Lovejoys Antique Web App - Report - Introduction to Computer Security - G6077

Cand No.: 249763

1) App URL: https://lovejoysantique249763.000webhostapp.com/
2) Code File Location:249763_CompSecCW
3) Panopto Recording Link: https://sussex.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=fcc3b0c3-fb54-4c2c-9bbf-b0d7013510c2
4) Testing User details:
    a) Admin Account : Email - admin@gmail.com , Password - Admin123%
    b) User Account : Email - user@gmail.com , Password - User123%

Task 0 - Self Reflection

Excellent (10-9 marks) Student must have gone beyond	Good (8-6 marks)	Average (5-3 marks)	Poor (2-0 marks)	Criteria
Policy has no flaw, and its implementation is excellent. Various mechanisms implemented to ensure password policy is secure.	Policy has no flaws, but implementation of policy is simple.	Password policy has very few flaws. However, different sections of policy are implemented and working.	Policy has many flaws for example password is not encrypted, and no salt applied. Password forgot policy has security flaws.	Password policy 10marks Password entropy, encrypted storage, security questions and recovery of password
Several countermeasures are implemented, and the quality of countermeasures are excellent.	Countermeasures are implemented in all the pages however quality of implementation is simple.	Implemented countermeasures only in some parts of the application.	Very little effort to implement countermeasures to avoid these vulnerabilities.	Vulnerabilities 10 marks SQL injection, XSS, CSRF, File Upload and any other obvious vulnerability.
All the requirements are implemented to authenticate users. Implementation quality is excellent.	All requirements are implemented to authenticate the user. However, quality of implementation is simple.	Only some obvious requirements are not implemented.	Lots of obvious authentication's requirements are not implemented.	Authentication 10 marks User identity management (registration and login etc), Email verification for registration, 2 factor authentications (PIN and or email).
Excellent implementation of countermeasures against these attacks.	No flaws in countermeasures however quality of implementation is simple.	Some flaws in countermeasures	Very little effort against these attacks.	Obfuscation/Common attacks 10 marks Brute force attack – Number of attempts Botnet attack – Captcha -------- Dictionary attack/Rainbow table attack
Claimed features are complex. The quality of achievement is excellent. No holes in the web application.	Claimed features are complex however quality of achievement/implementation could have been better. Very few flaws in the security of the application	Claimed features are somewhat complex and implementation could have been better. Some flaws in the security of the application	Minimal effort to implement some obvious security features like storing confidential information.	Deeper understanding, 10 marks Carry out your investigation and implement more security features to ensure that there no gaps in your application.

5 marks	5 marks	5 marks	5 marks	5 marks	10 marks	
List evaluation-Task6	Request evaluation – task 5	Request evaluation – task 4	Forgot password-Task3	Login-Task2	User registration/Database-Task1	Features of webs application
Completed	Completed	Completed	Completed	Completed	Completed	35/35

Up to 5 marks	0 marks	
Fully completed	Marking not completed	Self-reflection

Task 1 - User Registration

Registration feature code screenshots:

`registerForm.php`: Users are first given a form so they can register

```
<div class="registration-form">
    <form action="registerScript.php" method="POST">
        <h1>Register Your Details</h1>
        <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
        <input id="txtForename" name="txtForename" type="text" class="form-control" placeholder="Forename">
        <input id="txtSurname" name="txtSurname" type="text" class="form-control" placeholder="Surname">
        <input id="txtEmail1" name="txtEmail1" type="email" class="form-control" placeholder="Email Address">
        <input id="txtEmail2" name="txtEmail2" type="email" class="form-control" placeholder="Confirm Email Address">
        <input id="txtPassword1" name="txtPassword1" type="password" class="form-control" placeholder="Password">
        <input id="txtPassword2" name="txtPassword2" type="password" class="form-control" placeholder="Confirm Password">
        <input id="txtPhone" name="txtPhone" type="text" class="form-control" placeholder="Contact Number">
        <div class="security-question-group">
            <label for="security_question">Security Question:</label>
            <select name="security_question" id="security_question" class="form-control">
                <option value="Your first pet's name?">Your first pet's name?</option>
                <option value="Where was your first job?">Where was your first job?</option>
                <option value="Your favourite game?">Your favourite game?</option>
            </select>
            <label for="security_answer">Answer:</label>
            <input type="text" name="security_answer" id="security_answer" class="form-control" placeholder="Your Answer" required>
        </div>
        <div class="checkbox-group">
            <label class="custom-checkbox">
                Enable Two-Factor Authentication (Required)
                <input type="checkbox" name="enable_2fa" value="yes"required>
                <span class="checkmark"></span>
            </label>
        </div>
        <div class="g-recaptcha" data-sitekey="6LcLPi4pAAAAAHpkbr5ERkj_f8ISKjOuJvOC61Kq"></div>
        <input type="submit" value="Register" class="btn btn-primary">
    </form>
    <a href="index.php" class="login-link">Already have an account? Log in</a>
</div>
```

The user is then asked to set up Two-Factor Authentication, using Google's Authenticator App.

```
<h1>Lovejoy's Antique Store - 249763</h1>
<div class="setup-form">
    <h2>Setup Two-Factor Authentication (Do not leave this page before successfully setting up or you will not be able to log on)</h2>
    <p class="info-text">Step 1: Go to your app store and download the Google Authenticator App.</p>
    <p class="info-text">Step 2: Open your Google Authenticator App and tap the plus button on the bottom right side.</p>
    <p class="info-text">Step 3: Enter this setup key with your Google Authenticator app.</p>
    <div class="form-control">
        <strong>Your setup key:</strong> <?php echo htmlspecialchars($secretKey); ?>
    </div>
    <p class="info-text">Once you have set up your app, click the link below to verify your setup:</p>
    <a href="verify2FA.php" class="btn">Click here to verify setup</a>
</div>
```

After setting up their two-factor authentication, they will be asked to use the generated code, for verification.

```
<div class="verify-form">
    <h1>Lovejoy's Antique Store - 249763</h1>
    <h2>Complete Two-Factor Authentication Setup</h2>
    <form action="verify2FAScript.php" method="POST">
        <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
        <label for="verification_code">Enter the code from the app:</label>
        <input type="text" id="verification_code" name="verification_code" class="form-control" required>
        <input type="submit" value="Verify Code" class="btn">
    </form>
</div>
```

Database Table:

- **Database Server with two tables**

Table	Action	Rows	Type	Collation	Size	Overhead
evaluation_requests	Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_general_ci	32.0 KiB	-
users	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	16.0 KiB	-

- **Users table (Personal information blurred)**

ID	password	forename	surname	email	phone	role	secret_key	enabled2FA	security_question	security_answer	failed_attempts	lockout_time
1	$2y$10$/qZhOQPloWZFrg9NtL7DD.saZDuJwzITwSF[redacted]	Yash	Magane	yash.mag[redacted]@gmail.com	07467[redacted]	user	EDNHJVVD7[redacted]	no	Your favourite game?	$2y$10$OrAeNXBEYuxHQDIC50BSXe15k0CD7sGcc1a[redacted]	1	2023-12-13 19:14:11
2	$2y$10$SZhWT90gSXgUDopLsLDZ2.M8sV02P.nLnJ[redacted]	Yash	Magane	yash.ma[redacted]@gmail.com	07467[redacted]	admin	T2E6U3XP7[redacted]	no	Your favourite game?	$2y$10$/0BwHXWAA7fGul.4dY85.O38VWoVM3SEad[redacted]	0	NULL
3	$2y$10$AhQAHdq/rXbLwUkim/T2ZuZJJJ94eBkNnfcX[redacted]	Yash	Magane	yashma[redacted]gmail.com	074674[redacted]	user	V4JHIHGAQX[redacted]	no	Your first pet's name?	$2y$10$Li82qkWK54wV8hOGYxl1ced/pNtYP5k9.uS[redacted]	0	NULL
4	$2y$10$0/2T2dZYlpNcbzXV0Bx35.PZ4ddPAn1a/Z21[redacted]	Krisha	Kamlesh	krishaka[redacted]@gmail.com	07533[redacted]	user	6ICTM2WS7A[redacted]	no	Your favourite game?	$2y$10$cJPEwEijHRSFwyr2FZHVEeaPxETr/vpdVWH[redacted]	0	NULL
11	$2y$10$FaeVtVjcikSJLf7M1gtWKu81fGNOvqg5Mpx8[redacted]	Yash	Magane	yashi[redacted]@gmail.com	074674[redacted]	user	BQYMLOHMU[redacted]	no	Your favourite game?	$2y$10$WXIdGZIiq.hcQJSpdSWXtuhKBcvSujQJ6L0[redacted]	0	NULL
12	$2y$10$AZiaK3iTDzVCzRG27xvAP.rP/K8DBccNEdO[redacted]	Yash	Magane	yash[redacted]@gmail.com	075441[redacted]	user	J3JNZID52JH[redacted]	no	Your favourite game?	$2y$10$K2Ck7nSpGvIA/GCukTwadujBMT/SehxsLv[redacted]	6	2023-12-13 19:14:41
13	$2y$10$V91hopNuYf.iw5Jv.4KdA.4ddBeYpGcl.Fo[redacted]	Yash	Magane	ragini.p[redacted]@gmail.com	07544184[redacted]	user	FAFCNXAAUO[redacted]	no	Your favourite game?	$2y$10$7jx.6xXsZfzLD6y2/boSvuIDMdkIHt4aN[redacted]	0	NULL

- **Users table structure**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**ID**	int(11)			No	*None*		AUTO_INCREMENT
2	**password**	varchar(255)	utf8mb4_general_ci		No	*None*		
3	**forename**	varchar(255)	utf8mb4_general_ci		No	*None*		
4	**surname**	varchar(255)	utf8mb4_general_ci		No	*None*		
5	**email**	varchar(255)	utf8mb4_general_ci		No	*None*		
6	**phone**	varchar(15)	utf8mb4_general_ci		No	*None*		
7	**role**	enum('admin', 'user')	utf8mb4_general_ci		No	user		
8	**secret_key**	text	utf8mb4_general_ci		No	*None*		
9	**enabled2FA**	enum('yes', 'no')	utf8mb4_general_ci		No	no		
10	**security_question**	varchar(255)	utf8mb4_general_ci		Yes	*NULL*		
11	**security_answer**	varchar(255)	utf8mb4_general_ci		Yes	*NULL*		
12	**failed_attempts**	int(11)			No	0		
13	**lockout_time**	timestamp			Yes	*NULL*		

- **Evaluation Requests table**

id	user_id	details	contact_method	photo_path	created_at
5	2	Can I get a quote on this old vase please	email	uploads/1702345079_oldvase.jpg	2023-12-12 01:37:59
6	2	I have had this old player for a long time, Can yo...	phone	uploads/1702345183_veryOldObject.jpg	2023-12-12 01:39:43
7	2	My son got me this kettle many years back, I have ...	phone	uploads/1702345308_Kettle.jpg	2023-12-12 01:41:48
8	3	Can i get price check on this	email	uploads/1702346088_inbound7598637222892914851.jpg	2023-12-12 01:54:48
9	1	This is my old item, can you check it out and give...	phone	uploads/1702431380_Cup.jpg	2023-12-13 01:36:20
10	1	I have this wooden box, which has been with me for...	email	uploads/1702431576_Wooden Box.jpg	2023-12-13 01:39:36
11	1	Canfwniofw	email	uploads/1702490145_Full time post -1.png	2023-12-13 17:55:45
12	1	ifwniogw	email	uploads/1702491500_Wooden Box.jpg	2023-12-13 18:18:20
13	1	jdiwjfiojfw	email	uploads/1702493381_Cup.jpg	2023-12-13 18:49:41

- **Evaluation Requests table structure**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**id**	int(11)			No	*None*		AUTO_INCREMENT
2	**user_id**	int(11)			No	*None*		
3	**details**	text	utf8mb4_general_ci		No	*None*		
4	**contact_method**	enum('phone', 'email')	utf8mb4_general_ci		No	*None*		
5	**photo_path**	varchar(255)	utf8mb4_general_ci		Yes	*NULL*		
6	**created_at**	timestamp			No	current_timestamp()		

Reasons for Security:

- CSRF Protection: CSRF token generated and checked during form submission on all registration pages, including the setup2FA.php and verify2FA.php files.

```
<input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
```

```
if (!isset($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
```

- Password Hashing: function to create a hashed version of the user's password. '**PASSWORD_DEFAULT**', which is close to immune to brute force attacks as it is bcrypt.

```
// Hashing the password to store password hashed
$hashed_password = password_hash($password1, PASSWORD_DEFAULT);
```

- Furthermore, this function also creates a salt for each password, meaning if users have the same password it would still have a different hash, this is great against Rainbow Table attacks.
- Parameterised Query/ Prepared Statements: Preventing user's input from directly inserting in the SQL statement, to prevent SQL injection attacks.

```
// Insert into database including the salt and hashed password
$sql = "INSERT INTO users (password, forename, surname, email, phone, secret_key, security_question, security_answer) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
$stmt = $pdo->prepare($sql);
if ($stmt->execute([$hashedPassword, $forename, $surname, $email1, $phone, $secret, $securityQuestion, $securityAnswer])) {
```

- PDO Exceptions: PDO throws exceptions if an error occurs on the database, throwing exceptions ensures that no sensitive information is shown to the user.

```
catch (\PDOException $e) {
```

- Input Validation: Sanitising user input with '**HTMLSPECIALCHARS**' to mitigate XSS attacks.

```
$forename = htmlspecialchars($_POST['txtForename']);
$surname = htmlspecialchars($_POST['txtSurname']);
$email1 = htmlspecialchars($_POST['txtEmail1']);
$password1 = htmlspecialchars($_POST['txtPassword1']);
$password2 = htmlspecialchars($_POST['txtPassword2']);
$phone = htmlspecialchars($_POST['txtPhone']);
```

- PDO Connection: connection to database using PDO for secure connections.

```
try {
    $pdo = new PDO($dsn, $mysql_user, $mysql_password, $options);
} catch (\PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
```

- Two-Factor Authentication: Using Google's Authenticator App to verify the account.

```
if ($google2fa->verifyKey($secretKey, $verificationCode)) {
    header('Location: dashboard.php');
    exit;
} else {
```

- To prevent Botnet attacks and brute force attacks I have a strong password policy, which I will go into detail about in task 3.

```
if ($passwordStrengthError != '') {
    // Store the error message in the session
    $_SESSION['registration_errors'] = $passwordStrengthError;
```

- Security Questions:

```
<option value="Your first pet's name?">Your first pet's name?</option>
<option value="Where was your first job?">Where was your first job?</option>
<option value="Your favourite game?">Your favourite game?</option>
```

- Security Question: Answers to the security questions are hashed. This will be useful in task 3.

```
$securityQuestion = $_POST['security_question'];
$securityAnswer = password_hash($_POST['security_answer'], PASSWORD_DEFAULT);
```

- Captcha Version 2: I have used a captcha widget to prevent fake/bot accounts from being registered on my website.

```
        <div class="g-recaptcha" data-sitekey="6LceWS0pAAAAAJH4rwLi1a2OLJLxrwzNBdMy1z4K"></div>
```

Task 2 - Develop Secure Login Feature

Login feature code screenshots

index.php

```
<div class="login-form">
    <form action='loginScript.php' method='POST'>
        <h2>Login to Your Account</h2>
        <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
        <input name='txtEmail' type='text' class='form-control' placeholder='Email'>
        <input name='txtPassword' type='password' class='form-control' placeholder='Password'>
        <input name="2fa_code" type="text" class="form-control" placeholder="Two-Factor Authentication Code" required>
        <div class="g-recaptcha" data-sitekey="6LcLPi4pAAAAAHpkbr5ERkj_f8ISKjOuJvOC61Kq"></div>
        <input type='submit' value='Login' class='btn'>
        <a href='registerForm.php' class='register-link'>Not Registered Yet?</a>
        <a href='passRecovery.php' class='password-link'>Forgot Password?</a>
    </form>
</div>
```

Reasons for Security:

- CSRF Protection: CSRF token generated and checked during form submission
-

```
<input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
```

```
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    $_SESSION['error_message'] = 'CSRF token validation failed';
    header('Location: index.php');
    exit;
}
```

- Captcha Version 2: I have used a Captcha widget to prevent fake/bot accounts from logging on to my website.

```
$recaptchaResponse = $_POST['g-recaptcha-response'];
$recaptcha = file_get_contents('https://www.google.com/recaptcha/api/siteverify?secret=' . urlencode($recaptchaSecretKey)
$recaptcha = json_decode($recaptcha);
```

- PDO: Using a secure database connection such as PDO

```
try {
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
    throw new \PDOException($e->getMessage(), (int)$e->getCode());
}
```

- Combining PDO with Prepared Statements to prevent SQL Injections

```
$sql = "SELECT id, password, secret_key, role, failed_attempts, lockout_time FROM users WHERE email = :email";
$stmt = $pdo->prepare($sql);
$stmt->execute(['email' => $email]);
$user = $stmt->fetch();
```

- Using '**PASSWORD_VERIFY**', it is a secure function as it checks hashed passwords without exposing plain text passwords.

```
if (password_verify($password, $user['password'])) {
```

- Two-Factor Authentication: An additional layer of security is provided through 2FA, requiring users to verify their identity using a second method (like a time-based one-time password) besides just the password.

```
$google2fa = new Google2FA();
$isValid2FA = $google2fa->verifyKey($user['secret_key'], $_POST['2fa_code']);
```

- Account Lockout System: I created a way to only allow a user 5 attempts at failing to log on or the account locks for 30 minutes, having this prevents brute force attacks.

```
$remainingAttempts = 5 - $user['failed_attempts'] - 1; // Subtract 1 for the current attempt
$_SESSION['error_message'] = "Incorrect password. You have $remainingAttempts remaining attempts.";
header('Location: index.php');
exit;
```

- Error Handling: Error handling for both the database connection and the login process, I provide a clear response to the user to prevent script errors from exposing sensitive information.

```
$_SESSION['error_message'] = "Incorrect email or password.";
header('Location: index.php');
exit;
```

- Role-Based Dashboards: When the user logs on they are redirected to an admin or user dashboard, which allows restricted access to the list of evaluations page.

```
// Redirect based on user role
if ($_SESSION['role'] === 'admin') {
    header('Location: admin_dashboard.php');
    exit;
} else {
    header('Location: dashboard.php');
    exit;
}
```

- Input Validation: Sanitising user input with '**HTMLSPECIALCHARS**' to mitigate XSS attacks.

```
$email = htmlspecialchars($_POST['txtEmail']);
$password = htmlspecialchars($_POST['txtPassword']);
```

Task 3 - Implement Password Strength and Password Recovery

Password Policy: List of Elements

- Password Strength: Preventing brute force

```php
$passwordStrengthError = '';

if (strlen($password1) < 8) {
    $passwordStrengthError .= 'Password must be at least 8 characters long. ';
}

if (!preg_match('/[A-Z]/', $password1)) {
    $passwordStrengthError .= 'Password must include at least one uppercase letter. ';
}

if (!preg_match('/[a-z]/', $password1)) {
    $passwordStrengthError .= 'Password must include at least one lowercase letter. ';
}

if (!preg_match('/\d/', $password1)) {
    $passwordStrengthError .= 'Password must include at least one number. ';
}

if (!preg_match('/\W/', $password1)) {
    $passwordStrengthError .= 'Password must include at least one special character. ';
}

if ($passwordStrengthError != '') {
    // Store the error message in the session
    $_SESSION['registration_errors'] = $passwordStrengthError;
```

- Password and Security Question Answer Storage: Hashed using '**`PASSWORD_HASH`**'

```php
$hashed_password = password_hash($password1, PASSWORD_DEFAULT);
}

$securityQuestion = $_POST['security_question'];
$securityAnswer = password_hash($_POST['security_answer'], PASSWORD_DEFAULT);
```

- '**`PASSWORD_HASH`**' function automatically creates a salt, for users with the same password to prevent Rainbow Table attacks.
- Password Recovery: (**`passRecovery.php`**), includes CSRF Token hidden. User is asked for their email, security question they used, the answer to the question, 2FA code and tick the captcha.

```html
<div class="forgot-password-form">
    <h1>Forgot Password</h1>
    <form action="forgotPasswordScript.php" method="POST">
        <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
        <input type="email" name="email" class="form-control" placeholder="Enter your email" required>
        <select name="security_question" class="form-control">
            <option value="Your first pet's name?">Your first pet's name?</option>
                <option value="Where was your first job?">Where was your first job?</option>
                <option value="Your favourite game?">Your favourite game?</option>
        </select>
        <input type="text" name="security_answer" class="form-control" placeholder="Answer for security question" required>
        <input type="text" name="2fa_code" class="form-control" placeholder="2FA Code" required>
        <div class="g-recaptcha" data-sitekey="6LcLPi4pAAAAAHpkbr5ERkj_f8ISKjOuJvOC61Kq"></div>
        <input type="submit" value="Reset Password" class="btn">
    </form>
    <a href="index.php" class="login-link">Back to Login</a>
```

- The script then checks the CSRF token and verifies the Recaptcha

```php
if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die('CSRF token validation failed');
}
```

```php
$recaptchaResponse = $_POST['g-recaptcha-response'];
$recaptcha = file_get_contents('https://www.google.com/recaptcha/api/siteverify?secret='
. urlencode($recaptchaSecretKey) . '&response=' . urlencode($recaptchaResponse));
$recaptcha = json_decode($recaptcha);
```

- Retrieves user data using email provided, database connection using PDO

```php
try {
    $pdo = new PDO("mysql:host=$host;dbname=$db", $user, $pass);

    // Retrieve user data
    $stmt = $pdo->prepare("SELECT id, email, secret_key, security_question, security_answer FROM users WHERE email = ?");
    $stmt->execute([$_POST['email']]);
    $user = $stmt->fetch();
```

- Using '**PASSWORD_VERIFY**', it is a secure function as it checks hashed security question answers without exposing plain text answers.

```php
password_verify($_POST['security_answer'], $user['security_answer'])) {
```

- Redirection with Unique Token: Upon successful verification, the script generates a unique token and stores it in the session. The user is then redirected to a password reset page with this token, which helps to ensure that the password reset process is initiated by the authenticated user.

```php
if ($google2fa->verifyKey($user['secret_key'], $_POST['2fa_code'])) {
    // Redirect to reset password page with a unique token
    $_SESSION['reset_token'] = bin2hex(random_bytes(32));
    $_SESSION['reset_user_id'] = $user['id']; // Store user ID for password reset
    header('Location: resetPassword.php?token=' . $_SESSION['reset_token']);
```

- User is then redirected to the reset password form, which checks the CSRF and Unique Tokens.

```html
<div class="reset-password-form">
    <h1>Reset Your Password</h1>
    <form action="resetPasswordScript.php" method="POST">
        <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
        <input type="hidden" name="reset_token" value="<?php echo htmlspecialchars($_GET['token']); ?>">
        <input type="password" name="new_password" class="form-control" placeholder="New Password" required>
        <input type="password" name="confirm_new_password" class="form-control" placeholder="Confirm New Password" required>
        <button type="submit" class="btn">Reset Password</button>
    </form>
</div>
```

- The script for the form above also verifies both tokens and uses PDO to connect to the database
- Password matching with sanitisation

```php
if (htmlspecialcahrs($_POST['new_password']) !== htmlspecialcahrs($_POST['confirm_new_password'])) {
    die('Passwords do not match');
}
```

- Storing password hashed:

```php
try {
    $pdo = new PDO("mysql:host=$host;dbname=$db", $user, $pass);

    // Update user password
    $newPasswordHash = password_hash($_POST['new_password'], PASSWORD_DEFAULT);
    $stmt = $pdo->prepare("UPDATE users SET password = ? WHERE id = ?");
    $stmt->execute([$newPasswordHash, $_SESSION['reset_user_id']]);
```

Task 4 - Implement an 'Evaluation Request' web page

Evaluation Request feature code screenshots

This page is only accessible to users with a user role. When a user registers/logs on they will be redirected to the user dashboard: **`dashboard.php`**

```html
<div class="dashboard">
    <h1>Welcome!</h1>
    <a href="req_evaForm.php" class="dashboard-link">Request Evaluation</a>
    <a href="logout.php" class="dashboard-link">Logout</a>
</div>
```

Then they will be able to access the form below (**`req_evaForm.php`**)

```html
<div class="evaluation-form">
    <h1>Item Evaluation Request</h1>
    <form action="req_eva.php" method="POST" enctype="multipart/form-data">
        <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
        <label for="details" style="font-weight: bold; color: #8B4513;">Details of the Object:</label>
        <textarea id="details" name="details" class="form-control" placeholder="Details of the Object" required></textarea>
        <label for="contactMethod" style="font-weight: bold; color: #8B4513;">Preferred Contact Method:</label>
        <select id="contactMethod" name="contactMethod" class="form-control">
            <option value="phone">Phone</option>
            <option value="email">Email</option>
        </select>
```

Reasons for Security

- CSRF Token was hidden in the form and then validated in the script (**`req_eva.php`**),

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== $_SESSION['csrf_token']) {
        // Handle the error - CSRF token does not match or not set
        die('CSRF token validation failed.');
    }
}
```

- Session-based User Authentication: First there is a check if a user is logged in (`$_SESSION['loggedin']`) before processing the form, ensuring that only authenticated users can submit evaluation requests. Also redirection to the login page.

```
if (!isset($_SESSION['loggedin']) || $_SESSION['loggedin'] !== true) {
    header('Location: index.php');
    exit;
}
```

- Input Sanitisation: Employs `filter_input` with appropriate filters (`FILTER_SANITIZE_FULL_SPECIAL_CHARS` and `FILTER_SANITIZE_STRING`) to sanitize user input before processing, mitigating the risk of XSS

```
$details = filter_input(INPUT_POST, 'details', FILTER_SANITIZE_FULL_SPECIAL_CHARS);
$contactMethod = filter_input(INPUT_POST, 'contactMethod', FILTER_SANITIZE_STRING);
$userId = $_SESSION['user_id'];
```

- Secure Database Connection: Using PDO and `bindParam` for database operations, protecting against SQL injection attacks.

```
$sql = "INSERT INTO evaluation_requests (user_id, details, contact_method, photo_path)
    VALUES (:user_id, :details, :contact_method, :photo_path)";
$stmt = $pdo->prepare($sql);
$stmt->bindParam(':user_id', $userId, PDO::PARAM_INT);
$stmt->bindParam(':details', $details, PDO::PARAM_STR);
$stmt->bindParam(':contact_method', $contactMethod, PDO::PARAM_STR);
$stmt->bindParam(':photo_path', $photoPath, PDO::PARAM_STR);
```

Task 5 - Develop a feature that will allow customers to submit photographs

<u>Extension feature code screenshots</u>

Same as task 4 - This page is only accessible to users with a user role. When a user registers/logs on they will be redirected to the user dashboard:

```
<div class="dashboard">
    <h1>Welcome!</h1>
    <a href="req_evaForm.php" class="dashboard-link">Request Evaluation</a>
    <a href="logout.php" class="dashboard-link">Logout</a>
</div>
```

Then they will be able to access the form below (Full Evaluation Request Web page)

```
<div class="evaluation-form">
    <h1>Item Evaluation Request</h1>
    <form action="req_eva.php" method="POST" enctype="multipart/form-data">
        <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
        <label for="details" style="font-weight: bold; color: #8B4513;">Details of the Object:</label>
        <textarea id="details" name="details" class="form-control" placeholder="Details of the Object" required></textarea>
        <label for="contactMethod" style="font-weight: bold; color: #8B4513;">Preferred Contact Method:</label>
        <select id="contactMethod" name="contactMethod" class="form-control">
            <option value="phone">Phone</option>
            <option value="email">Email</option>
        </select>
        <label for="objectPhoto" style="font-weight: bold; color: #8B4513;">Image of your Object (jpg, png, jpeg, gif)</label>
        <input type="file" id="objectPhoto" name="objectPhoto" accept="image/*" class="form-control">

        <input type="submit" value="Submit Request" class="btn">
        <a href="dashboard.php" class="btn-back">Back to Dashboard</a>
    </form>
</div>
```

This is the full form for the Request Evaluation Page as it allows the user to upload images of their objects.

Reasons for Security:

- Handling File Upload: Using a unique file name for each upload to prevent file overwriting and potential code injection through file uploads.

```
if (isset($_FILES['objectPhoto']) && $_FILES['objectPhoto']['error'] == 0) {
    $targetDir = "uploads/"; // Directory where files will be stored
    // Creating a unique file name
    $fileName = time() . '_' . basename($_FILES["objectPhoto"]["name"]);
    $targetFilePath = $targetDir . $fileName;
    $fileType = strtolower(pathinfo($targetFilePath, PATHINFO_EXTENSION));
```

- Restricted file extensions: the allowed types are (jpg, png, jpeg, gif).

```
$allowTypes = array('jpg', 'png', 'jpeg', 'gif');
if (in_array($fileType, $allowTypes)) {
    // Upload file to the server
    if (move_uploaded_file($_FILES["objectPhoto"]["tmp_name"], $targetFilePath)) {
        // File upload success, path will be stored in the database
```

- File Naming Convention for Uploads: Generates a unique file name for each upload using the current timestamp, reducing the risk of file overwrites and potential conflicts.

```
    $fileName = time() . '_' . basename($_FILES["objectPhoto"]["name"]);
```

Task 6 – Request Listing Page

Request Listing Page feature code screenshots

This page has restricted access, only admin accounts can access the admin dashboard and see the listing page. You cannot register for an admin account for security reasons. You need access to the database to change account settings manually to be granted administrative privileges.

Admin users are first directed to their dashboard

```
<div class="dashboard">
    <h1>Welcome!</h1>
    <a href="list_eva_req.php" class="dashboard-link">List of Evaluation Requests</a>
    <br>
    <a href="logout.php" class="dashboard-link">Logout</a>
</div>
```

From there they can logout or view the List of Evaluation Requests.

```
<?php foreach ($requests as $request): ?>
<tr>
    <td><?php echo htmlspecialchars($request['id']); ?></td>
    <td><?php echo htmlspecialchars($request['user_id']); ?></td>
    <td><?php echo htmlspecialchars($request['details']); ?></td>
    <td><?php echo htmlspecialchars($request['contact_method']); ?></td>
    <td><?php
if (!empty($request['photo_path'])) {
    echo "<img src='/uploads/" . htmlspecialchars($request['photo_path'])
        . "' alt='Evaluation Photo' style='width:100px; height:auto;'>";
} else {
    echo "No photo";
}
?></td>
    <td><?php echo htmlspecialchars($request['created_at']); ?></td>
</tr>
<?php endforeach; ?>
```

Reasons for Security:

(**`list_eva_req.php`**)

- Database Connection: Using PDO to connect to the database to retrieve evaluation requests.

```
try {
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
```

- Redirect if not logged in or not an admin

```
if (!isset($_SESSION['loggedin']) || $_SESSION['loggedin'] !== true || $_SESSION['role'] !== 'admin') {
    header('Location: index.php');
    exit;
}
```

- Data Sanitisation when retrieving from database for XSS Attacks

```
<td><?php echo htmlspecialchars($request['id']); ?></td>
<td><?php echo htmlspecialchars($request['user_id']); ?></td>
<td><?php echo htmlspecialchars($request['details']); ?></td>
<td><?php echo htmlspecialchars($request['contact_method']); ?></td>
```

- Secure image display, '**HTMLSPECIALCHARS**' function sanitises the photo path to prevent XSS attacks

```
if (!empty($request['photo_path'])) {
    echo "<img src='" . htmlspecialchars($request['photo_path']) . "' alt='Evaluation Photo' style='width:100px; height:auto;'>";

} else {
    echo "No photo";
}
```

Deeper Understanding:

- For this page the <style> tag was isolated to reduce the risk of inline styling-related security issues, such as XSS attacks through style attributes

```
<head>
    <meta charset="UTF-8">
    <title>Admin Dashboard - Lovjoy's Antique Store</title>
    <style>
    body {
        font-family: 'Times New Roman', serif;
        background-color: #fdf6e3;
        color: #8B4513;
        padding: 20px;
        text-align: center;
    }
```

Task 7 – AWS Virtual Private Cloud settings screenshots