

Computer Security - Coursework

Syed F. Shah

231012

Words: 4028

University of Sussex - Informatics

### **Abstract**

The paper presents a website called Lovejoy Antique goods developed using PHP/HTML that allows users to register and log in to a platform for submitting evaluations of antique goods. The website features a user-friendly interface and a secure authentication system to protect user information. The registration process requires users to provide a unique username and password and their email address for verification. Once logged in, users can submit evaluations of antique items and admin accounts can view all total submissions. The website also includes a 2FA (2-factor authentication) and several security features to prevent any forms of SQL injection, XSS payloads/attacks etc. Overall, the website provides a convenient and secure platform for users to share their expertise and knowledge of antique items as required in the brief description. **All links to downloading and watching are at the very end, within the appendix on the last page on this report.**

*Keywords: secure, SQL injection, evaluation, XSS payload, 2FA*

<b>Abstract</b>	<b>2</b>
<b>Registry - Task 1</b>	<b>4</b>
Code Snippet	4
Security Features	6
Account Database	8
Summary of Securing the Registration	10
<b>Login - Task 2</b>	<b>11</b>
Code Snippet	11
Security Features	14
<b>Passwording - Task 3</b>	<b>17</b>
Code	17
Summary of Password Policy	21
Why is it Secure?	21
<b>Evaluation - Task 4</b>	<b>23</b>
Code Snippet	23
Security	24
<b>Photo Submission - Task 5</b>	<b>26</b>
Image File Upload Code	26
Security Features	27
<b>(Extension) Request Listing Page - Task 6</b>	<b>29</b>
Code Snippets	29
Security Features	31
<b>Self-Evaluation - Task 0</b>	<b>32</b>
<b>Appendix</b>	<b>34</b>
mainSystem.php	34
dbConfig.php	36
Link to Downloads & Viewing Testing	37

## Registry - Task 1

## Code Snippet

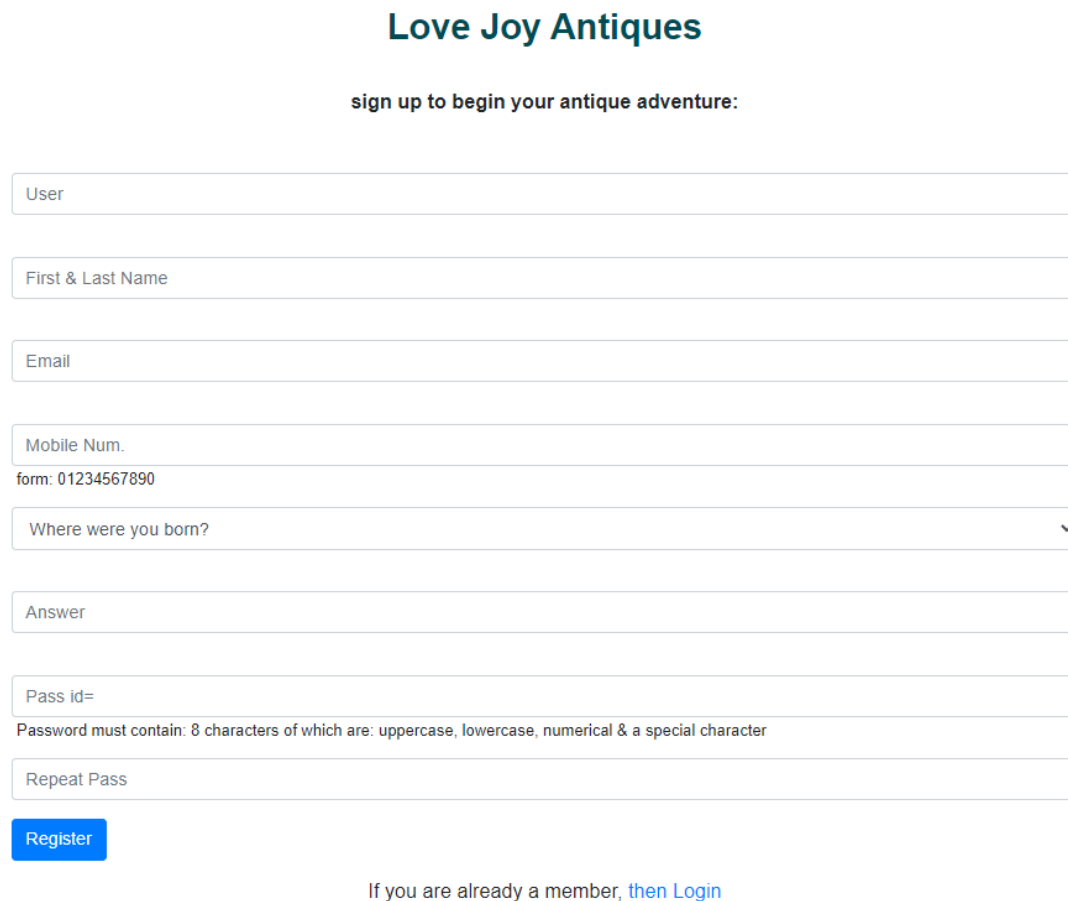
```

16 <h2><strong></p><p style="color:#004c54">Love Joy Antiques </strong><br></h2><br>
17 <p><strong>sign up to begin your antique adventure:</strong></p>
18 <form action="register.php" method="post" autocomplete="off">
19   <div class="form-group">
20     <label for="username"></label>
21   </div>
22   <!-- textbox for data entry -->
23   <div class="form-group">
24     <input type="text" name="username" minlength="3" placeholder="User" id="username" pattern="[A-Za-z0-9]+" class="form-control" required>
25     <label for="fullname"></label>
26   </div>
27   <!-- required 4 characters for fullname for valid input -->
28   <div class="form-group">
29     <input type="text" name="fullname" placeholder="First & Last Name" minlength="4" id="fullname" class="form-control" pattern="[A-Za-z -]+" required>
30     <label for="email"></label>
31   </div>
32   <!-- using email datatype to ensure @ is used correctly -->
33   <div class="form-group">
34     <input type="email" name="email" placeholder="Email" class="form-control" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$" required>
35     <label for="telephone"></label>
36   </div>
37   <!-- UK based telephone num system using regex -->
38   <div class="form-group">
39     <input type="tel" name="telephone" placeholder="Mobile Num." pattern="[0]{1}[0-9]{10}" class="form-control" required>
40     <label for="securityQuestion"></label>
41     <small>form: 01234567890</small>
42   </div>
43   <!-- security questions-->
44   <div class="form-group">
45     <select name="securityQuestion" class="form-control">
46       <option value="Where were you born?">Where were you born?</option>
47       <option value="What is your favourite video game?">What is your favourite video game?</option>
48       <option value="What is your first girlfriend/boyfriend called?">What is your first girlfriend/boyfriend called?</option>
49       <option value="What is your primary school called?">What is your primary school called?</option>
50       <option value="What is your favourite colour?">What is your favourite colour?</option>
51     </select>
52     <label for="securityAnswer"></label>
53   </div>
54   <!-- security questions-->
55   <div class="form-group">
56     <select name="securityQuestion" class="form-control">
57       <option value="Where were you born?">Where were you born?</option>
58       <option value="What is your favourite video game?">What is your favourite video game?</option>
59       <option value="What is your first girlfriend/boyfriend called?">What is your first girlfriend/boyfriend called?</option>
60       <option value="What is your primary school called?">What is your primary school called?</option>
61       <option value="What is your favourite colour?">What is your favourite colour?</option>
62     </select>
63     <label for="securityAnswer"></label>
64   </div>
65   <!-- allow for any type of answer for a security question as it is a personal question that allows for any answer as long as it makes sense to the user -->
66   <div class="form-group">
67     <input type="text" name="securityAnswer" placeholder="Answer" id="securityAnswer" pattern="[A-Za-z0-9 -]+" class="form-control" required>
68     <label for="password"></label>
69   </div>
70   <!-- password entry and confirmation of password. Few steps to ensure it is strong-->
71   <div class="form-group">
72     <input type="password" name="password" placeholder="Pass" id="password" class="form-control" minlength="8" required>
73     <label for="confirm_password"></label>
74     <small>Password must contain: 8 characters of which are: uppercase, lowercase, numerical & a special character</small>
75   </div>
76   <!-- retype the same password as before for confirmation-->
77   <div class="form-group">
78     <input type="password" name="confirm_password" placeholder="Repeat Pass" minlength="10" id="confirm_password" class="form-control" required>
79   </div>
80   <div class="form-group">
81     <input type="submit" class="btn btn-primary" value="Register">
82   </div>
83   <p>If you are already a member, <a href="loginAcc.php">then Login</a></p>

```

Figure 1 &amp; 2: Index.html Login

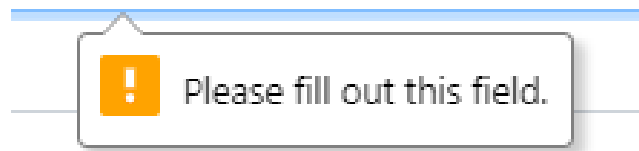
As seen by figures 1 & 2, it shows a html based index page designed to register the user's login. This result is as seen in figure 3.



The image shows a web registration form for 'Love Joy Antiques'. The form is titled 'Love Joy Antiques' in a teal font, followed by the instruction 'sign up to begin your antique adventure:'. The form contains several input fields: 'User', 'First & Last Name', 'Email', 'Mobile Num.' (with a placeholder 'form: 01234567890'), a dropdown menu for 'Where were you born?', 'Answer', 'Pass id=' (with a password requirement note: 'Password must contain: 8 characters of which are: uppercase, lowercase, numerical & a special character'), and 'Repeat Pass'. A blue 'Register' button is at the bottom left of the form. Below the button, there is a link: 'If you are already a member, [then Login](#)'.

*Figure 3: Index.html In demonstration*

Since this is just a simple html document supplemented with CSS stylesheets, there are no apparent security features present. However, in the instance that a user is to miss a certain part of information when typing in their details. They will be met with a warning telling them to either complete the missing information required e.g. missing user entry or password doesn't match the confirmation box. This is accomplished by using simple CSS which requires the user to have entered a piece of text into the box as seen in Figure 4.



*Figure 4: Validation in index.html*

### Security Features

In terms of security, there are a handful of features that were attempted to be implemented into the website as a means of preventing or reducing the effects like SQL Injection or XSS. SQL injection is a type of security vulnerability in which an attacker is able to execute malicious SQL code in a database. This can allow the attacker to gain access to sensitive information in the database, such as user passwords, or to manipulate data in the database. SQL injection is typically carried out by providing malicious input to an application that is then passed to the database, and it is a common method of attack used by hackers. An Example of this is below:

A screenshot of a web application titled "Vulnerability: SQL Injection". It shows a "User ID:" label above a text input field containing the malicious payload "a' OR ''=" and a "Submit" button. Below the input field, the application displays three rows of user data in red text, demonstrating the results of the SQL injection attack. The data rows are: 1) ID: a' OR ''=, First name: admin, Surname: admin; 2) ID: a' OR ''=, First name: Gordon, Surname: Brown; 3) ID: a' OR ''=, First name: Hack, Surname: Me.

Vulnerability: SQL Injection		
<b>User ID:</b>		
<input "="" type="text" value="a' OR ''="/>	<input type="button" value="Submit"/>	
ID: a' OR ''= First name: admin Surname: admin		
ID: a' OR ''= First name: Gordon Surname: Brown		
ID: a' OR ''= First name: Hack Surname: Me		

*Figure 5: SQL Injection Demo<sup>1</sup>*

<sup>1</sup> <https://pentestlab.blog/2012/09/18/sql-injection-exploitation-dvwa/>

In the same way, Cross-Site Scripting (XSS) is a type of web security vulnerability that allows attackers to inject malicious code into websites or web applications. This can allow the attacker to steal sensitive information, such as user login credentials, or even manipulate the website to perform actions on the attacker's behalf. XSS attacks can be divided into two main categories: non-persistent and persistent. Non-persistent XSS attacks occur when the malicious code is included in a request that is sent to the server, but is not permanently stored on the server. Persistent XSS attacks, on the other hand, involve injecting malicious code into the website or web application that is permanently stored on the server.

Both these methods can be prevented by implementing proper input validation and sanitization, using web application firewalls, and so. On the topic of my programmed solution. An example of prevention can be seen in figure 6

```
12 //PHP sanitizing. This is used to prevent the usage of special characters
13 //to inject into the text boxes.
14 $username = mysqli_real_escape_string($con, $_POST['username']);
15 $fullname = mysqli_real_escape_string($con, $_POST['fullname']);
16 $email = mysqli_real_escape_string($con, $_POST['email']);
17 $telephone = mysqli_real_escape_string($con, $_POST['telephone']);
18 $password = mysqli_real_escape_string($con, $_POST['password']);
19 $confirm_password = mysqli_real_escape_string($con, $_POST['confirm_password']);
20 $securityQuestion = mysqli_real_escape_string($con, $_POST['securityQuestion']);
21 $securityAnswer = mysqli_real_escape_string($con, $_POST['securityAnswer']);
22 //-----
```

*Figure 6: Sanitization in register.php*

In this figure, we notice that the use of sanitization has been used within the context of my registration check feature. The usage of “mysqli\_real\_escape\_string” is crucial as it is a function in the MySQLi extension in PHP that is used to create a legal SQL string that can be used in an SQL statement. This function is used to escape special characters in a string to prevent SQL injection attacks.

Furthermore, this can be seen in the context of HTML characters whereby “htmlspecialchars” can be used to convert predefined characters into HTML text to prevent XSS and is in the following figure:

```
//Same thing but for HTML characters
$username = htmlspecialchars($username);
$fullname = htmlspecialchars($fullname);
$email = htmlspecialchars($email);
$telephone = htmlspecialchars($telephone);
$password = htmlspecialchars($password);
$confirm_password = htmlspecialchars($confirm_password);
$securityQuestion = htmlspecialchars($securityQuestion);
$securityAnswer = htmlspecialchars($securityAnswer);
//-----
```

Figure 7: Sanitization of HTML in register.php

### Account Database

```
1 CREATE TABLE IF NOT EXISTS 'accounts'(
2     'id' int(11) NOT NULL AUTO_INCREMENT,
3     'username' varchar(255) NOT NULL,
4     'fullname' varchar(255) NOT NULL,
5     'password' varchar(255) NOT NULL,
6     'email' varchar(255) NOT NULL UNIQUE,
7     'telephone' varchar(255) NOT NULL,
8     'securityQuestion' varchar(100) NOT NULL,
9     'securityAnswer' varchar(100) NOT NULL,
10    'activation_code' varchar(60) NOT NULL,
11    'role' enum('admin','member') NOT NULL DEFAULT 'member',
12    'ip' varchar(255) NOT NULL,
13    '2FA_Code' smallint(1) NOT NULL DEFAULT 0,
14    PRIMARY KEY (id)
```

Figure 8: acDB.sql holding accounting information



The way in which the database functions is by taking in details from the PHP elements that prepare the data being entered for entry.

```
//otherwise close statement and then prepare to insert into the db as a new account
$stmt->close();
// insert into the table called accounts
//that contains all the information about the new user signing up
$stmt = $con->prepare('INSERT INTO accounts (username, fullname, password, email, telephone ,activation_code, securityQuestion, securityAnswer, ip)
//to create complete secrecy, using a hashed passwords
//prevents insider corruption or penetration
$passwordenc = password_hash($password, PASSWORD_DEFAULT);
```

*Figure 9: Preparing SQL in register.php*

```
//store data
$stmt->bind_param('ssssssss', $username, $fullname , $passwordenc, $email, $telephone , $activate, $securityQuestion, $securityAnswer, $ip)
$stmt->execute();
$stmt->close();
//-----
```

*Figure 10: Storing Data in register.php*

```
//Sending the email by calling function from 'mainSystem.php'
$subject = 'Activation for LoveJoy';
$activate_link = 'http://lovejantiques.000webhostapp.com/activate.php?email=' . $email . '&code=' . $activate;
$message = '<p>Activate here: <a href="' . $activate_link . '">' . $activate_link . '</a></p>';

$email_template = str_replace('%link%', $message, file_get_contents('activating.html'));

if (sendEmail($email, $email_template, $subject)) {
    echo 'Registration link sent.';
} else {
    header("Refresh:4; url=http://lovejantiques.000webhostapp.com/loginAcc.php");
    echo 'Error';
}
```

*Figure 11: Sending an Activation Email in register.php*

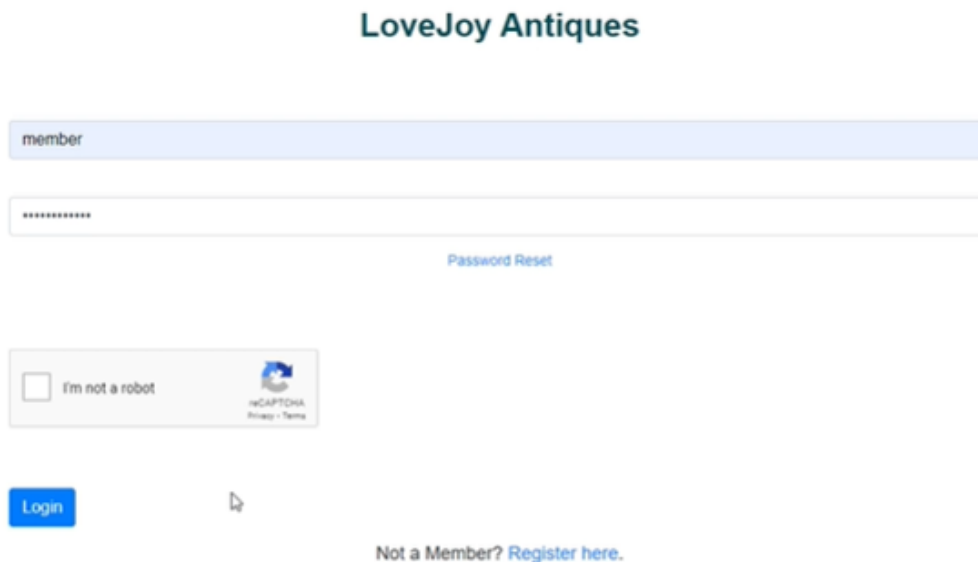
### Summary of Securing the Registration

To put it explicitly, the site features a set of security implementations that allow users to safely and securely register an account without worry of having their details exploited. This could be argued for the following reasons:

- The use of sanitizing the text entry, allows for a decrease in usage of XSS or SQL Injection
  - The way this works is by using the phrase “real\_escape\_string”, which removes any form of special characters within the entered string.
  - HTML is also affected in this sense as the use of “htmlspecialchars”, will convert special characters in alternative text. So if the user were to input &, the output would look similar to &amp, if a “ was inputted it would be converted to &quot thus protecting the integrity of the database.
    - A Common XSS attack which takes the form of  
`<script>alert('XSS')</script>`, would be changed to `&ltScript&gtAlert` and so, leading to have no effect against the entry field and getting detected and kicked by the validation used in the HTML fields.
- Further use of activating an email via an activation link sent to their address prevents the usage of fake emails as a way of just getting straight access to the site for malicious content.
  - The email used cannot be the same as the username or password either, stopping other accounts registered on the site to be compromised. This again is done using the “real\_escape\_string” method



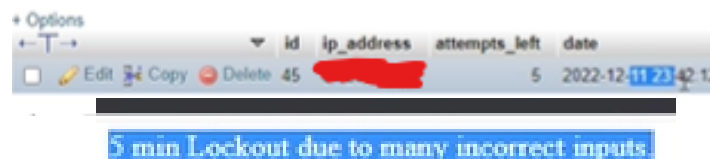
Figure 11 shows the html form used to submit a login entry onto the website. In a proven situation, this will appear as in figure 13 below:



The screenshot shows the 'LoveJoy Antiques' login page. It features a header with the site name. Below it is a login form with a text input field containing 'member', a password input field with masked characters, and a 'Password Reset' link. A reCAPTCHA widget is present with the text 'I'm not a robot' and a checkbox. A blue 'Login' button is at the bottom left, and a link 'Not a Member? Register here.' is at the bottom right.

*Figure 13: loginAcc.php in Real Instance*

Figure 11 further shows some of the unique countermeasures towards attacks that could occur on the site and will be examined towards the end of this section. These include CSRF mitigation and botnet strikes. Figure 12 is used to check whether or not the user is already logged into their account by checking the active session ID. In the instance that there is an active user already logged in, the account will be automatically launched to the Evaluation Request Page. If the user however isn't logged into an account and they present their details incorrectly, the next phase will be to lock the user out of their account and submit an entry for a set amount of time.



The screenshot shows a database table with columns: id, ip\_address, attempts\_left, and date. The first row has values 45, a redacted IP address, 5, and 2022-12-11 23:42:12. Below the table, a blue message box states: '5 min Lockout due to many incorrect inputs.'

id	ip_address	attempts_left	date
45	[REDACTED]	5	2022-12-11 23:42:12

5 min Lockout due to many incorrect inputs.

*Figure 14: login\_attempts Database (Botnet Prevention)*

```

3 //to prevent any form of dictionary and bruting attacks, the site will allow a certain amount of time before a lockout
4 //this is for 5 mins
5 $login_attempts = loginAttempts($con, FALSE);
6 if ($login_attempts == 0) {
7     exit('5 min Lockout due to many incorrect inputs.');
```

*Figure 15: login\_attempts output in logChk.php*

Figure 14 & 15 shows a database and code required to output text that contains the active information of the user attempting to login into their account. In the instance that they cannot get into their account correctly, they will be subject to a mandatory 5 minute lockout to prevent attempted brute force or botnet strikes. The database will also be able to save the active user ip to check if it is actually the user attempting to login at the time.

```

17 //-----
18 // check to see if the user actually submitted the captha correctly
19 if (isset($_POST['submit']) && $_POST['g-recaptcha-response'] != "") {
20     // skey provided by the google api
21     $secret = '6LeU6FkjAAAAABf8zx1Ds_l98KDIFpWykRjxLdM';
22     $verifyResponse = file_get_contents('https://www.google.com/recaptcha/api/siteverify?secret=' . $secret . '&response=' . $_POST['g-recaptcha-response']);
23     $responseData = json_decode($verifyResponse);
24     // if success is true, then carry on
25     if ($responseData->success) {
26         ;
27     } else {
28         //if in case of error, exit out and alert user to complete it
29         exit("captcha not completed correct, please redo and<br><a href='loginAcc.php'>login again</a>");
30     }
31 } else {
```

*Figure 16: confirmation of the captcha using the secret key in logChk.php*

```

36 //-----
37 // using real_escape_string helps to protect against cases of SQL injecting the site
38 $usernameentered = $usernameentered = mysqli_real_escape_string($con, $_POST['username']);
39 $usernameentered = htmlspecialchars($usernameentered);
40 $passwordentered = $passwordentered = mysqli_real_escape_string($con, $_POST['password']);
41 $passwordentered = htmlspecialchars($passwordentered);
42
```

*Figure 17: real\_escape\_string for SQL Injection Prevention in logChk.php*

```
43 //-----
44 //obtain details from the db and prepare to store the information into it
45 $stmt = $con->prepare('SELECT id, password, activation_code, role, ip, email FROM accounts WHERE username = ?');
46 $stmt->bind_param('s', $usernameentered);
47 $stmt->execute();
48
49 // store and check against the db
50 $stmt->store_result();
51 // see if the account is real
52 if ($stmt->num_rows > 0) {
53     $stmt->bind_result($id, $password, $activation_code, $role, $ip, $email);
54     $stmt->fetch();
55     $stmt->close();
56 }
```

*Figure 18: store details in acc DB*

## Security Features

After presenting the main parts of both the logChk.php & logAcc.php files which surround giving the user a comfortable and ease to mind of their login experience, the security features that they entail are going to be identified:

- The use of obfuscation plays a role in the coding element in preventing botnet and dictionary attempts
  - This is seen in the Login attempts database and function which actively tracks the number of attempts a user takes to login and if they bypass the amount of time allotted, the site will lock them out for a set period of 5 mins and then escalate to a total of 1 hour maximum.
  - Botnets are prevented on the other hand through the use of a google recaptcha system which can even identify if the captcha has even been attempted in the first place and shoot back an error requesting the user to please finish it.
- The site uses authenticator methods to ensure fake emails are used for malpractice.
  - If the user logging in doesn't match with the IP used at the time of account creation, they will be prompted for the completion of 2 Factor Authentication code completion, which will be sent to the original user's email address.

```

56 //-----
57 // if the account is real, then check the password to see if it is correct.
58 if (password_verify($passwordentered, $password)) {
59     // start by checking if the db displays their account as active
60     //if not, then prompt them to activation by resending an activating email
61     if ($activation_code != 'activated') {
62         echo 'Account not active, please do so <a href="reActive.php">here</a>';
63     } else if ($_SERVER['REMOTE_ADDR'] != $ip) {
64         // if the ip in the db doesn't NOT match current, then 2fa process will be prompted
65         $_SESSION['2FA'] = uniqid();
66         //2fa link will be sent to the user
67         $link = 'https://lovejantiques.000webhostapp.com/2fa.php?id=' . $id . '&email=' . $email . '&code=' . $_SESSION['2FA'];
68         header("location: $link");
69     } else {

```

Figure 19: IP Identified as not the same in logChk.php

```

8 //-----
9 //Check the ID, Email and Code are all correct in the 2FA email sent out
10 if (isset($_GET['id'], $_GET['email'], $_GET['code'], $_SESSION['2FA']) && $_SESSION['2FA'] == $_GET['code']) {
11     // as per usual
12     //using prepare as a method of preventing sql attacks
13     $stmt = $con->prepare('SELECT email, 2FA_code, role FROM accounts WHERE id = ? AND email = ?');
14     $stmt->bind_param('ii', $_GET['id'], $_GET['email']);
15     $stmt->execute();
16     //store results and check in db
17     $stmt->store_result();
18 //-----
19 // should there be an account matching the details
20 if ($stmt->num_rows > 0) {
21     //attach account code, email and the role of the account (admin or member)
22     //-----
23     $stmt->bind_result($email, $acc_code, $role);
24     $stmt->fetch();
25     $stmt->close();
26     //check the 2Fa code
27     if (isset($_POST['code'])) {
28 //-----
29         $codeEnt = mysqli_real_escape_string($con, $_POST['code']);
30         $codeEnt = htmlspecialchars($codeEnt);
31         if ($codeEnt == $acc_code) {
32             // if the code is legit, update ip address to the new location
33             $ip = $_SERVER['REMOTE_ADDR'];
34             $stmt = $con->prepare('UPDATE accounts SET ip = ? WHERE id = ?');
35             $stmt->bind_param('si', $ip, $_GET['id']);
36             $stmt->execute();
37             $stmt->close();
38 //

```

Figure 20: 2FA code sent to email in logChk.php

- As seen in the figures, once the user receives the 2FA code in their emails and completes the form on the site and submits the code received, the accounts database will update their new active IP address as their new default ip, Which in turn will be asked again for a new 2FA if their IP changes again.
  - This ensures that a person with malicious intent will not be able to login to the compromised account without the details of the 2FA pin code.

- Finally, apart from the use of 2FA & Captcha to prevent botnet attacks, the code uses `htmlspecialchars` & `real_escape_string` to sanitize and prevent attacks via SQL Injection. The use of token checking when identifying whether someone is logged in or not will also prevent CSRF (Cross-Site Requested Forgery) as the site will check to see accredited login requests versus forged. If an attack of this caliber were successful, it would leave the user's data extremely vulnerable.



## Passwording - Task 3

### Code

```
35 //checking the password matches all the listed requirements
36 //uppercase
37 $uppercase = preg_match('@[A-Z]@', $password);
38 //lowercase
39 $lowercase = preg_match('@[a-z]@', $password);
40 //numebers
41 $number = preg_match('@[0-9]@', $password);
42 //specialcharacters
43 $specialChars = preg_match('@[^\w]@', $password);
44 if(!$uppercase || !$lowercase || !$number || !$specialChars || strlen($password) < 10) {
45     exit("Password doesn't meet the requirements. <a href='index.html'>Return to Sign Up</a>");
46 }
```

*Figure 21: Register.php new password check*

- The Password requires entry of a:
  - Uppercase Letter (Line 37)
  - Lowercase Letter (Line 39)
  - Numerical Value (Line 41)
  - Special Character (Line 43)
- If there is an issue with any of the lines where it doesn't meet the requirements, the password will be rejected using the text (on line 45), and they will have to try again. This ensures that the password and entropy check of the registration phase is airtight and hard to bypass. It also ensures that the user has a strong account that is harder to compromise.

```

4 //-----
5 // If not empty when submitted
6 if (isset($_POST['email'])) {
7     //prevent xss, and check if email is inputted
8     $email = mysqli_real_escape_string($con, $_POST['email']);
9     $email = htmlspecialchars($email);
10    //preparing a statement prevents usage or at least prevenets
11    //some attempt of sql injection
12    $stmt = $con->prepare('SELECT * FROM accounts WHERE email = ?');
13    $stmt->bind_param('s', $email);
14    $stmt->execute();
15    $stmt->store_result();
16 //-----
17    if ($stmt->num_rows > 0) {
18        //sees if email is in db
19        $stmt->close();
20        // If Email does exist
21        $unqid = uniqid();
22        $stmt = $con->prepare('UPDATE accounts SET reset = ? WHERE email = ?');
23        $stmt->bind_param('ss', $unqid, $email);
24        $stmt->execute();
25        $stmt->close();
26    //-----
27        //create a new email to send to the user
28        $subject = 'pass reset';
29        $reset_link = 'http://lovejantiques.000webhostapp.com/resetpassword.php?email=' . $email . '&code=' . $unqid;
30        $message = '<p>reset password here: <a href="' . $reset_link . '">' . $reset_link . '</a></p>';
31    //-----
32        //email body and text
33        $email_template = str_replace('%link%', $message, file_get_contents('passwordRez.html'));
34        if (sendEmail($email, $email_template, $subject)) {
35            $msg = 'Reset Sent.';
36        } else {
37            $msg = 'Error';
38            header("Refresh:5; url=loginAcc.php");

```

Figure 22: reset password in passRez.php

- If in the instance the user forgets their password, they can choose to reset it.
  - Prevent SQL injection attempts in the textboxes using “htmlspecialchars” (line 9).
  - Preparing the statement to store into the database will also aid this (line 12)
  - (Line 17 to 34) Sending an email to the User with a generated reset link used to reset their password.
    - This newly generated link will include a new set of Password policies as there needs to be an assumption that in the worst case scenario, the password and account was compromised to ensure total security.

```

19 // If it is found that the user requesting the password actually exists in the database
20 // begin by fetching their security details and old password to make sure they don't copy the exact same
21 //password over again
22 if ($stmt->num_rows > 0) {
23     $stmt->bind_result($securityQuestion, $securityAnswer, $passwordOld, $username);
24     $stmt->fetch();
25     $stmt->close();
26     //If not empty
27 //-----
28     if (isset($_POST['new_password'], $_POST['confirm_password'])) {
29         //using real_escape_string prevents attacks against the code in SQL injection methods
30         $securityAnswerEnt = mysqli_real_escape_string($con, $_POST['securityAnswer']);
31         $securityAnswerEnt = htmlspecialchars($securityAnswerEnt);
32         $password = mysqli_real_escape_string($con, $_POST['new_password']);
33         $password = htmlspecialchars($password);
34         $confirm_password = mysqli_real_escape_string($con, $_POST['confirm_password']);
35         $confirm_password = htmlspecialchars($confirm_password);
36 //-----
37         //in case the user decides to use caps lock for the security question
38         //this will just make sure they match no matter what
39         $securityAnswer=strtolower($securityAnswer);
40         $securityAnswerEnt=strtolower($securityAnswerEnt);
41 //-----
42         //Checking password strength just like in registration
43         //uppercase
44         $uppercase = preg_match('@[A-Z]@', $password);
45         //lowercase
46         $lowercase = preg_match('@[a-z]@', $password);
47         //numbers
48         $number = preg_match('@[0-9]@', $password);
49         //if there is the use of a special character
50         $specialChars = preg_match('@^[^w]@', $password);
51

```

*Figure 23: entropy of resetpassword.php*

- The php code here is what occurs when the user clicks the reset password link received by their email address.
  - The php code takes information from the database and binds it to a new set of variables for cross comparison (line 22 to 26)
  - To prevent SQL injecting or XSS scripting, the same sanitizing methods have been used in appropriate places (Line 29 to 35)
  - To implement validation to prevent any common errors, text is converted to lower case to ensure no mistakes on entry (line 39 & 40)
  - The same requirements are then implemented from the registration page to check if the password follows the same basic guidelines (Line 44 to 50)

```

52 //-----
53
54 //Checking new password entry
55 if(!$uppercase || !$lowercase || !$number || !$specialChars || strlen($password) < 10 || strlen($password) > 100) {
56     $msg = 'Password must contain: 8 characters of which are: uppercase, lowercase, numerical & a special character.';
57 } else if ($confirm_password != $password) {
58     //Checking if new passwords match each other
59     $msg = 'new password doesnt match the confirmation';
60 } else if (strcmp($securityAnswer, $securityAnswerEnt) != 0) {
61     //checking to see if security answer is a match
62     $msg = 'answer for the security question is incorrect.';
63 //-----
64 } else {
65     //Store the new password in the database and update the accounts table
66     $stmt = $con->prepare('UPDATE accounts SET password = ?, reset = "" WHERE email = ?');
67 //-----
68     //encrypt the new password
69     $passwordenc = password_hash($password, PASSWORD_DEFAULT);
70     $stmt->bind_param('ss', $passwordenc, $_GET['email']);
71     $stmt->execute();
72     $stmt->close();
73 //-----
74     //close msgs
75     $msg = 'Password changed.';
76     header("Refresh: 5; url=http://lovejantiques.000webhostapp.com/loginAcc.php");

```

Figure 24: resetpassword.php cont.

- Cont. of resetpassword.php
  - Taking a look at the basic checks of the password (Line 55 to 59).
    - There is a character limit of 100 to prevent the overflowing of the database
  - Check to see that the security question is correct, otherwise default to an error (Line 62)
  - Make sure the new and the older password are different (Line 57).
    - Furthermore, this also checks to see if the new password is confirmed correctly
  - Encrypt the new password in the db (Line 69)
    - Do this by binding and preparing the SQL code to prevent malicious interference.
  - Print out successful change (Line 76)

### Summary of Password Policy

- Based on the code above, the password policy can be summarized as follows:
  - Must be at least 8 Characters long, 10 characters if the password needs to be reset
  - Should Include a single value minimum (1,2,3,4...)
  - Should have a special character (!,<,>,#,@)
  - Include an Uppercase (A,B,C,D...)
  - Include a lowercase (a,b,c,d...)
  - Must match the password confirmation
  - In the case of a password reset, MUST NOT be the same as a older/already used password
  - User ID  $\neq$  Password
- To improve password policy if the project were to be undertaken in another instance:
  - Look for passwords written in “LEET” format eg. Hello = H3110, Coding = c0d1n6 etc.
  - Check to see palindromes eg. Kayak, Racecar, Rotor, Civic etc.

### Why is it Secure?

- All Passwords saved are encrypted and cannot be view by the admin or anyone with access to the database, for example passwords would be saved as:



- The php coding uses Authenticating means of checking the user resetting their password is who they say they are (Fig 22, line 29)
  - Only way of direct entry would be for the user email to be compromised as well

- As well as the security question needs to be correct otherwise there can be no change to the password
  - Improvements could be made if the user also forgets their security question.
- SQL injection is prevented using `real_escape_string` in areas of data submission, also `htmlspecialchars` on the used HTML forms found in `forgotpassword.php`

## Evaluation - Task 4

## Code Snippet

```

<!-- if the user is an admin, they will see an option for the view requests -->
<?php if ($_SESSION['role'] == 'Admin'): ?>
    <li><a href="vReq.php"></i>view requests</a></li>
<?php endif; ?>
<!-- logout button for the user in the top left corner -->
<li style="float:left"><a href="logout.php"></i>logout</a></li>

'>
>w">
:lass = "col-md-12">
>r>
>r>
>r>
<h2><strong><p style="color:#004c54">Love Joy Antiques: </strong><br><small>request evaluation</small></h2><br>
<!-- check the details of the evaluation upon submitting -->
<form action="rChck.php" method="post" autocomplete="off" enctype="multipart/form-data">
    <div class="form-group">
        <u><p style="text-align:left">Select the image of the item</p></u>
        <!-- require a file to be selected from the user's images, as well as a descriptor -->
        <input type="file" name="antique_image" required>
    </div>
    <br>
    <div class="form-group">
        <u><p style="text-align:left">Describe Said Object</p></u>
        <!-- text to talk abotu the object details, which can also be resized -->
        <textarea autofocus name="desc" spellcheck minlength="10" rows="5" cols="50" placeholder="Object Details" required></textarea>
    </div>
    <br>
    <!-- radio button to choose between telephone number and email which will be saved
    in hte db-->
    <div class="form-group">
        <u><p style="text-align:left">Preferred Method of Contact</p></u>
        <input type="radio" id="telephone" value=0 name="contactChoice" required>
        <label style="margin-right: 30px" for="email">Email</label>
        <input type="radio" id="email" value=1 name="contactChoice" required>
        <label for="telephone">Mobile Phone</label><br>
    </div>
    <!-- close msg -->
    <div class="form-group">
        <input type="submit" name="submit" class="btn btn-primary" value="submit">

```

Figure 25: rEval.php (HTML form submission)

- The program begins by checking to see the role of the user logged in at the current time.
  - If they are an admin, they will default to view requests (task 6)
  - Otherwise, they will go the request evaluations page (as seen below)

**Love Joy Antiques:**  
request evaluation

Select the image of the item

Choose File No file chosen

Describe Said Object

Object Details

Preferred Method of Contact

☐ Email ☐ Mobile Phone

- The HTML document will prompt users to insert an image of their choice, and then describe their object. It will end with asking for a preferred method of contact.
  - Each of the elements of the page have validation involved, meaning that if the user were to accidentally miss anything and forget to input any results, the page would inform them to complete it before continuing
- Buttons are used to determine the difference between email (0) and mobile telephone (1), saving in the database accordingly (Figure 26)

	valid	description	contactDetail	image_url	userid
1	1	Capsan from a boat!	0	IMG-63966b4745d104.20961203.jpg	11
11	11	This is my favourite antique!	1	IMG-63966b60870979.86145852.jpg	11

Figure 26: rEval.php Database

## Security

```

5 //-----
6 //check to see if the user has submitted an image and text
7 if (isset($_POST['submit']) && isset($_FILES['antique_image'])) {
8     //gets the name of image
9     $img_name = $_FILES['antique_image']['name'];
10    //gets the size of the image
11    $img_size = $_FILES['antique_image']['size'];
12    //checks for any errors
13    $tmp_name = $_FILES['antique_image']['tmp_name'];
14    $error = $_FILES['antique_image']['error'];
15 //-----
16    //check the contact detail saved
17    $choice = $_POST['contactChoice'];
18 //-----
19    //using real_escape_string to prevent sql injection or XSS
20    $desc = mysqli_real_escape_string($con, $_POST['desc']);
21    $desc = htmlspecialchars($desc);
22 //-----

```

Figure 27: rEval.php Database



- As seen in figure 27, line 7 shows that the image data is being received (further discussion in section below).
  - However, the use of `_isset` and lack of `htmlentities` means that there is no need to worry about XSS attack or SQL injection due to the inability to submit any special characters
- This changes on lines 20 & 21 when special characters can be submitted into the description boxes.
  - Therefore common sanitizing methods used to remove any illegal characters are exercised here.
- Making sure the user accessing this page is actually a user is also extremely crucial.
  - This prevents anybody just accessing the page through the URL and submitting any random bits of data they have and flooding the system.
  - Instead they will be kicked back to the login screen and prompted to make an account.

## Photo Submission - Task 5

### Image File Upload Code

```

38         <form action="rChck.php" method="post" autocomplete="off" enctype="multipart/form-data">
39             <div class="form-group">
40                 <u><p style="text-align:left">Select the image of the item</p></u>
41                 <!-- require a file to be selected from the user's images, as well as a descriptor -->
42                 <input type="file" name="antique_image" required>
43             </div>

```

Figure 28: HTML of rEva.php for submitting an Image

- The set “input-type” is on image (line 42), which allows the user to upload an image
  - It will automatically name it antique\_image in the database and save it after it completes the required checks to see if it is valid for upload.

```

26 //-----
27 //in the instance that everything meets the requirements
28 //begin checking the image
29 if ($error == 0) {
30     //check to see if the image is too large for the entry image
31     //since most ripped images will be around 600 to 900kb in most instances
32     //1250000 kb or 1.25mb is a generous amount of memory allotted
33     if ($img_size > 1250000) {
34         echo 'file too large for entry (x < 1.25mb)';
35         header("Refresh:2; url=rEva.php");
36     }else {
37 //-----
38         //checking the file extension to ensure htat the right file type is being used
39         $img_ex = pathinfo($img_name, PATHINFO_EXTENSION);
40         //changes it to Lowercase for cross comparison
41         $img_ex_lc = strtolower($img_ex);
42 //-----
43         //THE ONLY FILE TYPES ALLOWED
44         $allowed_exs = array("jpg", "jpeg", "png");
45 //-----
46         //if the data recieved is an image format without any errors
47         if (in_array($img_ex_lc, $allowed_exs)) {
48             //give the file a name and uniqueid to attach to the user ID
49             $new_img_name = uniqid("IMG-", true).'.'.$img_ex_lc;
50             //attach the folder of storage to the file name
51             //all the images are stored in a folder called antiImages
52             $img_upload_path = 'antiImages/'.$new_img_name;
53 //-----
54             //take the image and move to hte folder
55             move_uploaded_file($tmp_name, $img_upload_path);
56 //-----
57             //Move image to the database, and assign unique
58             //user id's for each called evalID
59             //preping the statement allows for protection against SQL attempts
60             $stmt = $con->prepare("INSERT INTO evaluations (description, contactDetail, image_url, userid) VALUES (?, ?, ?, ?)");
61 //

```

Figure 29: rChck.php, checks submitted data entry

- Steps surrounding accepting an image in this code are as follow:
  - Checks to see if the image submitted is under or equal to the allotted memory of 1.25mb (Line 33)

- Check the file extension. Reject if not jpg, jpeg or png (Line 44)
  - If it is correct, then upload it to the folder called antiImages (Line 52)
- Prepare the upload to DB (Line 60)

This is seen in the live images below:



*Figure 30: Evaluation Database*

## Security Features

- There are a few security features that are used to prevent any forms of attacks that could take place when submitting an image to the site.
  - Htmlspecialchars or real\_escape\_string are not tools needing to be implemented due to the lack of text needed when actually submitting an image.
    - Meaning SQL injecting cannot occur within the image file upload itself
  - Furthermore, outside research suggests that keeping file uploads to around 1.25mb, can prevent DDOS-based attacks that involve uploading gigabytes of data that throttle database and site bandwidth.
    - Usage of the file-extension upload checker also ensures that files cannot be uploaded with malevolent scheming in mind (scripts, worms, RAT's etc.)
  - This section of the site cannot be accessed by just any random user.

- Typing in the URL extension of rEval.php, will kick the user to the login screen
- Furthermore, accounts must be correctly activated in-order to get started uploading images, meaning accounts must use a non-fake email address.

**(Extension) Request Listing Page - Task 6****Code Snippets**

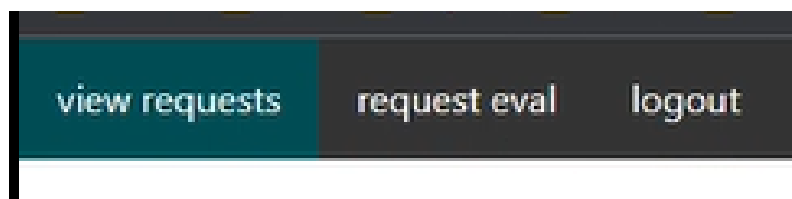
```

6 if ($_SESSION['role'] != 'Admin') {
7     header("Location: rEva.php");
8     exit;
9
10 //-----
11 }
12 ?>
13 <!DOCTYPE html>
14 <!-- view request-->
15 <html>
16     <head>
17         <meta charset="UTF-8">
18         <!-- header name-->
19         <title>view requests</title>
20         <link rel="stylesheet" href="s1sheet.css">
21         <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
22
23     </head>
24     <body class="loggedin">
25         <nav class="navtop">
26             <div>
27                 <ul>
28                     <!-- options on the top Left corner to move around the site -->
29                     <li><a href="vReq.php" class="active"></i>view requests</a></li>
30                     <li><a href="rEva.php">request eval</a></li>
31                     <li style="float:left"><a href="logout.php"></i>logout</a></li>

```

*Figure 31: vReq.php (HTML Segment)*

- This page will only ever be actively available for an admin user
  - This is check on line 6, which sees whether or not the user is currently an admin
  - Furthermore, since this is the page all admins are directed to immediately, they see more options to navigate from on their home screen [Lines 28 to 31](As seen below in live view as well)]

*Figure 32: vReq.php (Live View)*

```



41                                     <h2><strong><p style="color:#004c54">Love Joy Anti: </strong><br><small>all the active requests</small></h2>
42                                     <?php
43                                     //-----
44                                     //grab all the data and information needed from two different tables
45                                     //one is the account to get the user's name, and then all of the evaluation data
46                                     $sql = 'SELECT accounts.fullname, evaluations.description, evaluations.contactDetail, accounts.email, accounts.telephone,
47                                     $result = mysqli_query($con, $sql);
48                                     $results = mysqli_query($con, $sql);
49                                     //-----
50                                     //in the instance that there are no active request,
51                                     //tell the admin
52                                     if ($results->num_rows === 0) {
53                                         echo "<p>no active submission, come back another time</p>";
54                                         exit();
55                                     }
56                                     //-----
57
58                                     //if there are active requests,
59                                     //create a table of elements that will be used to display all the antiques
60                                     echo "<table>";
61                                     echo "<tr>";
62                                     //-----
63                                     //customer name
64                                     echo "<th>customer</th>";
65                                     //-----
66                                     //descript of antiques
67                                     echo "<th>attributes</th>";
68                                     //-----
69                                     //contact info
70                                     echo "<th>contact info</th>";
71                                     //-----
72                                     //image
73                                     echo "<th>image</th>";
74                                     //-----
75                                     echo "</tr>";
76                                     //-----
77                                     //printing rows
78                                     //of tables

```

Figure 33: vReq.php (Making a Table of Contents)

- After grabbing User data from the database (Lines 46 to 48), this information is referenced to create a table of active requests made by users on the platform.
  - Line 53 shows that if there are no active requests, then print text to inform the admin
  - Otherwise, Lines 57 to 78 show details of the user being made in HTML which is styled in tabular format as seen live below:

all the active requests

customer	attributes	contact info	image
test test	Capsan from a boat!	testjoymember@gmail.com	
test test	This is my favourite antique!	01234567890	

### Security Features

- A lot less features are required for this page due to the nature of those who can access it
  - The already set role for only admins to access it means that trust is already an existing factor that cannot be defined through the physical web page itself, but outside external circumstances.
  - Is no actual interaction on this page itself, just viewing ability
    - Could be improved to allow an admin to remove submissions on through the site itself without needing to interact with the database.
  - Typing the URL into the site will kick the user to the sign up page

## Self-Evaluation - Task 0

Excellent (10-9 marks)	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.	<b>Password policy 10marks</b>  Password entropy, encrypted storage, security questions and recovery of password  Implemented all the required elements for the password policy
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.	<b>Vulnerabilities 10 marks</b>  SQL injection, XSS, CSRF, File Upload and any other obvious vulnerability.  Implemented all the required elements for the Vulnerabilities
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.	<b>Authentication 10 marks</b>  User identity management (registration and login etc), Email verification for registration, 2 factor authentications (PIN and or email),  Implemented all the required elements for the Authentication



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.	<b>Obfuscation/Common attacks</b> <b>10 marks</b>  Brute force attack – Number of attempts  Botnet attack – Captcha  Implemented all the required elements for the Obfuscation but did not go beyond
Implementation of other security features has no flaws. No obvious security feature is ignored.	Several security features implemented. Implementation has flaws.	Other security features are implemented but obvious ones are ignored.	Very little effort to implement some obvious other security features like storage of confidential information.	<b>Other security features like confidentiality of important information</b> <b>10 marks</b>  Implemented all the required elements for the encryption and ensured sensitive material was kept a secret. Could have improved by making IP address encrypted as well.
Claimed features are complex. Quality of achievement is excellent.	Claimed features are complex however quality of achievement/implementation could have been better.	Claimed features are somewhat complex and implementation could have been better.	Claimed features are not complex and challenging.	<b>Deeper understanding, two extra web security</b> <b>10 marks</b>  Implemented main security features, but did not implement anything extra. Did include features that checked to see if the user just typed in certain URL's instead of being logged in.

## Appendix

### mainSystem.php

```

5
6 // connect to the file containing db elements
7 include_once 'dbConfig.php';
8 session_start();
9
10 // connect to db
11 $con = mysqli_connect(db_host, db_username, db_password, db_name);
12 if (mysqli_connect_errno()) {
13     exit('cannot connect to db: ' . mysqli_connect_error());
14 }
15 //-----
16 mysqli_set_charset($con, db_charset);
17
18 // function to check to see if the user is logged in
19 function check_loggedin($con, $redirect_file = 'loginAcc.php') {
20     if (!isset($_SESSION['loggedin'])) {
21         // If the user is not logged in redirect to the login page.
22         header('Location: ' . $redirect_file);
23         exit;
24     }
25 }
26 //-----
27 //using login attempts to prevent bruteforce the system
28 function loginAttempts($con, $update = TRUE) {
29     //get ip of failed attempt
30     $ip = $_SERVER['REMOTE_ADDR'];
31     $now = date('Y-m-d H:i:s');
32     if ($update) {
33         // -1 per time there is an incorrect attempt to login
34         $stmt = $con->prepare('INSERT INTO login_attempts (ip_address, `date`) VALUES (?,?) ON DUPLICATE KEY UPDATE attempts_left = attempts_left - 1, `date` = VALUES(`date`)');
35         $stmt->bind_param('ss', $ip, $now);
36         $stmt->execute();
37         $stmt->close();
38     }
39 }

```

*Figure 33: mainSystem.php*

- Lines 11 to 13, confirm whether the user is able to connect to a database
- Line 28 to 37 is a function concerning login-attempts and ensures that a user is tracked on the number of attempts they make on logging in.
  - Otherwise, to prevent botnet attacks, they are timed out for a certain period of time.

```

68 //-----
69 //new function for the sendEmail PHPmailer fucntion
70 function sendEmail($email, $email_template, $subject) {
71     //setting it in a function allow for easier emails to be sent out
72     //as it means that any case where an email is needed, this can jsut be called
73     //start by using require to establish what is needed for php mailer
74     require 'PHPMailer.php';
75     require 'Exception.php';
76     require 'SMTP.php';
77     require 'ljDetails.php';
78
79     //create new instance
80     $mail = new PHPMailer(true);
81     //set mailer to smtp
82     $mail->isSMTP();
83     //define the smpt domain
84     $mail->Host = 'smtp.gmail.com';
85     //set auth to true
86     $mail->SMTPAuth = true;
87     // set love joy sender email
88     $mail->Username = 'FahdLoveJoy@gmail.com';
89     //password needed to access the email through remote app
90     $mail->Password = 'vrjywtflndjiwyae';
91     //set tls encryption
92     $mail->SMTPSecure = 'tls';
93     //define the port
94     $mail->Port = 587;
95
96     //set sender email and the name to appear
97     $mail->setFrom(EMAIL, 'Ctznfour from Lovejoy');
98     //add the user address
99     $mail->addAddress($email);
100    //attach lovejoy email
101    $mail->addReplyTo('FahdLoveJoy@gmail.com');
102    //enable html
103    $mail->isHTML(true);
104    //allow for a custom message dependig on use case
105    $mail->Subject = $subject;
106    $mail->Body = $email_template;
107
108    //send the email
109    if(!$mail->send()) {
110        return 0;

```

Figure 34: mainSystem.php cont.

- Lines 68 to 110 are the code required to send emails via PHPMailer.
  - A tool used in PHP to send emails that is used within the website parameters several times for password resets, activating accounts, resending code, 2FA auth and so on. SMTP is set to Gmail to allow for a setup gmail account to be used to send out these emails.
  - The subject for the emails are set to a variable to allow for the message to be customized based on their purpose which changes on the PHP file.

**dbConfig.php**

```
1 <?php
2 //db host config settings
3 //-----
4 define('db_host', 'localhost');
5 define('db_username', 'id19786655_sfshah');
6 define('db_password', 'QWEasdZXC122!#');
7 define('db_name', 'id19786655_ss2168_compsec');
8 define('db_charset', 'utf8');
9 ?>
```

*Figure 35: dbConfig.php*

- Settings required to be able to access my database
  - In php file format as it removes the need to access every instance of database interaction, and just need to update one file in-case of issues with naming or formatting of the main database.

### **Link to Downloads & Viewing Testing**

**Site Link:** <https://lovejantiques.000webhostapp.com>

### **Dropbox**

- Dropbox of Files (Will Auto Download, so check downloads in chrome/or whatever browser is being used):

<https://www.dropbox.com/sh/dmu21eb4t9m1uc3/AABDtb3j1mBKEp6JZIy2nxPha?dl=1>

### **Video of User Testing**

- Entire Video of Testing with all the Features Demonstrated:

<https://sussex.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=0e891966-81da-4635-8800-af6801854d5f>