Restaurant Management System

6005 CEM Security

Group Members:
Damien Tan Lek Khee
Lim Pau Thing
Ooi Ying Jie

TABLE OF CONTENTS

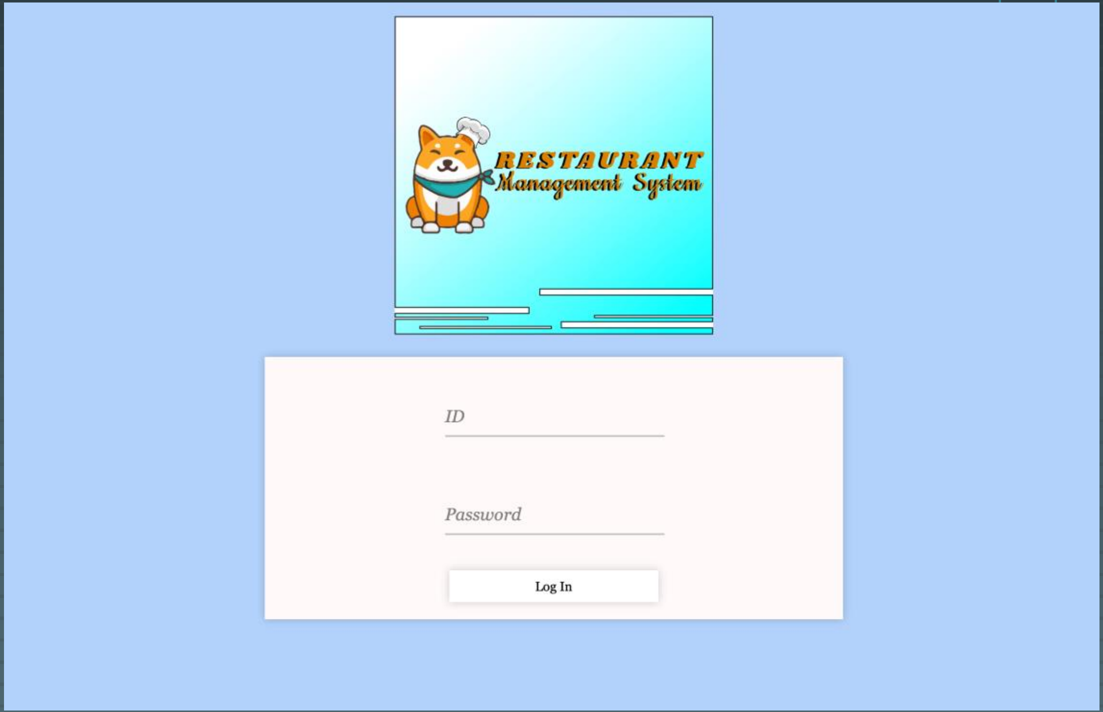
01

Introduction

System Overview

Restaurant Management System

- Facilitate ordering and reservation processes for restaurant staff.
- Staff can place food orders.
- Staff can manage customer reservations.



02

SQL Statement

Modify SQL Statements

- **SQL Injection:**
  - A common attack where malicious SQL code is inserted into user input fields.
  - Attackers exploit vulnerabilities in poorly constructed SQL queries, gaining unauthorized access to a database.
- **Preventing SQL Injection:**
  - Prepared statements act as a robust defense against SQL injection attacks.

Modify SQL Statements

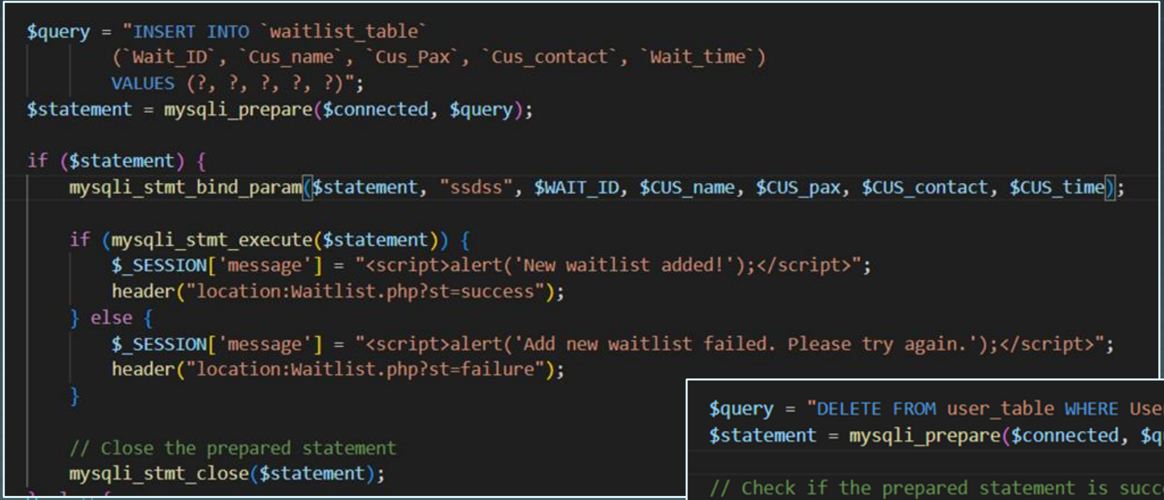
- **Use of Prepared Statements:**
  - Implementation employs '*mysqli_prepare*' to prepare an SQL statement for execution to counteract SQL injection attacks.

- **Parameter Binding and Data Type Specification:**
  - Parameters are binded using '*mysqli_stmt_bind_param*'.
  - Data type specification ensures accurate data processing.

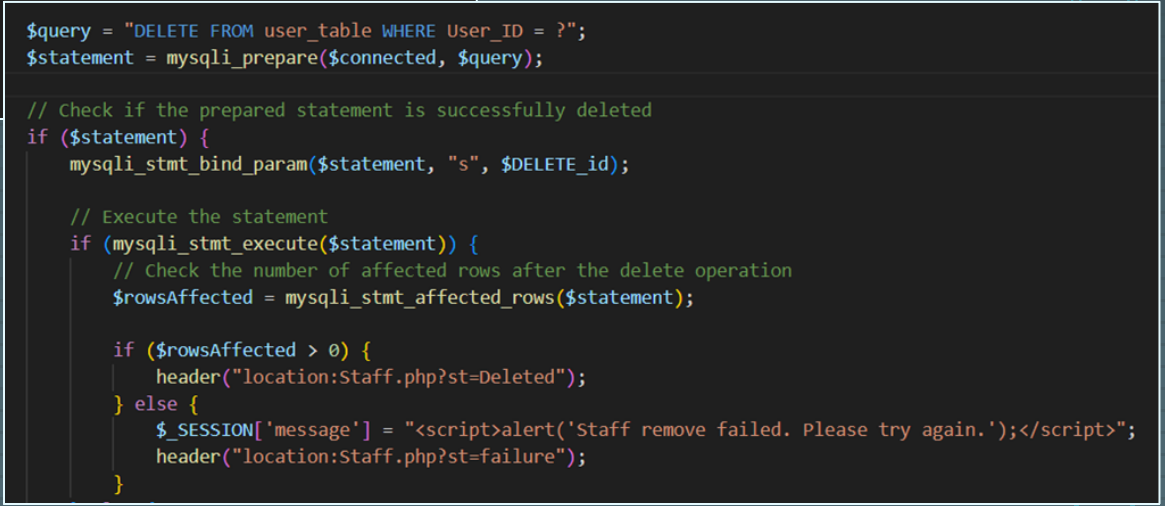
- **Query Execution:**
  - '*mysqli_stmt_execute'* is utilized to execute the prepared statement and fetch the data.
  - Enables safe processing of data, preventing malicious injections.

- **Operation Impact:**
  - '*mysqli_stmt_affected_rows*' is employed to return the number of rows affected by the executed statement.

Examples of SQL Statement



```php
$query = "INSERT INTO `waitlist_table`
        (`Wait_ID`, `Cus_name`, `Cus_Pax`, `Cus_contact`, `Wait_time`)
        VALUES (?, ?, ?, ?, ?)";
$statement = mysqli_prepare($connected, $query);

if ($statement) {
    mysqli_stmt_bind_param($statement, "ssdss", $WAIT_ID, $CUS_name, $CUS_pax, $CUS_contact, $CUS_time);

    if (mysqli_stmt_execute($statement)) {
        $_SESSION['message'] = "<script>alert('New waitlist added!');</script>";
        header("location:Waitlist.php?st=success");
    } else {
        $_SESSION['message'] = "<script>alert('Add new waitlist failed. Please try again.');</script>";
        header("location:Waitlist.php?st=failure");
    }

    // Close the prepared statement
    mysqli_stmt_close($statement);
```



```php
$query = "DELETE FROM user_table WHERE User_ID = ?";
$statement = mysqli_prepare($connected, $query);

// Check if the prepared statement is successfully deleted
if ($statement) {
    mysqli_stmt_bind_param($statement, "s", $DELETE_id);

    // Execute the statement
    if (mysqli_stmt_execute($statement)) {
        // Check the number of affected rows after the delete operation
        $rowsAffected = mysqli_stmt_affected_rows($statement);

        if ($rowsAffected > 0) {
            header("location:Staff.php?st=Deleted");
        } else {
            $_SESSION['message'] = "<script>alert('Staff remove failed. Please try again.');</script>";
            header("location:Staff.php?st=failure");
        }
```

04

Input Validation

Input Validation and Error Handling

- Implemented thorough input validation to mitigate security risks, specifically addressing concerns like SQL injection.

- Enhanced error handling codes to ensure clear error messages while safeguarding confidential data.

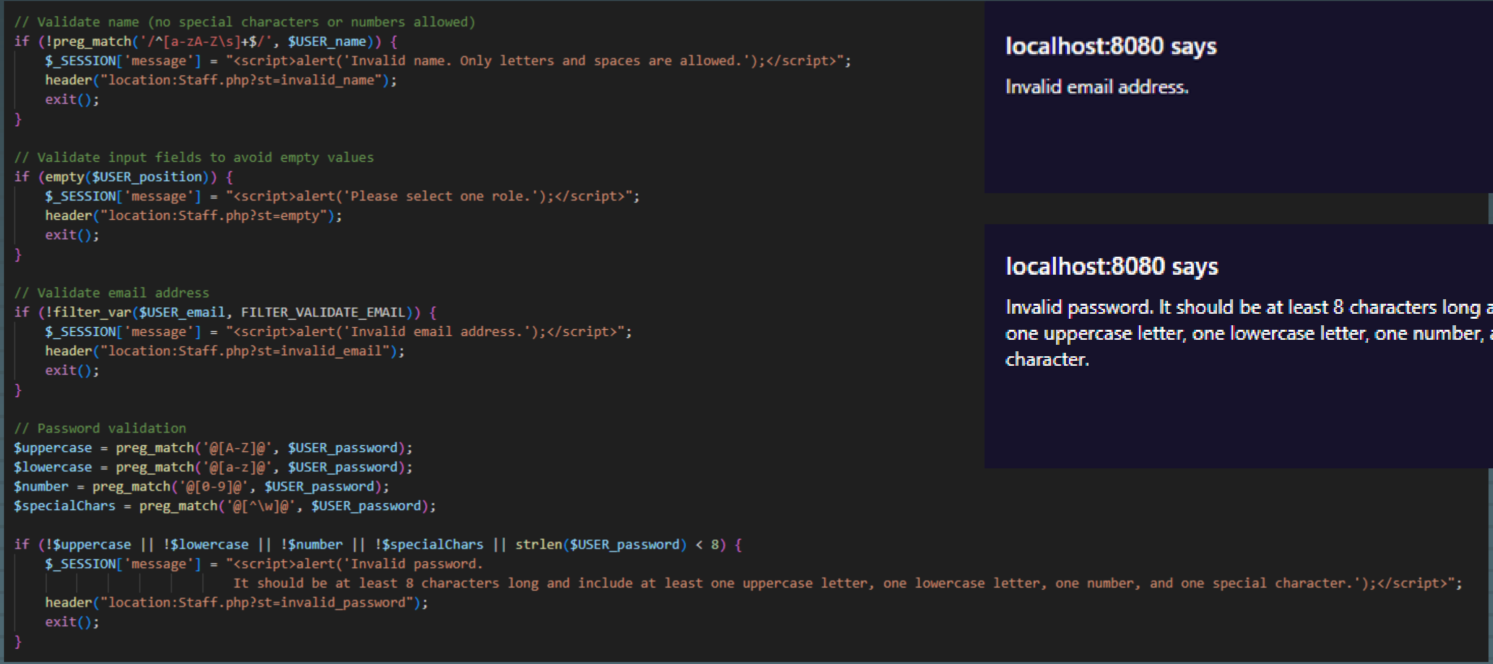
Reasons

- **Preventing Security Risks**
  - Ensures that user input is thoroughly checked before interacting with the database, reducing the risk of malicious attacks

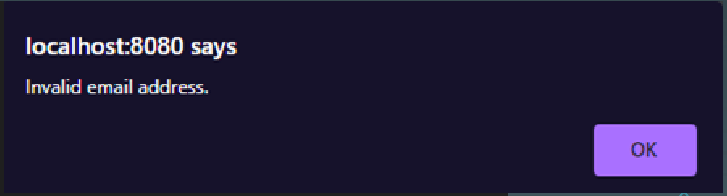
- **Clear Error Handling**
  - Helps in diagnosing issues without compromising data confidentiality

Examples of Input Validation

When a user attempts to register with an invalid email address or a password that doesn't meet the criteria, the system responds with a clear error message, guiding the user on the correct input format.

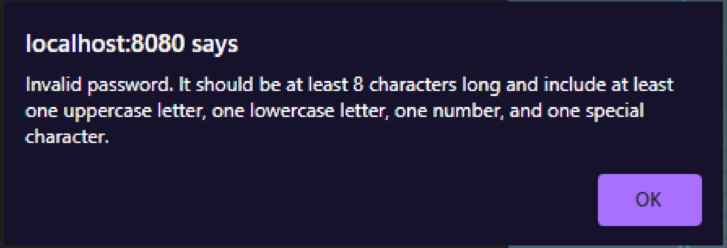


```php
// Validate name (no special characters or numbers allowed)
if (!preg_match('/^[a-zA-Z\s]+$/', $USER_name)) {
    $_SESSION['message'] = "<script>alert('Invalid name. Only letters and spaces are allowed.');</script>";
    header("location:Staff.php?st=invalid_name");
    exit();
}

// Validate input fields to avoid empty values
if (empty($USER_position)) {
    $_SESSION['message'] = "<script>alert('Please select one role.');</script>";
    header("location:Staff.php?st=empty");
    exit();
}

// Validate email address
if (!filter_var($USER_email, FILTER_VALIDATE_EMAIL)) {
    $_SESSION['message'] = "<script>alert('Invalid email address.');</script>";
    header("location:Staff.php?st=invalid_email");
    exit();
}

// Password validation
$uppercase = preg_match('@[A-Z]@', $USER_password);
$lowercase = preg_match('@[a-z]@', $USER_password);
$number = preg_match('@[0-9]@', $USER_password);
$specialChars = preg_match('@[^\w]@', $USER_password);

if (!$uppercase || !$lowercase || !$number || !$specialChars || strlen($USER_password) < 8) {
    $_SESSION['message'] = "<script>alert('Invalid password.
                            It should be at least 8 characters long and include at least one uppercase letter, one lowercase letter, one number, and one special character.');</script>";
    header("location:Staff.php?st=invalid_password");
    exit();
}
```



localhost:8080 says

Invalid email address.



localhost:8080 says

Invalid password. It should be at least 8 characters long and include at least one uppercase letter, one lowercase letter, one number, and one special character.

05 Password Hashing Algorithm

Password Hashing Algorithm

- Integrated a robust and trustworthy hashing algorithm to enhance the security of stored passwords.

- Improved existing password hashing codes to counter the risk associated with storing passwords in plaintext.

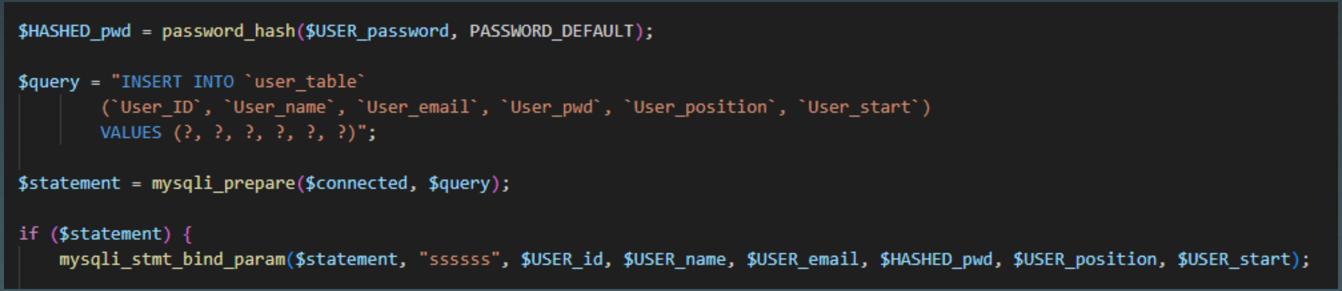
Reasons

- **Protecting Sensitive Data**
  - Ensures that passwords are securely stored, reducing the likelihood of unauthorized access even if the database is compromised.

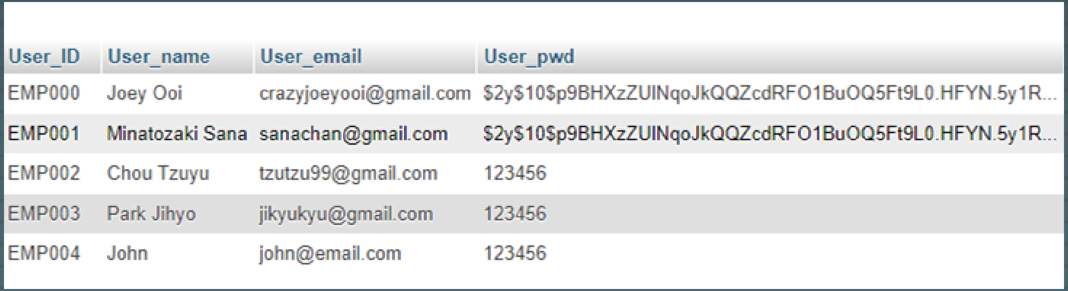
- **Lowering Security Risks**
  - Using a tested algorithm adds an extra layer of security, making it harder for attackers to decipher stored password information.

Examples of Password Hashing

- This function securely hashes the user's input password using a one-way cryptographic algorithm.



```
$HASHED_pwd = password_hash($USER_password, PASSWORD_DEFAULT);

$query = "INSERT INTO `user_table`
        (`User_ID`, `User_name`, `User_email`, `User_pwd`, `User_position`, `User_start`)
        VALUES (?, ?, ?, ?, ?, ?)";

$statement = mysqli_prepare($connected, $query);

if ($statement) {
    mysqli_stmt_bind_param($statement, "ssssss", $USER_id, $USER_name, $USER_email, $HASHED_pwd, $USER_position, $USER_start);
```

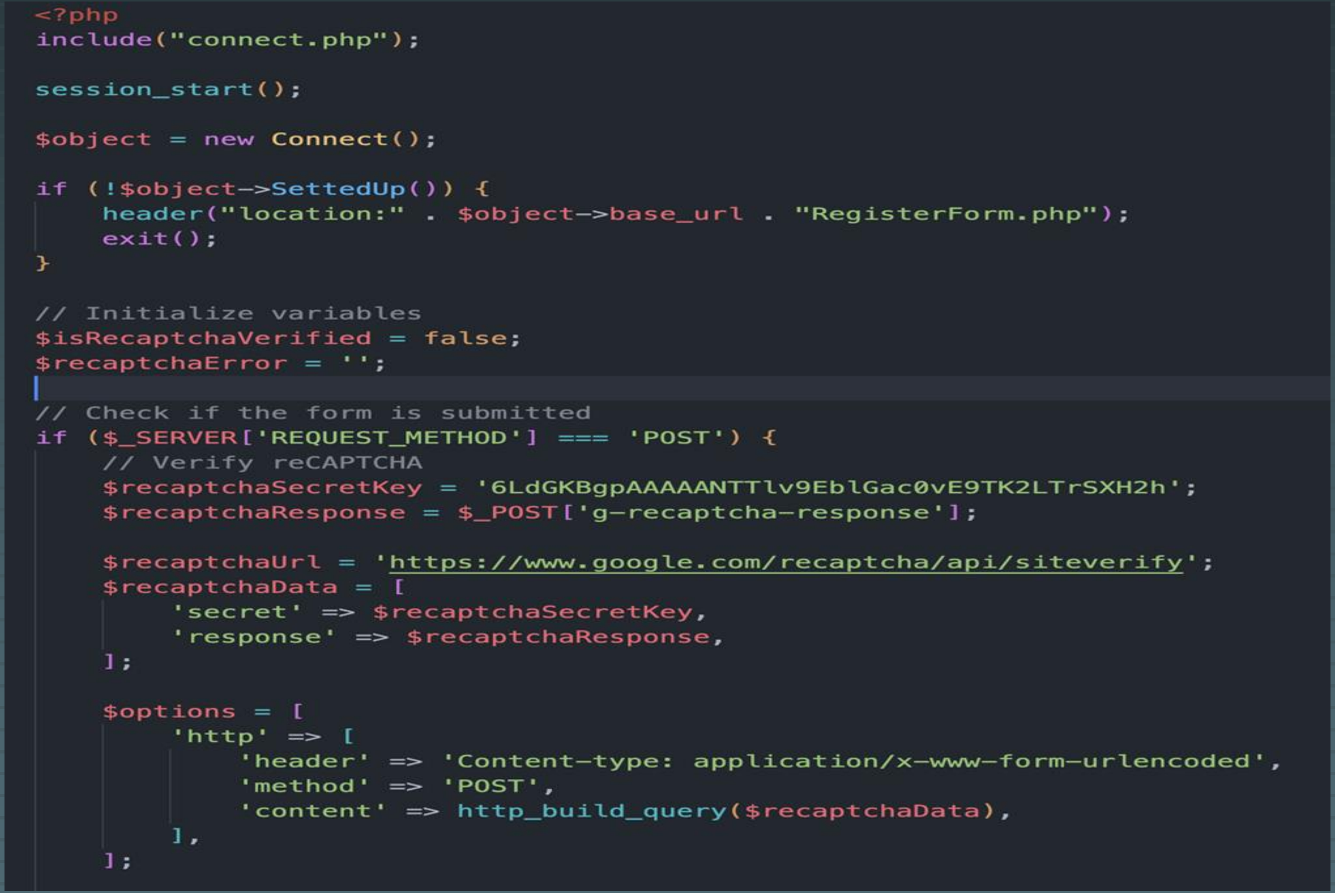


User_ID	User_name	User_email	User_pwd
EMP000	Joey Ooi	crazyjoeyooi@gmail.com	$2y$10$p9BHXzZUINqoJkQQZcdRFO1BuOQ5Ft9L0.HFYN.5y1R...
EMP001	Minatozaki Sana	sanachan@gmail.com	$2y$10$p9BHXzZUINqoJkQQZcdRFO1BuOQ5Ft9L0.HFYN.5y1R...
EMP002	Chou Tzuyu	tzutzu99@gmail.com	123456
EMP003	Park Jihyo	jikyukyu@gmail.com	123456
EMP004	John	john@email.com	123456

06

ReCAPTCHA

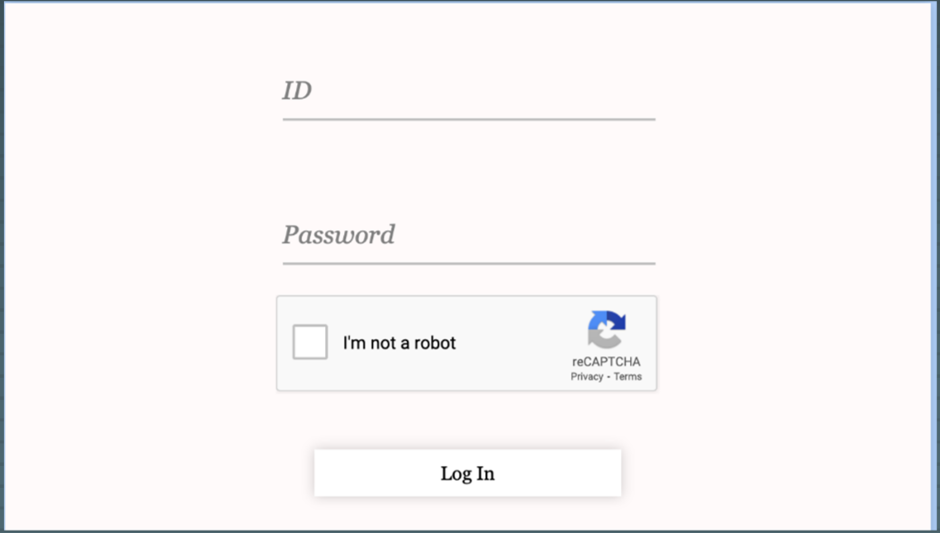
ReCAPTCHA Algorithm



```php
<?php
include("connect.php");

session_start();

$object = new Connect();

if (!$object->SettedUp()) {
    header("location:" . $object->base_url . "RegisterForm.php");
    exit();
}

// Initialize variables
$isRecaptchaVerified = false;
$recaptchaError = '';

// Check if the form is submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Verify reCAPTCHA
    $recaptchaSecretKey = '6LdGKBgpAAAAANTTlv9EblGac0vE9TK2LTrSXH2h';
    $recaptchaResponse = $_POST['g-recaptcha-response'];

    $recaptchaUrl = 'https://www.google.com/recaptcha/api/siteverify';
    $recaptchaData = [
        'secret' => $recaptchaSecretKey,
        'response' => $recaptchaResponse,
    ];

    $options = [
        'http' => [
            'header' => 'Content-type: application/x-www-form-urlencoded',
            'method' => 'POST',
            'content' => http_build_query($recaptchaData),
        ],
    ];
```

ReCAPTCHA Algorithm

- Mitigate automated login attempts and enhance overall authentication security.
- Provided an additional layer of defense against malicious bots and potential brute-force attacks.



07 Overall Achievements

Overall Achivements

- Mitigate the risk of SQL Injection
- Significantly lowered the likelihood of unauthorized data access
- Strengthened password security by storing only hashed representations
- Enhanced user authentication
- Additional layer of protection against automated attacks

Thank You

GitHub URL

Link: https://github.com/CrazyJoey4/Security

User Manual

1. System Overview:
The Restaurant Management System is a comprehensive solution designed to streamline various operations within
a restaurant. It encompasses staff management, tax management, payment processing, food order and category
management, food inventory control, table management, and waitlist management for available tables.

2. System Requirements:
Web server (XAMPP recommended)
PHP 7.0 or later
MySQL database (PHPMyAdmin for administration)
Modern web browser (Chrome, Firefox, Safari)

3. Installation Instructions:
Download and install XAMPP from https://www.apachefriends.org/index.html.
Launch XAMPP and start the Apache server and MySQL database.
Download the Restaurant Management System files.
Place the system files in the 'htdocs' folder within the XAMPP installation directory.
Import the provided database file into PHPMyAdmin.
4. System Access:
Open your web browser and navigate to localhost web page.
The login page will be displayed.
Enter the provided username and password to access the system.

5. Troubleshooting:
If you encounter any issues, ensure that XAMPP services are running.
Check the PHP and MySQL versions for compatibility.
Verify the database import process.