INT6005CEM

BCSCUN

CS1

Lau En Sin (15731047)

Arief bin Abdul Latib (15730888)

Bryan Yeoh Seng Sheng (15730936)

Hsia Hao Ze (15730992)

Khor Cojean (15731014)

NECESSARY LINKS

GITHUB LINK

- **Original System GitHub Link**
https://github.com/m-2199015/Online-Computer-Store.git

- **Enhanced System GitHub Link**
https://github.com/Ensin2004/INT6005CEM_group_assignment_G23.git

USER MANUAL

- https://github.com/Ensin2004/INT6005CEM_group_assignment_G23/tree/main/User%20Manual

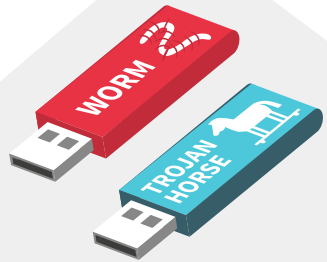


TABLE OF CONTENTS

01.

SYSTEM OVERVIEW

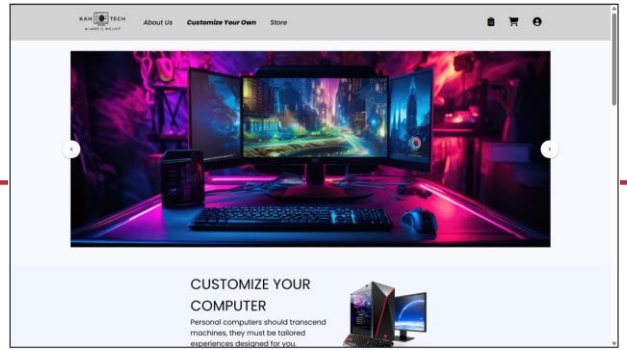
OUR SYSTEM - KAHTECH

KAHTECH

Online computer store paltform for customers in to browse PC components, and place order.

Features

Explore accessories, view product details and customize PC builds



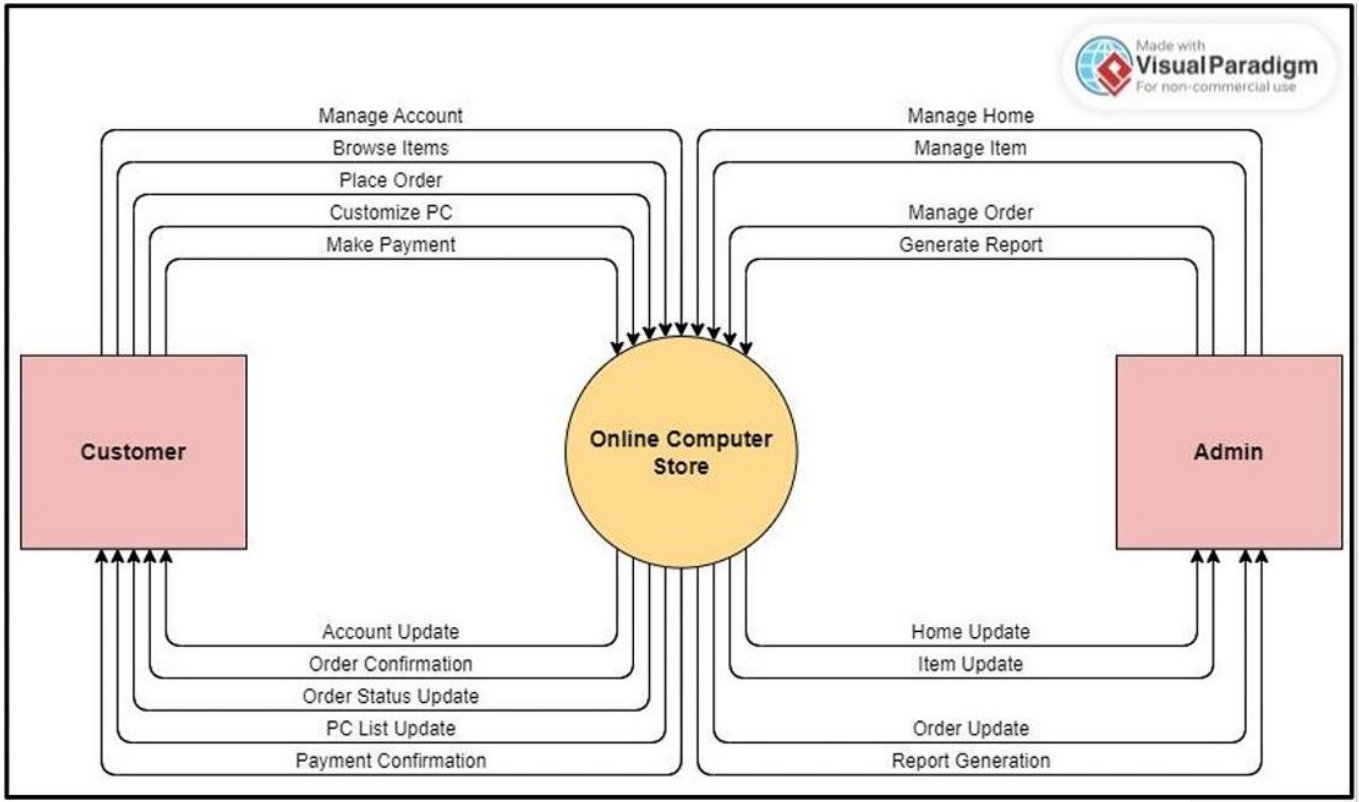
User roles

User and Admin

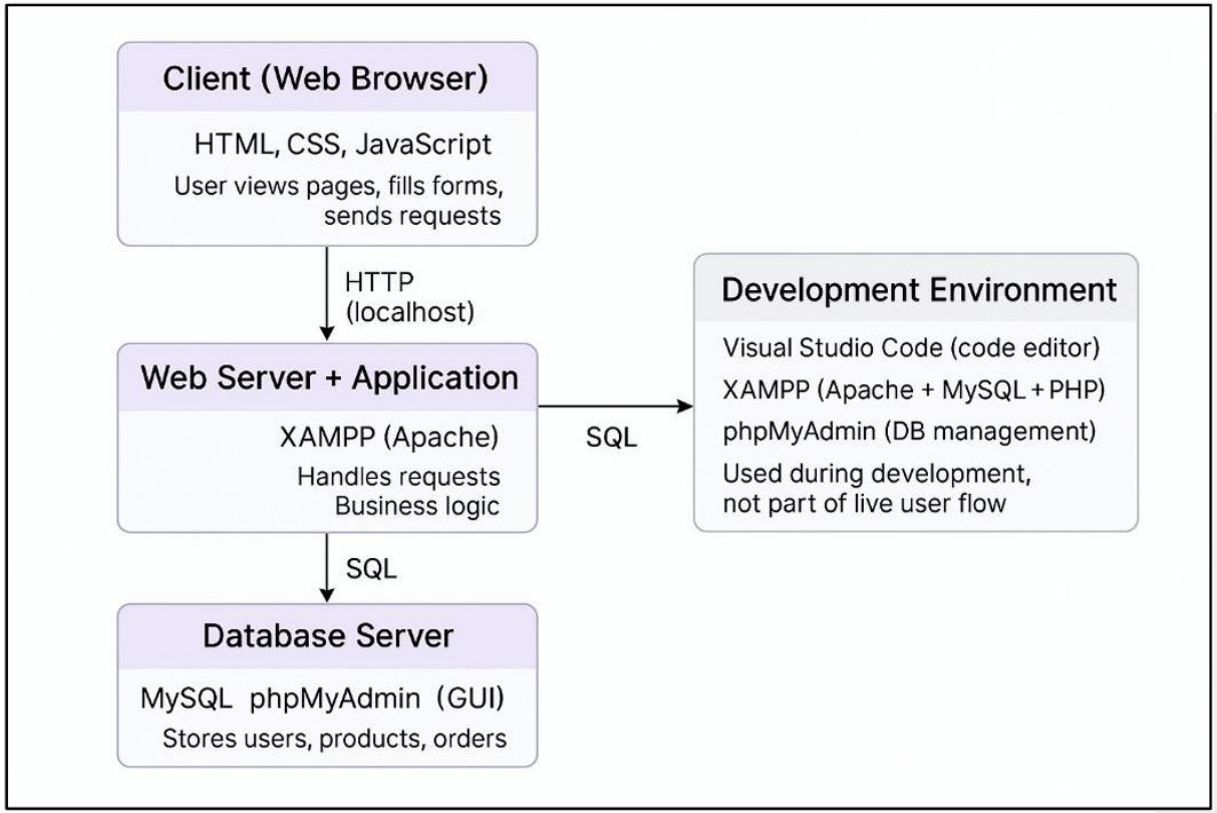
Ordering & Admin

Users can order with TNG e-Wallet or COD, Admins manage products, stock orders and reports

SYSTEM FUNCTIONALITIES



SYSTEM ARCHITECTURE



02.

SYSTEM ANALYSIS

SYSTEM ANALYSIS

01

Authentication

Lack of login rate limiting, super admin and forgot password features.

02

Password Security

Lack of strong password and password hashing implementation.

03

Email Account Handling

Lack of email verification and backup email implementation.

04

Data Privacy & User Education

Lack of data privacy policy and user education.

SYSTEM ANALYSIS

05

Session Management

Lack of session timeout, protection against session fixation and CSRF attacks.

06

Audit Logging

Lack of audit logging feature.

07

Error Handling

Lack of page and system error handling.

08

Input Validation & Cleaning

Lack of input validation, auto-trim, whitespace check and length input limit.

SYSTEM ANALYSIS

09

SQL Injection

Lack of protection against error-based, union-based and blind SQL injection.

10

Cross-Site Scripting (XSS)

Lack of protection against reflected, stored and DOM-based XSS.

11

Cryptography

Lack of protection against data-in-transit and data-at-rest.

12

Secure Cookies

Lack of secure cookies implementation.

SYSTEM ANALYSIS

13

Digital Certificate

Lack of digital certificate deployment.

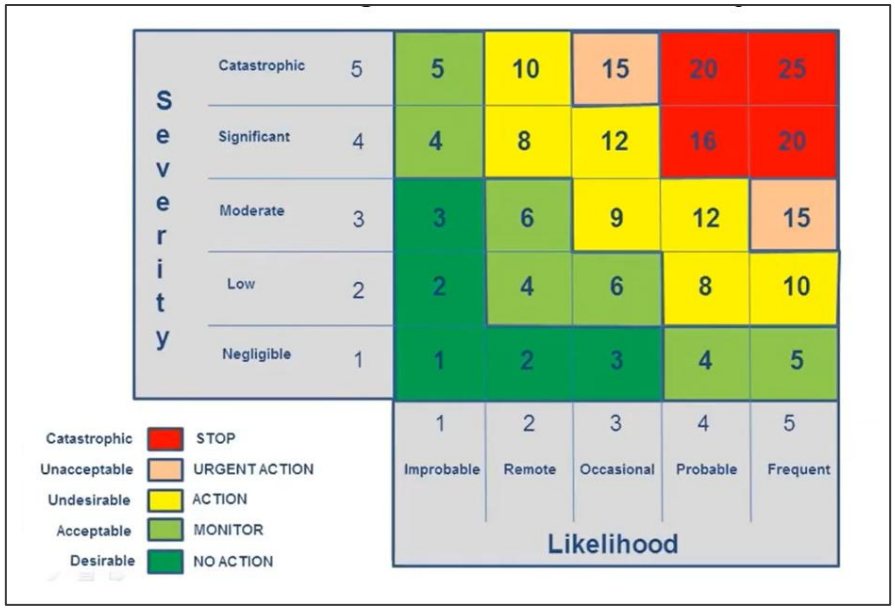
03.

RISK ASSESSMENT

Risk Assessment

$$Risk\ Score = Likelihood \times Severity$$

- **Likelihood**: 1 (Improbable) to 5 (Frequent)
- **Severity**: 1 (Negligible) to 5 (Catastrophic)



Risk Assessment

Vulnerability	Likelihood (1–5)	Severity (1–5)	Risk Score/ Level (L×S)	Reason
Lack of login rate-limiting	5	5	**25** (Catastrophic - STOP)	Unlimited brute-force attempts allow passwords to be guessed easily.
Missing super admin control	4	5	**20** (Catastrophic - STOP)	A compromised manager will gain full system control, no principle of least privilege.
No forgot-password feature	3	3	**9** (Action)	Users risk permanent lockout; more social-engineering opportunities.
Weak password rules	5	5	**25** (Catastrophic - STOP)	Simple passwords are easily cracked by brute force.
No password hashing	5	5	**25** (Catastrophic - STOP)	Plaintext passwords lead to instant account takeover.

Risk Assessment

No email verification	5	4	**20** (Catastrophic - STOP)	Fake accounts, bots, spam and DDOS-style flooding.
No backup email	3	3	**9** (Action)	Permanent lockout if primary email fails.
Missing privacy policy	3	3	**9** (Action)	Unclear consent and possible compliance issues.
No session timeout	4	5	**20** (Catastrophic - STOP)	Unattended sessions expose admin functions.
Potential session fixation	5	5	**25** (Catastrophic - STOP)	Attacker can pre-set the victim's session ID.
No CSRF protection	5	5	**25** (Catastrophic - STOP)	Forced actions possible without user intent.
No audit logging	3	3	**9** (Action)	Hard to detect misuse or suspicious activity.

Risk Assessment

Insecure error messages	4	4	**16** (Unacceptable - Urgent Action)	Server information leaks assist targeted attacks.
Missing input validation	5	5	**25** (Catastrophic - STOP)	Opens the door to SQLi, XSS and malformed data.
SQL Injection	5	5	**25** (Catastrophic - STOP)	Full database compromise is possible.
XSS (Reflected & Stored)	5	5	**25** (Catastrophic - STOP)	Script execution leads to session theft and data manipulation.
Cryptographic failures (HTTP)	5	5	**25** (Catastrophic - STOP)	Credentials travel in clear text; easily intercepted.
Cryptographic failures (plain DB)	5	5	**25** (Catastrophic - STOP)	Full exposure of names, emails, phone, address.
Insecure cookies	5	5	**25** (Catastrophic - STOP)	Session hijacking becomes trivial.
No SSL certificate	5	5	**25** (Catastrophic - STOP)	MITM, sniffing and data tampering during transit.

04.

SECURITY RECOMMENDATION

Security Recommendation

Area	Recommendation
1. Authentication	• Apply login rate limiting to slow down brute-force attempts. • Introduce a dedicated super-admin role to prevent privilege misuse. • Provide a secure forgot-password flow with OTP verification.
2. Password Security	• Enforce strong password rules with minimum length and required character types. • Hash all passwords using Argon2id with high-cost parameters before storing.
3. Multi-Factor Authentication & Email Control	• Verify user email during registration through a one-time code. • Add a backup email feature to support secure account recovery with MFA protection.
4. Data Privacy & User Awareness	• Display clear privacy information on data usage. • Remind users about safe password handling and account hygiene.
5. Session Management & CSRF Protection	• Implement automatic session timeout after inactivity. • Regenerate session IDs after login to prevent fixation. • Protect all sensitive forms with a server-generated CSRF token.
6. Logging & Monitoring	• Record all admin actions and store before/after states for traceability. • Restrict log visibility to super-admins only.

Security Recommendation

7. Error Handling	• Use a unified error page for unexpected failures. • Remove sensitive details from all error outputs to avoid information leakage.
8. Input Validation	• Validate field lengths and content for all user inputs. • Trim whitespace, reject invalid characters and give clear feedback for corrections.
9. SQL Injection Prevention	• Sanitize incoming data and escape special characters where needed. • Use parameterized queries (prepare–bind–execute) for all database operations.
10. XSS Mitigation	• Encode output in the correct context using htmlspecialchars, urlencode and safe link builders. • Ensure no untrusted input is reflected back without encoding.
11. Digital Certifiate Deployment	• Enable HTTPS with a valid SSL/TLS certificate. • Force HTTP→HTTPS redirection. • Enable HSTS for strict secure communication.
12. Cryptography	• Protect all traffic with HTTPS/TLS 1.3 using a valid certificate. • Encrypt sensitive fields at rest with AES-256-GCM, using secure key handling.
13. Browser Cookie Security	• Mark session cookies as Secure, HttpOnly and SameSite=Strict to reduce hijacking risks.

05.

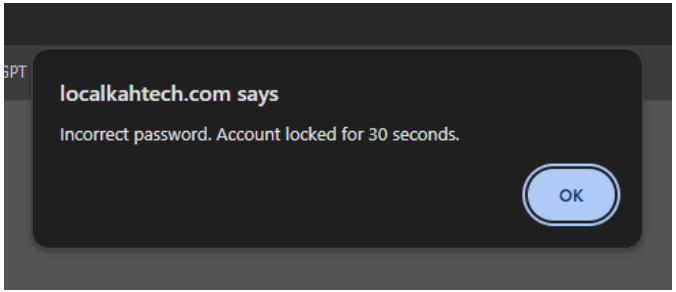
SECURITY IMPLEMENTATION

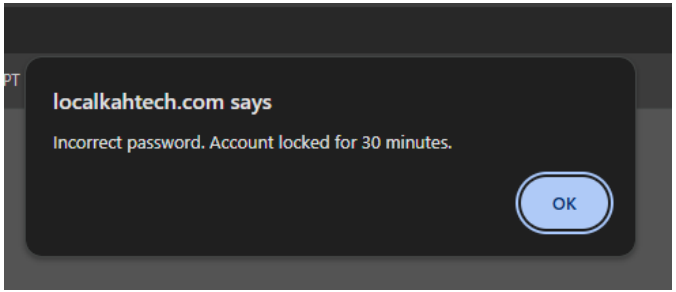
05.

SECURITY IMPLEMENTATION (1. AUTHENTICATION)

Login Rate Limiting

```php
$current_time = date("Y-m-d H:i:s");

// Check whether account is locked or not
if (!is_null($row['lock_until']) && $row['lock_until'] > $current_time) {
    $remaining = strtotime($row['lock_until']) - time();
    echo "<script> alert('Account is locked. Please try again after {$remaining} seconds.'); window.location.href='../login.php'; </script>";
    exit;
}

// Check whether password is correct or not
if (!password_verify($password, $row['pwd'])) {
    $wrong_pwd_count = $row['wrong_pwd_count'] + 1;
    $lock_until = null;
    $lock_message = "";

    // Account lock time
    if ($wrong_pwd_count < 6) {
        $lock_until = null;
        $lock_message = "Incorrect password.";
    } elseif ($wrong_pwd_count == 6) {
        $lock_until = date("Y-m-d H:i:s", strtotime("+30 seconds"));
        $lock_message = "Incorrect password. Account locked for 30 seconds.";
    } elseif ($wrong_pwd_count == 7) {
        $lock_until = date("Y-m-d H:i:s", strtotime("+1 minute"));
        $lock_message = "Incorrect password. Account locked for 1 minute.";
    } elseif ($wrong_pwd_count == 8) {
        $lock_until = date("Y-m-d H:i:s", strtotime("+5 minutes"));
        $lock_message = "Incorrect password. Account locked for 5 minutes.";
    } elseif ($wrong_pwd_count == 9) {
        $lock_until = date("Y-m-d H:i:s", strtotime("+10 minutes"));
        $lock_message = "Incorrect password. Account locked for 10 minutes.";
    } elseif ($wrong_pwd_count > 9) {
        $lock_until = date("Y-m-d H:i:s", strtotime("+30 minutes"));
        $lock_message = "Incorrect password. Account locked for 30 minutes.";
    }

    // Update database
    mysqli_query(
        $conn,
        "UPDATE users SET wrong_pwd_count = '$wrong_pwd_count', lock_until = " . ($lock_until ? "'$lock_until'" : "NULL") . " WHERE id = '{$row['id']}'"
    );

    // Display messages
    echo "<script> alert('$lock_message'); window.location.href='../login.php'; </script>";

} else {
```





Protect the system from brute-force and automated password-guessing attacks.

Super Admin Role Setup

```php
$_SESSION['ID'] = $row['id'];
$_SESSION['AdminName'] = $row['admin_name'];
$_SESSION['role'] = $row['role'];
$_SESSION['status'] = $row['account_status'];
```

```php
<?php if (isset($_SESSION['role']) && $_SESSION['role'] === 'super_admin') { ?>
    <a class="menu_button" href="managers.php">
        <i class="fa-solid fa-users-gear"></i>
        <p class="menu_text">MANAGERS</p>
    </a>
<?php } ?>
```

```php
if (isset($_GET['id']) && isset($_GET['action'])) {
    $adminId = $_GET['id'];
    $action = $_GET['action'];

    if ($action === 'ban') {
        $sql = "UPDATE admins SET account_status = 'banned' WHERE id = ?";
    } elseif ($action === 'unban') {
        $sql = "UPDATE admins SET account_status = 'active' WHERE id = ?";
```

```php
if (isset($row['account_status']) && strtolower(string: $row['account_status']) === 'banned') {
    echo "<script>alert('Your account has been banned. Please contact the system administrator.'); window.history.go(-1);</script>";
    exit();
}
```

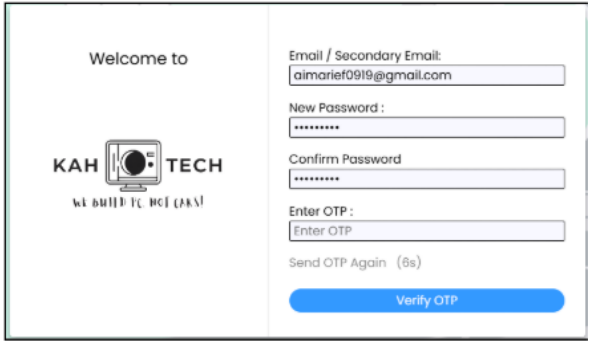
Role Based Access Control, managers do not have access to “managers” page, only admins are able to access it and manage managers.

Forgot Password Mechanism

```php
$checkEmail = mysqli_num_rows(result: mysqli_query(mysql: $conn, query: "SELECT email FROM users
WHERE LOWER(email) = LOWER('$email') OR LOWER(secondary_email) = LOWER('$email');"));
```

```php
if ($checkEmail == 0) {
    echo "<script>alert('Email not exists'); window.history.go(-1);</script>";
    exit;
```

```php
$otp = rand(min: 10000, max: 99999);
$otpExpiresAt = time() + 60;
// Send OTP via email
$to = $email;
$from = "kahtechpng@gmail.com";
$fromName = "KAHTECH";
$message = $otp . " is your OTP.";
$subject = "Secondary Email Verification";
$header = 'From: ' . $fromName . ' <' . $from . '>';

mail(to: $to, subject: $subject, message: $message, additional_headers: $header)
```



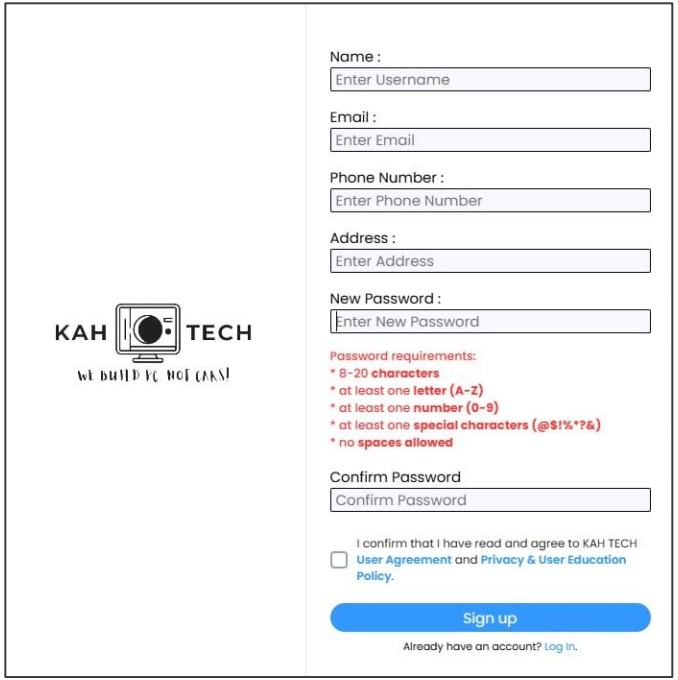
Avoid permanent lock out when user forget the password, also prevent social engineering risk especially for those who store their login information on another platform or physically.

05.

SECURITY IMPLEMENTATION (2. PASSWORD SECURITY)

Strong Password Validation

```
<script>
    document.getElementById('newPassword').addEventListener('input', validatePassword);
    document.getElementById('confirmPassword').addEventListener('input', validatePassword);

    function validatePassword() {
        var password = document.getElementById('newPassword').value;
        var confirmPassword = document.getElementById('confirmPassword').value;
        var pwd_validation_container = document.getElementById('pwd_validation_container');
        var pwd_confirmation = document.getElementById('pwd_confirmation');
        var pwd_character = document.getElementById('pwd_character');
        var pwd_letter = document.getElementById('pwd_letter');
        var pwd_number = document.getElementById('pwd_number');
        var pwd_symbol = document.getElementById('pwd_symbol');
        var submit_btn = document.getElementById('submit_btn');

        const passwordRegex = /^(?=.*[a-zA-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,20}$/;

        if (!passwordRegex.test(password) || confirmPassword != password) {
            submit_btn.disabled = true;

            if (!passwordRegex.test(password)) {
                pwd_validation_container.style.display = "block";
                pwd_confirmation.style.display = "none";

                if (password.length < 8 || password.length > 20) {
                    pwd_character.style.display = "block";
                } else {
                    pwd_character.style.display = "none";
                }

                if (!/[a-zA-Z]/.test(password)) {
                    pwd_letter.style.display = "block";
                } else {
                    pwd_letter.style.display = "none";
                }

                if (!/\d/.test(password)) {
                    pwd_number.style.display = "block";
                } else {
                    pwd_number.style.display = "none";
                }

                if (!/[@$!%*?&]/.test(password)) {
                    pwd_symbol.style.display = "block";
                } else {
                    pwd_symbol.style.display = "none";
                }
            } else {
                pwd_validation_container.style.display = "none";
                pwd_confirmation.style.display = "block";
            }
        } else {
            pwd_validation_container.style.display = "none";
            pwd_confirmation.style.display = "none";
            submit_btn.disabled = false;
        }
    }
</script>
```



Enforced strong password validation, requiring 8-20 characters, one letter, one number, one special character and no spaces allowed.

Password Hashing & Salting

```php
$ARGON_OPTS = [
    'memory_cost' => 131072, // 128 MB
    'time_cost'   => 3,      // 3 iterations
    'threads'     => 1
];
```

```php
$password = password_hash($_POST["newPassword"], PASSWORD_ARGON2ID, $ARGON_OPTS);
```

16	Kamen Rider	p22014743@student.newinti.edu.my	019-8284731	Seri Melati, George Town	Kamen123@
17	Tan Mei Mei	p22014743@student.newinti.edu.my	017-2374621	Air Itam Dam	Tanmeimei123@
19	Junzo	junzobryn28@gmail.com	0123456789	bryan 123, at jalan bryan	$argon2id$v=19$m=131072,t=3,p=1$NkxTbjJWTktORjZEZE

```
$argon2id$v=19$m=131072,t=3,p=1$NkxTbjJWTktORjZEZE5Kbw$HI/rZQBEULzlF+3Y8KpgczH5z0VVUMBRYvGEWM8dfEg
```

Meta Data | Salt | Hash

Converts passwords into irreversible hashes with unique salts instead of plaintext so stolen database data cannot reveal the original passwords

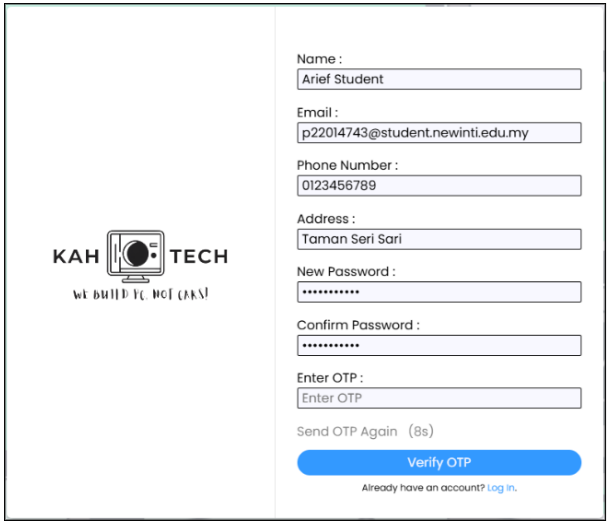
05.

SECURITY IMPLEMENTATION (3. MFA & EMAIL CONTROL)

Email Verification

```php
$checkResult = mysqli_num_rows(result: mysqli_query(mysql: $conn,
query: "SELECT user_name FROM users WHERE LOWER(user_name) = LOWER('$name');"));

$checkEmail = mysqli_num_rows(result: mysqli_query(mysql: $conn,
query: "SELECT email FROM users WHERE LOWER(email) = LOWER('$email');"));

$checkSecondaryEmail = mysqli_num_rows(result: mysqli_query(mysql: $conn,
query: "SELECT secondary_email FROM users WHERE LOWER(secondary_email) = LOWER('$email');"));
```

```php
$otp = rand(min: 10000, max: 99999);
$otpExpiresAt = time() + 60;
// Send OTP via email
$to = $email;
$from = "kahtechpng@gmail.com";
$fromName = "KAHTECH";
$message = $otp . " is your OTP.";
$subject = "New Account Sign Up";
$header = 'From: ' . $fromName . ' <' . $from . '>';

mail(to: $to, subject: $subject, message: $message, additional_headers: $header)
```



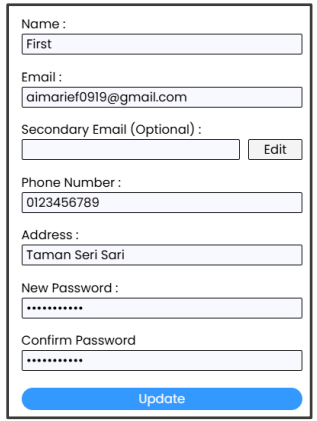
Username and email addresses that are used cannot be used again and OTP will be sent for verification so one email can only sign up for one time to prevent any spamming of bot accounts which may lead to DDOS attack.

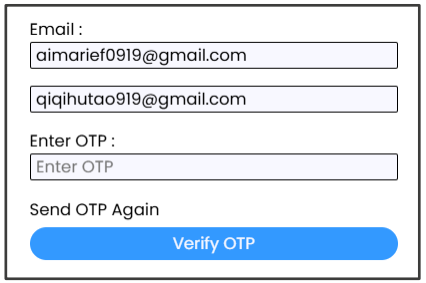
Backup Email / Recovery with MFA

```php
$checkEmail = mysqli_num_rows(result: mysqli_query(mysql: $conn, query: "SELECT email FROM users
WHERE LOWER(email) = LOWER('$email') OR LOWER(secondary_email) = LOWER('$email');"));
$query = mysqli_query(mysql: $conn, query: "SELECT * FROM users WHERE email = '$firstEmail'");
$row = mysqli_fetch_assoc(result: $query);

if (mysqli_num_rows(result: $query) == 0 || $row['pwd'] != $confirmPassword) {
    echo "<script> alert('Wrong password'); window.history.go(-1); </script>";
    exit;
}
```

```php
if ($checkEmail != 0) {
    echo "<script>alert('Email already exists'); window.history.go(-1);</script>";
    exit;
```

```php
    $noAutoSend   = isset($_POST['no_autosend']) || isset($_GET['no_autosend']);
    $otp = $_POST['otp'] ?? ($_GET['otp'] ?? null);
    $otpExpiresAt = $_POST['otp_expires_at'] ?? ($_GET['otp_expires_at'] ?? null);

    $firstEmail      = htmlspecialchars(string: $_POST['primaryEmail']     ?? $_GET['primaryEmail']     ?? '');
    $email           = htmlspecialchars(string: $_POST['secondaryEmail']   ?? $_GET['secondaryEmail']   ?? '');
    $confirmPassword = htmlspecialchars(string: $_POST['confirmPassword']  ?? $_GET['confirmPassword']  ?? '');

    // Send OTP via email
    $to = $email;
    $from = "kahtechpng@gmail.com";
    $fromName = "KAHTECH";
    $subject = "Secondary Email Verification";
    $header = 'From: ' . $fromName . ' <' . $from . '>';

  if ((($otp === null) && !$noAutoSend) || isset($_POST['resend'])) {
$otp = rand(min: 10000, max: 99999);
$otpExpiresAt = time() + 60; // seconds
$message = $otp . " is your OTP.";
@mail(to: $to, subject: $subject, message: $message, additional_headers: $header);
```





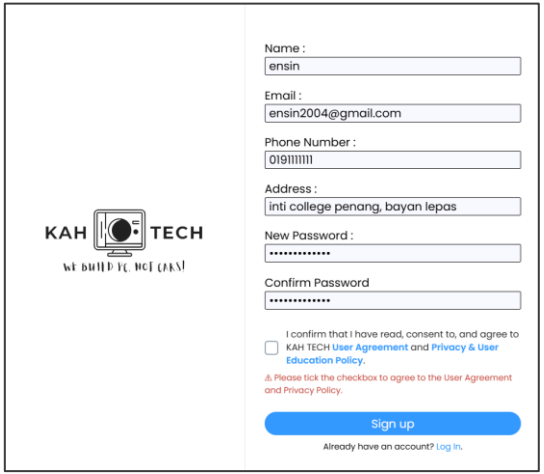
If the main email has problems, the backup allows users to have a second way to prove ownership which prevent users from being permanently unavailable from the system.

05.

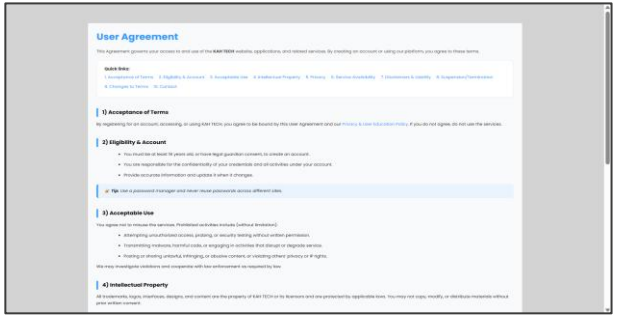
SECURITY IMPLEMENTATION (4. DATA PRIVACY & USER EDUCATION)

Data Privacy & User Education

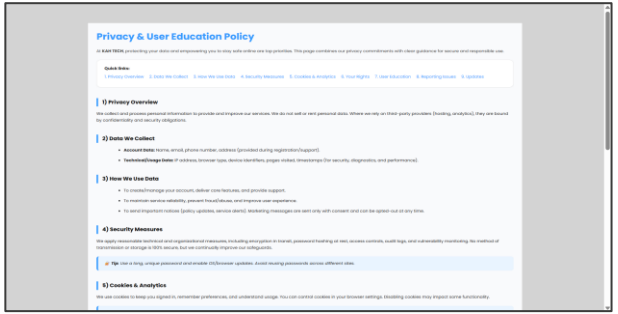
```
<!-- === Consent checkbox === -->
<div class="consent" style="margin-top:10px;">
    <label style="display:flex; gap:8px; align-items:flex-start; line-height:1.4;">
        <input type="checkbox" id="agree_terms" style="margin-top:3px;">
        <span>
            I confirm that I have read, consent to, and agree to KAH TECH
            <a href="user-agreement.php" target="_blank" rel="noopener">User Agreement</a>
            and
            <a href="privacy-education.php" target="_blank" rel="noopener">Privacy & User Education Policy</a>.
        </span>
    </label>
    <p id="agree_error" style="display:none; color:#c0392b; margin-top:6px;">
        Please tick the checkbox to agree before signing up.
    </p>
</div>
```



User agreement



Privacy & user education



These controls ensure every user clearly understands the terms, privacy practices and safety guidelines before creating an account

05.

SECURITY IMPLEMENTATION (5. SESSION MANAGEMENT & CSRF PROTECTION)

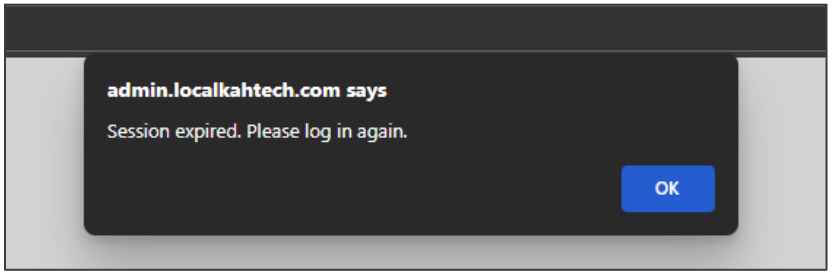
Session Timeout

sessionTimeout.js

```javascript
// Timeout minutes
const timeoutMinutes = 5;

// Timeout seconds
const timeoutSeconds = timeoutMinutes * 60;

// Idle time counter in seconds
let idleTime = 0;

// Session flag
let sessionExpired = false;

// Reset idle time counter
function resetIdleTime() {
    idleTime = 0;
}

// Increment idle time every 10 seconds
const idleInterval = setInterval(() => {
    if (sessionExpired) return;

    idleTime += 10;

    // Log out if idle time = timeout time
    if (idleTime >= timeoutSeconds) {
        sessionExpired = true;
        window.location.href = "includes/logoutAccount.php?timeout=1";
    }
}, 10000);

// Detect user activity and reset idle time counter
document.addEventListener("mousemove", resetIdleTime);
document.addEventListener("click", resetIdleTime);
document.addEventListener("scroll", resetIdleTime);
document.addEventListener("keydown", resetIdleTime);
```

security.php

```php
// Session Timeout Check
$timeoutSeconds = 300;

if (isset($_SESSION['LastActivity']) && (time() - $_SESSION['LastActivity']) >= $timeoutSeconds) {
    echo "<script> window.location.href='includes/logoutAccount.php?timeout=1'; </script>";
    exit;
}

$_SESSION['LastActivity'] = time();
```

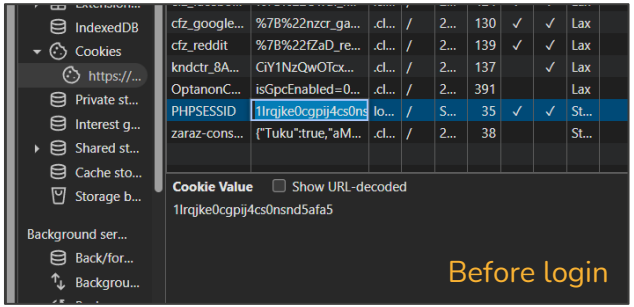


Protect the system from risks associated with unattended devices, shared computers and session misuse.

Session Regeneration

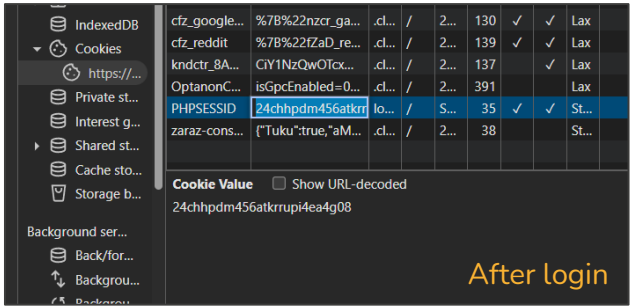
loginuser.php

```
includes > loginuser.php > ...
88                  mysqli_query(
89                      $conn,
90                      "UPDATE users SET wrong_pwd_count = 0, lock_until = NULL WHERE id = '{$row['id']}'"
91                  );
92
93                  // Regenerate session ID (prevent session fixation)
94                  session_regenerate_id(true);
95
96                  // Update session
97                  $_SESSION['ID'] = $row['id'];
98                  $_SESSION['UserName'] = $row['user_name'];
99
100                 // Display messages
101                 echo "<script> alert('Log in successfully'); window.location.href='../index.php'; </script>";
```



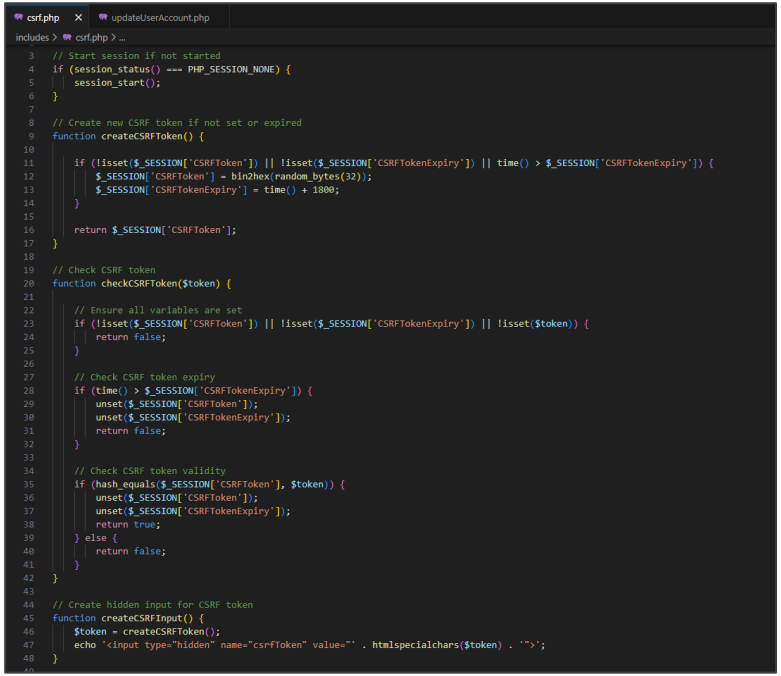
logoutAccount.php

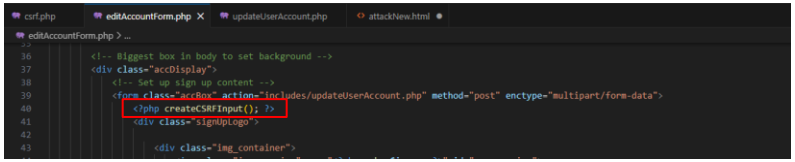
```
includes > logoutAccount.php > ...
25                  $params['httponly']
26              );
27          }
28
29          // 3. Destroy the session on the server
30          session_unset();
31          session_destroy();
32
33          // 4. Start new empty session with new session id
34          session_start();
35          session_regenerate_id(true);
36
37          // 5. Redirect with alert
38          echo "<script>alert('Log out successfully'); window.location.href='../index.php';</script>";
```

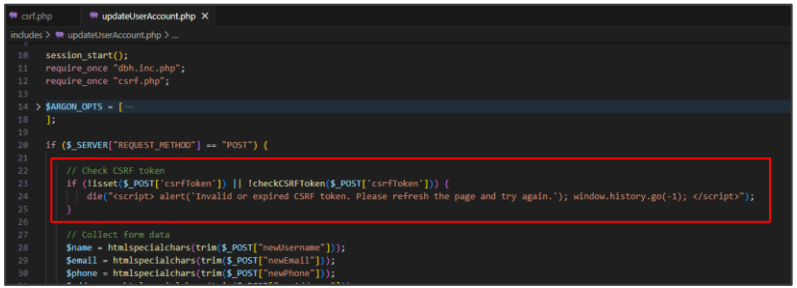


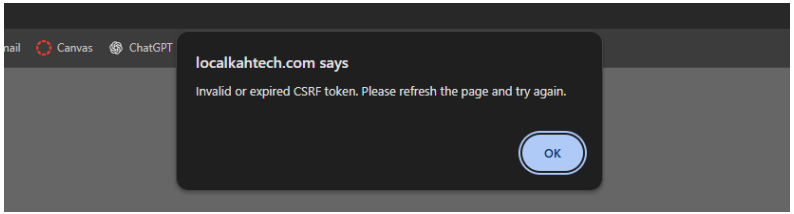
Attackers can no longer reuse or fix the session ID on the victim's device, mitigate the risk of session fixation attacks.

CSRF Token Implementation





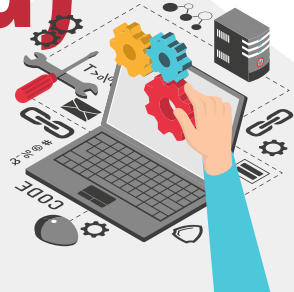




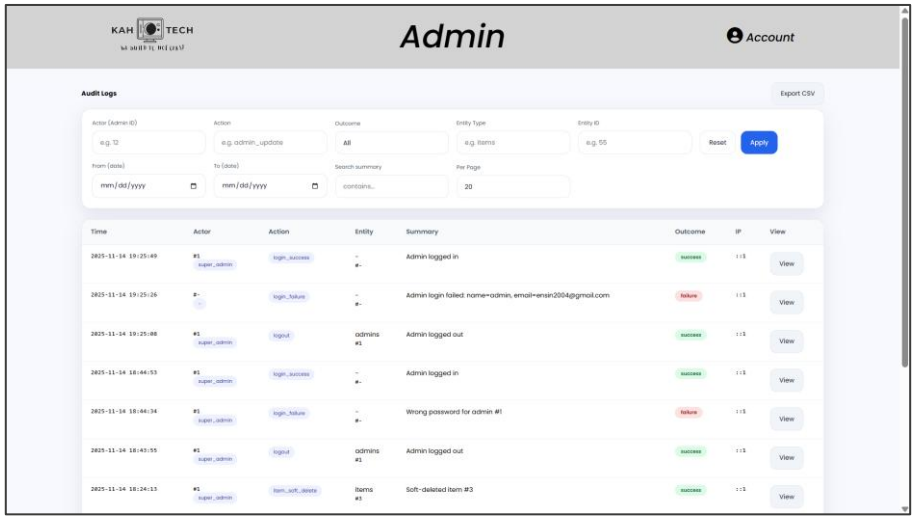
CSRF token prevents request from unauthorized websites to be executed.

05.

SECURITY IMPLEMENTATION (6. LOGGING & MONITORING)



Logging & Monitoring



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	actor_admin_id	int(11)			Yes	NULL			Change Drop More
3	actor_role	enum('manager', 'super_admin')	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	action	varchar(64)	utf8mb4_general_ci		No	None			Change Drop More
5	entity_type	varchar(64)	utf8mb4_general_ci		Yes	NULL			Change Drop More
6	entity_id	int(11)			Yes	NULL			Change Drop More
7	summary	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
8	before_json	longtext	utf8mb4_bin		Yes	NULL			Change Drop More
9	after_json	longtext	utf8mb4_bin		Yes	NULL			Change Drop More
10	outcome	enum('success', 'failure')	utf8mb4_general_ci		No	success			Change Drop More
11	ip_address	varchar(45)	utf8mb4_general_ci		Yes	NULL			Change Drop More
12	user_agent	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
13	created_at	datetime			No	current_timestamp()			Change Drop More

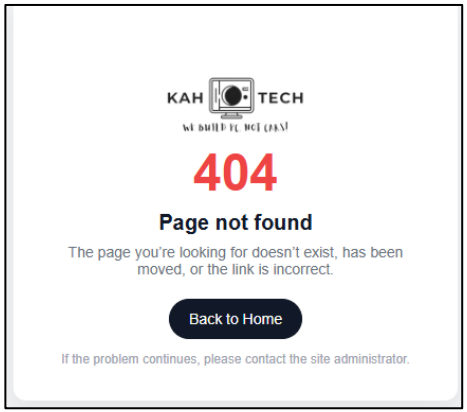
Strengthens accountability, supports incident investigation and ensures all critical actions are fully traceable.

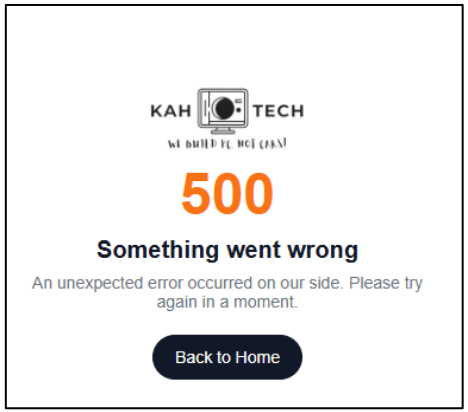
05.

SECURITY IMPLEMENTATION (7. ERROR HANDLING)

Unified Error Page

```
<VirtualHost *:443>
    ServerName www.localkahtech.com
    ServerAlias localkahtech.com
    DocumentRoot "C:/xampp/htdocs/INT6005CEM_group_assignment_G23/Online Computer Store"

    ServerSignature Off
    ErrorDocument 404 /errors/404.php
    ErrorDocument 500 /errors/500.php

    Alias /Image "C:/xampp/htdocs/INT6005CEM_group_assignment_G23/Image"

    <Directory "C:/xampp/htdocs/INT6005CEM_group_assignment_G23/Image">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    SSLEngine on
    SSLCertificateFile "conf/ssl.crt/server.crt"
    SSLCertificateKeyFile "conf/ssl.key/server.key"

    Header always set Strict-Transport-Security "max-age=31536000"
</VirtualHost>
```






```php
function handleErrorAndExit($message = 'Unexpected error during sign up.') {
    error_log('[SIGNUP ERROR] ' . $message);
    http_response_code(500);
    header("Location: ../errors/500.php");
    exit;
}
```

Provides a single safe fallback screen so unexpected failures show controlled messages without exposing system details.

Suppress Sensitive Details

```
display_errors=Off

; The display of errors which occur during PHP's startup sequence are handled
; separately from display_errors. We strongly recommend you set this to 'off'
; for production servers to avoid leaking configuration details.
; Default Value: On
; Development Value: On
; Production Value: Off
; https://php.net/display-startup-errors
display_startup_errors=Off

; Besides displaying errors, PHP can also log errors to locations such as a
; server-specific log, STDERR, or a location specified by the error_log
; directive found below. While errors should not be displayed on productions
; servers they should still be monitored and logging is a great way to do that.
; Default Value: Off
; Development Value: On
; Production Value: On
; https://php.net/log-errors
log_errors=On
```

```
24999  59 2025] [php:warn] [pid 51472:tid 1936] [client ::1:2988] PHP Warning:  Undefined variable $img3_file_name in C:\\xampp\\htdocs\\Online-Computer-Store-main\\Admin Online Compu
25000  59 2025] [php:warn] [pid 51472:tid 1936] [client ::1:2988] PHP Warning:  Undefined variable $img4_file_name in C:\\xampp\\htdocs\\Online-Computer-Store-main\\Admin Online Compu
25001  59 2025] [php:warn] [pid 51472:tid 1936] [client ::1:2988] PHP Warning:  Undefined variable $img5_file_name in C:\\xampp\\htdocs\\Online-Computer-Store-main\\Admin Online Compu
25002  59 2025] [php:error] [pid 51472:tid 1936] [client ::1:2988] PHP Fatal error:  Uncaught mysqli_sql_exception: Unknown column 'abc' in 'field list' in C:\\xampp\\htdocs\\Online-C
25003  22 2025] [php:error] [pid 51472:tid 1944] [client 127.0.0.1:54030] script 'C:/xampp/htdocs/INT6005CEM_group_assignment_G23/Online Computer Store/hello.php' not found or unable
25004  38 2025] [php:error] [pid 51472:tid 1944] [client 127.0.0.1:54030] script 'C:/xampp/htdocs/INT6005CEM_group_assignment_G23/Online Computer Store/hello.php' not found or unable
25005  21 2025] [php:notice] [pid 51472:tid 1920] [client 127.0.0.1:31054] [LOGIN ERROR] Table 'computer_store.bryan' doesn't exist, referer: https://localkahtech.com/login.php
25006
```

Hides internal system information so errors reveal only safe messages, preventing attackers from learning anything useful for exploitation

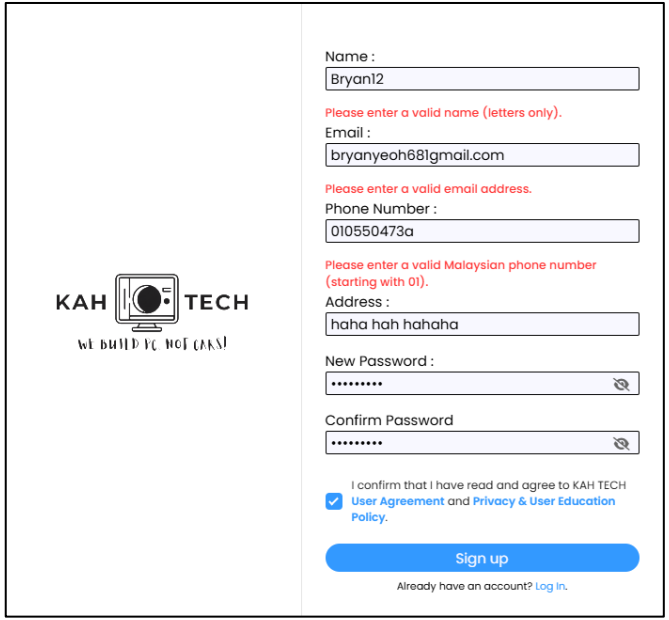
05.

SECURITY IMPLEMENTATION (8. INPUT VALIDATION & CLEANING)

Input Validation

```
// ------------ Name validation ------------
function validateName() {
    const name = nameInput.value.trim();
    const error = document.getElementById('nameError');

    // Letters, spaces, apostrophes, dots, hyphens; length 2-50
    const nameRegex = /^[A-Za-z\s'.-]{2,50}$/;

    if (name.length === 0) {
        error.style.display = "none";
        isNameValid = false;
    } else if (nameRegex.test(name)) {
        error.style.display = "none";
        isNameValid = true;
    } else {
        error.style.display = "block";
        isNameValid = false;
    }

    updateSubmitButton();
}

// ------------ Email validation ------------
function validateEmail() {
    const error = document.getElementById('emailError');
    const email = emailInput.value.trim();

    if (email.length === 0) {
        error.style.display = "none";
        isEmailValid = false;
    } else if (emailInput.validity.valid) {
        error.style.display = "none";
        isEmailValid = true;
    } else {
        error.style.display = "block";
        isEmailValid = false;
    }

    updateSubmitButton();
}
```

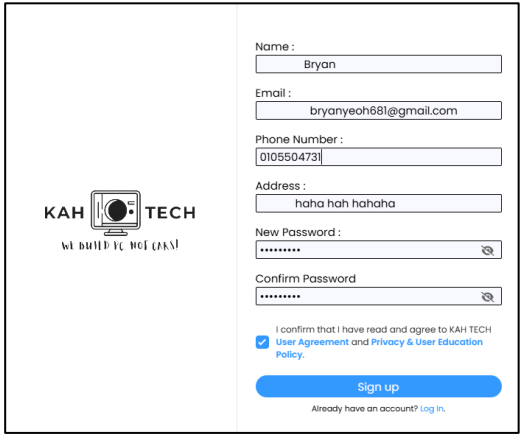
```
// ------------ Malaysian phone validation ------------
function validatePhone() {
    const phone = phoneInput.value.trim();
    const error = document.getElementById('phoneError');

    // Simple Malaysian format:
    // - starts with 0
    // - total 10 or 11 digits
    const phoneRegex = /^01\d{8,9}$/;

    if (phone.length === 0) {
        error.style.display = "none";
        isPhoneValid = false;
    } else if (phoneRegex.test(phone)) {
        error.style.display = "none";
        isPhoneValid = true;
    } else {
        error.style.display = "block";
        isPhoneValid = false;
    }

    updateSubmitButton();
}
```



Input validation checks whether submitted data follows the required rules so only safe, expected values enter the system, blocking harmful or invalid inputs.

Trim & Whitespace Checks

```php
$name = htmlspecialchars(trim($_POST["newUsername"]));
$email = htmlspecialchars(trim($_POST["newEmail"]));
$phone = htmlspecialchars(trim($_POST["newPhone"]));
$address = htmlspecialchars(trim($_POST["newAddress"]));
$password = password_hash($_POST["newPassword"], PASSWORD_ARGON2ID, $ARGON_OPTS);
```



19 Junzo	junzobryn28@gmail.com	0123456789	bryan 123, at jalan bryan	$argon2id$v=19$m=131072,t=3,p=1$NkxTbjJWTktORjZEZE...
20 Bryan	bryanyeoh681@gmail.com	0105504731	haha hah hahaha	$argon2id$v=19$m=131072,t=3,p=1$ZEV1T1ZCUktVQ1Q4aD...

Trim and whitespace checks remove extra spaces and hidden characters so the system processes clean, accurate input and blocks sneaky manipulation.

Input Limiting

```
<div class="signUpInfo">
    <label for="Username">Name :</label>
    <input required type="text" id="Username" name="newUsername" maxlength="50" placeholder="Enter
    Username">
    <p class="limit-warning" id="nameLimit">Character limit reached (50)</p>
    <p class="validation-error" id="nameError">Please enter a valid name (letters only).</p>

    <label for="Email">Email :</label>
    <input required type="email" id="Email" name="newEmail" maxlength="100" placeholder="Enter Email">
    <p class="limit-warning" id="emailLimit">Character limit reached (100)</p>
    <p class="validation-error" id="emailError">Please enter a valid email address.</p>

    <label for="Phone">Phone Number :</label>
    <input required type="text" id="Phone" name="newPhone" maxlength="11" placeholder="Enter Phone
    Number">
    <p class="limit-warning" id="phoneLimit">Character limit reached (11)</p>
    <p class="validation-error" id="phoneError">Please enter a valid Malaysian phone number (starting
    with 01).</p>

    <label for="Address">Address :</label>
    <input required type="text" id="Address" name="newAddress" maxlength="200" placeholder="Enter
    Address">
    <p class="limit-warning" id="addressLimit">Character limit reached (200)</p>

    <label for="newPassword">New Password :</label>
    <div class="pwd-wrapper">
        <input required type="password" id="newPassword" name="newPassword" maxlength="20"
        placeholder="Enter New Password">
        <i class="fa-solid fa-eye-slash toggle-eye" onclick="togglePassword('newPassword', this)"></i>
    </div>
    <p class="limit-warning" id="newPwdLimit">Character limit reached (20)</p>
    <div class="pwd_validation_container" id="pwd_validation_container">
        <p>Password requirements: </p>
        <p class="pwd_validation" id="pwd_character">* 8-20 <b>characters</b></p>
        <p class="pwd_validation" id="pwd_letter">* at least one <b>letter (A-Z)</b></p>
        <p class="pwd_validation" id="pwd_number">* at least one <b>number (0-9)</b></p>
        <p class="pwd_validation" id="pwd_symbol">* at least one <b>special characters (@$!%*?&)</b></
        p>
        <p class="pwd_validation" id="pwd_space">* no <b>spaces allowed</b></p>
    </div>

    <label for="confirmPassword">Confirm Password</label>
    <div class="pwd-wrapper">
        <input required type="password" id="confirmPassword" name="confirmPassword" maxlength="20"
        placeholder="Confirm Password">
        <i class="fa-solid fa-eye-slash toggle-eye" onclick="togglePassword('confirmPassword', this)
        "></i>
    </div>
    <p class="limit-warning" id="confirmPwdLimit">Character limit reached (20)</p>
    <p class="pwd_confirmation" id="pwd_confirmation">Password not match</p>
```

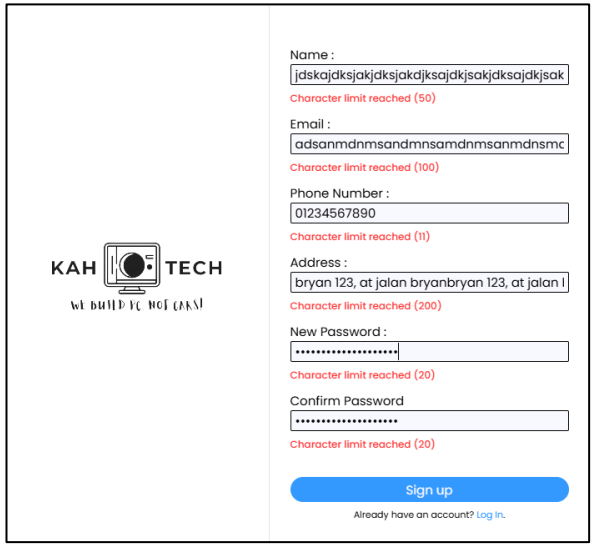
```
// Field Limit Warning
function setupLimitWarning(inputId, warningId, max) {
    const input = document.getElementById(inputId);
    const warning = document.getElementById(warningId);

    input.addEventListener('input', () => {
    if (input.value.length === max) {
        warning.style.display = "block";
    } else {
        warning.style.display = "none";
    }
    });
}

// Initialize limit warnings for all fields
setupLimitWarning('Username', 'nameLimit', 50);
setupLimitWarning('Email', 'emailLimit', 100);
setupLimitWarning('Phone', 'phoneLimit', 11);
setupLimitWarning('Address', 'addressLimit', 200);
setupLimitWarning('newPassword', 'newPwdLimit', 20);
setupLimitWarning('confirmPassword', 'confirmPwdLimit', 20);

// ------------ Attach validation listeners ------------
nameInput.addEventListener('input', validateName);
emailInput.addEventListener('input', validateEmail);
phoneInput.addEventListener('input', validatePhone);
```

```
function validatePrice(input) {
    const warning = document.getElementById("priceWarning");
    if (parseFloat(input.value) > parseFloat(input.max)) {
        input.value = input.max;
        warning.style.display = "block";
    } else {
        warning.style.display = "none";
    }
}

function validateStock(input) {
    const warning = document.getElementById("stockWarning");
    if (parseInt(input.value) > parseInt(input.max)) {
        input.value = input.max;
        warning.style.display = "block";
    } else {
        warning.style.display = "none";
    }
}

// Field Limit Warning
function setupLimitWarning(inputId, warningId, max) {
    const input = document.getElementById(inputId);
    const warning = document.getElementById(warningId);

    input.addEventListener('input', () => {
    if (input.value.length === max) {
        warning.style.display = "block";
    } else {
        warning.style.display = "none";
    }
    });
}

// Initialize limit warnings for all fields
setupLimitWarning('name', 'nameLimit', 255);
setupLimitWarning('description', 'descriptionLimit', 999);
```

Input limiting restricts data size and format so only safe, valid inputs are accepted, blocking malicious or oversized submissions.

Input Limiting

User Sign Up Form Limit

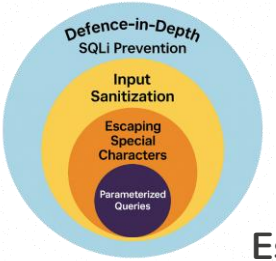


Admin Add Item Form Limit

05.

SECURITY IMPLEMENTATION (9. SQL INJECTION PREVENTION)

SQL Prevention



Input Sanitization

```php
/**
 * Basic sanitization only (no validation here):
 * - trim
 * - strip HTML tags
 * - remove special symbols (keeps letters/numbers/space/-/_/./,)
 */
3 references
function sanitize_basic(?string $s): string {
    if ($s === null) return '';
    $s = trim(string: strip_tags(string: $s));
    // allow: a-z, A-Z, 0-9, space, dash, underscore, dot, comma
    $s = preg_replace(pattern: '/[^a-zA-Z0-9 \-\_\.\,]/u', replacement: '', subject: $s);
    // collapse multiple spaces
    $s = preg_replace(pattern: '/\s+/u', replacement: ' ', subject: $s);
    return $s;
}
```

```php
// Sanitize inputs
$category_raw = $_GET['category'] ?? null;
$search_raw   = $_GET['search']   ?? null;

$category = $category_raw !== null ? sanitize_basic(s: $category_raw) : null;
$search   = $search_raw   !== null ? sanitize_basic(s: $search_raw)   : null;
```

Escaping Special Characters

```php
// utf8mb4 for safety
$conn->set_charset(charset: 'utf8mb4');
```

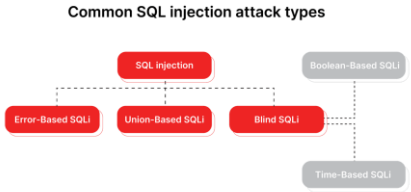
```php
$like_esc = $conn->real_escape_string(string: "%{$search}%");
$sql_legacy = "
  SELECT items.id, items.item_name, items.price, items.stock_qty, items.description,
         items.image1, items.image2, items.image3, items.image4, items.image5,
         categories.category_name
  FROM items
  LEFT JOIN categories ON items.category_id = categories.id
  WHERE item_status = 'Active'
    AND (
      LOWER(items.item_name)             LIKE LOWER('{$like_esc}')
      OR LOWER(items.description)        LIKE LOWER('{$like_esc}')
      OR LOWER(categories.category_name) LIKE LOWER('{$like_esc}')
    )
";
```

Parameterized Queries

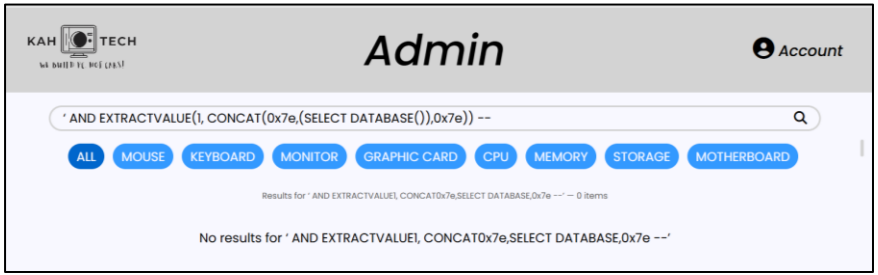
```php
$like = "%{$search}%";
$sql = "
  SELECT items.id, items.item_name, items.price, items.stock_qty, items.description,
         items.image1, items.image2, items.image3, items.image4, items.image5,
         categories.category_name
  FROM items
  LEFT JOIN categories ON items.category_id = categories.id
  WHERE item_status = 'Active'
    AND (
      LOWER(items.item_name)             LIKE LOWER(?)
      OR LOWER(items.description)        LIKE LOWER(?)
      OR LOWER(categories.category_name) LIKE LOWER(?)
    );
";
$stmt = $conn->prepare(query: $sql);

if ($stmt) {
    $stmt->bind_param(types: "sss", var: &$like, vars: &$like, $like); // escaping not needed for prepared statements
    $stmt->execute();
    $itemResult = $stmt->get_result();
    $stmt->close();
```

Together, these layered controls ensure all user inputs are safely handled and malicious SQL commands are fully prevented from reaching or manipulating the database

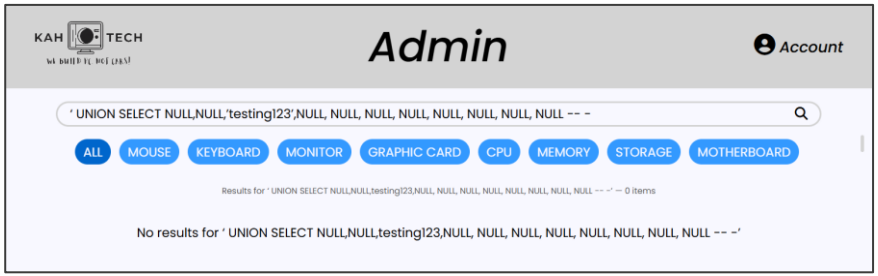
SQL Prevention



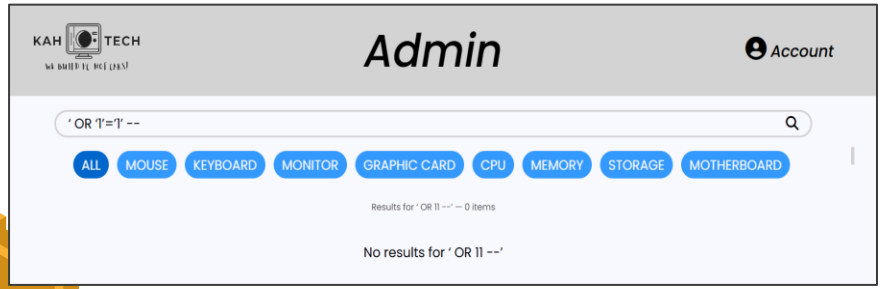
Error based SQL Injection



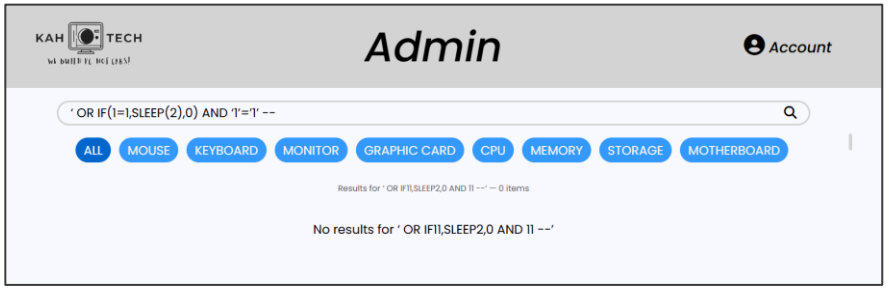
Union based SQL Injection



Blind SQLi Injection (Boolean)



Blind SQLi Injection (Time)



05.

SECURITY IMPLEMENTATION (10. XSS MITIGATION)

XSS Mitigation

```php
/**
 * Output-encoding helpers (XSS mitigation by context)
 */

10 references
function e(?string $s): string {            // HTML text/attribute
    return htmlspecialchars(string: $s ?? '', flags: ENT_QUOTES, encoding: 'UTF-8');
}

2 references
function q(string $s): string {             // URL query parameter
    return urlencode(string: $s);
}

1 reference
function qp(string $s): string {            // URL path segment
    return rawurlencode(string: $s);
}
```

```php
$catHref = "store.php?category=" . q(s: $catName);
$cls = $isSelected ? 'category_button selected' : 'category_button';
echo '<a class="'.$cls.'" href="'.$catHref.'">' .
     strtoupper(string: e(s: $catName)) . '</a>';
```

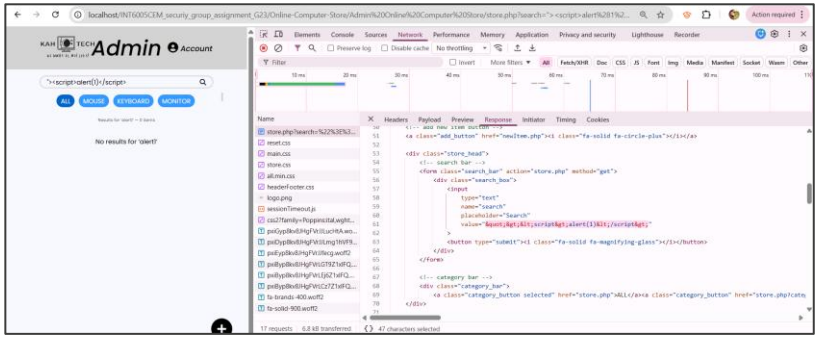
```php
<!-- search bar -->
<form class="search_bar" action="store.php" method="get">
    <div class="search_box">
        <input
            type="text"
            name="search"
            placeholder="Search"
            value="<?php echo isset($_GET['search']) ? e(s: $_GET['search']) : ''; ?>"
        >
        <button type="submit"><i class="fa-solid fa-magnifying-glass"></i></button>
    </div>
</form>
```

```php
$id    = (int)$row['id']; // numeric only
$name  = e(s: $row['item_name']);
$desc  = e(s: $row['description']);
$price = e(s: $row['price']);
// guard filename, then URL-encode for path usage
$img1f = basename(path: $row['image1'] ?? '');
$imgSrc = "../Image/" . qp(s: $img1f);
$stock = (int)$row['stock_qty'];
$detailsHref = "itemDetails.php?item={$id}";
```

Ensure that any malicious input is safely neutralised and displayed as harmless text, preventing scripts from ever running in the browser

XSS Mitigation

Script Tag



Attribute Injection

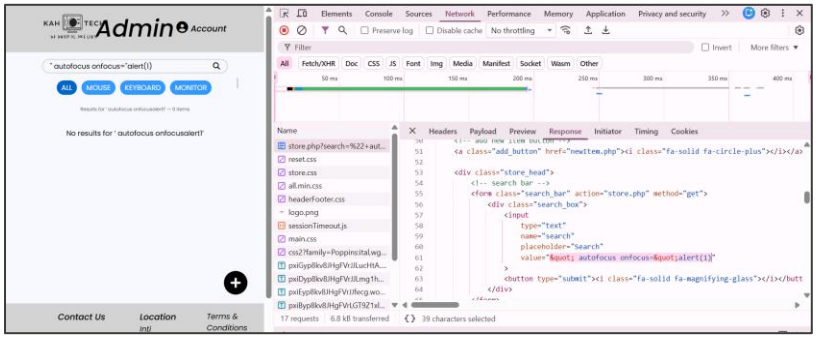
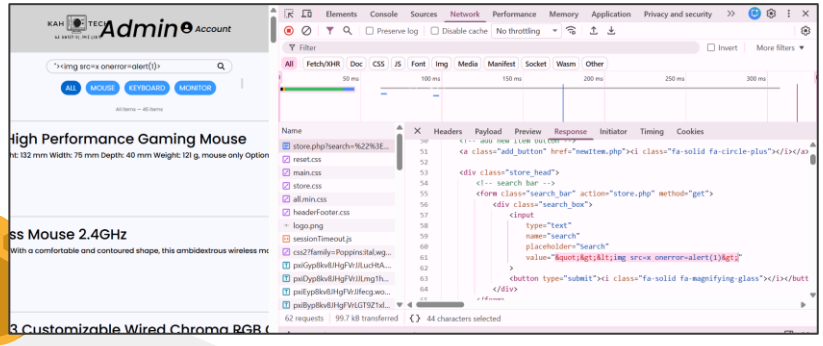
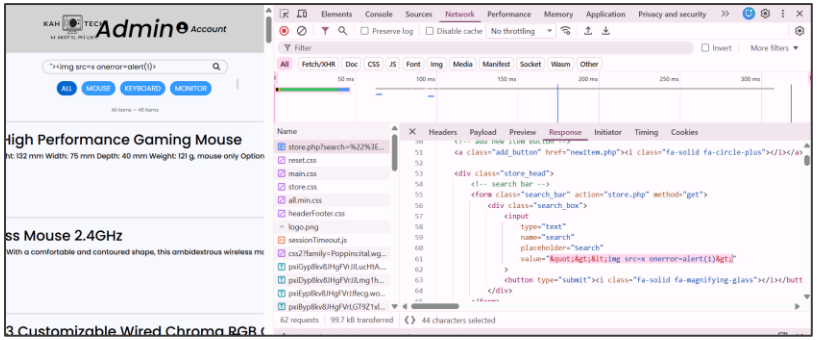


Image Error Handler



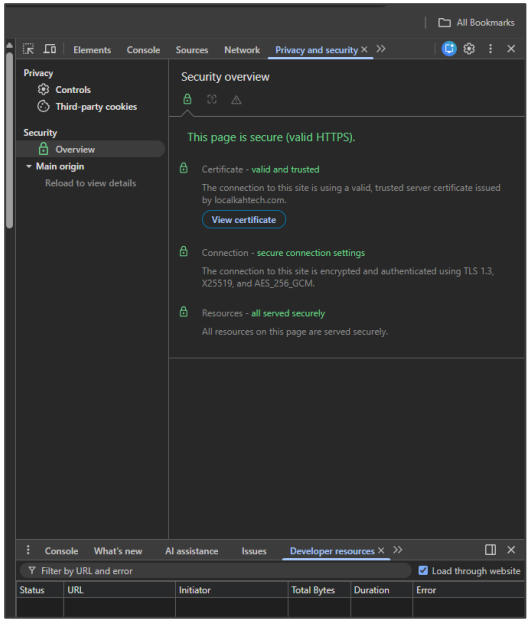
JavaScript URL

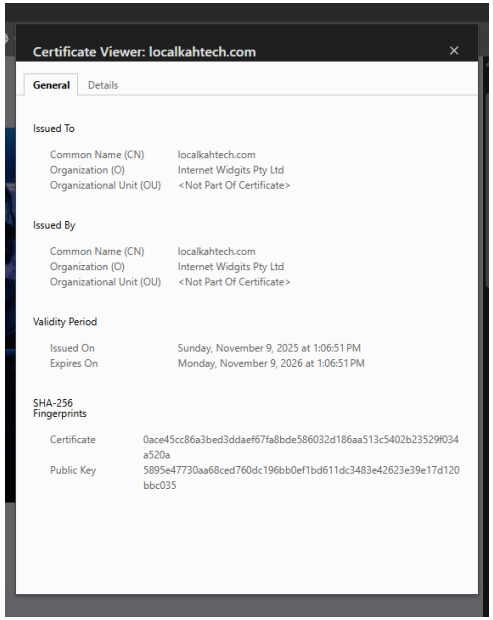


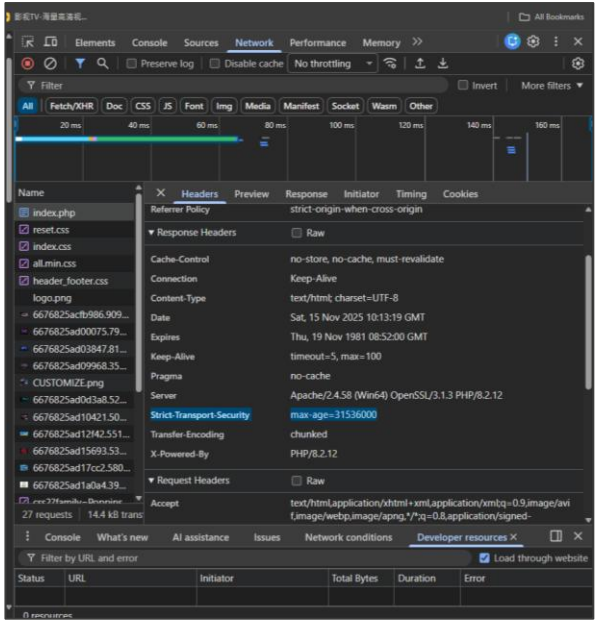
05.

SECURITY IMPLEMENTATION (11. DIGITAL CERTIFICATE DEPLOYMENT)

Digital Certificate Deployment





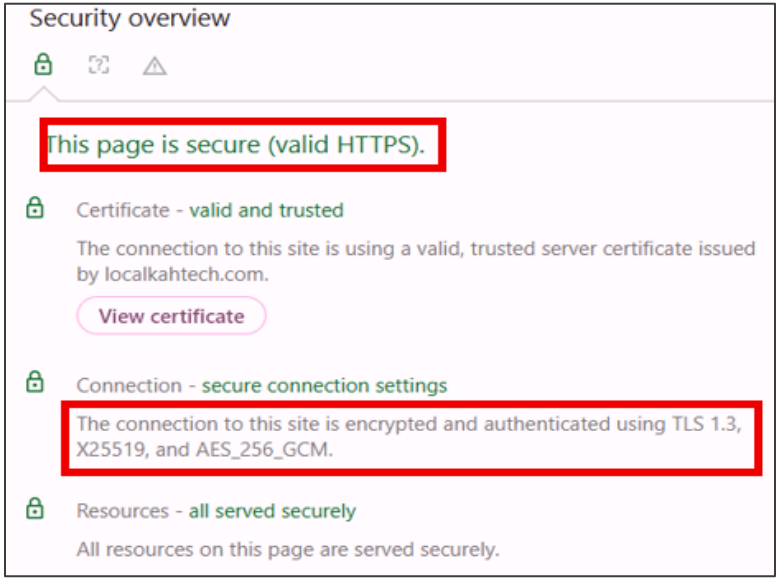


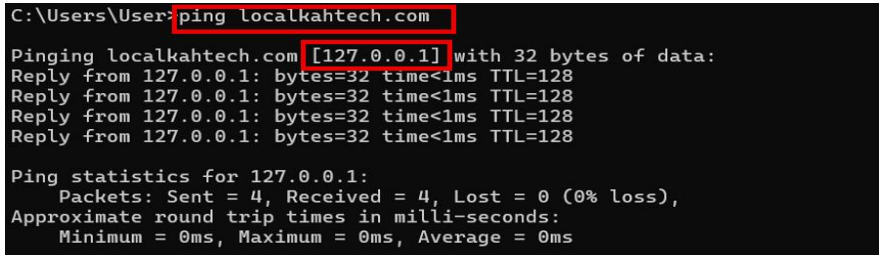
Provides authenticated and encrypted communication, significantly improving overall data confidentiality and integrity.

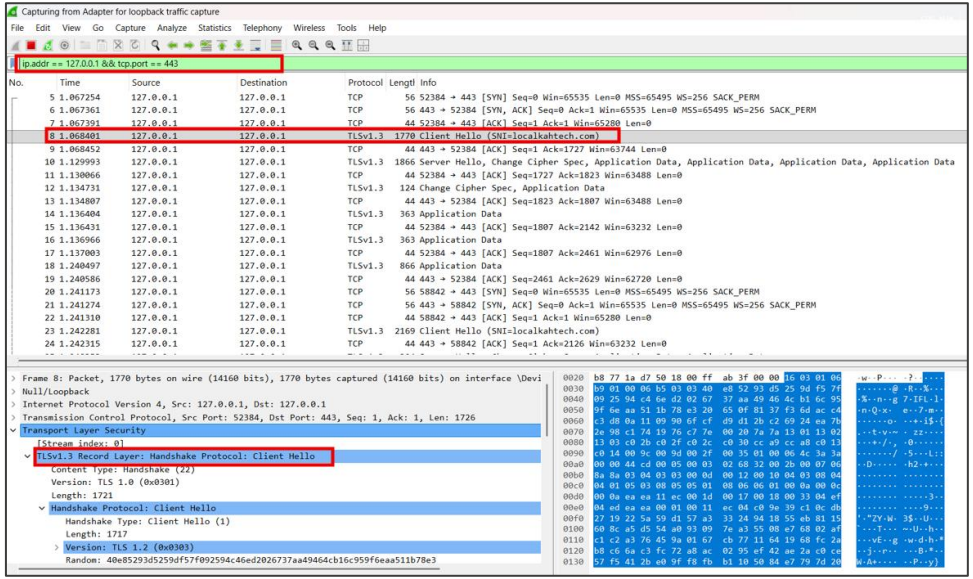
05.

SECURITY IMPLEMENTATION (12. CRYPTOGRAPHY IMPLEMENTATION)

Data-in-Transit (HTTPS/ TLS1.3)







TLS 1.3 setup ensures all traffic fully encrypted and protected in transit, keeping sensitive data secure without needing extra application-level encryption.

Data-at-Rest (AES-256-GCM)

Before & After:

#	Name	Type	Collation	Attributes	Null	Default
1	id	int(11)			No	None
2	user_name	varchar(25)	utf8mb4_general_ci		No	None
3	email	varchar(255)	utf8mb4_general_ci		No	None
4	secondary_email	varchar(255)	utf8mb4_general_ci		Yes	NULL
5	phone	varchar(15)	utf8mb4_general_ci		No	None
6	user_address	varchar(255)	utf8mb4_general_ci		No	None
7	pwd	varchar(255)	utf8mb4_general_ci		No	None
8	user_image	varchar(255)	utf8mb4_general_ci		Yes	no_profile_pic.png
9	wrong_pwd_count	int(11)			Yes	0
10	lock_until	datetime			Yes	NULL

#	Name	Type	Collation	Attributes	Null	Default
1	id	int(11)			No	None
2	user_name	varchar(255)	utf8mb4_general_ci		No	None
3	email	varchar(255)	utf8mb4_general_ci		No	None
4	secondary_email	varchar(255)	utf8mb4_general_ci		Yes	NULL
5	ph					
6	use					
7	pw					
8	use					
9	wr					
10	loc					

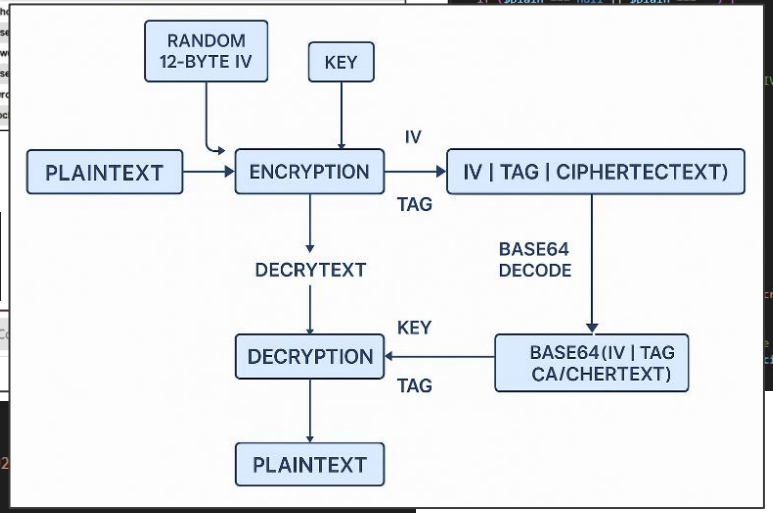
Key:

```php
<?php
echo bin2hex(string: random_bytes(length: 32));
?>
```

localhost/INT6005CEM_securiy_group_assignment_G23/Online-C

60d4d9476c34544becd52ff72cdfaec819202e5d715b97648f723c69cdeb32ec

```php
// 32-byte encryption key
// Generate in keygen.php
const ENC_KEY_HEX = '60d4d9476c34544becd52ff72cdfaec8192

// Convert hex to binary key
const ENC_METHOD = 'aes-256-gcm';

2 references
function get_enc_key(): string {
    return hex2bin(string: ENC_KEY_HEX);
}
```



Encryption:

```php
/**
 * Encrypt a single field using AES-256-GCM.
 * Returns base64(iv + tag + ciphertext).
 */
8 references
function encrypt_field(?string $plain): string {
    if ($plain === null || $plain === '') {
```

```
IV for GCM
```

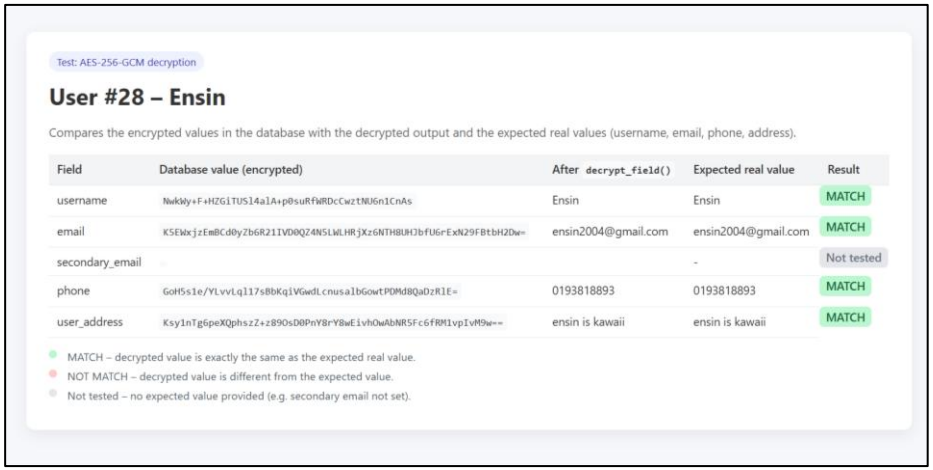
```
cryption failed');
```

```
e blob
cipher);
```

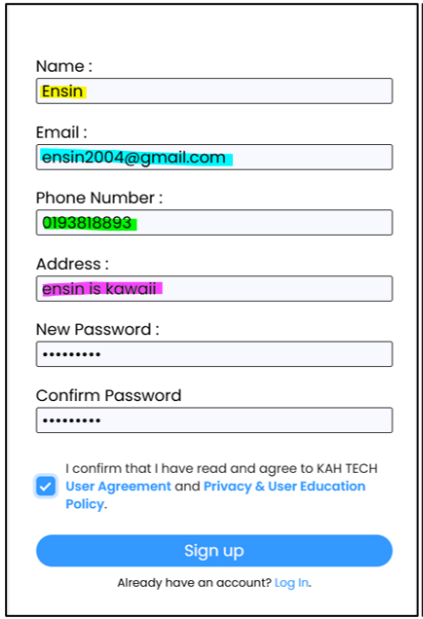
```php
/**
 * Decrypt a field stored as base64(iv + tag + ciphertext).
 */
27 references
function decrypt_field(?string $encoded): string {
    if ($encoded === null || $encoded === '') {
        return '';
    }

    $blob = base64_decode(string: $encoded, strict: true);
    if ($blob === false || strlen(string: $blob) < 12 + 16) {
        return '';
    }

    $iv   = substr(string: $blob, offset: 0, length: 12);
    $tag  = substr(string: $blob, offset: 12, length: 16);
    $cipher = substr(string: $blob, offset: 28);

    $plain = openssl_decrypt(
        data: $cipher,
        cipher_algo: ENC_METHOD,
        passphrase: get_enc_key(),
        options: OPENSSL_RAW_DATA,
        iv: $iv,
        tag: $tag
    );

    return $plain === false ? '' : $plain;
}
```

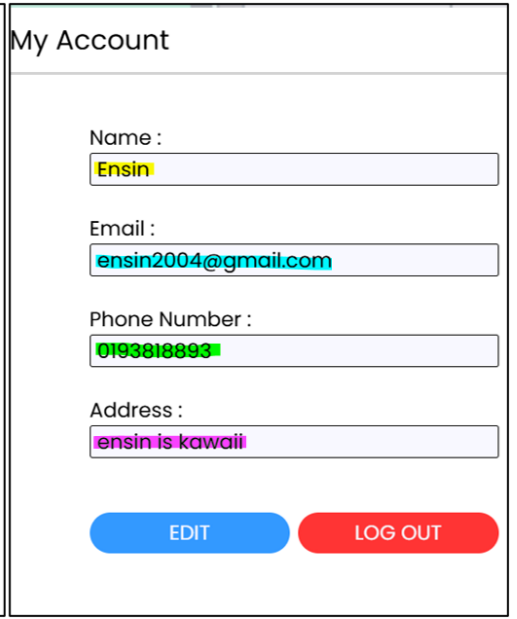
Data-at-Rest (AES-256-GCM)

Code-Based Verification Against Database Value



UI-Level Check





AES-256-GCM setup ensures sensitive database fields remain unreadable if leaked while still decrypting safely and accurately when the system needs them.

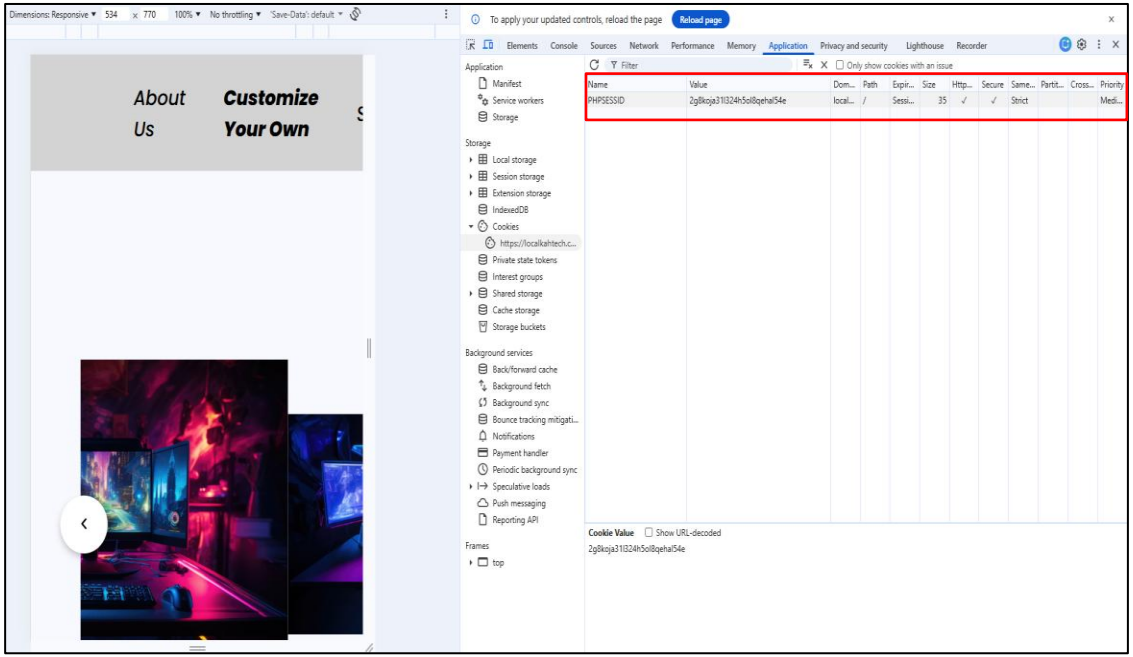
05.

SECURITY IMPLEMENTATION (13. BROWSER COOKIE)

Browser Cookie Security

```php
session_set_cookie_params([
    'lifetime' => 0,        // expires when browser closes
    'path' => '/',
    'secure' => true,       // only over HTTPS
    'httponly' => true,     // JS cannot access it
    'samesite' => 'Strict' // strong CSRF protection
]);
```

Browser cookies are configured with strict security settings to protect from theft, sniffing and CSRF attacks



06.

DISCUSSION & SUMMARY

06.

DISCUSSION & SUMMARY (DISCUSION)

OWASP TOP 10 COVERAGE

ACHIEVED	PARTIALLY ACHIEVED	NOT APPLICABLE
A01 – Broken Access Control	A04 – Insecure Design	A10 – Server-Side Request Forgery (SSRF)
A02 – Cryptographic Failures	A06 – Vulnerable & Outdated Components	
A03 – Injection (SQLi)	A08 – Software & Data Integrity Failures	
A05 – Security Misconfiguration		
A07 – Identification & Authentication Failures		
A09 – Security Logging & Monitoring Failures		

REMAINING VULNERABILITIES AFTER CONTROL

A01

Broken Access Control

- New pages/APIs may skip role checks
- Risk of exposing sensitive IDs in URLs
- **Fix:** Follow consistent access control rules

A02

Cryptographic Failures

- AES/TLS depends on proper key storage
- Require encryption and strict access to backups
- **Fix:** Improve key handling & backup encryption

A03

Injection

- Legacy code may use unsafe SQL
- **Fix:** Enforce prepared statements in code reviews

A04

Insecure Design

- Controls exist but no formal threat modelling
- **Fix:** Use simple design/threat checklist

REMAINING VULNERABILITIES AFTER CONTROL

A05

Security Misconfiguration

- Configurations may drift during deployment
- New servers may miss error pages or HSTS
- **Fix:** Use deployment checklists to prevent drift

A06

Vulnerable & Outdated Components

- Libraries may become outdated silently
- No automated dependency monitoring
- **Fix:** Add automated scanning and patch cycles

A07

Identification & Authentication Failures

- Strong passwords + rate limiting exist
- Still vulnerable to credential stuffing and no MFA for admins
- **Fix:** Enforce MFA for admins; offer MFA for users

A08

Software & Data Integrity Failures

- No code-signing or pipeline integrity checks
- **Fix:** Harden DevOps pipeline and use signed releases

REMAINING VULNERABILITIES AFTER CONTROL

A09

Security Logging & Monitoring Failures

- Logs not analysed centrally
- Suspicious activity may be detected late
- **Fix:** Centralize logs / SIEM

A10

SSRF

- Not currently applicable
- Future features (webhooks, URL fetchers) could introduce SSRF
- **Fix:** Apply SSRF-safe design for URL-fetching features

FUTURE ENHANCEMENT

Full MFA for Admins

- Enable MFA for all admin login
- Use TOTP + controlled recovery
- Reduces password compromise

Automated Dependency Scanning

- Track all libraries/component
- Use Composer/NPM/Git scans
- Catch CVEs before deployment

Secure SDLC & Checklists

- Use security checklists in reviews
- Quick threat-modelling for major changes
- Prevents new vulnerabilities

1 — 2 — 3 — 4 — 5

CSP & Browser Hardening

- Add CSP to block untrusted scripts
- Set security headers (nosniff, DENY, referrer policy)
- Mitigates XSS + clickjacking

Centralized Logging & Alerts

- Send app + server logs to SIEM/ELK
- Dashboards + alerts for anomalies
- Faster threat detection

06.

DISCUSSION & SUMMARY (SUMMARY)

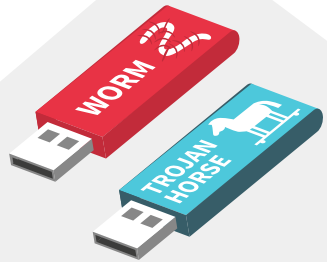
SUMMARY

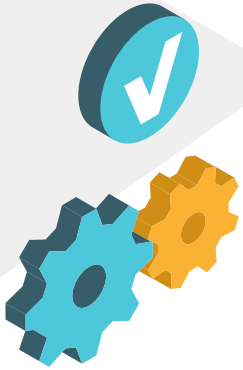
CONCLUSION

- System now addresses most OWASP Top 10 risks
- Major OWASP risks addressed (SQLi, XSS, weak sessions, no TLS, plaintext passwords)
- Implemented Argon2id, TLS 1.3, CSRF, secure cookies, logging & safer queries
- Overall system now far more secure and closer to production-ready

LEARNING OUTCOME ACHIEVED

- Hands-on experience with SQLi, XSS, session abuse & crypto issues
- Implemented real defenses: Argon2id, AES-GCM, TLS, CSRF, secure sessions
- Improved risk prioritization (Likelihood × Severity)
- Gained skills with Burp, Wireshark & dev tools





THANK YOU