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](#_Toc151665285)

[1.0 Introduction and Overview 7](#_Toc151665286)

[2.0 Design 8](#_Toc151665287)

[2.1 Potential security issues (for each element of design) 8](#_Toc151665288)

[2.2 Recommendations (for each potential security issues) 9](#_Toc151665289)

[3.0 Implementation 10](#_Toc151665290)

[3.1 Security Considerations and Implementation 10](#_Toc151665292)

[3.2 Security Techniques and Methods 11](#_Toc151665293)

[4.0 Discussion 31](#_Toc151665294)

[4.1 Key Findings and Issue Resolution 31](#_Toc151665296)

[4.2 Continuous Improvement and Maintenance of Security 32](#_Toc151665297)

[5.0 Summary 34](#_Toc151665298)

[5.1 Conclusion 34](#_Toc151665300)

[5.2 Learning Outcome Achieved 34](#_Toc151665301)

[6.0 References 35](#_Toc151665302)

[7.0 Appendix 36](#_Toc151665303)

**Table of Figures**

[**Figure 1**: Food Ordering and Management System 7](#_Toc151671350)

[**Figure 3**: Input Validation (Login Check) 11](#_Toc151671351)

[**Figure 4**: Input Validation (Category Add) 11](#_Toc151671352)

[**Figure 5**: Input Validation (Category Edit) 11](#_Toc151671353)

[**Figure 6**: Input Validation (Food Edit) 12](#_Toc151671354)

[**Figure 7**: Input Validation (Staff Edit) 12](#_Toc151671355)

[**Figure 8**: Input Validation (Table Edit) 12](#_Toc151671356)

[**Figure 9**: Input Validation (Profile Edit) 13](#_Toc151671357)

[**Figure 10**: Input Validation (Waitlist Add) 13](#_Toc151671358)

[**Figure 11**: Input Validation (Waitlist Edit) 14](#_Toc151671359)

[**Figure 12**: Password Verification & Hashing 14](#_Toc151671360)

[**Figure 13**: SQL Statement Modifications (Category Update) 15](#_Toc151671361)

[**Figure 14**: SQL Statement Modifications (Category Delete) 16](#_Toc151671362)

[**Figure 15**: SQL Statement Modifications (Food Insert) 17](#_Toc151671363)

[**Figure 16**: SQL Statement Modifications (Food Update) 17](#_Toc151671364)

[**Figure 17**: SQL Statement Modifications (Food Delete) 18](#_Toc151671365)

[**Figure 18**: SQL Statement Modifications (Login Check) 18](#_Toc151671366)

[**Figure 19**: SQL Statement Modifications (Staff Delete) 19](#_Toc151671367)

[**Figure 20**: SQL Statement Modifications (Staff Insert) 19](#_Toc151671368)

[**Figure 21**: SQL Statement Modifications (Staff Update) 20](#_Toc151671369)

[**Figure 22**: SQL Statement Modifications (Order Delete) 20](#_Toc151671370)

[**Figure 23**: SQL Statement Modifications (Order Update) 21](#_Toc151671371)

[**Figure 24**: SQL Statement Modifications (Order Insert) 21](#_Toc151671372)

[**Figure 25**: SQL Statement Modifications (Table Delete) 22](#_Toc151671373)

[**Figure 26**: SQL Statement Modifications (Table Update) 22](#_Toc151671374)

[**Figure 27**: SQL Statement Modifications (Table Insert) 23](#_Toc151671375)

[**Figure 28**: SQL Statement Modifications (Tax Delete) 23](#_Toc151671376)

[**Figure 29**: SQL Statement Modifications (Tax Insert) 24](#_Toc151671377)

[**Figure 30**: SQL Statement Modifications (Tax Update) 24](#_Toc151671378)

[**Figure 31**: SQL Statement Modifications (Wait Insert) 25](#_Toc151671379)

[**Figure 32**: SQL Statement Modifications (Wait Update) 25](#_Toc151671380)

[**Figure 33**: SQL Statement Modifications (Wait Delete) 26](#_Toc151671381)

[**Figure 34**: SQL Statement Modifications (Profile Update) 27](#_Toc151671382)

[**Figure 35**: reCAPTCHA Integration 28](#_Toc151671383)

[**Figure 36**: reCAPTCHA Container 29](#_Toc151671384)

[**Figure 37**: reCAPTCHA Integration (Login Check) 30](#_Toc151671385)

# Introduction and Overview

Due to its handling of sensitive data and user interactions, the food ordering and reservation management system created for restaurant staff places a high priority on security. The security measures put in place during the system's design and development to guarantee data integrity, availability, and privacy are described in this report. Robust security measures, such as error handling, input validation, vulnerability protection, and CAPTCHA integration, have been implemented to protect sensitive data and guarantee the application's correct operation when staff members use it for reservations and meal orders.

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.

**b. Passwords Stored in PlainText**

Without adequate password hashing mechanisms, there is a chance that sensitive data could be accessed by unauthorized parties.

**c. SQL Injection Vulnerabilities in 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.

## 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.

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 addresses security risks by incorporating robust input validation to prevent malicious attacks like SQL injection. Error handling codes are also enhanced to ensure clear messages without compromising 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**

Security is improved by modifying SQL statements to mitigate vulnerability to SQL injection. Prepared statements and restricted access for authorized personnel contribute significantly to lowering the risk of unauthorized data access. Regular updating of database access credentials further enhances overall system security.

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

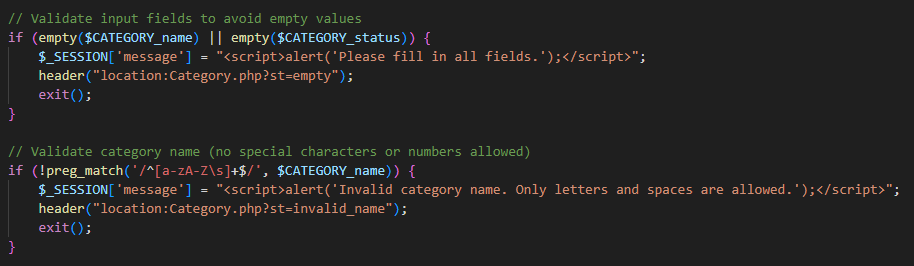
1. **Comprehensive Input Validation and Error Handling (Ying Jie)**

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 (Ying Jie)**



A screen shot of a computer

Description automatically generated

**Figure 11**: Password Verification & Hashing

(*PHP Password Hashing Tutorial (with Examples) - Alex Web Develop* n.d.)

This PHP code uses 'password\_hash' with crypt to securely hash passwords. The hashed password is stored and compared during authentication. Successful verification logs login time and grants access to the dashboard. Failure results in a redirect with a status and message. The approach enhances security by ensuring proper user credentials for dashboard access.

1. **Modifications of SQL Statements (Pau Thing)**

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

(*Sql - PHP Hash Password on Login Page - Stack Overflow* n.d., *How to Prevent SQL Injection with Prepared Statements | TechTarget* n.d., *How to Prevent SQL Injection Vulnerabilities: How Prepared Statements Work* n.d.)

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

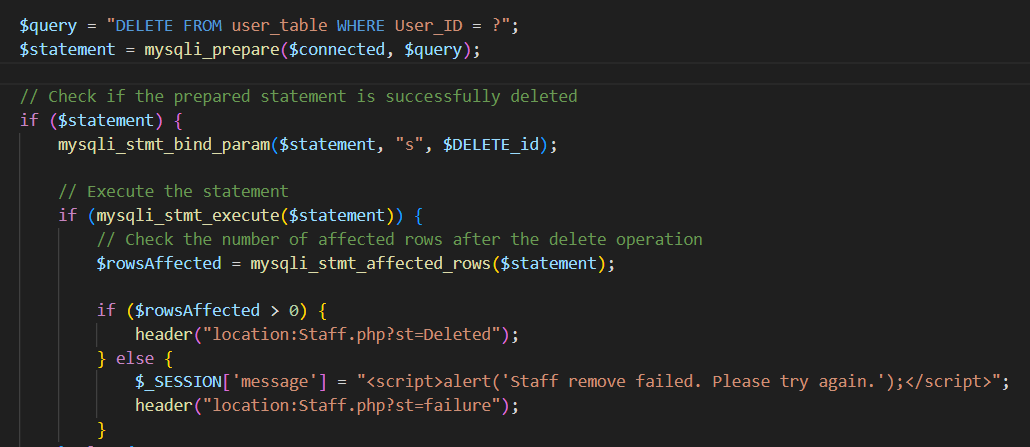
Description automatically generated

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

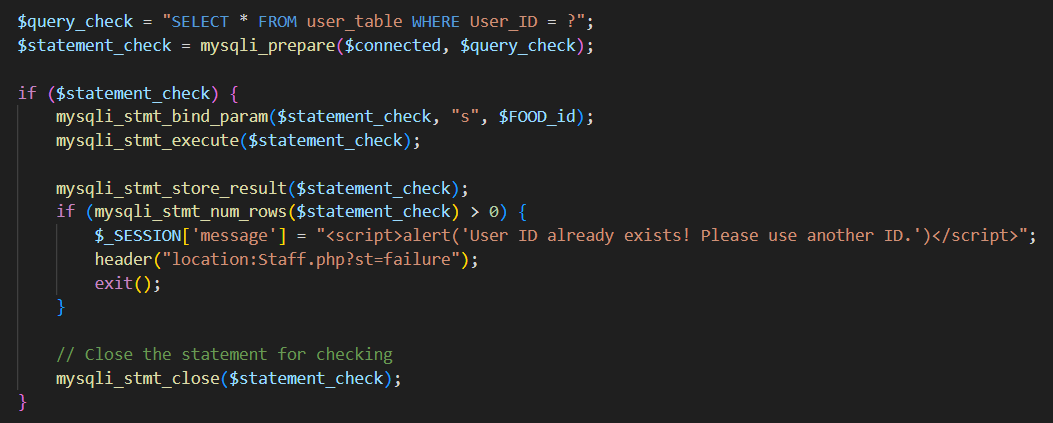
A screen shot of a computer code

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)

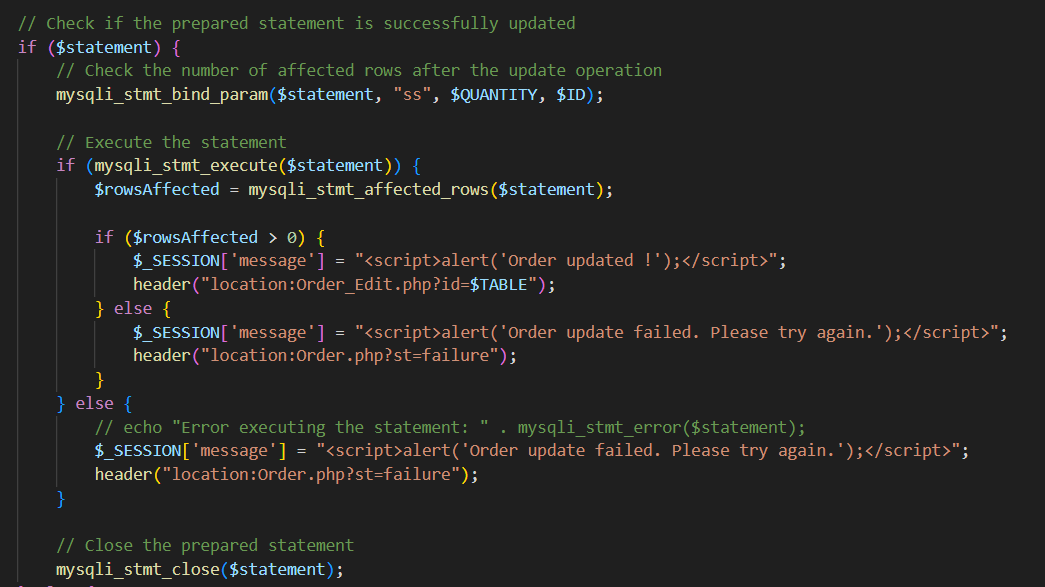
A screen shot of a computer program

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)

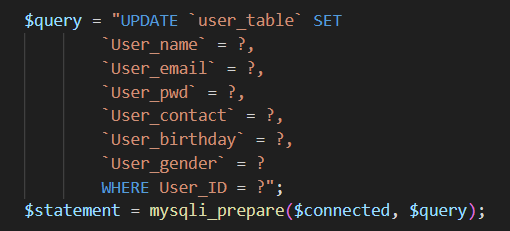
A screen shot of a computer program

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)

By employing prepared statements 'mysqli\_prepare' to thwart SQL injection, the implementation places a strong emphasis on SQL security. Security is improved by parameter binding and data type specification 'mysqli\_stmt\_bind\_param’. Accurate data processing is ensured by essential data types. "mysqli\_stmt\_execute" allows for safe and sanitised query execution. ‘mysqli\_stmt\_affected\_rows’ evaluates operation impact, improving security and control over 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

This code validates the application setup and checks user reCAPTCHA response using the supplied secret key. If successful, it sets $isRecaptchaVerified to true; otherwise, assigns an error to $recaptchaError. This enhances security by ensuring form submissions are from actual users, preventing automated scripts.

A black background with green and orange text

Description automatically generated

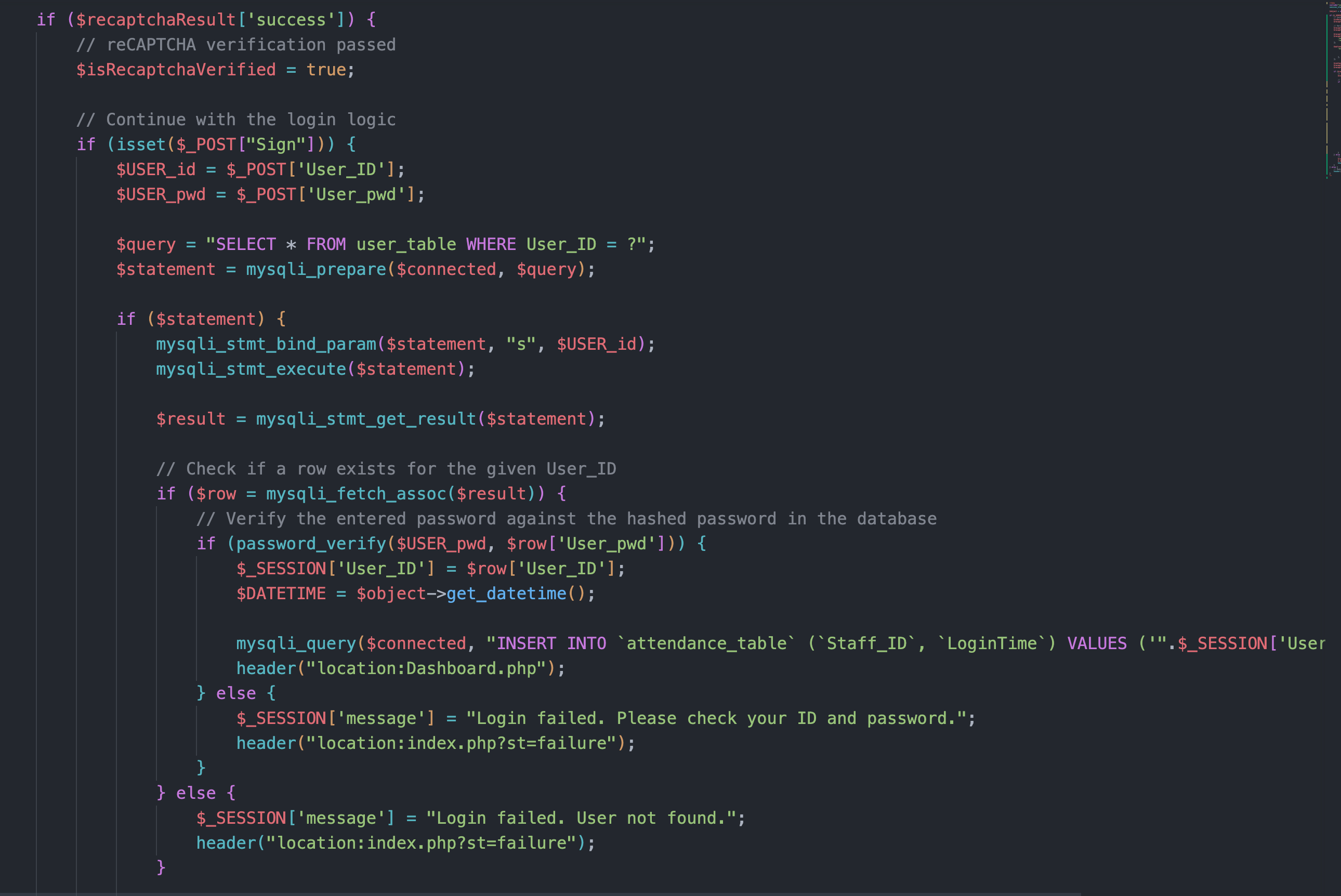
**Figure 35**: reCAPTCHA Container

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

A screen shot of a computer program

Description automatically generated

(*How to Implement a CAPTCHA Solution on Your Website* n.d., *ReCAPTCHA  |  Google for Developers* n.d.)



A computer screen with colorful text

Description automatically generated

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

This PHP script strengthens user authentication by validating reCAPTCHA before login. Secure authentication involves comparing user-supplied credentials with hashed passwords in the database. Successful logins update attendance records and redirect to the dashboard. Failures display error messages and redirect users. The implementation enhances security with reCAPTCHA validation and secure 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 defences.

**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

1. *How to Implement a CAPTCHA Solution on Your Website* (n.d.) available from <https://www.a2hosting.com/blog/implement-captcha-solution-website/> [23 November 2023]

*2. How to Prevent SQL Injection Vulnerabilities: How Prepared Statements Work* (n.d.) available from <https://www.securityjourney.com/post/how-to-prevent-sql-injection-vulnerabilities-how-prepared-statements-work> [23 November 2023]

*3. How to Prevent SQL Injection with Prepared Statements | TechTarget* (n.d.) available from <https://www.techtarget.com/searchsecurity/feature/How-to-prevent-SQL-injection-with-prepared-statements> [23 November 2023]

*4. PHP Password Hashing Tutorial (with Examples) - Alex Web Develop* (n.d.) available from <https://alexwebdevelop.com/php-password-hashing/> [23 November 2023]

*5. ReCAPTCHA  |  Google for Developers* (n.d.) available from <https://developers.google.com/recaptcha> [23 November 2023]

*6. Sql - PHP Hash Password on Login Page - Stack Overflow* (n.d.) available from <https://stackoverflow.com/questions/75901160/php-hash-password-on-login-page> [23 November 2023]

# Appendix

**Github Link:** [**https://github.com/CrazyJoey4/Security**](https://github.com/CrazyJoey4/Security)  
**Turnitin**

As the cover page was forgotten to be removed, the plagiarism rate is 15%.

A screen shot of a cell phone

Description automatically generated