INTI International College Penang School of Engineering and Technology 3+0 Bachelor of Science (Hons) in Computer Science, in collaboration with Coventry University, UK 3+0 Bachelor of Science (Hons) in Computing, in collaboration with Coventry University, UK

Coursework cover sheet

Section A - To be completed by the student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Full Name:  Group Member 1: **Damien Tan Lek Khee**  Group Member 2: **Lim Pau Thing**  Group Member 3: **Ooi Ying Jie** | | | | |
| CU Student ID Number:  Group Member 1: **12672844**  Group Member 2: **12672877**  Group Member 3: **12672888** | | | | |
| Semester | : | 10 | | |
| Lecturer | : | **R.K.Krishnamoorthy** | | |
| Module Code and Title | : | **INT6005CEM Security** | | |
| Assignment No. / Title | : | **Secure Application Development** | | **50%** of Module Mark |
| Hand out date: **5/10/23** | | | Due date: **23/11/23** | |
| Penalties: No late work will be accepted. If you are unable to submit coursework on time due to extenuating circumstances, you may be eligible for an extension. Please consult the lecturer. | | | | |
| Declaration: I/we the undersigned confirm that I/we have read and agree to abide by the University regulations on plagiarism and cheating and Faculty coursework policies and procedures. I/we confirm that this piece of work is my/our own. I/we consent to appropriate storage of our work for plagiarism checking.  Signature(s):  Group Member 1:  Group Member 2:  Group Member 3: | | | | |

Section B - To be completed by the module leader

|  |  |  |
| --- | --- | --- |
| Intended learning outcomes assessed by this work:   1. Develop and evaluate software that addresses the most common and most severe security concerns. 2. Critically evaluate the security of an IT ecosystem. | | |
| Marking scheme | Max | Mark |
| 1. **Report**   *Refer to the section on Marking Scheme on detailed breakdown.* | **60** |  |
| 1. **System Functionality**   *Refer to the section on Marking Scheme on detailed breakdown.* | **40** |  |
| **Total** | **100** |  |
| **Lecturer’s Feedback** | | |
| **Internal Moderator’s Feedback** | | |

# Table of Contents

[Table of Contents 4](#_Toc151546576)

[1.0 Introduction and Overview 6](#_Toc151546577)

[2.0 Design 7](#_Toc151546578)

[2.1 Potential security issues (for each element of design) 7](#_Toc151546579)

[2.2 Recommendations (for each potential security issues) 8](#_Toc151546580)

[3.0 Implementation 9](#_Toc151546581)

[3.1 Security Considerations and Implementation 9](#_Toc151546583)

[3.2 Security Techniques and Methods 10](#_Toc151546584)

[4.0 Discussion 17](#_Toc151546585)

[4.1 Key Findings and Issue Resolution 17](#_Toc151546587)

[4.2 Continuous Improvement and Maintenance of Security 17](#_Toc151546588)

[5.0 Summary 18](#_Toc151546589)

[5.1 Conclusion 18](#_Toc151546591)

[5.2 Learning Outcome Achieved 18](#_Toc151546592)

[6.0 References 19](#_Toc151546593)

[7.0 Appendix 20](#_Toc151546594)

**Table of Figures**

[**Figure 1**: Food Ordering and Management System 6](#_Toc151546562)

[**Figure 2**: Input Validation (Login) 10](#_Toc151546563)

[**Figure 3**: Input Validation (Category Add) 10](#_Toc151546564)

[**Figure 4**: Input Validation (Category Edit) 11](#_Toc151546565)

[**Figure 5**: Password Verification & Hashing 11](#_Toc151546566)

[**Figure 6**: Update Category 12](#_Toc151546567)

[**Figure 7**: Delete Category 13](#_Toc151546568)

[**Figure 8**: Create Food Menu 14](#_Toc151546569)

[**Figure 9**: Update Food Menu 14](#_Toc151546570)

[**Figure 10**: Delete Food Menu 15](#_Toc151546571)

[**Figure 11**: Login Check 16](#_Toc151546572)

# Introduction and Overview

The system enhanced is a food ordering and reservation management system created for restaurant staff is to expedite ordering and reservation handling inside the company. It is critical to give security considerations top priority during the design and development phases of the system because it handles sensitive data and involves user interactions. The security measures built into the system to guarantee the privacy, availability, and integrity of data are described in this report.

Employees can make reservations and place food orders through the system, so strong security measures are needed to safeguard private information and guarantee that the application works as intended. Security considerations cover a wide range of topics, such as CAPTCHA integration, error handling, input validation, and vulnerability protection.

A screenshot of a login screen

Description automatically generated

**Figure 1**: Food Ordering and Management System

# Design

## Potential security issues (for each element of design)

**a. Inadequate Input Validation and Error Handling**

Inadequate input validation raises a security risk because it can result in SQL injection or other types of attacks. Malicious inputs have the potential to impair system performance or jeopardize the underlying database if they are not properly validated.

**b. Passwords Stored in PlainText**

Passwords stored in plaintext present a serious security risk. Without adequate password hashing mechanisms, there is a chance that sensitive data could be accessed by unauthorized parties.

**c. SQL Injection Vulnerabilities in Statements**

SQL injection attacks may be possible on the system due to insecure SQL statements. Malicious actors may alter database queries in the absence of parameterized queries or prepared statements, which could result in unauthorized data access.

**d. Absence of CAPTCHA**

The absence of CAPTCHA could leave the system vulnerable to automated attacks. The integration of CAPTCHA into the user registration and login procedures is necessary to resolve these problems.

## Recommendations (for each potential security issues)

1. **Inadequate Input Validation and Error Handling**

Implementation of thorough input validation codes to sanitize user inputs and prevent SQL injection or other malicious attacks. Additionally, error handling codes will be modified and added, if necessary, thus providing user-friendly messages without revealing sensitive information.

1. **Passwords Stored in PlainText**

Integration of reliable algorithm to enhance existing password hashing codes.

1. **SQL Injection Vulnerabilities in Statements**

Modifications of existing SQL statements by adding parameterized queries or prepared statements. Constant updates of database access credentials and restrict access to authorized personnel only.

1. **Absence of CAPTCHA**

Implementation of CAPTCHA codes hence preventing automated attacks during user login.

# Implementation



## Security Considerations and Implementation

1. **Comprehensive Input Validation and Error Handling**

The implementation includes the incorporation of comprehensive input validation codes in response to the potential security risk associated with inadequate input validation. These codes are made to efficiently sanitize user inputs, reducing the possibility of malicious attacks like SQL injection. Additionally, current error handling codes will be enhanced and changed as needed to guarantee the delivery of messages that are easy to understand without jeopardizing confidential data.

1. **Integration of Reliable Password Hashing Algorithm**

The implementation concentrates on the integration of a trustworthy algorithm to improve the current password hashing codes to counter the risk associated with passwords stored in plaintext. This guarantees the security of sensitive data by lowering the possibility of unauthorized access using a reliable and tested hashing algorithm.

1. **Modifications of SQL Statements**

By altering the current SQL statements, the potential vulnerability to SQL injection attacks caused by insecure SQL statements is mitigated. Incorporating prepared statements or parameterized queries lowers the likelihood of unauthorized data access considerably. The system's overall security is further strengthened by the implementation of restricted access that is only permitted to authorized personnel and the routine updating of database access credentials.

1. **Integration of CAPTCHA Codes**

The implementation plan calls for the integration of CAPTCHA codes to reduce the risk associated with the lack of CAPTCHA. By adding this, automated attacks against the user registration and login processes are thwarted.

## Security Techniques and Methods

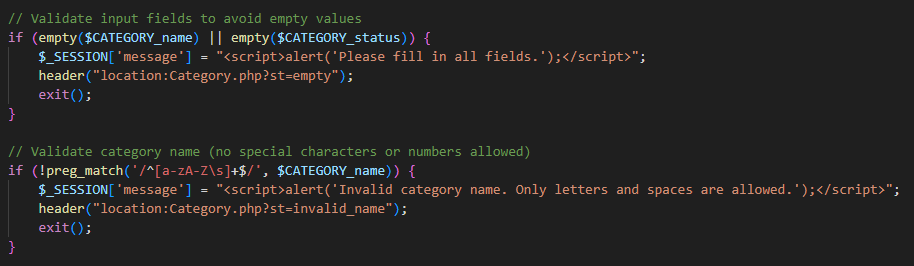
1. **Comprehensive Input Validation and Error Handling**

A screen shot of a computer code

Description automatically generated

**Figure 2**: Input Validation (Login Check)

Ensures that both $USER\_id and $USER\_pwd must not be empty, hence reducing the risk of unauthorized access.



**Figure 3**: Input Validation (Category Add)

Ensure that all $CATEGORY\_name and $CATEGORY\_status must not be empty. Special characters or numbers are considered invalid.

A computer screen shot of a code

Description automatically generated

**Figure 4**: Input Validation (Category Edit)

$CATEGORY\_status must be selected during the editing process to maintain data accuracy.

A computer screen shot of a black screen

Description automatically generated

**Figure 5**: Input Validation (Food Edit)

Ensure that $FOOD\_status must be selected.

A computer screen with text

Description automatically generated

**Figure 6**: Input Validation (Staff Edit)

Ensures that the $POSITION must be selected to maintain data accuracy.

A screen shot of a computer code

Description automatically generated

**Figure 7**: Input Validation (Table Edit)

Ensure that $STATUS must be selected.



**Figure 8**: Input Validation (Profile Edit)

Ensures that user information such as name, email, password, contact information, birthday, and gender be chosen or entered. Data security is improved by specific restrictions on special characters and acceptable formats for phone numbers and emails.

A screen shot of a computer code

Description automatically generated

**Figure 9**: Input Validation (Waitlist Add)

Ensure that $CUS\_contact must be a valid phone number.

A screen shot of a computer code

Description automatically generated

**Figure 10**: Input Validation (Waitlist Edit)

Similar to Waitlist add page, the $CONTACT must also be a valid phone number during the editing process.

1. **Integration of Reliable Password Hashing Algorithm**



A screen shot of a computer

Description automatically generated

**Figure 11**: Password Verification & Hashing

1. **Modifications of SQL Statements**

A computer screen shot of a black background with white text

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Figure 12**: SQL Statement Modifications (Category Update)

A black background with colorful text

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Figure 13**: SQL Statement Modifications (Category Delete)

A computer screen with text on it

Description automatically generated

A black background with white text

Description automatically generated

A computer screen with text

Description automatically generated

**Figure 14**: SQL Statement Modifications (Food Insert)

A computer code with many colorful text

Description automatically generated with medium confidence

A screen shot of a computer program

Description automatically generated

**Figure 15**: SQL Statement Modifications (Food Update)

A screen shot of a computer program

Description automatically generated

A black background with white text

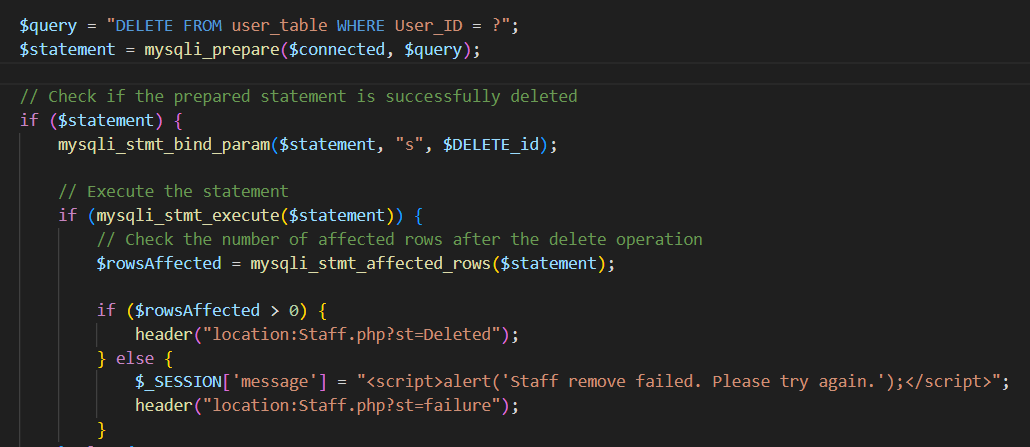
Description automatically generated

**Figure 16**: SQL Statement Modifications (Food Delete)

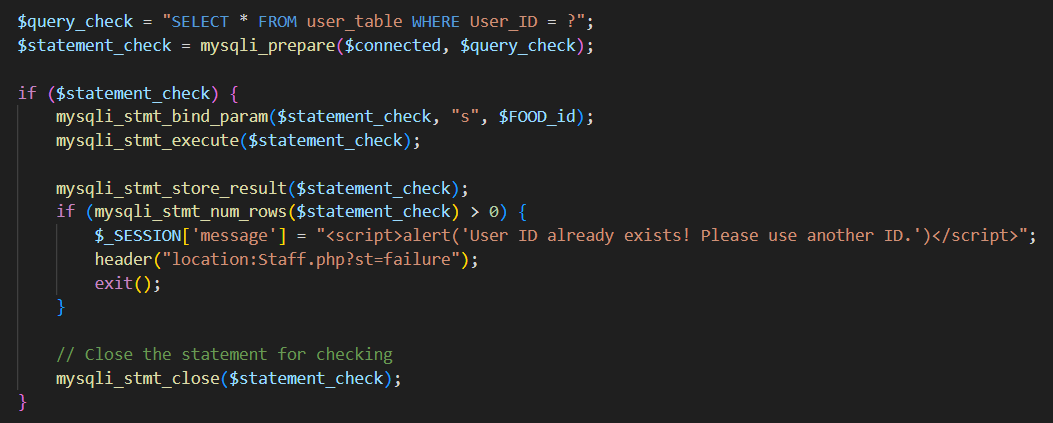
A screen shot of a computer code

Description automatically generated

**Figure 17**: SQL Statement Modifications (Login Check)



**Figure 18**: SQL Statement Modifications (Staff Delete)



A computer screen with text

Description automatically generated

**Figure 19**: SQL Statement Modifications (Staff Insert)

A screen shot of a computer program

Description automatically generated

**Figure 20**: SQL Statement Modifications (Staff Update)

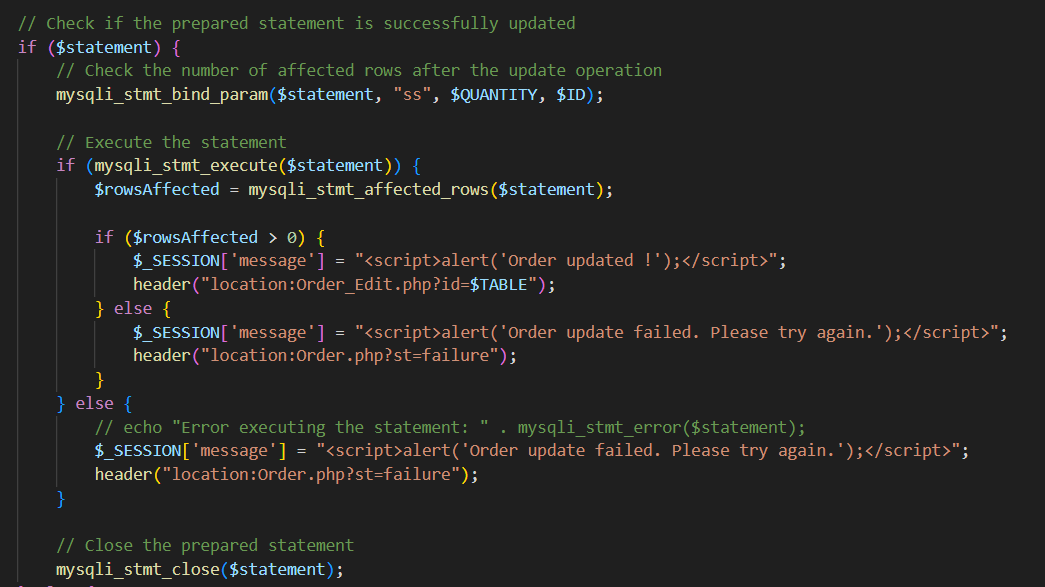
A screen shot of a computer program

Description automatically generated

**Figure 21**: SQL Statement Modifications (Order Delete)

A computer screen shot of a black background with white text

Description automatically generated



**Figure 22**: SQL Statement Modifications (Order Update)

A screen shot of a computer program

Description automatically generated

**Figure 23**: SQL Statement Modifications (Order Insert)

A computer screen shot of a program code

Description automatically generated

**Figure 24**: SQL Statement Modifications (Table Delete)

A screen shot of a computer program

Description automatically generated

**Figure 25**: SQL Statement Modifications (Table Update)

A screen shot of a computer program

Description automatically generated

**Figure 26**: SQL Statement Modifications (Table Insert)

A computer screen shot of a program code

Description automatically generated

**Figure 27**: SQL Statement Modifications (Tax Delete)

A computer screen shot of text

Description automatically generated

**Figure 28**: SQL Statement Modifications (Tax Insert)

A black background with white text

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Figure 29**: SQL Statement Modifications (Tax Update)

A computer screen shot of text

Description automatically generated

**Figure 30**: SQL Statement Modifications (Wait Insert)

A screen shot of a computer program

Description automatically generated

**Figure 31**: SQL Statement Modifications (Wait Update)

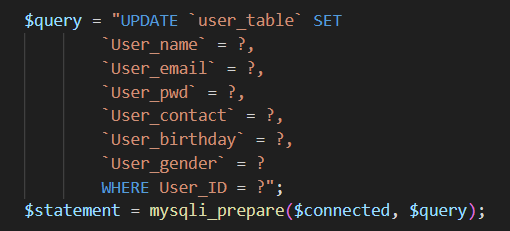
A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Figure 32**: SQL Statement Modifications (Wait Delete)



A screen shot of a computer program

Description automatically generated

**Figure 33**: SQL Statement Modifications (Profile Update)

The implementation relies heavily on the use of the 'mysqli\_prepare' function to generate prepared statements, which improves SQL query security by guarding against SQL injection attacks. The 'mysqli\_stmt\_bind\_param' function, which performs parameter binding and specifies the data types for each parameter, strengthens these prepared statements even more. To guarantee that the data being processed is accurate, data types like "i" for integers, "s" for strings, and "d" for decimals are essential. The 'mysqli\_stmt\_execute' function is then used to execute these prepared statements, resulting in secure and sanitised queries. The function 'mysqli\_stmt\_affected\_rows' is utilised to evaluate the effects of operations, especially in UPDATE or DELETE queries. This function enables the system to confirm if any rows were impacted, adding an extra degree of security and control to database operations.

1. **Integration of CAPTCHA Codes**

A screen shot of a computer program

Description automatically generated  
A screen shot of a computer code

Description automatically generated

**Figure 34**: reCAPTCHA Integration

Checks to see if the application setup is complete after initializing the required database connection and session. After the form is submitted, it retrieves the user's reCAPTCHA response and uses the secret key supplied to confirm it with the Google reCAPTCHA API. The $isRecaptchaVerified variable is set to true if the verification is successful; if not, an error message is assigned to $recaptchaError. The web application's security is improved by this implementation, which verifies that form submissions come from actual users and are not the result of automated scripts.

A black background with green and orange text

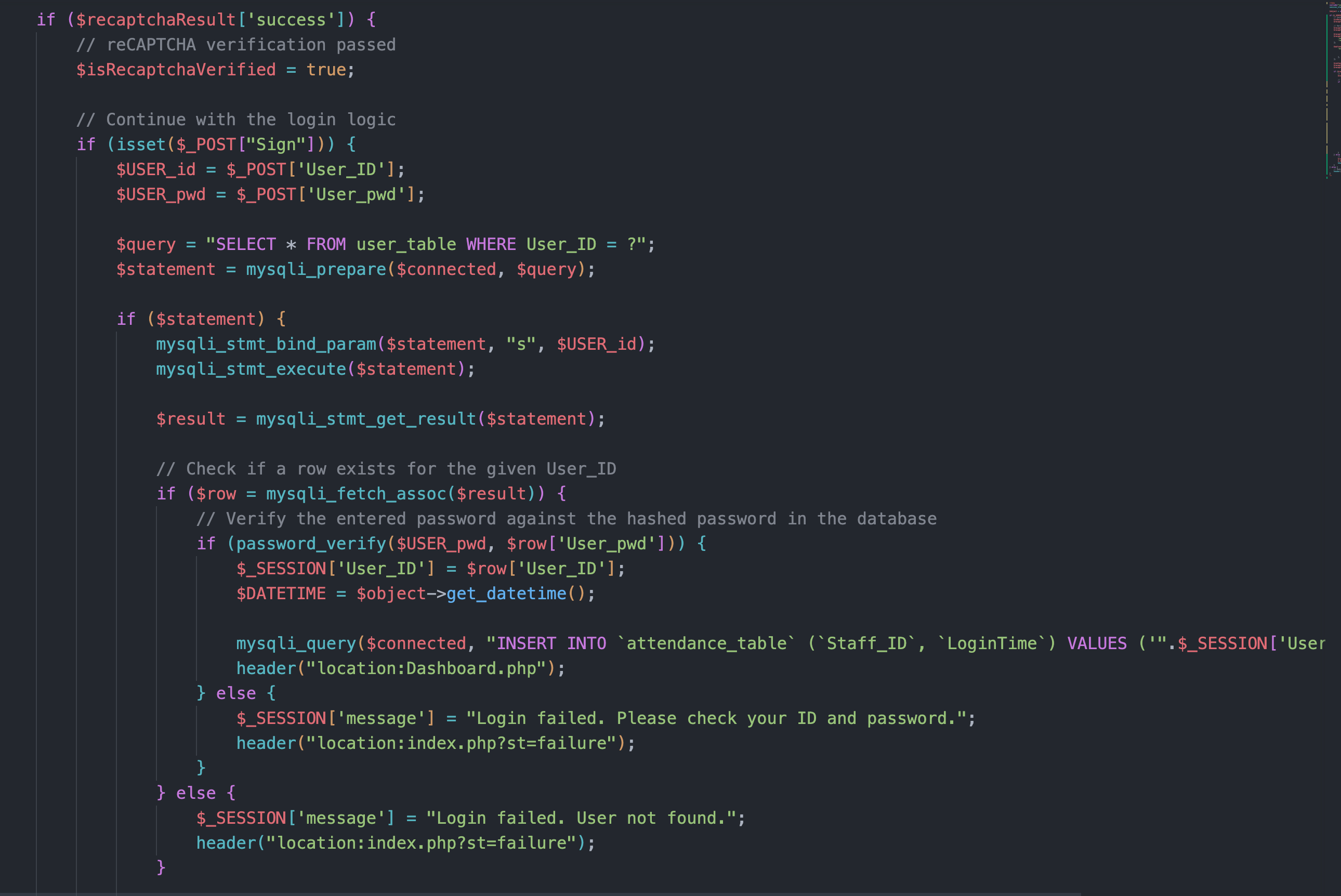
Description automatically generated

**Figure 35**: reCAPTCHA Container

The reCAPTCHA container contains the checkbox where user could complete the CAPTCHA.

A screen shot of a computer program

Description automatically generated



A computer screen with colorful text

Description automatically generated

**Figure 36**: reCAPTCHA Integration (Login Check)

To confirm that the user is authentic, this PHP script first validates the Google reCAPTCHA response when the form is submitted. If the reCAPTCHA verification is successful, the login process continues. Secure authentication is then made possible by comparing the credentials that the user supplied with the hashed password stored in the database. When a user successfully logs in, their attendance record is updated and they are redirected to the dashboard; when they don't, the relevant error messages and redirects are displayed. The implementation contributes to a strong login mechanism by strengthening security through reCAPTCHA validation and protecting user authentication through hashed password comparison.

# Discussion



## Key Findings and Issue Resolution

**Overview of Key Findings:**

During the review and enhancement of the system, several key findings were identified, focusing on SQL statements, password handling, and the implementation of reCAPTCHA. These findings played a crucial role in fortifying the security posture of the system.

**Resolved Issues and Improvements Made:**

**SQL Statement Modification**

* + Identified and rectified potential SQL injection vulnerabilities by modifying SQL statements.
  + Implemented parameterized queries to enhance database interaction security.
  + Addressed potential points of unauthorized access through strengthened SQL defenses.

**Password Hashing**

* + Improved the security of user credentials by implementing advanced password hashing techniques.
  + Migrated from conventional hashing to more secure algorithms, enhancing resistance against password-related attacks.
  + Ensured that stored passwords are not susceptible to compromise, even in the event of a data breach.

**reCAPTCHA Implementation**

* + Integrated reCAPTCHA to mitigate automated login attempts and enhance overall authentication security.
  + Provided an additional layer of defense against malicious bots and potential brute-force attacks.
  + Established a user-friendly approach to differentiating between genuine users and automated scripts during login attempts.

## Continuous Improvement and Maintenance of Security

**Continuous Improvement**

**Regular Security Audits**

* + Arrange regular security audits to find and handle new threats.
  + Perform vulnerability assessments to find and fix possible security flaws early on.
  + Stay up to date on industry standards and best practises to make sure security measures meet the most recent requirements.

**Employee Training and Awareness**

* + Continuously educate staff on evolving security threats and preventive measures.
  + Foster a security-conscious culture within the organization to promote collective responsibility for safeguarding sensitive information.

**Real-Time Monitoring and Incident Response**

* + Implement real-time monitoring tools to promptly detect and respond to security incidents.
  + Establish an incident response plan, ensuring a swift and effective reaction to any security breaches.
  + Regularly update and rehearse the incident response plan to maintain its effectiveness.

**Adaptive Authentication Measures**

* + Explore multifactor authentication (MFA) options to add an extra layer of identity verification.
  + Consider implementing adaptive authentication, dynamically adjusting security measures based on risk assessments and user behaviour.

**Maintenance of Security**

**Regular Software Updates**

* + Keep all software components, including the web application, database, and server, up to date.
  + Apply security patches promptly to address known vulnerabilities and ensure the latest security features are active.

**Data Backup and Recovery**

* + Implement a robust data backup strategy to prevent data loss in case of unexpected incidents.
  + Regularly test data recovery procedures to verify their effectiveness and efficiency.

**User Account Management**

* + Regularly review and audit user accounts to ensure only necessary personnel have access.
  + Disable or remove inactive accounts promptly to minimize potential security risks.

**Documentation and Knowledge Sharing**

* + Maintain comprehensive documentation on security configurations and procedures.
  + Facilitate knowledge sharing among the development and security teams to promote a collaborative security approach.

# Summary



## Conclusion

In conclusion, the systematic approach to security upgrades, encompassing SQL statement modifications, advanced password hashing, and the integration of reCAPTCHA, marks a pivotal step towards fortifying the system against potential threats. The dedication to resolving identified issues and instituting ongoing security measures reflects our commitment to maintaining a resilient and secure platform.

## Learning Outcome Achieved

**Improved Knowledge of Database Security**

* By modifying SQL statements, we have gained a deeper understanding of how to protect databases from common vulnerabilities, particularly SQL injection.

**Useful Applications of Advanced Password Hashing**

* The switch to more secure password hashing methods has given us useful knowledge about how to protect user credentials.

**Including reCAPTCHA to Boost Authentication**

* In addition to reducing the possibility of automated attacks, the effective integration of reCAPTCHA shows that we can easily integrate sophisticated security features.

**Strategies for Constant Improvement**

* Our commitment to adapting to changing security challenges is demonstrated by the implementation of continuous improvement strategies, such as frequent security audits and employee training.

# References

# Appendix