

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: Hneah Guey Ling	
CU Student ID Number: 13445654	
Semester: August 2023	
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: <u>Hneah</u>	

Section B - To be completed by the module leader

<p>Intended learning outcomes assessed by this work:</p> <ol style="list-style-type: none"> 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
<ol style="list-style-type: none"> 1. Report Refer to the section on Marking Scheme on detailed breakdown. 2. System Functionality Refer to the section on Marking Scheme on detailed breakdown. 	<p>60</p> <p>40</p>	
Total	100	
<p><u>Lecturer's Feedback</u></p>		
<p><u>Internal Moderator's Feedback</u></p>		

Table of Contents

1.0 Introduction and System Overview4

2.0 Design6

 2.1 Potential Security Issues Identified.....6

 2.2 Applied Security Features in the System 10

3.0 Implementation 12

4.0 Discussion 18

 4.1 Key findings..... 18

 4.2 Continuous Improvement and Maintenance23

5.0 Summary23

6.0 References.....24

7.0 Appendix25

1.0 Introduction and System Overview

The selected system for assessing security is a warehouse management system, which is an employer project with MY E TCM SDN. BHD. The system is implemented as a web-based application using Laravel framework. There are three modules in the system: the admin module, the purchasing staff module, and the warehouse staff module.

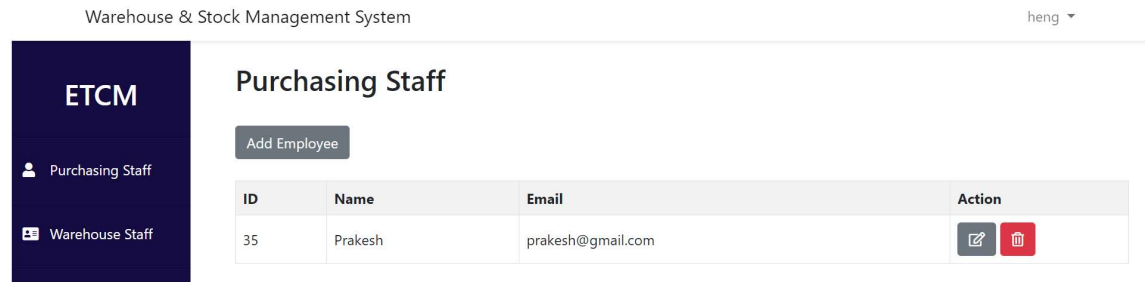


Figure 1-1 Interface for the admin

Figure 1-1 above shows the interface for the admin in the warehouse management system. After successful login, the admin can add, delete, view or update the information of purchasing staff and warehouse staff.

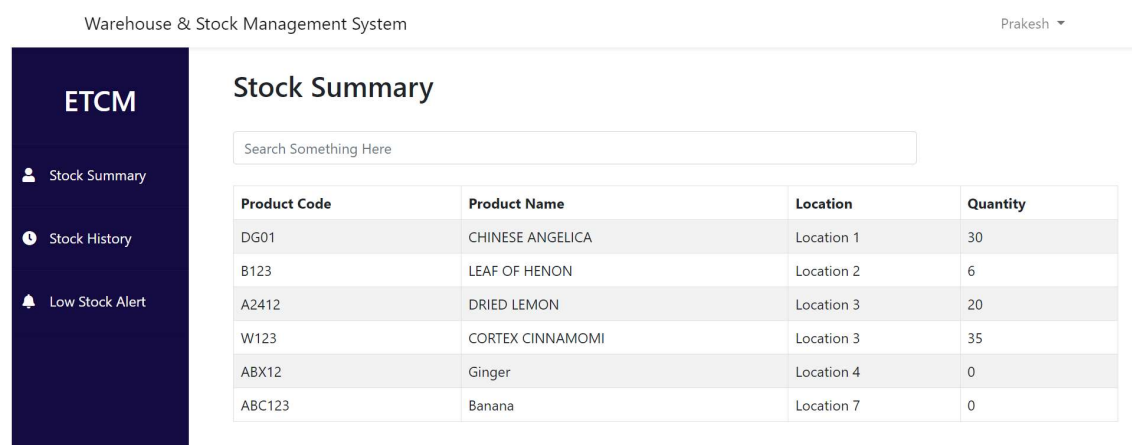


Figure 1-2 Interface for the purchasing staff

Figure 1-2 above shows the interface for the purchasing staff. The purchasing staff is able to view the stock summary, including the total number of products stored in each location. Information such as how many stocks are received and how many are issued can also be viewed in the stock history page. The low stock alert allows staff to identify when stocks are running low and need to be purchased.

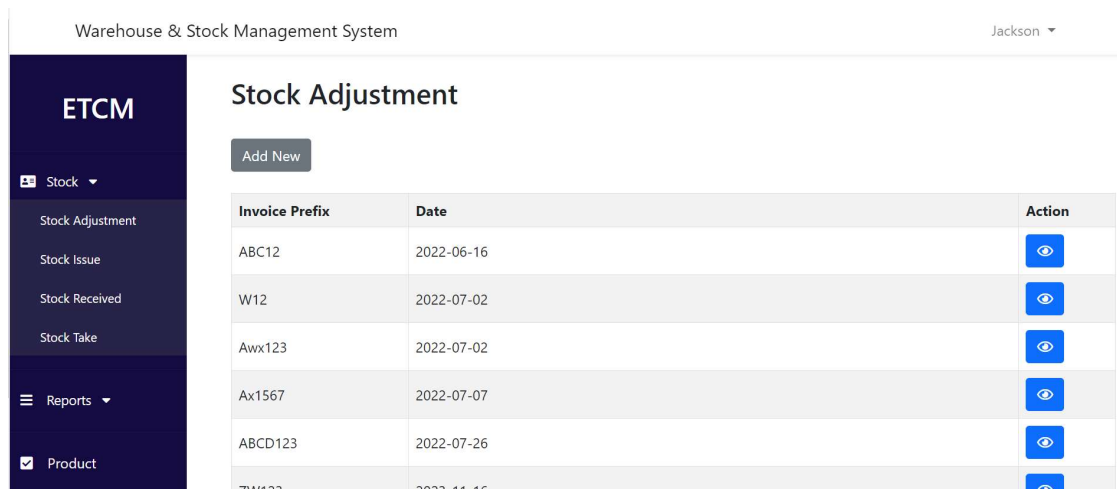


Figure 1-3 Interface for the warehouse staff

Figure 1-3 above shows the interface for warehouse staff. The warehouse staff can record stock in when stock is received as well as stock out when stock is issued. Stock adjustment can be made when there are cases such as stock disposal, stock spoilt or returned order. The stock take can be recorded when the staff conduct the scheduled stock count. Besides, the warehouse staff can also view the reports of stock summary and stock history. The products can be added, deleted, viewed or updated by the staff in the system.



2.0 Design

2.1 Potential Security Issues Identified

The potential security issues in the existing system will be identified and presented along with the recommendations for addressing these issues.

1. No Input Sanitization

id	name	email	email_verified_at
3	ali	ali@gmail.com	NULL
31	Lee	<script>malicious code</script>	NULL
34	Jackson	jackson@gmail.com	NULL
35	Prakesh	prakesh@gmail.com	NULL
36	heng	heng@gmail.com	NULL

Figure 2-1 Script saved in database

Figure 2-1 shows the data stored in the database of the warehouse management system. There is no sanitization in the user input and scripts can be saved easily. The attacker can inject malicious code written in HTML and JavaScript into the database and the malicious script will run every time when the user calls the appropriate function. The system is vulnerable to cross-site scripting (XSS) attack where the malicious input can get into the output if not taken care properly.

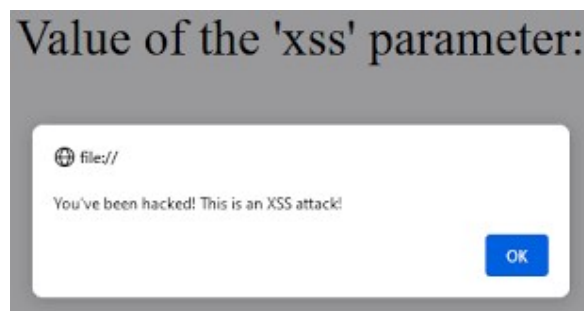


Figure 2-2 Pop-up dialog box by attacker (PVS-Studio, 2021)

The attacker might display advertisements to users, trick users into entering their credentials, or get them to click on malicious links when the script is executed. Figure 2-2 shows how attacker can display message to users by embedding JavaScript code with alert() function into the system.

Therefore, it is recommended to use `filter_var()` function to sanitize an input by removing potentially unsafe characters.

2. No Validation

The system does not implement strong validation rules for the user input. The system does not check for null value and ensuring the input is in correct format. The system does not ensure that the input is in the correct format. For instance, it does not validate that an email ends with ".com". Users are able to save weak passwords, which could pose a security risk.

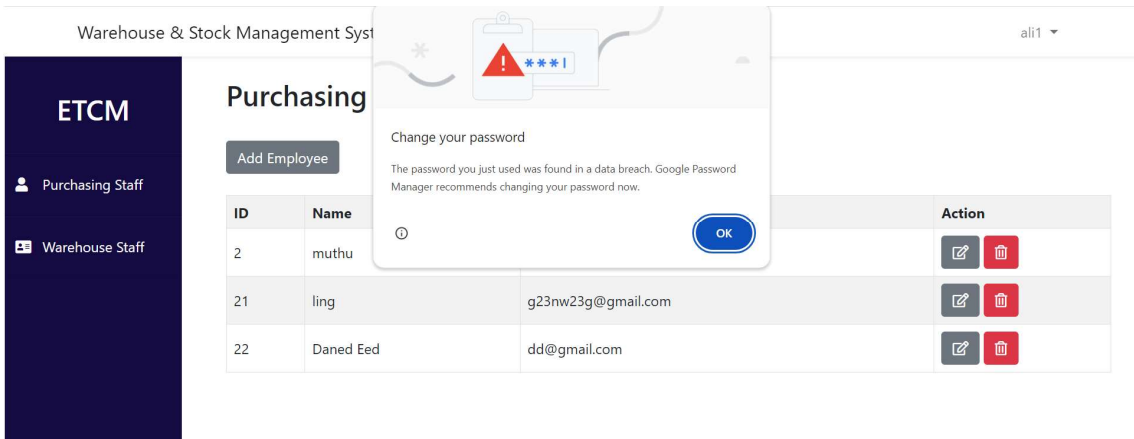


Figure 2-3 Google's warning

As shown in figure 2-3, the system allows weak password and warned by Google. Weak passwords are easily guessable or found using brute-force attacks. Hence, the attackers can hack the system and get unauthorised access.

SQL injection and XSS attacks may also occur if user inputs are not properly validated. For instance, if a user enters ' or '1'=1 in the email input, and the system fails to validate whether the email contains '@' symbol and ends with '.com', it could allow the user to bypass security measures and login into the system successfully.

Proper input validation and password strength requirements are required to enhance the security of the system.

3. Improper Error Handling

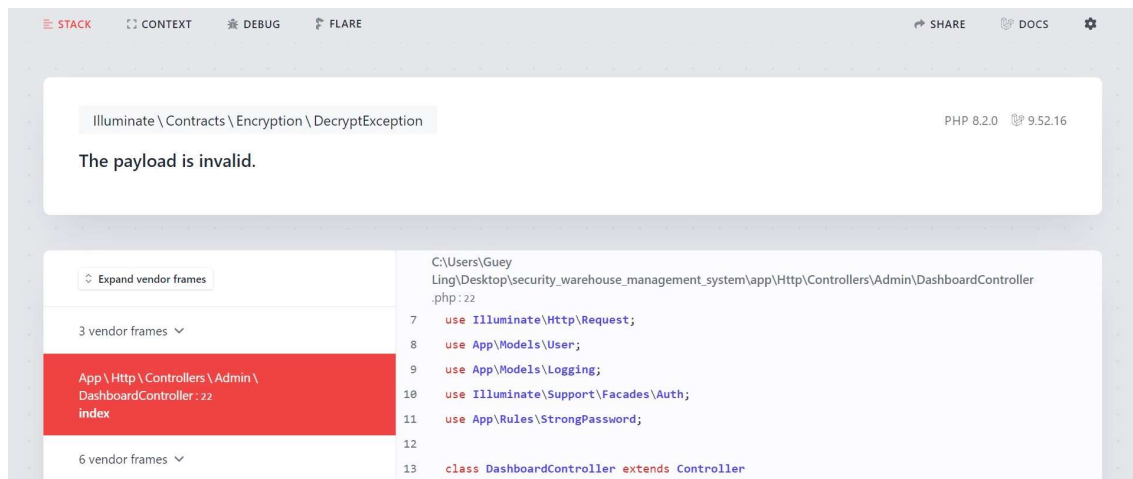


Figure 2-4 Error page

The existing system does not handle error properly. Instead of displaying the error message to the users, it reveals the code that leads to the error directly as shown in figure 2-4 above. The code might expose sensitive information about the system such as the internal data structures, password settings, and even database queries. Attackers can exploit this information to identify vulnerabilities and launch attacks.

It is suggested to use try catch exception handling method to display an error message when exception occurred. Furthermore, the debug mode should be set to false in order to reduce the information exposure.

4. No Logging

The current system lacks logging, making it difficult to trace data modifications and distinguish between authorized staff and potential hackers.

It is recommended to record actions, timestamps, and user identifiers for security purposes. When unusual or suspicious activity like unauthorized staff modifications occurred, it can be identified and trigger alerts. This allows security teams to investigate and respond promptly.

5. Lack of Data Encryption

The sensitive information of the users like phone number, identification number, and passport number are not encrypted before storing in the database. The data stored in the database can be viewed directly if the database is hacked or if unauthorized access is gained. A data breach of sensitive information may lead to legal issues, especially regarding data protection and privacy regulations like the Personal Data Protection Act (PDPA). The organizations might face fines and lawsuits and other regulatory consequences.

It is important to encrypt sensitive data using strong cryptographic algorithms, such as Advanced Encryption Standard (AES), Elliptic Curve Cryptography (ECC), or Rivest-Shamir-Adleman (RSA) to ensure the highest level of security and data protection.

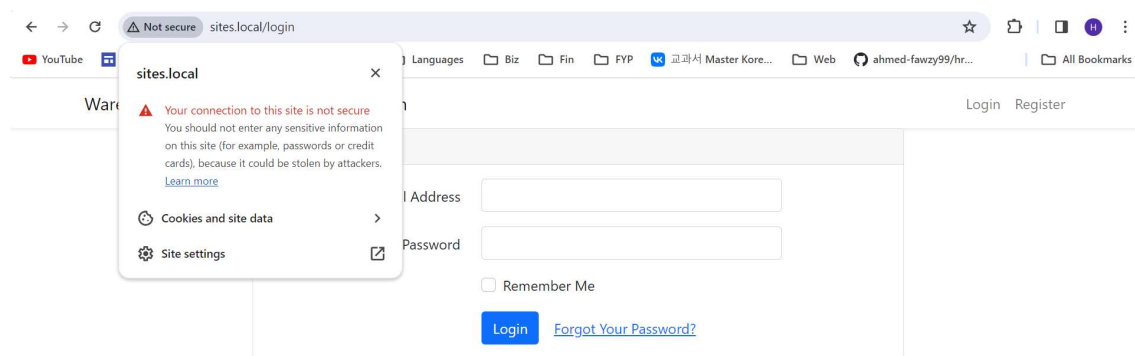


Figure 2-5 HTTP used for the web application

Figure 2-5 above shows that Hypertext Transfer Protocol (HTTP) is used for the web application. The data is unencrypted during transmission and this will lead to security threats such as man-in-the-middle attacks and data manipulation. The sensitive data can be viewed or accessed by unauthorized people (Yasar, 2022).

The application must use Hypertext Transfer Protocol Secure (HTTPS) during production. The usage of HTTP in the local development may cause mixed content issues (Joel van Bergen, 2019). Resources are loaded successfully during development and testing in HTTP environment but when the protocol changes to HTTPS, the browsers may block non-secure content causing problems with resource loading and functionality.

Therefore, it is a best practice to test and develop the application with HTTPS locally to prevent potential issues when transitioning to a production environment which utilizes

HTTPS. Encryption of data during transmission may enhance the data integrity and security of the system.

2.2 Applied Security Features in the System

This section will discuss the security features that are already implemented in the current system.

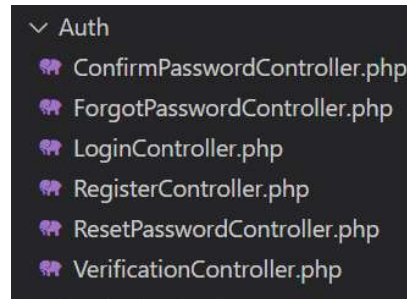


Figure 2-6 Controllers generated for user authentication

This authentication process is managed by Laravel's authentication middleware and controllers to verify the users. When the command 'php artisan make:auth' is run, the authentication scaffolding is generated as shown in figure 2-6 and Laravel will check if the input email and password match the records stored in the 'users' table.

```
'throttle' => 'Too many login attempts. Please try again in :seconds seconds.',
```

Figure 2-7 'throttle' key in the auth.php file

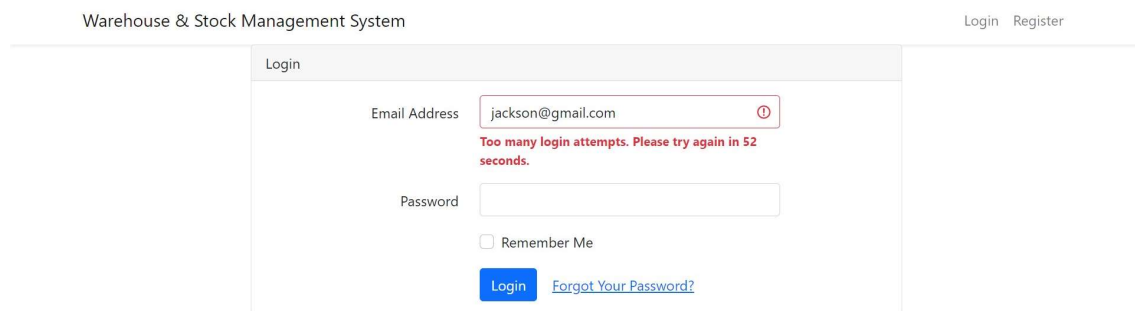
A screenshot of a web application's login page. The page has a header with 'Warehouse & Stock Management System' on the left and 'Login Register' on the right. The main content area is titled 'Login'. It contains a form with two input fields: 'Email Address' and 'Password'. The 'Email Address' field contains the text 'jackson@gmail.com' and has a red error message below it: 'Too many login attempts. Please try again in 52 seconds.' Below the password field is a checkbox labeled 'Remember Me'. At the bottom of the form are two buttons: a blue 'Login' button and a blue link 'Forgot Your Password?'. The entire form is enclosed in a light gray border.

Figure 2-8 Error message displayed

When the users exceed the maximum number of login attempts, the system will be temporarily locked out for 60 seconds. The configuration provided in auth.php which is shown in figure 2-7 is related to rate limiting for the login functionality. Rate limiting controls the

number of attempts the users can make in a given time period to avoid brute-force attacks and denial of services. The message defined in the 'throttle' key will be displayed to the users as shown in figure 2-8 above.

```
Route::get('/admin_deletewarehouse/{id}', 'App\Http\Controllers\Admin\DashboardController@deleteWarehouse')->middleware('role:admin');
Route::get('/admin_editwarehouse/{id}', 'App\Http\Controllers\Admin\DashboardController@showDataWarehouse')->middleware('role:admin');
Route::post('/admin_editwarehouse', 'App\Http\Controllers\Admin\DashboardController@editWarehouse')->middleware('role:admin');

Route::get('/purchasingstaff_dashboard', 'App\Http\Controllers\PurchasingStaff\DashboardController@index')->middleware('role:purchasing_staff');
Route::get('/purchasingstaff_stockhistory', 'App\Http\Controllers\PurchasingStaff\DashboardController@stockHistory')->middleware('role:purchasing_staff');
Route::get('/purchasingstaff_lowstockalert', 'App\Http\Controllers\PurchasingStaff\DashboardController@lowStockAlert')->middleware('role:purchasing_staff');
Route::get('/purchasingstaff_search', 'App\Http\Controllers\PurchasingStaff\DashboardController@search')->middleware('role:purchasing_staff');

Route::get('/warehousestaff_dashboard', 'App\Http\Controllers\WarehouseStaff\DashboardController@index')->middleware('role:warehouse_staff');
Route::get('/warehousestaff_stocktake', 'App\Http\Controllers\WarehouseStaff\DashboardController@stockTake')->middleware('role:warehouse_staff');
Route::get('/warehousestaff_addnewstocktake', 'App\Http\Controllers\WarehouseStaff\DashboardController@addNewStockTake')->middleware('role:warehouse_staff');
Route::post('/warehousestaff_selectlocation', 'App\Http\Controllers\WarehouseStaff\DashboardController@selectLocation')->middleware('role:warehouse_staff');
```

Figure 2-9 User authorization

Users' authorization with role-based access control is implemented in the system. There are three types of users set up for the system, which are the admin, the purchasing staff and the warehouse staff. Each of them has permission to perform different functions. A custom middleware is created to check a user's role before granting access to specific routes as shown in figure 2-9.

```
<td>{{ $item->id }}</td>
<td>{{ $item->name }}</td>
<td>{{ $item->email }}</td>
```

Figure 2-10 Proper output encoding

Figure 2-10 shows the proper output encoding method, which is double curly braces {{ }} used within the system. This method encodes and outputs the content securely as the variable or expression placed inside the double curly braces will be escaped to avoid XSS attacks. If there are any HTML or scripts in the variable, the code will be displayed as text and not executed.



3.0 Implementation

The potential security threats are identified and suggestions are provided in the previous section. This section will discuss the actual implementation taken to address the issues mentioned above.

1. Input Sanitization

```
$item = new Stockadjustment;
$item->invoice_prefix = filter_var($request->invoice_prefix, FILTER_SANITIZE_STRING);
$item->date = filter_var($request->date, FILTER_SANITIZE_STRING);
$item->description = filter_var($request->description, FILTER_SANITIZE_STRING);
$item->name = $request->get('name')[$key];
$item->code = filter_var($request->get('code')[$key], FILTER_SANITIZE_STRING);
$item->location = $request->get('location')[$key];
$item->quantity_available = $request->get('quantity_available')[$key];
$item->new_quantity = filter_var($request->get('new_quantity')[$key], FILTER_SANITIZE_NUMBER_INT);
$item->quantity_adjusted = $request->get('quantity_adjusted')[$key];
$item->remark = filter_var($request->get('remark')[$key], FILTER_SANITIZE_STRING);
// mention other fields here
$item->save();
```

Figure 3-1 Code snippet for input sanitization

Figure 3-1 displays an essential code snippet exemplifying the practice of input sanitization. The implementation of 'FILTER_SANITIZE_STRING' meticulously eradicates HTML tags from string inputs, while 'FILTER_SANITIZE_NUM_INT' rigorously filters out all characters except digits, plus, and minus signs, ensuring the acceptance of only essential numeric data. By leveraging the 'filter_var()' function with these filters, the code establishes a robust defense mechanism that significantly curtails the attack surface. This reduction in the scope of permissible input not only fortifies the application against injection attacks but also acts as a barrier against potentially malicious inputs that could otherwise pose significant security risks. Consequently, this fortified defense mechanism minimizes the susceptibility to exploitable vulnerabilities, safeguarding the system's integrity and shielding it from potential security breaches.

2. Input Validation

```
public function editWarehouse(Request $req){
    $req->validate([
        'name'=>'required',
        'email'=>'required | email',
        'password'=>['nullable', 'min:8', new StrongPassword],
        'phone_no'=>'min:10',
        'ic_passport_no'=>'min:12'
    ]);
}
```

Figure 3-2 Code snippet for input validation

Figure 3-2 shows the code snippet for input validation. The input is validated before storing in the database. This is a crucial step in ensuring that users provide the correct data with the right format, thus upholding data integrity and maintaining the quality of the data within the system. Some of the validation rules include required fields cannot be left null, a minimum number of digits is required, email input must contain the '@' symbol.

```
public function passes($attribute, $value)
{
    return preg_match('/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]+$/ ', $value);
}

/**
 * Get the validation error message.
 *
 * @return string
 */
public function message()
{
    return 'The :attribute must contain at least one uppercase letter, one lowercase letter, or';
}
```

Figure 3-3 Code snippet for strong password validation

A custom validation rule class is generated to specify password criteria, mandating that passwords must include at least one uppercase letter, one lowercase letter, one number, and one special character. If the password that is entered fails to meet these criteria, the system will display the error message defined in the message() function. This validation is essential to ensure that user passwords are sufficiently complex and not easily vulnerable to brute-force attacks. Furthermore, by incorporating this custom validation rule and displaying a specific error message through the message() function when the password criteria are not met, the system actively guides users towards creating stronger passwords, fostering a security-conscious user culture.

3. Error Handling

```
try{
    $users = new User;
    $users->name= filter_var($req->name, FILTER_SANITIZE_STRING);
    $users->email=$req->email;
    $users->password=Hash::make($req->password);
    $users->role="warehouse_staff";
    $users->phone_no=encrypt(filter_var($req->phone_no, FILTER_SANITIZE_STRING));
    $users->ic_passport_no=encrypt(filter_var($req->ic_passport_no,FILTER_SANITIZE_STRING));
    $users->save();

    $logs = new Logging;
    $logs->admin_name = Auth::user()->name;
    $logs->action="Added new warehouse staff";
    $logs->employee_name = $req->name;
    $logs->save();
} catch (\Exception $e) {
    return redirect()->back()->with('error','Data is not added successfully. Please use another email. ');
}
```

Figure 3-4 Code snippet for error handling

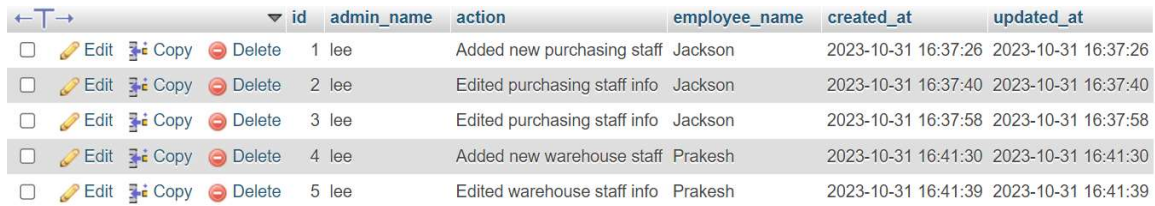
In the code snippet shown in figure 3-4, error handling is meticulously structured with the 'try' and 'catch' blocks. Operations prone to errors, like database data saving, are enclosed within the 'try' block to execute within a controlled environment. If an exception arises during this process, the 'catch' block intervenes, executing predefined actions to manage the encountered exception and display an appropriate error message. By handling potential exceptions in a structured manner, the system reduces the risk of exposing sensitive information or system vulnerabilities to users or potential attackers in case of unexpected failures. Thus, this robust error-handling design not only enhances user experience through informative error messages but also bolsters the system's security by mitigating the potential risks associated with unhandled exceptions.

```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:/INm/1UI7oAJ17EvTkpvkTT9C7C81FhQcJgKjCH8DuU=
4 APP_DEBUG=false
5 APP_URL=http://localhost
6
```

Figure 3-5 Debug option in the .env file

By setting the debug mode to false, as depicted in figure 3-5, the system adopts a secure approach by suppressing the display of code snippets or detailed error messages to users in case of system errors. This preventive measure shields sensitive information from being exposed to potential attackers or unintended users, reducing the risk of disclosing internal workings of the system that could be exploited for malicious purposes.

4. Logging



		id	admin_name	action	employee_name	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	lee	Added new purchasing staff	Jackson	2023-10-31 16:37:26	2023-10-31 16:37:26
<input type="checkbox"/>	Edit Copy Delete	2	lee	Edited purchasing staff info	Jackson	2023-10-31 16:37:40	2023-10-31 16:37:40
<input type="checkbox"/>	Edit Copy Delete	3	lee	Edited purchasing staff info	Jackson	2023-10-31 16:37:58	2023-10-31 16:37:58
<input type="checkbox"/>	Edit Copy Delete	4	lee	Added new warehouse staff	Prakesh	2023-10-31 16:41:30	2023-10-31 16:41:30
<input type="checkbox"/>	Edit Copy Delete	5	lee	Edited warehouse staff info	Prakesh	2023-10-31 16:41:39	2023-10-31 16:41:39

Figure 3-6 Logging information

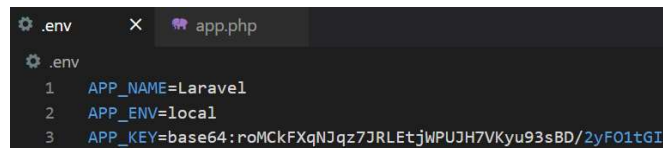
When the admin adds or updates any staff information, the records will be saved into the 'loggings' table as shown in figure 3-6 above. This meticulous logging mechanism not only preserves historical data but also acts as a forensic trail, enabling swift identification and investigation of suspicious activities. By referencing the 'loggings' table, the organizations can pinpoint the exact source and time of any unauthorized modifications, facilitating rapid response and ensuring the system's integrity and security.

5. Encryption

```
'key' => env('APP_KEY'),  
'cipher' => 'AES-256-CBC',
```

Figure 3-7 Encryption algorithm of the system stated in config/app.php

The Advanced Encryption Standard (AES) algorithm is used to encrypt and decrypt the sensitive information in the warehouse management system as shown in figure 3-7 above. The key length is 256 bits, which means that the secret key used is a 256-bit binary value. There are 2 raised to the power of 256 possible combinations of 0s and 1s for the secret key. Hence, it is highly impractical for an attacker to get the correct key through a brute force attack (Biering, 2018). As the key is longer, it provides a higher level of security compared to AES-128 and AES-192.



```
.env  
1 APP_NAME=Laravel  
2 APP_ENV=local  
3 APP_KEY=base64:roMckFXqNJqz7JRLEtjWPUJH7VKyu93sBD/2yFO1tGI=
```

Figure 3-8 Encryption key used in the system stated in .env file

As AES operates as a symmetric encryption algorithm, it employs a single secret key for both encryption and decryption processes. The secret key in this project is stored in the .env file as shown in figure 3-8 above.

```
$users = User::find($req->id);  
$users->name= filter_var($req->name, FILTER_SANITIZE_STRING);  
$users->email=$req->email;  
if (isset($req->password))  
$users->password=Hash::make($req->password);  
$users->phone_no=encrypt(filter_var($req->phone_no, FILTER_SANITIZE_STRING));  
$users->ic_passport_no=encrypt(filter_var($req->ic_passport_no,FILTER_SANITIZE_STRING));  
$users->role=$req->role;  
$users->save();
```

Figure 3-9 Code snippet for encryption

Figure 3-9 shows the code snippet for encryption. The phone number and identification number or passport number of the users are encrypted before storing in the database. This converts the text into an unreadable form to ensure the confidentiality and security of the stored information. By encrypting sensitive data, the system shields it from unauthorized access or interception, significantly reducing the risk of exposure in the event of a data breach or unauthorized database access. This practice aligns with security best practices, fortifying the system against potential threats and maintaining the privacy and integrity of sensitive user information within the database.

```
$user->phone_no = decrypt($user->phone_no);  
$user->ic_passport_no = decrypt($user->ic_passport_no);
```

Figure 3-10 Code snippet for decryption

Within the code snippet displayed in figure 3-10, the decryption functionality is showcased, responsible for retrieving and converting the encrypted data stored in the database back into a readable format for user display. The decryption process utilizes the secret key, previously employed during encryption, to reverse the encryption algorithm and restore the information to its original readable form. This pivotal step allows authorized users or system components to access and view the sensitive information securely stored in an encrypted state. By employing the secret key for decryption, the system ensures that only authorized entities with access to the correct decryption key can decipher and view the sensitive data, maintaining the confidentiality and integrity of the information throughout its storage and retrieval processes. This approach adheres to security best practices, enabling the secure handling and presentation of sensitive data while mitigating risks associated with unauthorized access or data exposure.

6. HTTPS

HTTPS is implemented for the warehouse management system by generating a secure sockets layer (SSL) certificate for the localhost and installing the certificate into the system.

 server	3/11/2023 12:18 PM	Security Certificate	2 KB
 server.key	3/11/2023 12:17 PM	KEY File	2 KB

Figure 3-11 SSL certificate and server key created

After configurations, the SSL certificate and server key is created successfully as shown in figure 3-11 above.

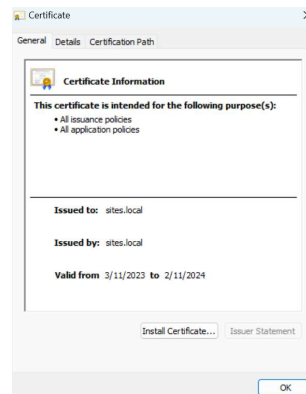


Figure 3-12 Installation page for the SSL certificate

Once the certificate is generated, it is installed into the local system as shown in figure 3-12. The certificate is then added to the Trusted Root Certification Authorities certificate store to ensure that the computer trusts the certificate.

```
<VirtualHost *:443>
    DocumentRoot "C:\Users\Guey Ling\Desktop\security_warehouse_management_system\public"
    ServerName sites.local
    ServerAlias *.sites.local
    SSLEngine on
    SSLCertificateFile "crt/sites.local/server.crt"
    SSLCertificateKeyFile "crt/sites.local/server.key"
</VirtualHost>
```

Figure 3-13 configurations in httpd-xampp.conf

The configuration depicted in Figure 3-13 is tailored to securely serve the warehouse management system over HTTPS on the sites.local domain. It operates by listening on port 443, designated for HTTPS traffic. The 'SSLCertificateFile' and 'SSLCertificateKeyFile' settings are specified to include the necessary certificate and private key. These components facilitate encryption in the communication between the server and the client, ensuring a secure and encrypted data exchange.

4.0 Discussion

4.1 Key findings

1. Input Sanitization

Warehouse & Stock Management System

Jackson ▾

ETCM

Stock ▾

Reports ▾

Product

Add New Stock Adjustment

Invoice Prefix

F123

Date

02/11/2023

📅

Description

Description

Code	Name	Location	Current Qty	Adjusted Qty	Variance	Remark	Action
W123	CORTEX CINNAMOMI	Location 3	35	<script>2</	Variance	good	<div>-</div> <div>+</div>

Confirm

Back

Figure 4-1 Script entered in the user input

Figure 4-1 shows a script typed in the user input, 'Adjusted Qty'. The 'Adjusted Qty' is the latest quantity of products in the warehouse after adding the stock quantity due to returned order or subtracting the stock quantity due to stock spoil, stock disposal, and damaged. Therefore, the expected input is an integer instead of string or scripts. If the system accepts the non-integer value, it will lead to data integrity issue and the system unable to process that input as intended.

Warehouse & Stock Management SystemJackson ▾

ETCM

Stock ▾

Reports ▾

Product

View Stock Adjustment

Invoice Prefix

F123

Date

2023-11-02

Description

Code	Name	Current Qty	Adjusted Qty	Variance	Location	Remark
W123	CORTEX CINNAMOMI	35	2	33	Location 3	good

Back

Figure 4-2 Sanitized input is displayed

The input data is sanitized successfully as shown in figure 4-2 above. The original input which includes tags are removed, remaining the number ‘2’ only. This can avoid users from entering incorrect data or malicious code.

2. Input Validation

Warehouse & Stock Management System

ETCM

Purchasing Staff

Warehouse Staff

Logging and Analytics

Add New Warehouse Staff

The name field is required.

The email must be a valid email address.

The password must be at least 8 characters.

The password must contain at least one uppercase letter, one lowercase letter, one number, and one special character.

The phone no must be at least 10 characters.

The ic passport no must be at least 12 characters.

Name:

Email:

testing

Password:

123

Phone Number:

0123

IC Number/ Passport Number:

0

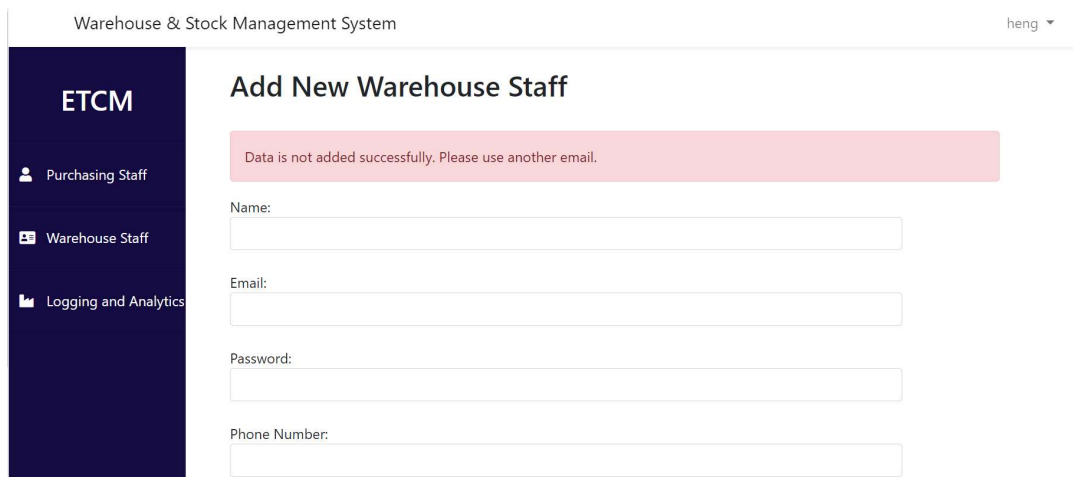
Add

Cancel

Figure 4-3 Validation for user input

Figure 4-3 shows that the user input must be validated before being stored in the database. When the input does not comply to the criterial and fails to be validated, the error message will be displayed to prompt users to enter the correct input. The system also ensures the registered password is complex enough to avoid the account from being hacked easily.

3. Error Handling



The screenshot shows a web application titled "Warehouse & Stock Management System" with a user profile "heng". On the left is a dark blue sidebar with the logo "ETCM" and three menu items: "Purchasing Staff", "Warehouse Staff", and "Logging and Analytics". The main content area is titled "Add New Warehouse Staff". At the top of this area is a pink error message box that reads: "Data is not added successfully. Please use another email." Below this are four input fields labeled "Name:", "Email:", "Password:", and "Phone Number:", each with a corresponding text input box.

Figure 4-4 Exception handled by displaying error message

When exception occurs, such as duplication of registered email, the system will catch the errors and handle it as shown in figure 4-4 above. The system will display error message to inform users what went wrong.

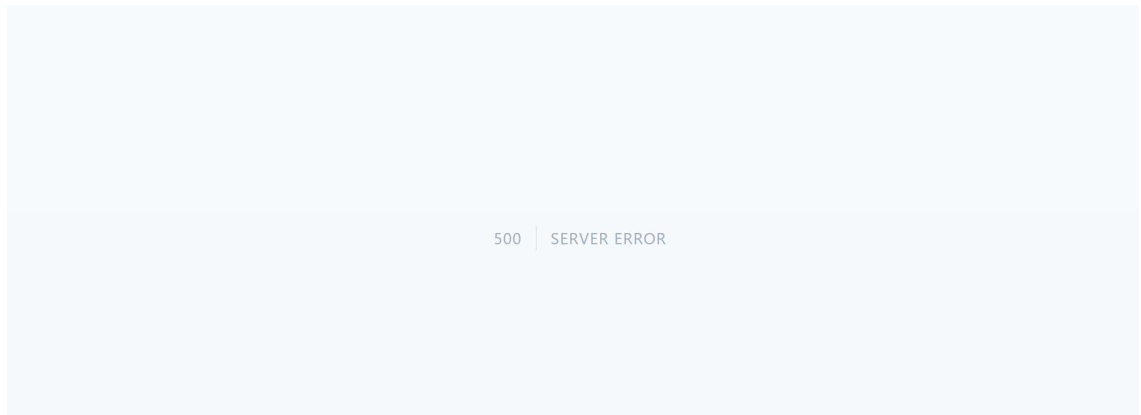


Figure 4-5 Error page with status code

After setting the `app_debug` to false, the application will only display the status code as shown in figure 4-5 instead of revealing the code information when any errors occurred.

4. Logging

Warehouse & Stock Management System

heng ▾

ETCM

Purchasing Staff

Warehouse Staff

Logging and Analytics

Logging and Analytics Information

Admin Name	Action Done	Employee Name	Time
lee	Edited warehouse staff info	Prakesh	2023-10-31 16:41:39
lee	Added new warehouse staff	Prakesh	2023-10-31 16:41:30
lee	Edited purchasing staff info	Jackson	2023-10-31 16:37:58
lee	Edited purchasing staff info	Jackson	2023-10-31 16:37:40
lee	Added new purchasing staff	Jackson	2023-10-31 16:37:26

Figure 4-6 Logging page

Figure 4-6 above shows the logging page where admin can view the information. The admin actions on adding or modifying staff data will be recorded together with the timestamp. This helps the admin to track if any suspicious activities occurred and respond to it effectively.

5. Encryption

updated_at	role	phone_no	ic_passport_no
2022-05-15 14:18:59	admin	NULL	NULL
2023-10-31 16:25:05	admin	NULL	NULL
2023-10-31 16:37:58	warehouse_staff	eyJpdil6ljlCOFIDOWxFRGw0MnhoYWpMNUVSVHc9PSIsInZhbH...	eyJpdil6lmtMcTMvSjhwOWZ5aEg5MjRybHYvQ3c9PSIsInZhbH...
2023-10-31 16:41:39	purchasing_staff	eyJpdil6lnhjcS96K0INZlJEZzBvN09peVdqWFE9PSIsInZhbH...	eyJpdil6llhXbmlZVllyZmlySnkzSFhaYWNHWUE9PSIsInZhbH...
2023-11-02 01:34:14	admin	NULL	NULL

Figure 4-7 Encrypted data in the database

Figure 4-7 shows the sensitive data, which are the phone number and identification number or passport number of the staff are encrypted into an unreadable format and stored in the database.

Warehouse & Stock Management System heng ▾

ETCM

- 👤 Purchasing Staff
- 🏠 Warehouse Staff
- 📊 Logging and Analytics

Edit Purchasing Staff

Name:

Email:

Password:

Phone Number:

IC Number/ Passport Number:

Figure 4-8 Decrypted data

The data is decrypted as shown in figure 4-8 above for authorized users to view or edit the details securely.

6. HTTPS

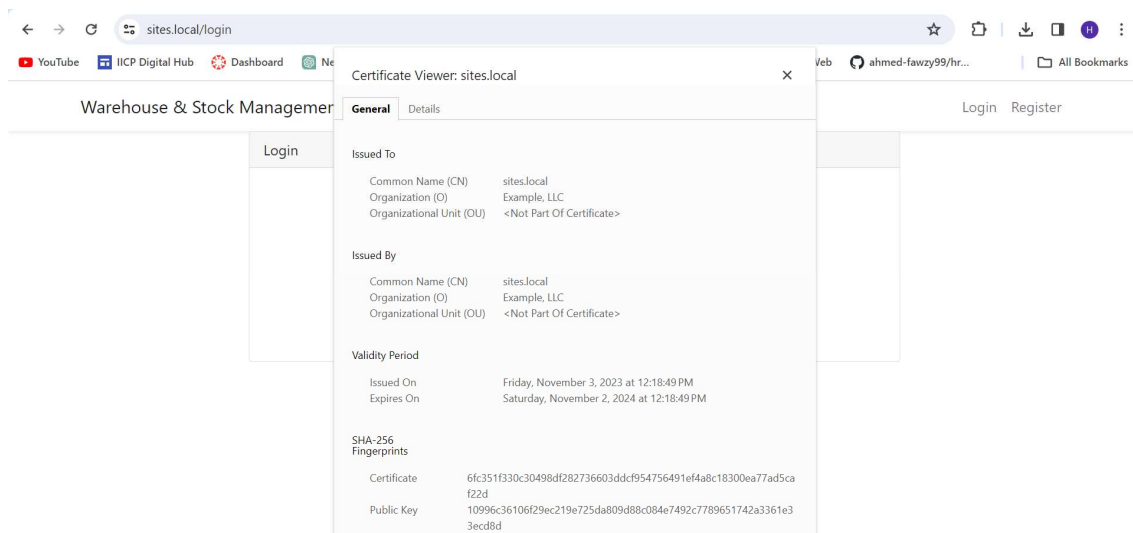


Figure 4-9 Certificate viewer

HTTPS is implemented on the web application successfully. The certificate is valid for one year as shown in figure 4-9 above. The data transmitted between the browser and the server is encrypted, making it more secure.

4.2 Continuous Improvement and Maintenance

The cryptographic key, which is the secret key stored in the .env file must be changed periodically because prolonged key usage increases the risk of unauthorized access and data breaches. When the attackers get the secret key, data can be decrypted efficiently. The .env file can be encrypted to provide additional security level, ensuring that the attackers cannot access the settings and credentials stored inside the file easily.

Continuous and thorough monitoring of user access, data transfers, and system logs is essential. Intrusion detection systems (IDS) and intrusion prevention systems (IPS) can be deployed for prompt threat recognition and mitigation. Furthermore, it is vital to consistently update and patch both the system and its components to address known vulnerabilities in order to maintain the security. In addition, the data must be backed up frequently to ensure data availability in the event of system downtime or cyberattacks.

5.0 Summary

The project has been successfully completed, resulting in the development of a web application that incorporates robust security measures. Authentication and authorization mechanisms have been implemented to guarantee that only authorized users can perform actions aligned with their assigned roles. The input is validated and sanitized before being stored in the system, and the output is encoded before being shown to users. Error messages will be displayed to the users if any errors occur to inform the user.

The system uses a secured protocol, HTTPS to encrypt all data during transmission. AES algorithm is also used to encrypt and decrypt data in the database. This is essential to ensure the system complies with the data protection and privacy regulation, PDPA. Logging information is stored for security purposes.

6.0 References

Bi Irie guy-cedric, Toa. (2018). A Comparative Study on AES 128 BIT AND AES 256 BIT. INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING. volume 6. 30-33. 10.26438/ijsrcse/v6i4.3033.

Jo-el van Bergen. (2019). What is mixed content? | Articles. [online] Available at: <https://web.dev/articles/what-is-mixed-content> [Accessed 3 Nov. 2023].

PVS-Studio. (2021). XSS: attack, defense - and C# programming. [online] Available at: <https://pvs-studio.com/en/blog/posts/csharp/0857/> [Accessed 2 Nov. 2023].

Yasar, K. (2022). What Is a Man-in-the-Middle Attack (MitM)? - Definition from IoTAgenda. [online] IoT Agenda. Available at: <https://www.techtarget.com/iotagenda/definition/man-in-the-middle-attack-MitM>.

 feedback studio®