyFile "C:\xampp\apache\conf\ssl.key\kyawzin.dom.key"
         <FilesMatch "\.(cgi|shtml|phtml|php)$">
         SSLOptions +StdEnvVars
         </FilesMatch>
        ErrorLog "logs/www.kyawzin.dom-error.log"
    CustomLog "logs/www.kyawzin.dom-access.log" common
</VirtualHost>
```

Figure 20 Configuration of Port 443



Certificate Viewer: kyawzin.dom

General Details

Issued To

Common Name (CN)	kyawzin.dom
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	Dev ssl
Organization (O)	ssl.indexnl.com
Organizational Unit (OU)	Dev ssl supply

Validity Period

Issued On	Saturday, February 3, 2024 at 4:42:57 PM
Expires On	Sunday, February 4, 2029 at 4:42:57 PM

SHA-256 Fingerprints

Certificate	bcab5878c5407e9fc02013bb3937c2a4f52b9696c78f2ebe47e4da275c5a9241
Public Key	1a317f8f30bc5cd2677bab3806c4e031691e2c6af81ca613794ebbcbbb989f11

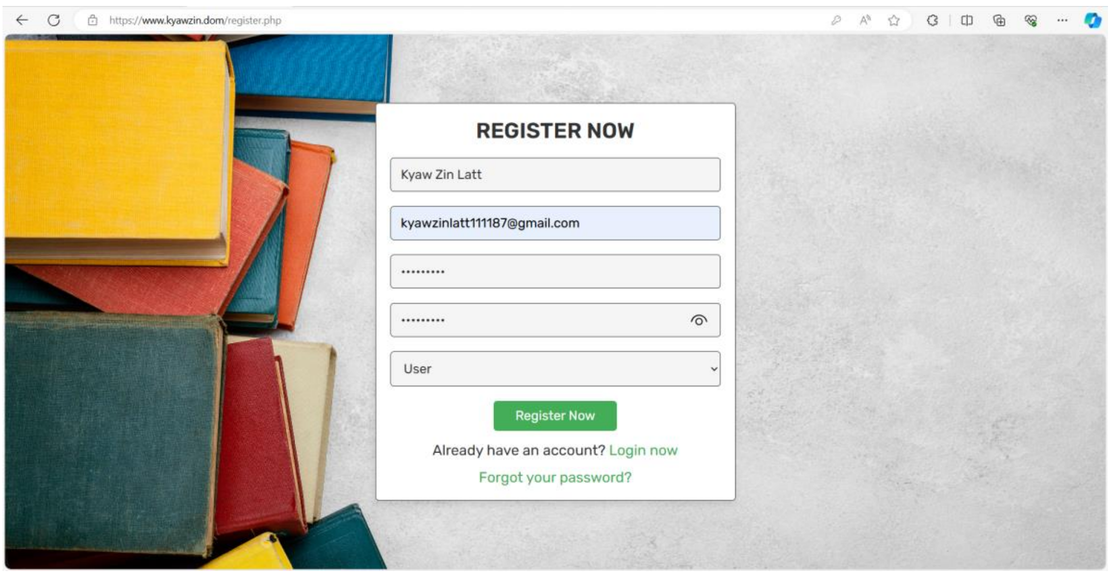
Figure 21 SSL Certificate in my Bookworm Webpage

4.2 Functionalities Testing

The testing of implemented functionalities will be undergone here.

4.2.1 Registration, Login, OTP Verification and Password Hashing

As I have inserted forms such as registration and login, and verification with two factor authentication while users' registration and login. Additionally, users' registered passwords are stored securely in the database. The below screenshots will demonstrate these functionalities.


https://www.kyawzin.dom/register.php

REGISTER NOW

Kyaw Zin Latt

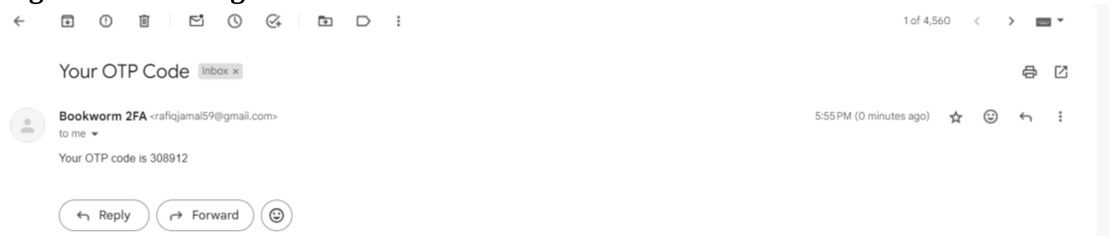
kyawzinlatt111187@gmail.com

•••••••••

•••••••••

User

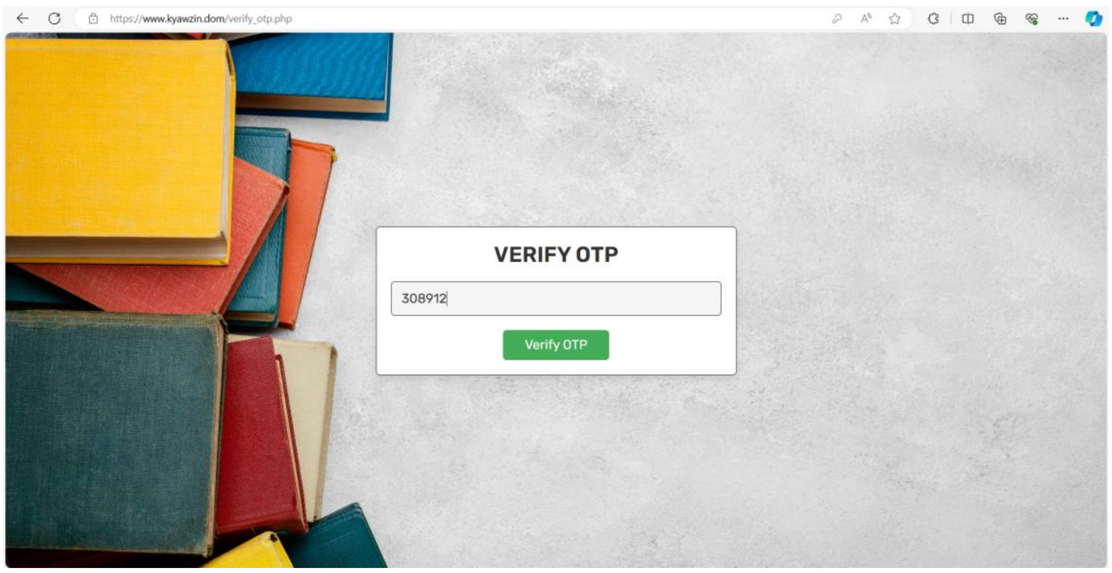
Figure 22 User Registration


Your OTP Code Inbox ×

Bookworm 2FA <rafiqjamal59@gmail.com>
to me
5:55 PM (0 minutes ago)

Your OTP code is 308912

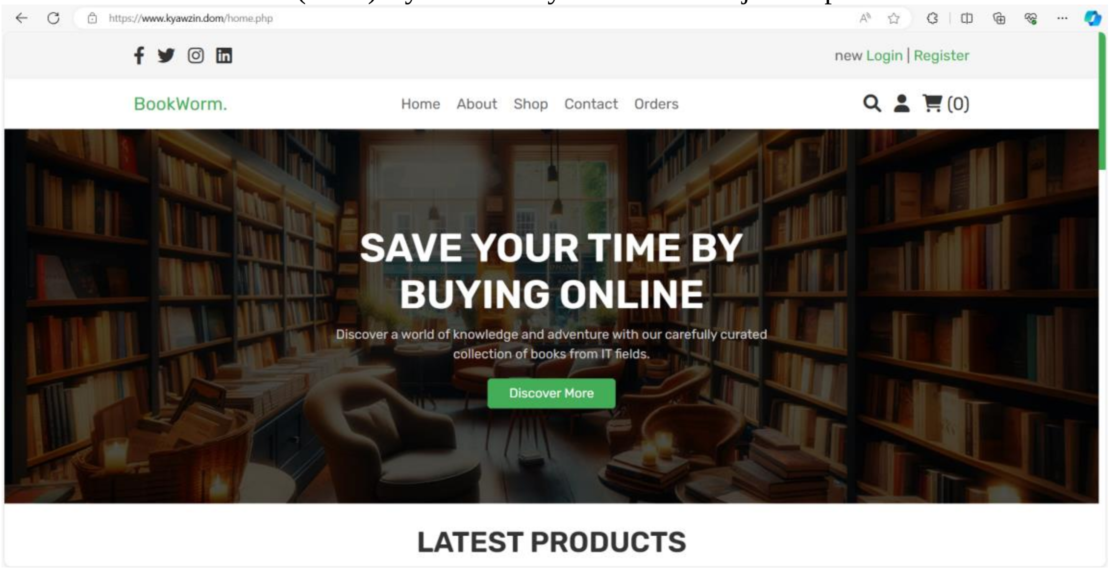
Figure 23 Receiving OTP in their Gmail


https://www.kyawzin.dom/verify_otp.php

VERIFY OTP

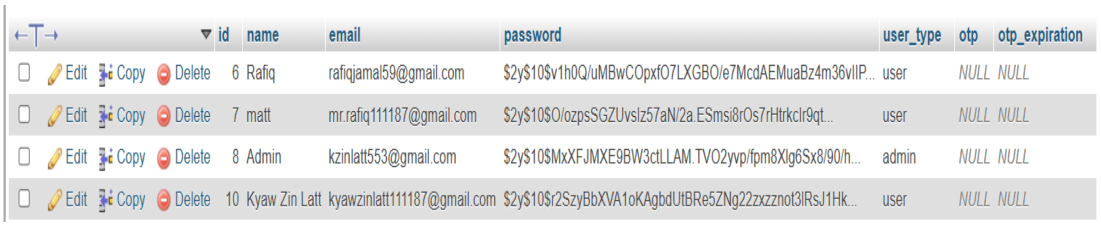
308912

Figure 24 OTP Verification


https://www.kyawzin.dom/home.php
new Login | Register
BookWorm.
Home About Shop Contact Orders
(0)
SAVE YOUR TIME BY BUYING ONLINE
Discover More
LATEST PRODUCTS

Discover a world of knowledge and adventure with our carefully curated collection of books from IT fields.

Figure 25 Access to the Webpage after Authentications



	id	name	email	password	user_type	otp	otp_expiration
Edit Copy Delete	6	Rafiq	rafiqjamal59@gmail.com	$2y$10$v1h0Q/uMBwCOpxfO7LXGBO/e7McdAEMuaBz4m36vIIP...	user	NULL	NULL
Edit Copy Delete	7	matt	mr.rafiq111187@gmail.com	$2y$10$O/ozpsSGZUvslz57aN/2a.ESmsi8rOs7rHtrkclr9qt...	user	NULL	NULL
Edit Copy Delete	8	Admin	kzinlatt553@gmail.com	$2y$10$MxXFJMXE9BW3ctLLAM.TVO2yvp/fpm8Xlg6Sx8/90/h...	admin	NULL	NULL
Edit Copy Delete	10	Kyaw Zin Latt	kyawzinlatt111187@gmail.com	$2y$10$r2SzyBbXVA1oKAgbdUtBRe5ZNg22zxzznot3lRsJ1Hk...	user	NULL	NULL

Figure 26 Passwords are Hashed in Database

4.2.2 Forgot-password Feature

In case users have forgotten their password, they can recover their account with their gmail. The password reset code will be sent to their gmail account and they can set a new password. The following pictures are the demonstration of password reset feature.

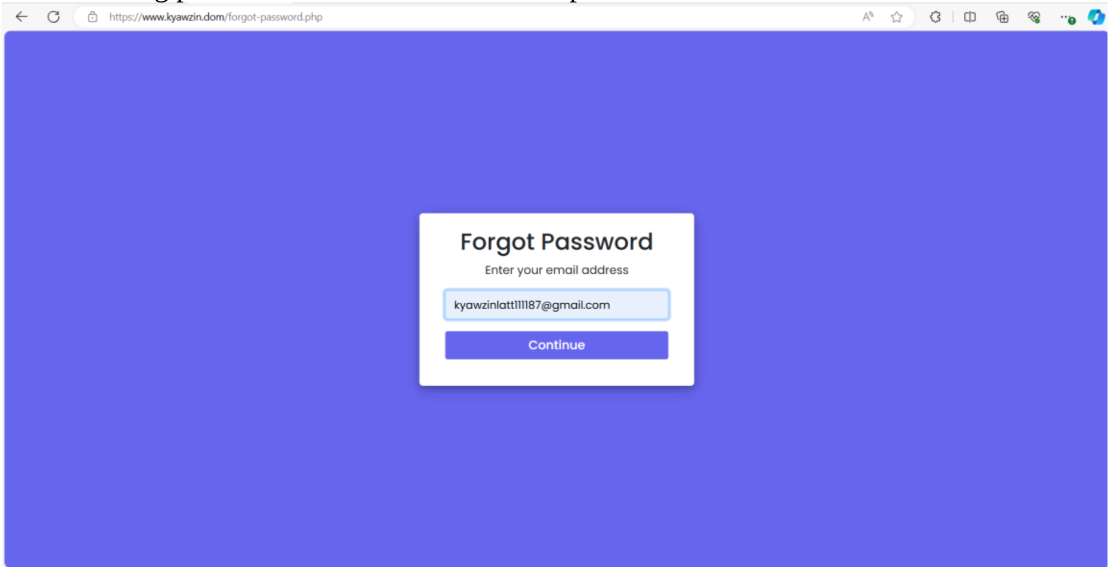
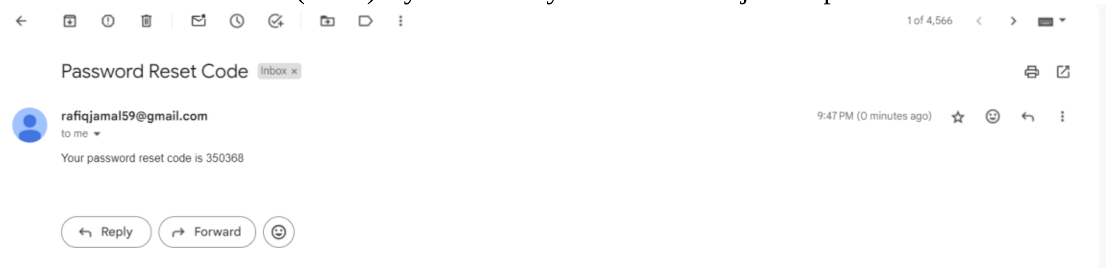

https://www.kyawzin.dom/forgot-password.php
Forgot Password
Enter your email address
kyawzinlatt111187@gmail.com
Continue

Figure 27 Recovering Password with Gmail



Password Reset Code

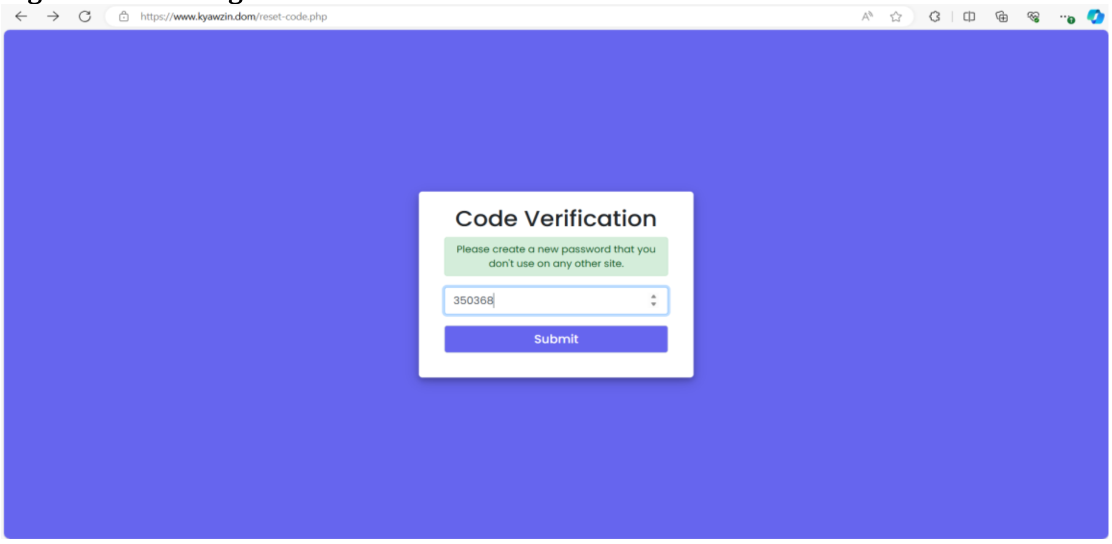
rafiqjamal59@gmail.com

to me

9:47 PM (0 minutes ago)

Your password reset code is 350368

Figure 28 Receving OTP



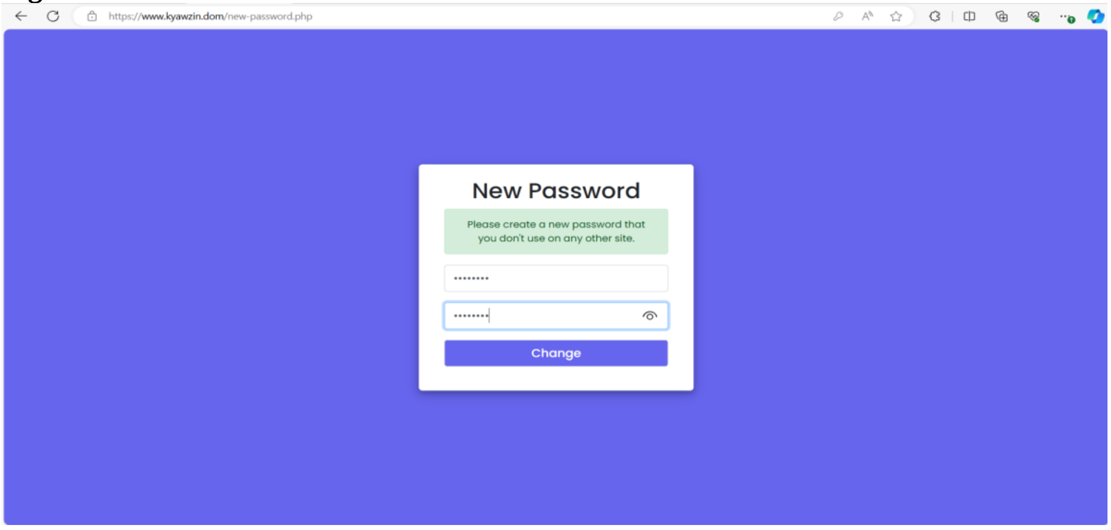
https://www.kyawzin.dom/reset-code.php

Code Verification

Please create a new password that you don't use on any other site.

350368

Figure 29 Verification with OTP

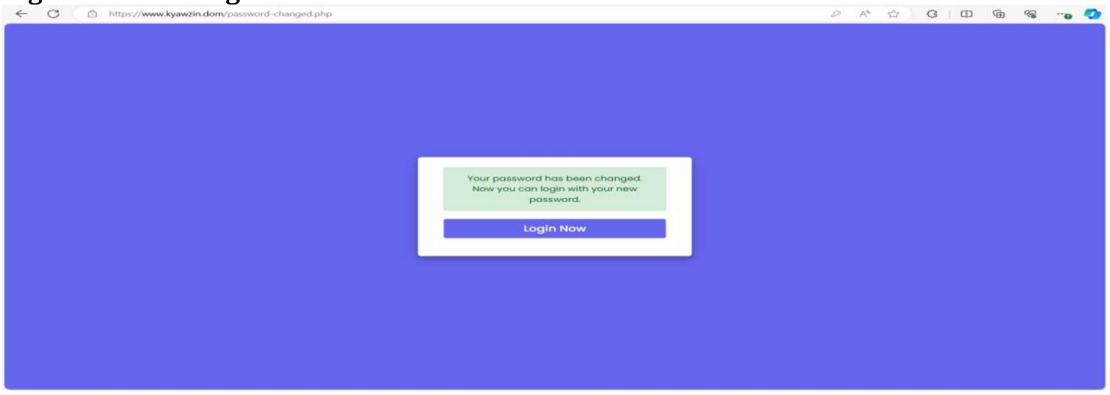


https://www.kyawzin.dom/new-password.php

New Password

Please create a new password that you don't use on any other site.

Figure 30 Entering New Password



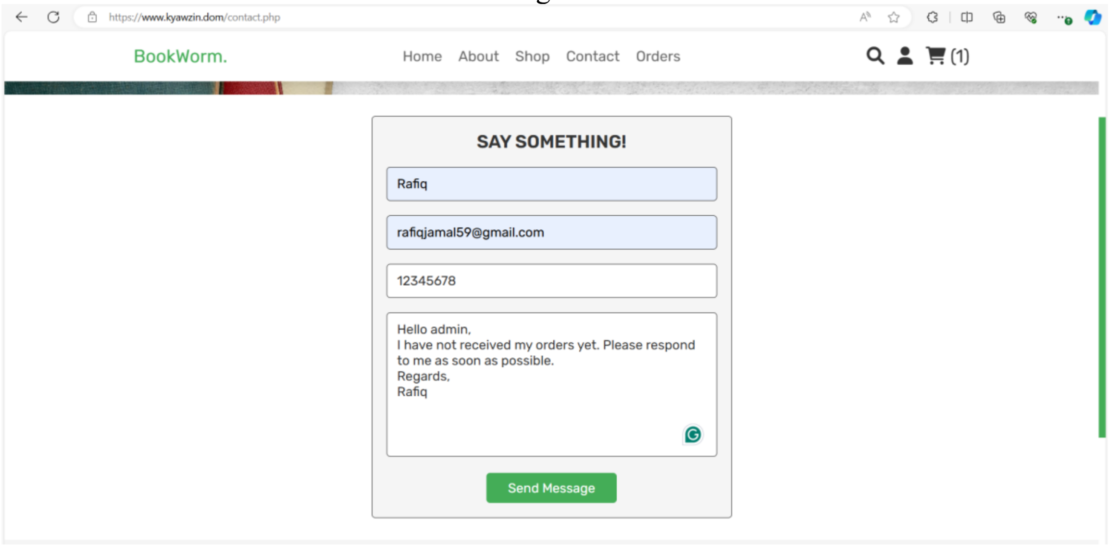
https://www.kyawzin.dom/password-changed.php

Your password has been changed. Now you can login with your new password.

Figure 31 Successfully Changed Password

4.2.3 Storing Encrypted Messages in Database

Messages are stored securely in the database. In case the database has been compromised, malicious actors cannot read the users' messages.


https://www.kyawzin.dom/contact.php

BookWorm. Home About Shop Contact Orders (1)

SAY SOMETHING!

Rafiq

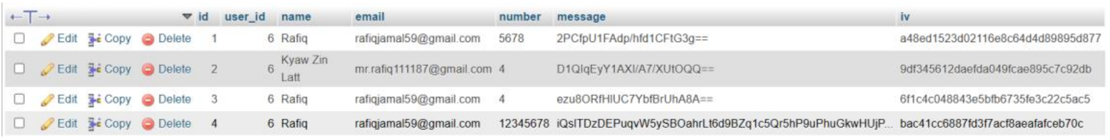
rafiqjamal59@gmail.com

12345678

Hello admin,
I have not received my orders yet. Please respond to me as soon as possible.
Regards,
Rafiq

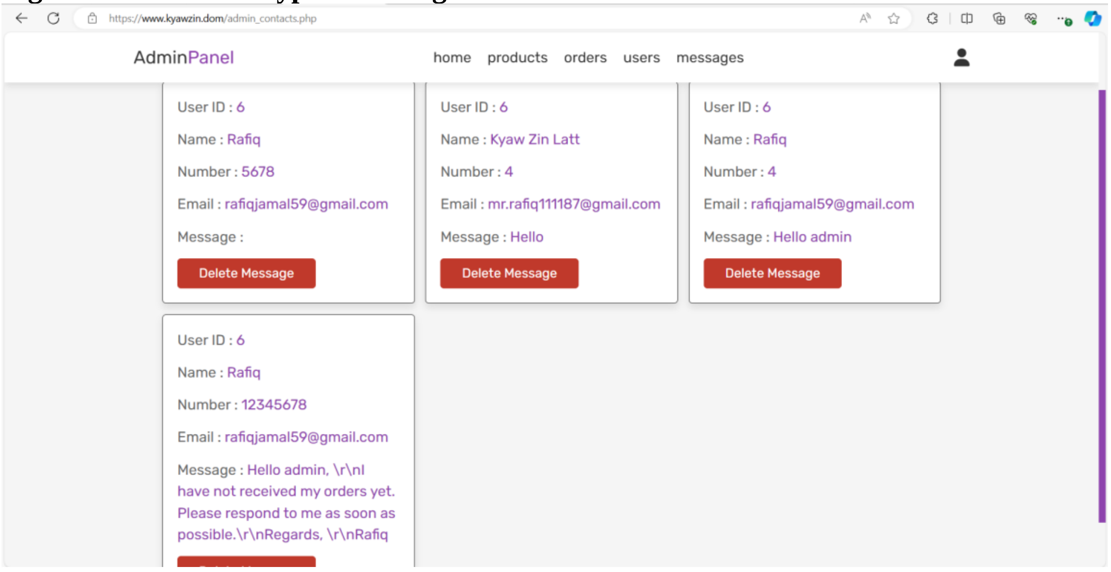
Send Message

Figure 32 User Sending Message



	id	user_id	name	email	number	message	iv
Edit Copy Delete	1	6	Rafiq	rafiqjamal59@gmail.com	5678	2PCfpU1FAdp/hfd1CFtG3g==	a48ed1523d02116e8c64d4d89895d877
Edit Copy Delete	2	6	Kyaw Zin Latt	mr.rafiq111187@gmail.com	4	D1QIqEyY1AXI/A7/XUtOQQ==	9df345612daefda049fcae895c7c92db
Edit Copy Delete	3	6	Rafiq	rafiqjamal59@gmail.com	4	ezu8ORfHIUC7YbfBrUhA8A==	6f1c4c048843e5bfb6735fe3c22c5ac5
Edit Copy Delete	4	6	Rafiq	rafiqjamal59@gmail.com	12345678	iQsITDzDEPuqvW5ySBOahrLt6d9BZq1c5Qr5hP9uPhuGkwHUjP...	bac41cc6887fd3f7acf8aeafafceb70c

Figure 33 Stores Encrypted Message with IV


https://www.kyawzin.dom/admin_contacts.php

AdminPanel home products orders users messages

User ID : 6
Name : Rafiq
Number : 5678
Email : rafiqjamal59@gmail.com
Message :
Delete Message

User ID : 6
Name : Kyaw Zin Latt
Number : 4
Email : mr.rafiq111187@gmail.com
Message : Hello
Delete Message

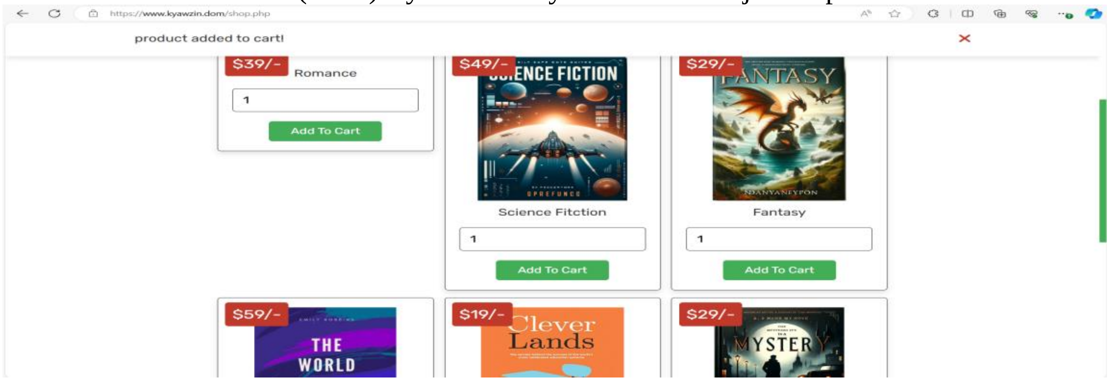
User ID : 6
Name : Rafiq
Number : 4
Email : rafiqjamal59@gmail.com
Message : Hello admin
Delete Message

User ID : 6
Name : Rafiq
Number : 12345678
Email : rafiqjamal59@gmail.com
Message : Hello admin, \r\nI have not received my orders yet. Please respond to me as soon as possible.\r\nRegards, \r\nRafiq

Figure 34 Reading Messages from Admin Panel

4.2.4 Shopping Books from Bookworm

Users can shop a variety of books from the website by adding into the cart. There is also the checkout procedure when a user wants to pay for the items in their cart. Users can also see their order details from the order tab. Admin can see users' details and order details from admin panel.


https://www.kyawzin.dom/shop.php

product added to cart!

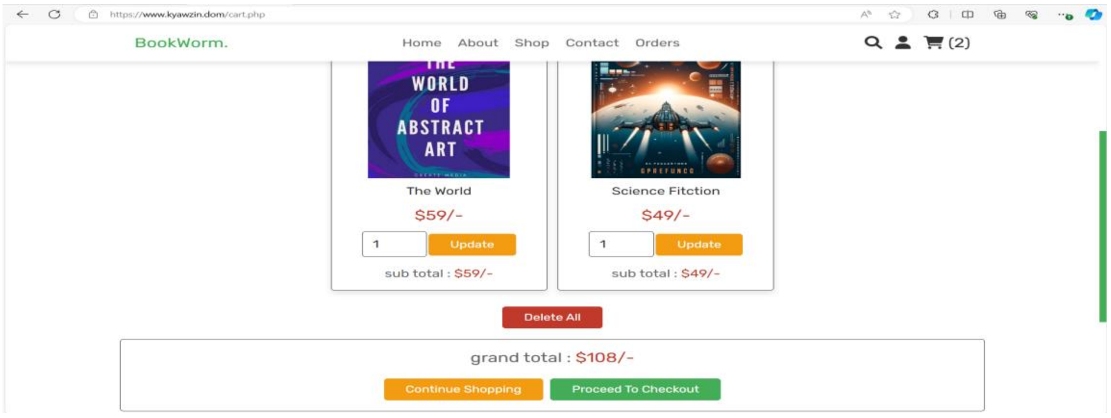
$39/- Romance
1
Add To Cart

$49/- Science Fitction
1
Add To Cart

$29/- Fantasy
1
Add To Cart

$59/-
$19/-
$29/-

Figure 35 Adding Products into Cart


https://www.kyawzin.dom/cart.php

BookWorm. Home About Shop Contact Orders (2)

The World
$59/-
1 Update
sub total : $59/-

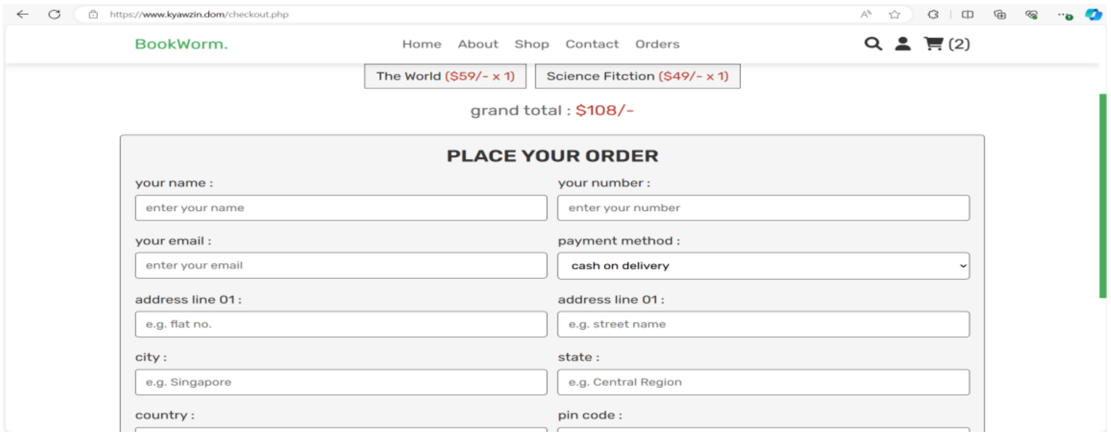
Science Fitction
$49/-
1 Update
sub total : $49/-

Delete All

grand total : $108/-

Continue Shopping Proceed To Checkout

Figure 36 Proceeding to Checkout from Cart


https://www.kyawzin.dom/checkout.php

BookWorm. Home About Shop Contact Orders (2)

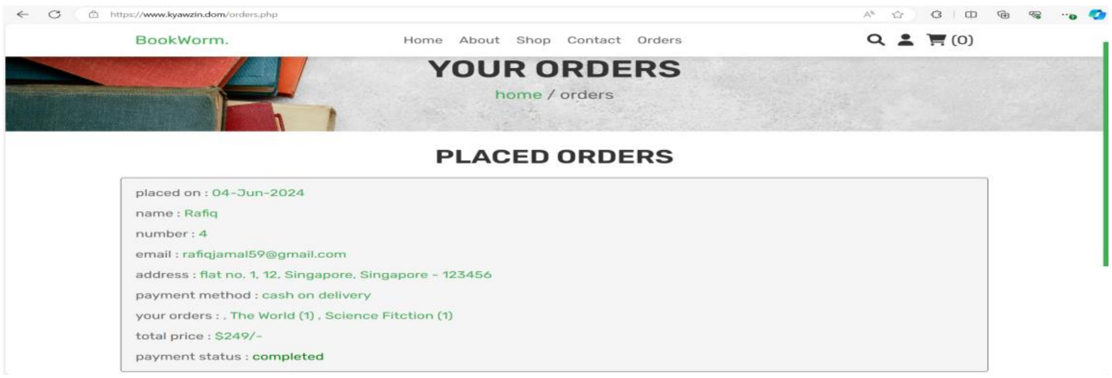
The World ($59/- x 1) Science Fitction ($49/- x 1)

grand total : $108/-

PLACE YOUR ORDER

your name : enter your name	your number : enter your number
your email : enter your email	payment method : cash on delivery
address line 01 : e.g. flat no.	address line 01 : e.g. street name
city : e.g. Singapore	state : e.g. Central Region
country :	pin code :

Figure 37 Entering Details for Order Confirmation


https://www.kyawzin.dom/orders.php

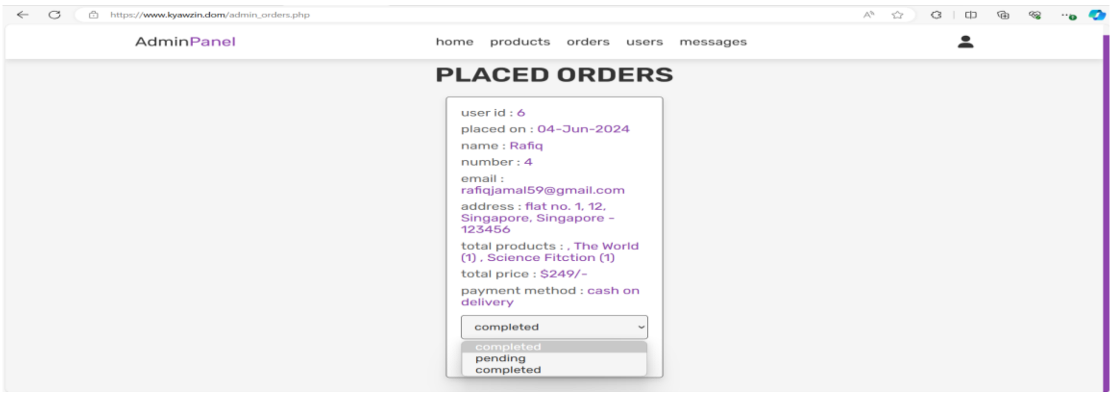
BookWorm. Home About Shop Contact Orders (0)

YOUR ORDERS
home / orders

PLACED ORDERS

placed on : 04-Jun-2024
name : Rafiq
number : 4
email : rafiqjamal59@gmail.com
address : flat no. 1, 12, Singapore, Singapore - 123456
payment method : cash on delivery
your orders : , The World (1) , Science Fitction (1)
total price : $249/-
payment status : completed

Figure 38 Order Details in User Interface

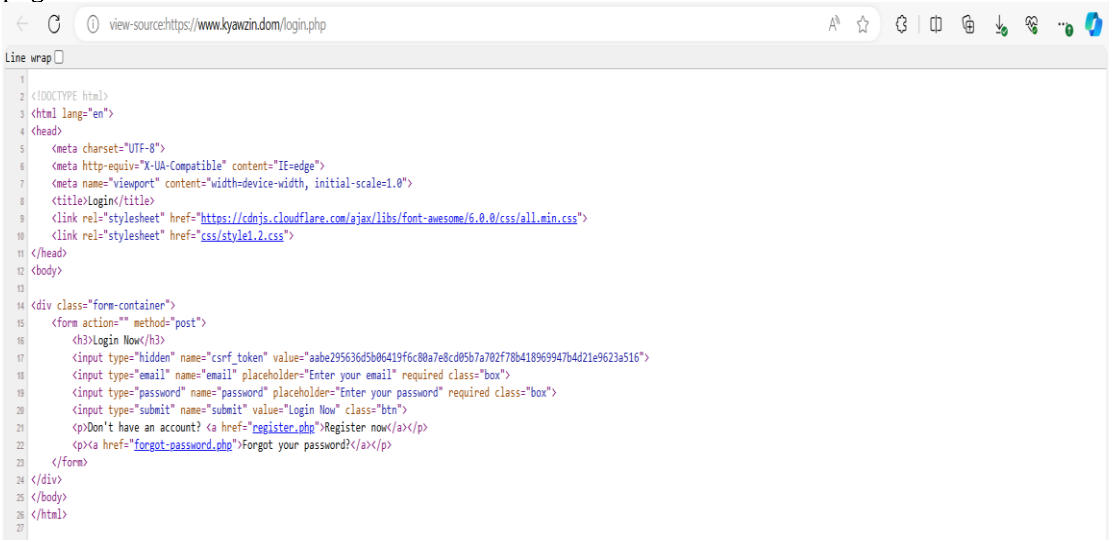


user id : 6
placed on : 04-Jun-2024
name : Rafiq
number : 4
email : rafiqjamal59@gmail.com
address : flat no. 1, 12, Singapore, Singapore - 123456
total products : , The World (1) , Science Fitction (1)
total price : $249/-
payment method : cash on delivery

Figure 39 Managing Order in Admin Panel

4.2.5 CSRF Token Generation

As CSRF tokens generation is implemented, the CSRF token will be seen when viewing the page source.



```html

<!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>Login</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">
    <link rel="stylesheet" href="css/style1.2.css">
</head>
<body>

<div class="form-container">
    <form action="" method="post">
        <h3>Login Now</h3>
        <input type="hidden" name="csrf_token" value="aabe295636d5b06419f6c80a7e8cd05b7a702f78b418969947b4d21e9623a516">
        <input type="email" name="email" placeholder="Enter your email" required class="box">
        <input type="password" name="password" placeholder="Enter your password" required class="box">
        <input type="submit" name="submit" value="Login Now" class="btn">
        <p>Don't have an account? <a href="register.php">Register now</a></p>
        <p><a href="forgot-password.php">Forgot your password?</a></p>
    </form>
</div>
</body>
</html>
```

Figure 40 CSRF Token

4.3 Penetration Testing

Penetration testing will be performed on this section. Common attacks such are cross-site scripting and SQL injection are demonstrated. Moreover, automated testing with OWASP ZAP has also been conducted.

4.3.1 SQL Injection

Due to the implementation of SQL injection detection system, when a user tries to input sql injection statements, the warning of input validation and error alerts will be shown.

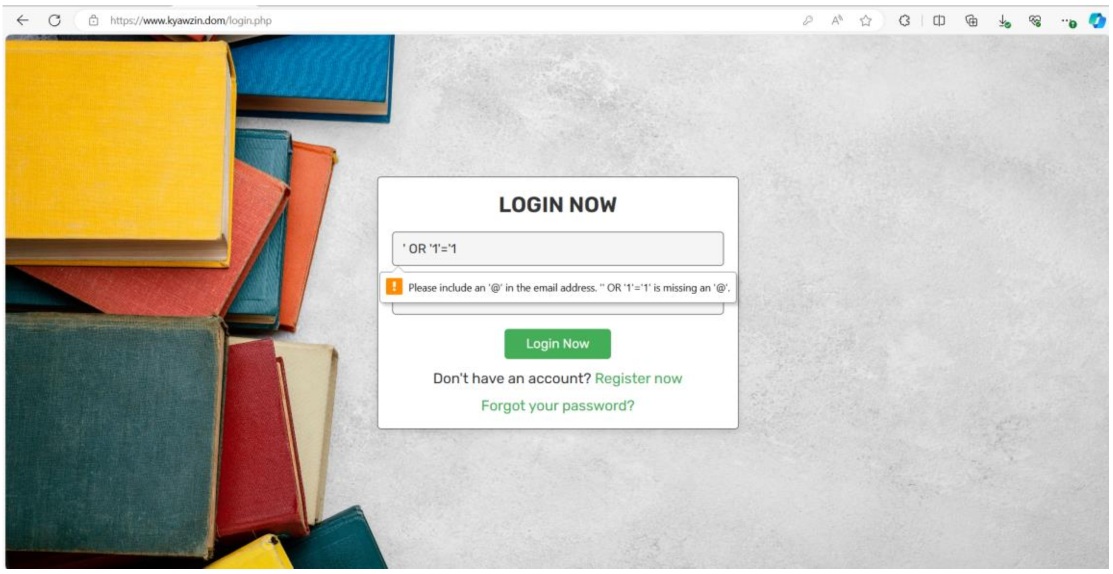

https://www.kyawzin.dom/login.php
LOGIN NOW
' OR '1'='1
Please include an '@' in the email address. '' OR '1'='1' is missing an '@'.
Login Now
Don't have an account? Register now
Forgot your password?

Figure 41 Warning Message

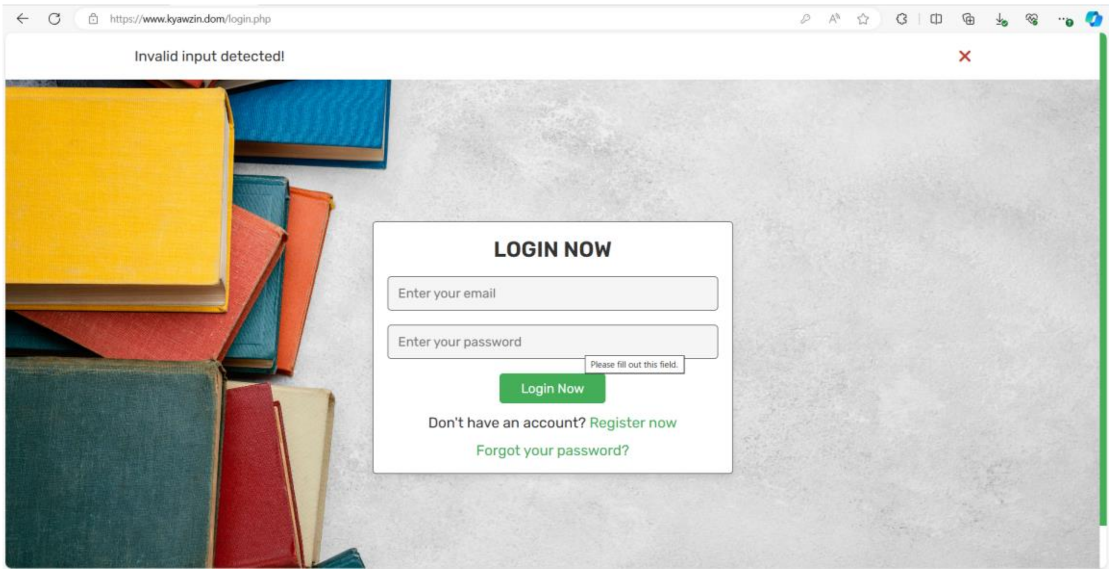
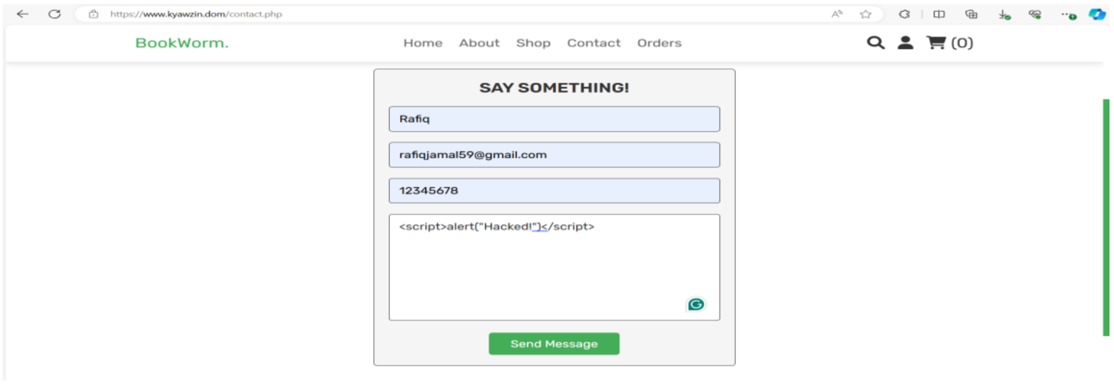


Figure 42 SQLi Attack Alert

4.3.2 XSS Attack

The XSS attack also does not work due to the implementation of input validation and code sanitizations.



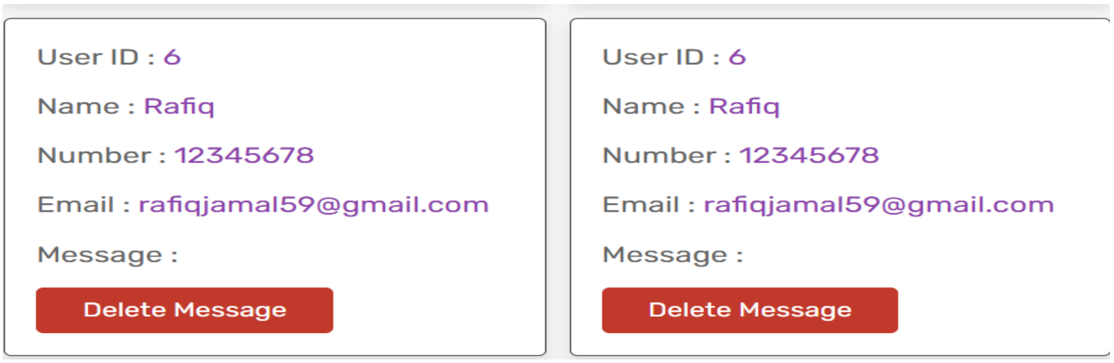
Rafiq

rafiqjamal59@gmail.com

12345678

```
<script>alert("Hacked!")</script>
```

Figure 43 Performing XSS Attack



User ID : 6

Name : Rafiq

Number : 12345678

Email : rafiqjamal59@gmail.com

Message :

User ID : 6

Name : Rafiq

Number : 12345678

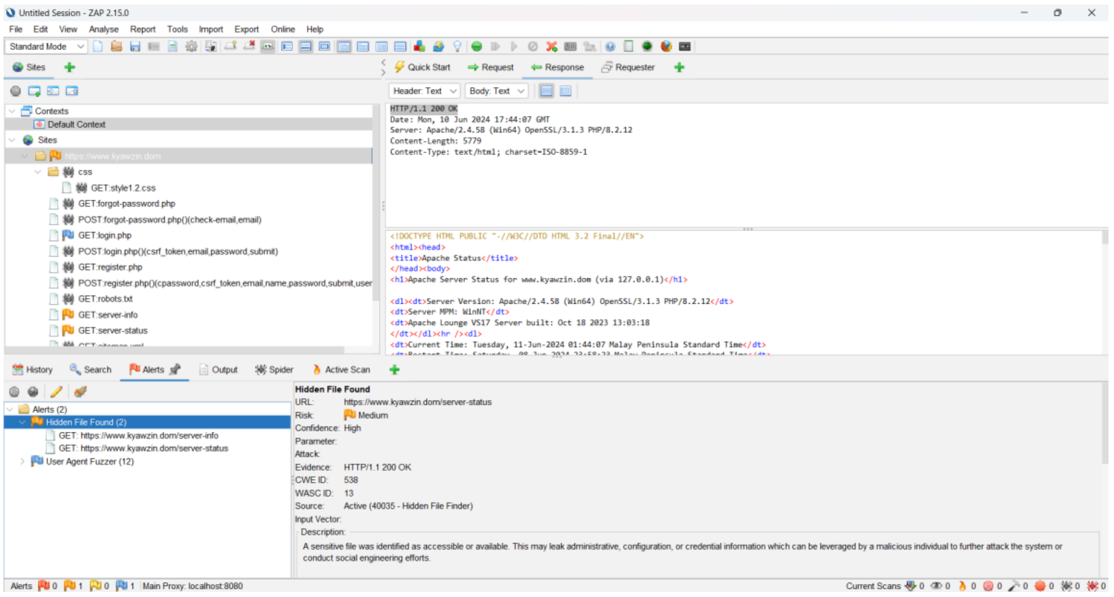
Email : rafiqjamal59@gmail.com

Message :

Figure 44 XSS Attack Fails

4.3.3 Automated Testing with OWASP ZAP

I have utilized OWASP ZAP automated penetration testing tool to carry out the security testing. I will not explain in depth the testing, however, during the testing only medium risk is found.



```
Contexts
  Default Context
Sites
  https://www.kyawzin.dom
    css
      GET:style1.2.css
    GET:forgot-password.php
    POST:forgot-password.php()(check-email,email)
    GET:login.php
    POST:login.php()(csrf_token,email,password,submit)
    GET:register.php
    POST:register.php()(cpassword,csrf_token,email,name,password,submit,user
    GET:robots.txt
    GET:server-info
    GET:server-status
```

```
HTTP/1.1 200 OK
Date: Mon, 10 Jun 2024 17:44:07 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
Content-Length: 5779
Content-Type: text/html; charset=ISO-8859-1
```

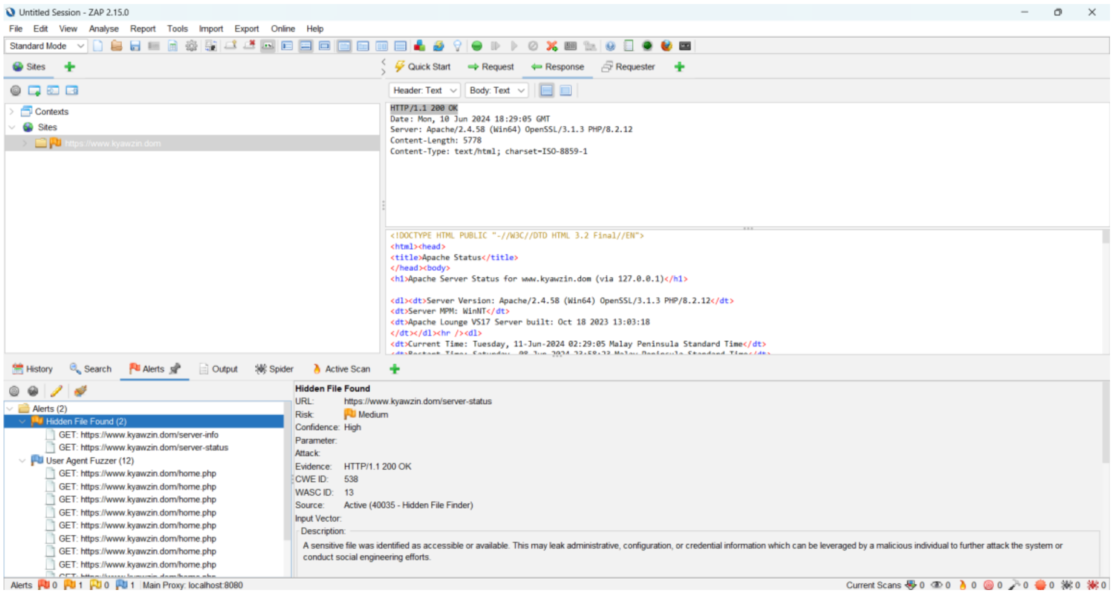
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html><head>
<title>Apache Status</title>
</head><body>
<h1>Apache Server Status for www.kyawzin.dom (via 127.0.0.1)</h1>

<dl><dt>Server Version: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12</dt>
<dt>Server MPM: WinNT</dt>
<dt>Apache Lounge VS17 Server built: Oct 18 2023 13:03:18
</dt></dl><hr /><dl>
<dt>Current Time: Tuesday, 11-Jun-2024 01:44:07 Malay Peninsula Standard Time</dt>
```

```
Alerts (2)
  Hidden File Found (2)
    GET: https://www.kyawzin.dom/server-info
    GET: https://www.kyawzin.dom/server-status
  User Agent Fuzzer (12)
```

```
Hidden File Found
URL:          https://www.kyawzin.dom/server-status
Risk:         Medium
Confidence:   High
Parameter:
Attack:
Evidence:     HTTP/1.1 200 OK
CWE ID:       538
WASC ID:      13
Source:       Active (40035 - Hidden File Finder)
Input Vector:
Description:
A sensitive file was identified as accessible or available. This may leak administrative, configuration, or credential information which can be leveraged by a malicious individual to further attack the system or conduct social engineering efforts.
```

```
Alerts 0 1 0 1   Main Proxy: localhost:8080          Current Scans 0 0 0 0 0 0 0 0
```

Figure 45 Testing on login.php



```
Contexts
Sites
  https://www.kyawzin.dom
```

```
HTTP/1.1 200 OK
Date: Mon, 10 Jun 2024 18:29:05 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
Content-Length: 5778
Content-Type: text/html; charset=ISO-8859-1
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html><head>
<title>Apache Status</title>
</head><body>
<h1>Apache Server Status for www.kyawzin.dom (via 127.0.0.1)</h1>

<dl><dt>Server Version: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12</dt>
<dt>Server MPM: WinNT</dt>
<dt>Apache Lounge VS17 Server built: Oct 18 2023 13:03:18
</dt></dl><hr /><dl>
<dt>Current Time: Tuesday, 11-Jun-2024 02:29:05 Malay Peninsula Standard Time</dt>
```

```
Alerts (2)
  Hidden File Found (2)
    GET: https://www.kyawzin.dom/server-info
    GET: https://www.kyawzin.dom/server-status
  User Agent Fuzzer (12)
    GET: https://www.kyawzin.dom/home.php
    GET: https://www.kyawzin.dom/home.php
    GET: https://www.kyawzin.dom/home.php
    GET: https://www.kyawzin.dom/home.php
    GET: https://www.kyawzin.dom/home.php
    GET: https://www.kyawzin.dom/home.php
    GET: https://www.kyawzin.dom/home.php
    GET: https://www.kyawzin.dom/home.php
```

```
Hidden File Found
URL:          https://www.kyawzin.dom/server-status
Risk:         Medium
Confidence:   High
Parameter:
Attack:
Evidence:     HTTP/1.1 200 OK
CWE ID:       538
WASC ID:      13
Source:       Active (40035 - Hidden File Finder)
Input Vector:
Description:
A sensitive file was identified as accessible or available. This may leak administrative, configuration, or credential information which can be leveraged by a malicious individual to further attack the system or conduct social engineering efforts.
```

```
Alerts 0 1 0 1   Main Proxy: localhost:8080          Current Scans 0 0 0 0 0 0 0 0
```

Figure 46 Testing on home.php

5. CONCLUSION

In conclusion, the development of Bookworm web application facilitates the necessity of security for e-commerce websites in this advanced technology era. During the building and implementation of this web application, I have mainly focused on confidentiality, integrity and availability (CIA) of user data, the security features of website, convenient of shopping experience for users, and safeguarding of users' transactions. This is my final year project report and I have practiced all the knowledge which I have learned throughout my classes. With these reflections of my learning journey, I have successfully implemented robust security features, functionalities and various cybersecurity measures.

REFERENCES

Hatzivasilis, G. (2017). Password-Hashing Status. *Cryptography*, *1*(2), 10. https://doi.org/10.3390/cryptography1020010

Hatzivasilis, G. (2020). Password Management: How Secure Is Your Login Process?. In *Handbook of Cybersecurity* (pp. 159-173). Springer. https://doi.org/10.1007/978-3-030-62433-0_10

Fournaris, A., Tselios, C., Haleplidis, E., Athanasopoulos, E., Dionysiou, A., Mitropoulos, D., Louridas, P., Christou, G., Athanatos, M., Hatzivasilis, G., Georgopoulos, K., Kalogeros, C., Kotselidis, C., Vogl, S., Hamon, F., & Ioannidis, S. (2023). Providing security assurance & hardening for open source software/hardware: The SecOPERA approach. *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 80-86. https://doi.org/10.1109/CAMAD59638.2023.10478410

Riabov, V. (2006). Simple Mail Transfer Protocol (SMTP). In M. Pagani (Ed.), *Encyclopedia of Internet Technologies and Applications* (pp. 572-578). IGI Global.

Nurhandhi, M., & Suhendar, A. (2023). Improving Firebase BaaS Service Security in Counseling Chat Applications: AES-256 and CBC Approach for End-to-End Encryption. *JISA (Jurnal Informatika dan Sains)*, 6(2), 153-160. https://doi.org/10.31326/jisa.v6i2.1783

Mailgun. (n.d.). Which SMTP port should I use? Understanding ports 25, 465, & 587. Retrieved from https://www.mailgun.com/blog/email/which-smtp-port-understanding-ports-25-465-587/

CodingWithElias. (n.d.). *Online Book Store*. GitHub. Retrieved June 11, 2024, from https://github.com/codingWithElias/online-book-store

Laudon, K. C., & Traver, C. G. (2020). *E-commerce 2020: Business, Technology, and Society*. Pearson.

Schneier, B. (2015). *Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World*. W.W. Norton & Company.

Bonneau, J., Herley, C., Van Oorschot, P. C., & Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *2012 IEEE Symposium on Security and Privacy* (pp. 553-567). IEEE. https://doi.org/10.1109/SP.2012.44

Rescorla, E. (2000). *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley.

Daemen, J., & Rijmen, V. (2002). *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer.

Silva, M. J. (2014). *SQL Performance Explained*. SQLPerformance.com.

OWASP Foundation. (n.d.). *OWASP Top Ten*. Retrieved from https://owasp.org/www-project-top-ten/

McClure, S., Scambray, J., & Kurtz, G. (2009). *Hacking Exposed Web Applications* (3rd ed.). McGraw-Hill.