Applied Information Technology And Computer Science Vol. 4 No. 1 (2023) 224-235
© UniversitiTun Hussein Onn Malaysia Publisher's Office

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Client Management System with Two Factor Authentication and Anti Input Injection for Asian Life Travels Sdn Bhd

Sanjay Ramadas[1], **Zubaile Abdullah**[1]*,

[1]Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, 86400, Parit Raja, Batu Pahat, Johor
MALAYSIA

*Corresponding Author Designation

DOI: https://doi.org/10.30880/aitcs.2023.04.01.014
Received 17 September 2022; Accepted 27 May 2023; Available online 30 June 2023

Abstract: A Client Management System (CMS) is a system that keeps track of a company's particular interactions with its customers. sales, marketing, and support teams. One of the key issues that needs immediate attention with the majority of existing Client Management System is security issues. This project focuses on developing a secure Client Management System for a travel and tour business provider called Asian Life Travels Sdn Bhd. The business provider has been providing variety of tour and travel services around Malaysia. The business provider deals with plenty of client's data and is in need of a system that can manage these data in a secure manner. So, if the selected CMS system leaks data, it may be a disastrous situation for both client and organization.

Keywords: Client Management System, Anti SQL Injection, Anti Cross Site Injection,

1. Introduction

Client data are records containing important data about a company's relationship with its clients. A Client Management System is a system that maintains track of a company's specific interactions with their clients.

Asian Life Travels Sdn Bhd is currently using a traditional way of managing the client's data. These methods could be potentially dangerous as it can affect the confidentiality, integrity, and authenticity of the data. Anyone who has access to the logbook and Excel sheets can manipulate the data which could compromise not only the business provider but also client's classified information.

The best approach to overcome this problem is to develop a client management system in a secure environment where only the required personnel have controlled access to it. One key feature to implement this security mechanism is by using Two Factor Authentication(2FA) by push notification

**Corresponding author: zubaile@uthm.edu.my*

to the system. Two-factor authentication, also known as multi-factor authentication, is a technique of establishing access to an online account or computer system that requires the user to give two distinct types of credentials. Two-factor authentication is intended to prevent unwanted users from accessing an account using only a compromised password. Apart from that, a secure system should contain a secure method to handle user inputs. Insecure way of handling user inputs results in dangerous input injection attacks such as SQL Injection and Cross Site Scripting. Implementing security features such as input validation and sanitization is a good approach to mitigate the risks. Hence, the objectives of this project are:

- Design a secure client management system with two factor authentication and anti-input injection.
- Develop a secure client management system using web-based approach.
- Test the security of developed client management system using OWASP authentication.

The significance of using two factor authentication and anti-input injection for this system is 2FA provides an additional layer of security used to ensure only authenticated users gain access to the client management system. Initially, the user will enter their username and a password as usual. Then, rather than gaining access straight away, they will be required to provide additional information where a push notification will be sent to email. This authentication method combats against common cyber-attacks such as brute force attacks. Brute force attack may manage to find a working username and password, but the attacker does not know what other authentication factors the 2FA system requires and does not have those credentials. Anti-input injection validates and sanitizes input from users and prevents malicious input to the system. Malicious input can alter and damage the system and good input validation practice mitigates this. This feature ensures confidentiality, integrity, and authenticity of data in a system.

2. Related Work

2.1 Structured Query Language Injection (SQLi)

Structured Query Language (SQL) used to communicate with a database. SQL can be used to read, update, add, and delete information held within the database. By introducing arbitrary malicious SQL code into a database query, an adversary can get complete control over the web application database [1]. Accessing backend database, attacker can breach information that may include any number of items, including sensitive company data, user lists or private customer details. SQL injection can have a massive impact on a company's economy at large. A successful attack could result in the attacker viewing user lists, deleting entire tables, and, in some situations, gaining administrative rights to a database, all of which are incredibly detrimental to a business [2].

2.2 SQL Injection Prevention

The malicious usage of SQL queries to craft SQL Injection attacks is becoming more complex and dangerous. Prevention methods should be applied to further ensure that SQL Injection can be mitigated. Among the prevention measures are input validation, character-escaping functions, and parameterized queries.

Input validation is a procedure that verifies if the type of input given by the user is permitted. Input validation ensures that the data is of the correct type, length, and format. Only the values that pass the validation procedure can be used. It assists in preventing any commands from being injected into the input string.

Next techniques that can be applied is to implement character escaping functions. Injection attacks frequently rely on the attacker's ability to craft an input that closes the SQL statement's argument string prematurely. This is the reason attempted SQL injection attacks frequently use the characters (') or ("

). Use character-escaping functions provided by each database management system for user-supplied input (DBMS). Typically, doubling up the quote character by replacing (') with (") which means treat this quote as part of the string, not the end of the string. This is done to ensure that the DBMS does not confuses with SQL query.

Another way of prevention is to do parametrized queries. Parameterized queries allow pre-compile a SQL query and then provide the parameters to run it. The database can detect the code and differentiate it from the input data using this way. This approach helps mitigate a SQL injection attack by automatically quoting user input and ensuring that the given input does not modify the purpose.

2.3 Cross Site Scripting (XSS)

Cross Site Scripting (XSS) Attacks are as of now the most well-known security issues in current web applications. Cross-site Scripting (XSS) is a type of code injection attack that occurs on the client side. By embedding malicious code in a legitimate web page or online application, the attacker intends to run malicious scripts in the victim's web browser. Cross-site scripting flaws allow an attacker to impersonate a target user and execute any operations that the user is capable of, as well as access any of the user's data. If the target user has privileged access to the system, the attacker may be able to take complete control of the app's functionality and data [3].

2.4 Cross Site Scripting Attack Prevention

There is no single strategy for mitigating cross-site scripting, and several types of web applications require distinct levels of protection. The fundamental problem of XSS is caused by the lack of separation between code and data. It is important to implement security measures early in the application's development life cycle [4].

The first line of defense against XSS is input sanitization. Whenever accepting any data, ensure the format of the data is as expected. In effect, this whitelists data is to ensure that the application does not accept any random code and ensures that only expected data is inputted [5]. It should be detected by several functions which have been built in the original source code. It can be used to detect the data length, the valid data, and regular expression.

Validation is the process of putting in place rules that restrict a user from filling out a form with data that does not fit specific criteria. Input validation ensures that only secure data is entered into an information system. User input should be validated as soon as the other party's data is received. Validation criteria for an input that asks for the user's "Last Name," for example, should only let the user submit data that contains alphanumeric characters. Any tags or characters often used in cross-site scripting, such as "<script>" tags, can be rejected using validation rules.

The next technique is processing the output data through output filtering. If the application uses data from user input, it is still included in the response. As a result, it can still be exploited to launch an attack by injecting malicious code into the user input response. To mitigate this scenario, we should implement HTML encoding to filter the malicious code. HTML encoding is done by using the HTML character entity reference to replace the native characters. This ensures that the malicious code to be converted into a safe character and treats them as the normal text format and ensures the malicious command cannot be executed by the web browser. This ensures that malicious code is encoded to a safe character and treated as normal text, as well as ensuring that the malicious command is not executed by the web browser.

2.5 Concept of Two Factor Authentication (2FA)

Two-factor authentication (also known as 2FA) is a security mechanism that involves two different forms of identification to gain access to something. Two-factor authentication (2FA) protects accounts from being hacked. Two-factor authentication is a security feature that prevents unwanted users from getting access to an account using only a stolen password. Users may be more vulnerable to password breach than they know, especially if they use the same password across many websites [6]. An account protected by two-factor authentication often requires an individual to authenticate using something they know, usually a password and something they have, such as a mobile phone or hardware token [7].

2.6 Study of Existing System

Two existing systems are compared with the proposed system. The two existing systems are Nimble Customer Relationship Management and ODOO Client Management System. ODOO is a client management system that runs on the web. ODOO, which has over five million members, offers strong customer management tools. Nimble is client management solution for teams and individuals that is built on the industry leading. It is chiefly used for managing relationships with potential and existing clients.

2.7 Comparison of Existing System with Proposed System

Existing systems that have been evaluated are crucial for comparing with the proposed system. The comparison's result is crucial for determining the differences between the systems. The suggested system shares several concepts with the previously stated system. The differences between the proposed system and the existing systems are shown in a comparison table.

Table 1: Comparison of Existing System with Proposed System

Features	ODOO Client Management System	Nimble Customer Relationship Management	Proposed System
Password complexity	✕	✕	✓
Email verification	✕	✕	✓
Prevention of input injection attack	✓	✕	✓
Input validation	✕	✓	✓
Two Factor Authentication	✓	✕	✓

3. Methodology

Object-oriented software is a type of computer programming that focuses on a specific outcome. It is the concept that objects that include structured fields of data are structured in procedural code, often known as methods. OOSD is a practical approach of designing a software system that keeps the attention on the problem's objects throughout the development process. The early focus on objects in OOSD, along with the goal of developing a useful model, results in a system image that is adaptable, reusable, dependable, and intelligible [8].

3.1 System Requirement Analysis

Through the requirement analysis process, four significant requirements for the proposed system have been identified. The first project requirement is to have a system that organizes the client’s personal information. Second is to ensure an easier and efficient flow and storage of client’s data. Next requirement is to have a secure method to access the client’s data through internet. The last requirement is to have multiple steps to verify and authenticate as legitimate users.

3.2 Security Requirement Analysis

"Process for Attack Simulation and Threat Analysis" is what PASTA stands for. PASTA threat modelling combines an attacker's perspective of a business with risk and impact analysis to produce a full picture of the dangers to goods and applications, their vulnerability to attack, and the factors that influence risk and priority choices for remedies. PASTA threat modelling enables stakeholders from throughout the organization to understand how cybersecurity threats affect their goals, and how their goals influence the organization's cybersecurity decisions. This modelling enables a full picture of the hazards that an organization could face to be obtained. This includes the risks of such threats becoming assaults, as well as the objectives that those threats may affect. [9]

3.3 Object-Oriented Design Phase

The OOSD approach to design refines the model produced in the requirements analysis phase into a software architecture and defines a language-specific representation of that architecture. The purpose of preliminary design is to define the system's software architecture. The early design provides a language-independent description of the proposed system. To do this, the top-level architecture is based on the application model produced during the requirements analysis. The proposed system design is simplified in this project by using a Unified Modelling Language (UML) diagram. Among the types of UML diagrams used are class diagram, sequence diagram, use case diagram and activity diagram. In addition to that, two data models are used which are schema and data dictionary. To further enhance the understanding of the proposed system wireframe of the user interface (UI) is also included.

Figure 1: System Architecture of Proposed System

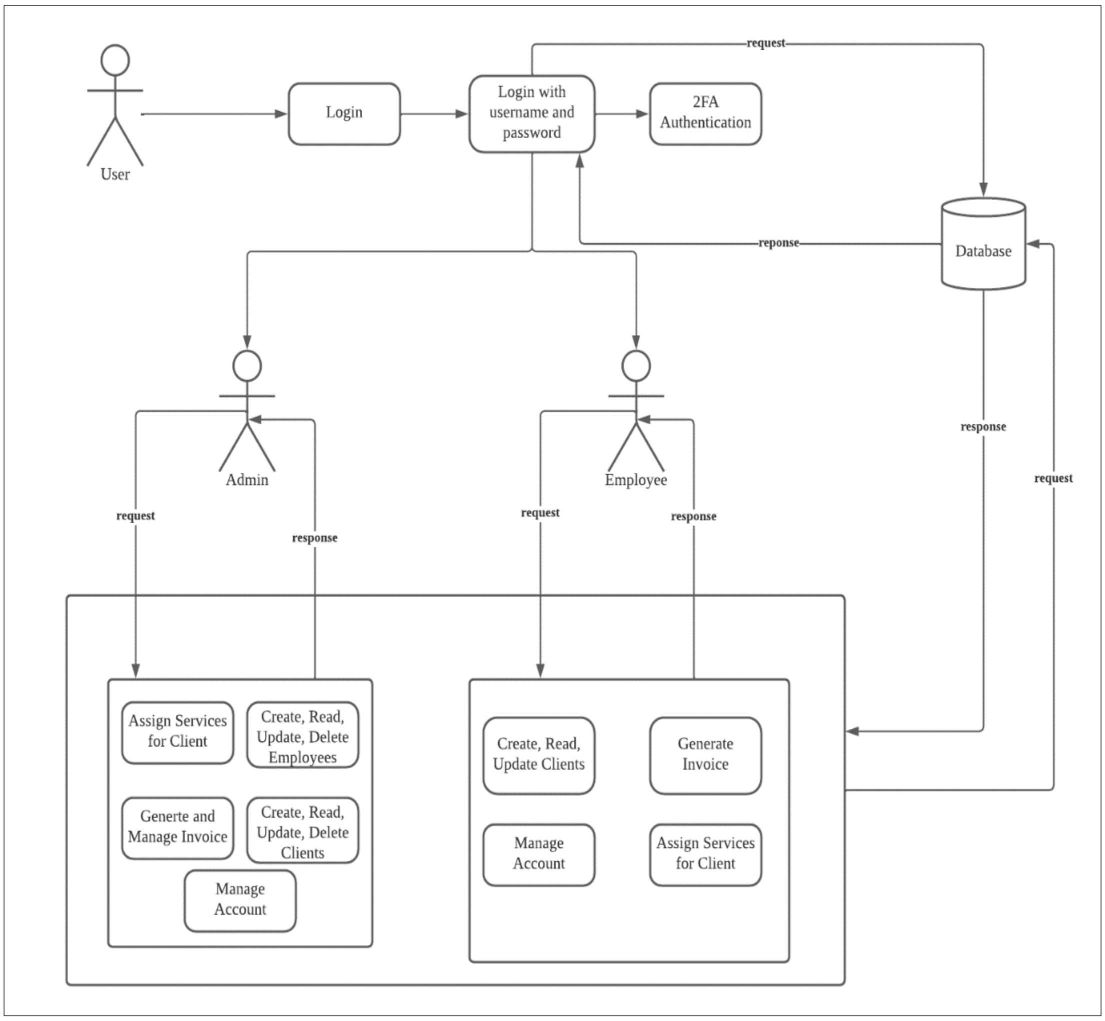
User
Login
Login with username and password
2FA Authentication
request
reponse
Database
Admin
Employee
request
response
request
response
response
request
Assign Services for Client
Create, Read, Update, Delete Employees
Generte and Manage Invoice
Create, Read, Update, Delete Clients
Manage Account
Create, Read, Update Clients
Generate Invoice
Manage Account
Assign Services for Client

Figure 1 shows the system architecture of the proposed system. In the system architecture, there are two main users who are user admin and employee. The user can choose to login as either Admin or Employee. With that, the system makes the confirmation with the information provided by checking the database once the email and the password matched. A one-time password is sent to the users as part of 2FA to authenticate them to the system. After authentication, admins and employees can perform their tasks, respectively.

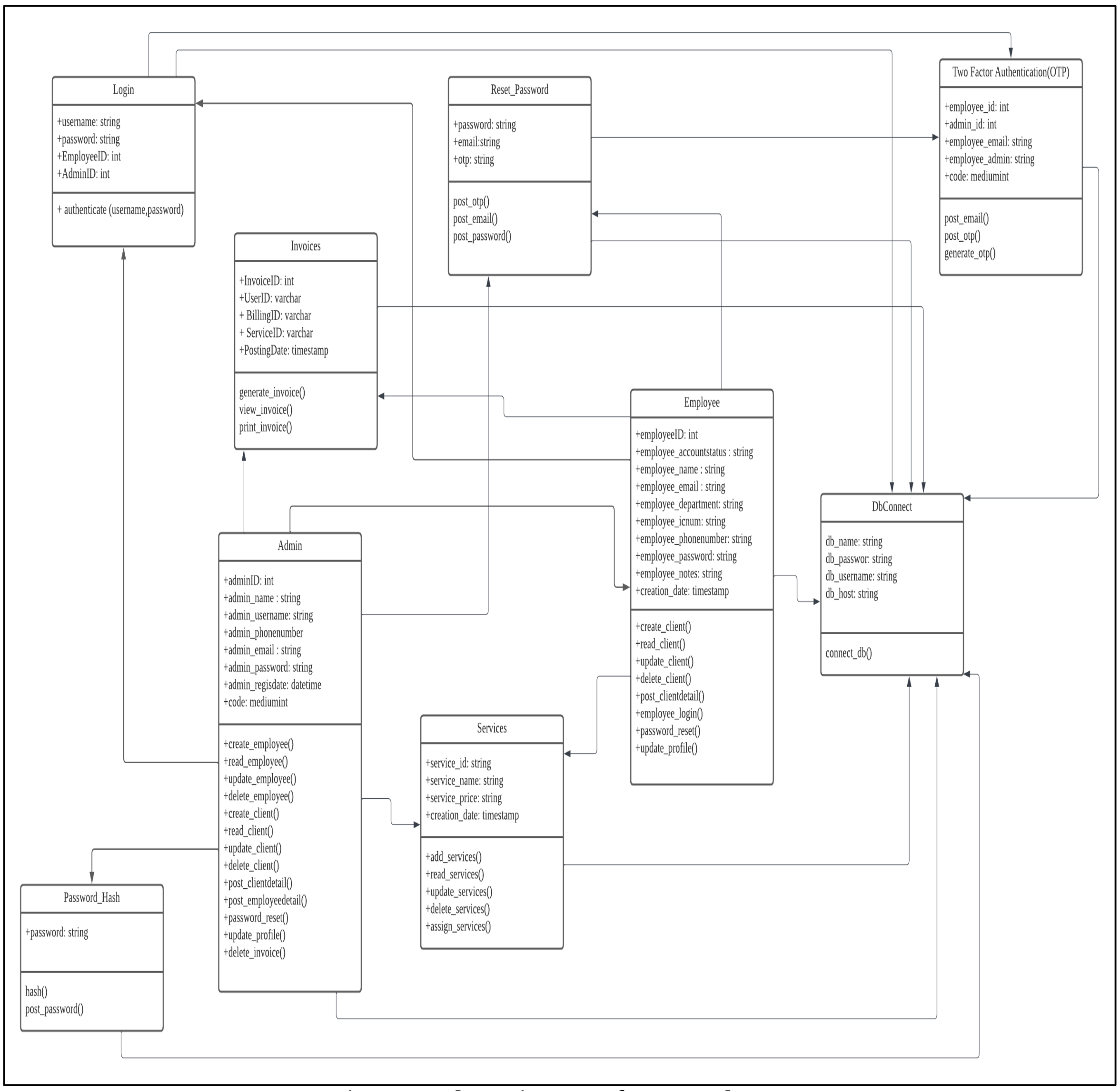

Figure 2: Class Diagram of Proposed System

Figure 2 shows the class diagram of the proposed Client Management System. There are nine classes in the system which are ‘Login,’ ‘Employee,’ ‘Services, ‘TwoFactorAuthentication,’ ‘Admin,’ ‘Reset_Password, ‘Password_Hash,’ ‘DBConnect’ and ‘Invoices.’ The attributes and elemets in the classes are set to public so that the classes can interact with each other.

3.4 Object-Oriented Implementation Phase

After the proposed system has been created, the implementation phase begins. The choice taken during the design phase allows developers to have a seamless implementation process because all they must do now is follow what was specified during the design phase. The proposed system's development phase includes class design, database tables, user interface, database setup, and how the database interacts with the web application's features. Verify that all the classes and database tables are connected. Aside from that, the user interface must be linked to classes. The system was programmed and constructed during this period. All the preceding concepts were converted into programming codes at this step. All the system designs were expressed in programming languages so that they could be implemented on computers. The system's front-end is built in Hypertext Markup Language (HTML), alongside Cascading Stylesheet (CSS) and JavaScript while the back end is implemented in the PHP programming language. Visual Studio IDE was used to develop the application, which was hosted on Apache. MySQL was chosen as the database because it was simple to use and handy.

3.5 Object-Oriented Testing Phase

The testing phase focuses on research and discovery. During the testing phase, developers check to see if their code and programming meet the needs of the client. Among the testing methods utilized are Functional testing and Security testing [10].

Table 2 shows the functional testing result for login module. Table 3 shows the functional testing result for admin module. Table 4 shows the functional testing employee module. Table 5 depicts the security testing for input injection attacks.

Table 2: Functional Testing Login Module

No	Test Case	Expected Result	Actual Output
1	User does not fill in all the required fields	System prompts that entered details are invalid	As expected
2	User enters the incorrect username or password	System prompts that entered details are invalid	As expected
3	User enters the correct id and password	The system will log into the user to its dashboard	As expected

Table 3: Functional Testing Admin Module

No	Test Case	Expected Result	Actual Output
1	User admin leaves all input field empty for client details	System does input validation and returns error message that input field should not be empty	As expected
2	User admin enters incorrect input pattern in client details	System does input validation and returns error to check the format	As expected
3	User admin enters invalid characters in input client details	System does input validation and returns error to check the input	As expected
4	User admin enters erroneous email format	System does input validation and returns error to check the format	As expected
5	User admin fil in all the input field for client details and submits	System accepts the input and new client record will be created	As expected
6	User admin views available clients in the system	System displays the clients available	As expected
7	User admin updates existing client details	Clients’ details are successfully updated and stored	As expected
8	User admin assigns services requested by client	Clients are successfully assigned with the required service	As expected
9	User admin deletes client records from system	Client records are removed from the system	As expected
10	User admin leaves all input field empty for employee details when registering new employee	System does input validation and returns error message that input field should not be empty	As expected

Table 3: (cont)

No	Test Case	Expected Result	Actual Output
11	User admin enters incorrect input pattern in employee details	System does input validation and returns error to check the format	As expected
12	User admin enters invalid characters in input employee details	System does input validation and returns error to check the input	As expected
13	User admin fil in all the input field for employee details and submits	System accepts the input and new employee record will be created	As expected
14	User admin views available employee in the system	System displays the employees available	As expected
15	User admin updates existing employee details	Employees' details are successfully updated and stored	As expected
16	User admin deletes employee records from system	Employee records are removed from the system	As expected

Table 4: Functional Testing Employee Module

No	Test Case	Expected Result	Actual Output
1	User employee leaves all input field empty for client details	System does input validation and returns error message that input field should not be empty	As expected
2	User employee enters incorrect input pattern in client details	System does input validation and returns error to check the format	As expected
3	User employee enters invalid characters in input client details	System does input validation and returns error to check the input	As expected
4	User employee enters erroneous email format	System does input validation and returns error to check the format	As expected
5	User employee fil in all the input field for client details	System accepts the input and new client record will be created	As expected
6	User employee views available clients in the system	System displays the clients available	As expected
7	User employee updates existing client details	Clients' details are successfully updated and stored	As expected
8	User employee assigns services requested by client	Clients are successfully assigned with the required service	As expected

Table 5: Security Testing for Input Messaging

No	Malicious Input	Vulnerable System	Secured Developed System
1	Inputting [‘or 1=1 –] at Login field	The login bypasses	Unable to login to the system
2	Inputting [admin' or '1'='1'] at Login field	The login bypasses	Unable to login to the system
3	Input <SCRIPT>alert('XSS');</SCRIPT> on Search field	The system reflects the malicious script	The system converts the input to normal HTML entity
4	Input <IFRAME SRC=# onmouseover="alert(document. cookie)"></IFRAME> on Search field	The system is prone to the script and returns session ID	The system converts the input to normal HTML entity

4. Results and Discussion

4.1 User Acceptance Form Result

The user acceptance of the system is collected through an online survey questionnaire using Google Form that has been distributed to respondents for user acceptance testing. There are five respondents involved. The result of the user acceptance form is separated into three graphs. First are the results on the system design testing, second is about the system functionality testing result and third is about the system security testing result.

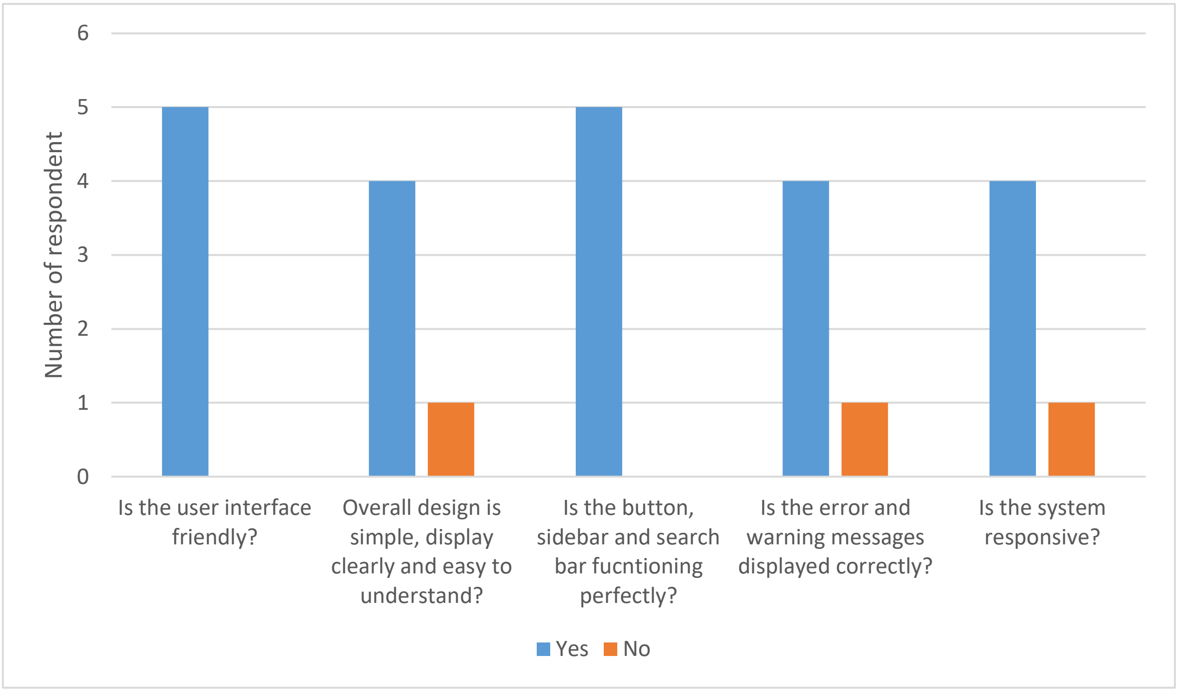

Figure 3: Testing Result for System Design

Figure 3 shows the system design testing result. The Y axis shows the number of respondents, and the X axis shows the characteristics of design of the system. All the five respondents agree that the system is user friendly. Four out of five respondents agree the overall design is simple, the display is clear and easy to understand, and one respondent disagree. All five respondents agree the button, sidebar, and search bar functions well. Four out of five respondents agree that error and warning messages displayed correctly, and one respondent disagrees. Four out of five respondents agree that the system is responsive, and one respondent disagree.

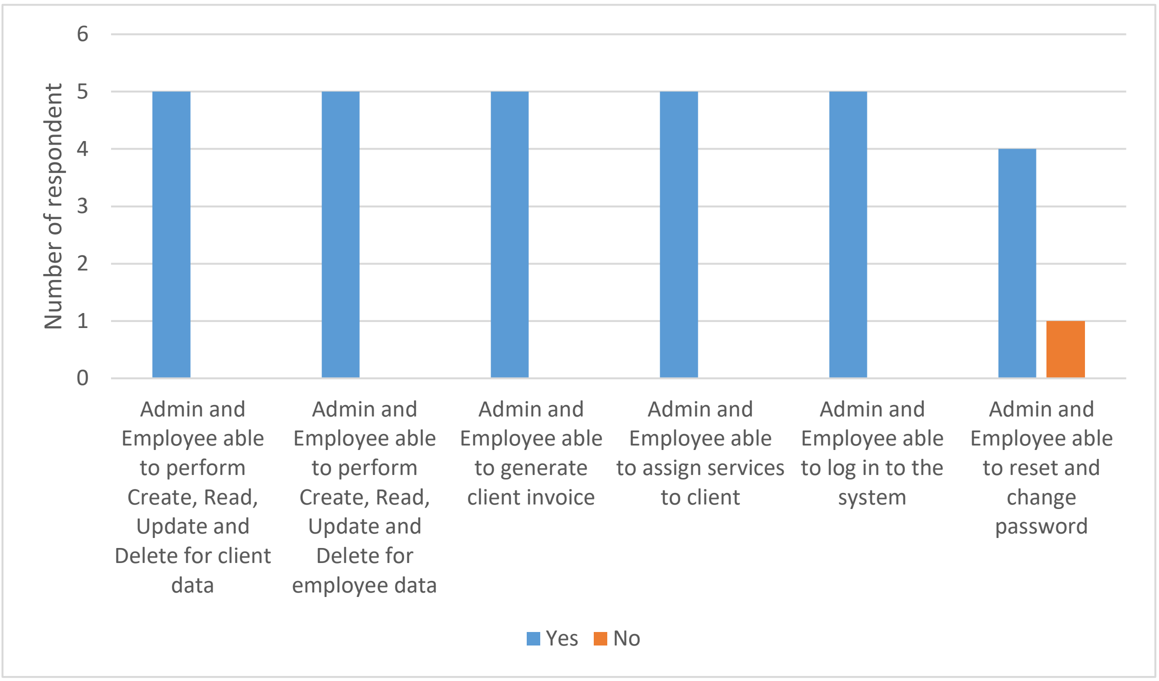

Figure 4: Testing Result for System Functionality

Figure 4 shows the system functionality testing result. The Y axis shows the number of respondents, and the X axis shows the characteristics of functionality of the system. All five respondents agree that admin and employees can perform CRUD function for clients and admin can perform CRUD functions for employees. All five respondents agree that admin and employees can generate client invoices. All respondents agree that admin and employee able to assign services to client. All respondents agree that admin and employee can login to the system. Four out of five respondents can reset, and change password and one respondent disagree.

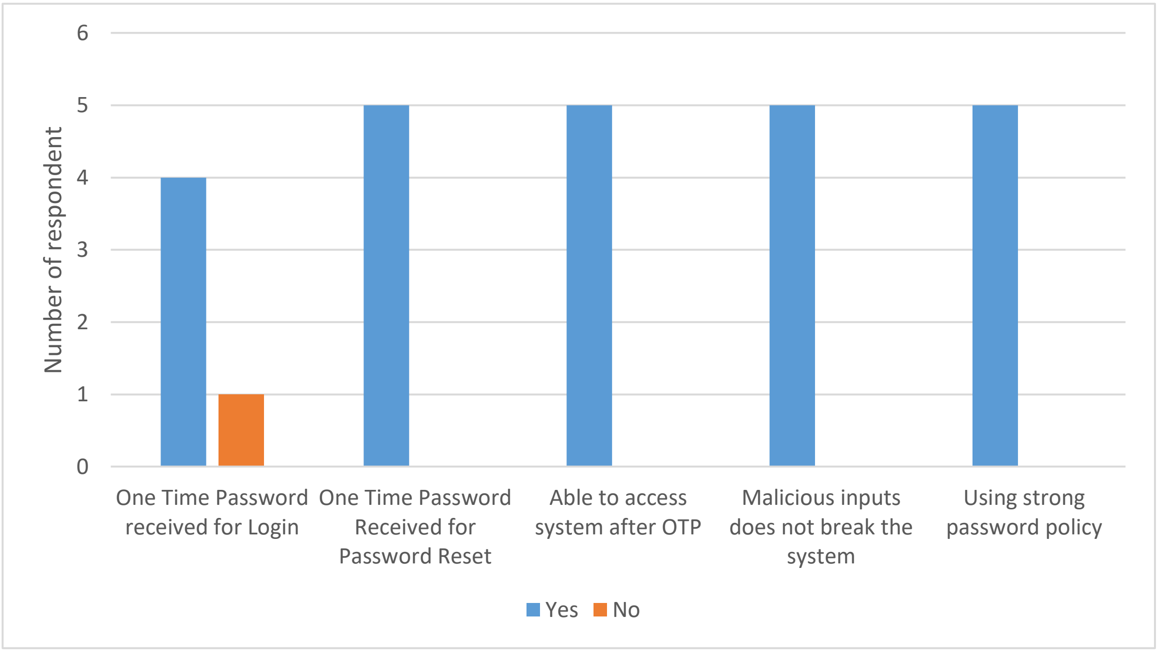

Figure 5: Testing Result for System Security

Figure 5 shows the security functionality testing result. The Y axis shows the number of respondents, and the X axis shows the characteristics of security of the system. Three out of five respondents can receive OTP for Login and two could not. All respondents agree that they can receive and use OTP for password reset. All respondents agree that they can access the system after OTP. All respondents confirm that malicious inputs do not break the system. All respondents agree that password policy implemented with strong password.

5. Conclusion

Users could utilize the client management system to manage and manage client details in a secure environment. The system enforces a strong password policy, which requires users to establish strong passwords. The system additionally uses password hash for the user's password with SHA algorithm. The system uses prepared statements as part of the approach to combat SQL injection attacks. Also, the utilization of html functions ensures Cross Site Scripting attack to be prevented. One Time Password is used as part of the two-factor authentication and provides an extra layer of security.

Users could utilize the client management system to manage and manage client details in a secure environment. The system enforces a strong password policy, which requires users to establish strong passwords. The system additionally uses password hash for the user's password with SHA algorithm. The system uses prepared statements as part of the approach to combat SQL injection attacks. Also, the utilization of html functions ensures Cross Site Scripting attack to be prevented. One Time Password is used as part of the two-factor authentication and provides an extra layer of security.

Acknowledgment

I would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

References

[1] Porup, J., 2021. What is SQL injection? How these attacks work and how to prevent them. [online] CSO Online. Available at: <https://www.csoonline.com/article/3257429/what-is-sql-injection-how-these-attacks-work-and-how-to-prevent-them.html> [Accessed 31 December 2021].

[2] Learning Center. 2021. What is SQL Injection | SQLI Attack Example & Prevention Methods | Imperva. [online] Available at: <https://www.imperva.com/learn/application-security/sql-injection-sqli/> [Accessed 31 December 2021].

[3] Academy, W. and scripting, C., 2021. What is cross-site scripting (XSS) and how to prevent it? | Web Security Academy. [online] Portswigger.net. Available at: <https://portswigger.net/web-security/cross-site-scripting> [Accessed 31 December 2021].

[4] Cloudflare.com. [Online]. Available: https://www.cloudflare.com/learning/security/threats/cross-site-scripting/. [Accessed: 31-Dec-2021].

[5] Goteleport.com. 2021. Preventing XSS Attacks. [online] Available at: <https://goteleport.com/blog/xss-attacks/> [Accessed 31 December 2021].

[6] Investopedia. 2021. Two-Factor Authentication (2FA). [online] Available at: <https://www.investopedia.com/terms/t/twofactor-authentication-2fa.asp> [Accessed 31 December 2021].

[7] Reese, K., Smith, T., Dutson, J., Armknecht, J., Cameron, J. and Seamons, K., 2021. A Usability Study of Five {Two-Factor} Authentication Methods. [online] Usenix.org. Available at:

<https://www.usenix.org/conference/soups2019/presentation/reese> [Accessed 31 December 2021].

[8] Colbert, E., 1989. The object-oriented software development method: a practical approach to object-oriented development. Proceedings of the conference on Tri-Ada '89 Ada technology in context: application, development, and deployment - TRI-Ada '89.

[9] Cynance. 2021. PASTA Threat Modelling - The Complete Cyber Security Meal. [online] Available at: <https://www.cynance.co/pasta-threat-modelling/> [Accessed 31 December 2021].

[10] Agarwal, M., 2021. How to Perform Web Application Testing Effectively? [online] Learn Programming and Software Testing. Available at: <https://www.techbeamers.com/web-application-testing/> [Accessed 31 December 2021].