Report: AWS VPC & Lovejoy’s Antiques Site

Student #246518

14.12.2023

Introduction to Computer Security

Table of Contents

Important Information

Website URL: https://matriarchal-balls.000webhostapp.com/

Source Code (OneDrive): matriarchal-balls source code.zip

Source Code Backup (Google Drive):
https://drive.google.com/file/d/1TZG1ldofux8aoSme1Vzi_45JkY1FSARC/view?usp=drive_link

Video Recording (Panopto):
https://sussex.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=99a8a08a-a6a7-41e5-8eba-b0d800d0bd4a

Backup Recording (Youtube):

https://youtu.be/274PQUHFFHQ

Sample Users:

- Admin Account
  - Email: td336@sussex.ac.uk
  - Password: Q!I0u76M99£9}vT!-;5
  - Security Question Answer: Chichester 1 LT
- Standard Account
  - Email: rgpnhxbbcoytnfxclq@cazlp.com
  - Password: q%9b@d)RR<;[?2G3n<$j?
  - Security Question Answer: Computer Science

Feel free to register your own account(s). You'll have to verify your email address. You're also free to activate 2 factor authentication on your account. There may be a chance that my API Credit runs out if you test this feature too much, so please do it in moderation.

Something to Note

000Webhost's servers have been found to sometimes be quite unreliable. Any ERR_CONNECTION_TIMED_OUT errors are due to 000webhost and not my code getting stuck in an inescapable loop or taking too long to respond. There's a chance that you may be registered in an invalid state if this occurs while registering. If it does, please register with another email.

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** • Passwords are Stored encrypted with salting & hashing. • Passwords are not stored anywhere in plaintext. • Passwords must follow the password policy (see task 3 for details). • 5 Incorrect login attempts will block a user’s IP from making login attempts for 5 minutes. • Users are blocked from using frequently used passwords. • Passwords can be reset by providing an email. o A one-time use token is used to reset the user’s password. o This token expires after 10 minutes. • When registering, users must give an answer to a security question (which they choose), which will be asked every time the user logs in (unless 2fa is turned on.). • Upon Registration & Password Resetting, Troy Hunt’s PWN’d Password API is called to see if the password has been exposed in a data breach. If it has, the user is told to select another 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 attacks are mitigated through using prepared SQL Statements & parameter binding. • All user input is sanitised using a filter to prevent XSS. • All forms include a CSRF Token which is checked during form submission to prevent CSRF attacks. • File uploads have been made secure. o Any uploaded files must be under 5MB. o File extensions are checked to be either JPEG or PNG. o Files are stored on the web server rather than the database to improve fetch times. o Files are renamed to random, high entropy strings to prevent attackers from viewing the files by luck from the web server.
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** • Users must verify their email addresses before they are able to log into the site. • Users can opt-in to use SMS-based 2 factor Authentication. o Users can disable 2 factor Authentication when logged in. o Phone numbers are checked. (+44 format) o Users with 2FA Enabled must enter their phone number every time they want to log in. o Activating 2FA disables the need for security questions when logging in. • Session tokens are used to keep track of users, rather than storing credentials.

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** • reCAPTCHA V3 Is used to prevent bot attacks. • Incorrect login attempts are rate limited and will block any IPs which input too many incorrect logins. • Evaluation Requests are limited to one per 5 minutes to prevent spam. • Rainbow Table & Dictionary attacks are prevented by using salting and hashing.
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** • All HTTP requests are redirected to be HTTPS requests to prevent Man in the Middle Attacks. • Enforced input validation by limiting database interactions to the a column's varchar limit, enhancing data integrity and security. •Disabled directory listing to stop potential attackers from accessing and listing the server's pages. • HTTP(s) Requests without corresponding files or directories are redirected to the index file. • Database, API, and Email credentials are stored within a separate file from the web root to prevent unauthorised access through the web.

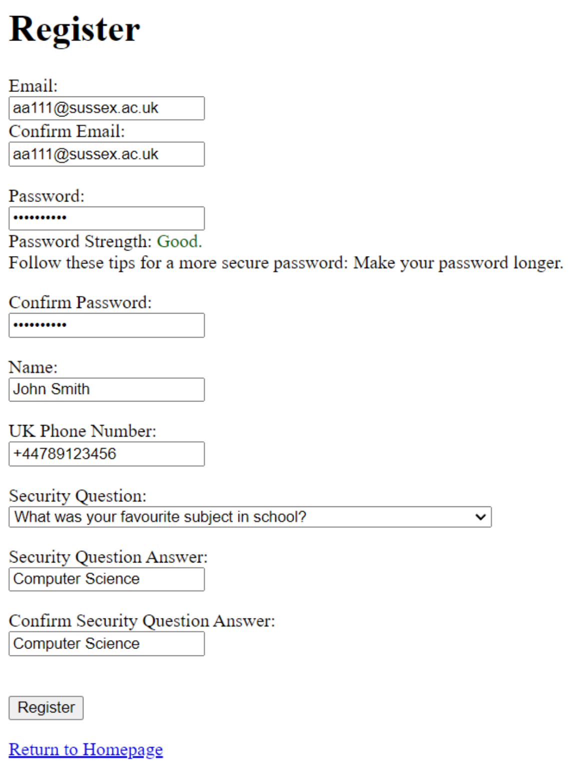
5 marks	**5 marks**	**5 marks**	**5 marks**	**5 marks**	**10 marks**	
List evaluation-Task6 Completed	Request evaluation – task 5 Completed	Request evaluation – task 4 Completed	Forgot password-Task3 Completed	Login-Task2 Completed	User registration/Database-Task1 Completed	**Features of webs application**

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

Excellent (9 to 10)	**Good (6 to 8)**	**Average (3 to 5)**	**Poor (0-2)**	
Everything is implemented as in the infrastructure	Very little mistakes in the implementation	Few mistakes in implementation	Very little attempt.	**Virtual Private Cloud & Security groups**

Task 1 – User Registration

Below is a screenshot of my site's registration page.



Register

Email:
aa111@sussex.ac.uk
Confirm Email:
aa111@sussex.ac.uk

Password:
••••••••••
Password Strength: Good.
Follow these tips for a more secure password: Make your password longer.

Confirm Password:
••••••••••

Name:
John Smith

UK Phone Number:
+44789123456

Security Question:
What was your favourite subject in school?

Security Question Answer:
Computer Science

Confirm Security Question Answer:
Computer Science

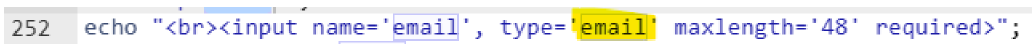
Register

Return to Homepage

Figure 1: A screenshot of the site's registration form, filled.

Users can register to the site by filling the above form. Each part of the form contains the following security & validation features:

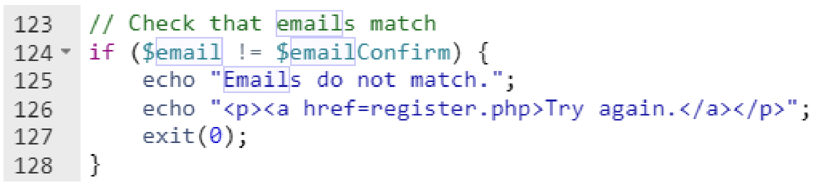
- **Emails**
  - Only valid emails are allowed to be submitted.



```
echo "<br><input name='email', type='email' maxlength='48' required>";
```

Figure 2: Code which checks for emails. (register.php)

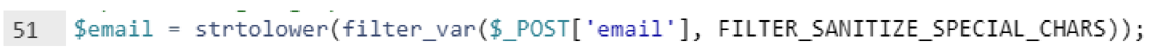
  - Users must confirm their email using a second box.
  - The email and email conformation boxes must match, otherwise registration is stopped.



```
// Check that emails match
if ($email != $emailConfirm) {
    echo "Emails do not match.";
    echo "<p><a href=register.php>Try again.</a></p>";
    exit(0);
}
```

Figure 3: Emails being checked to be identical. (registerCheck.php)

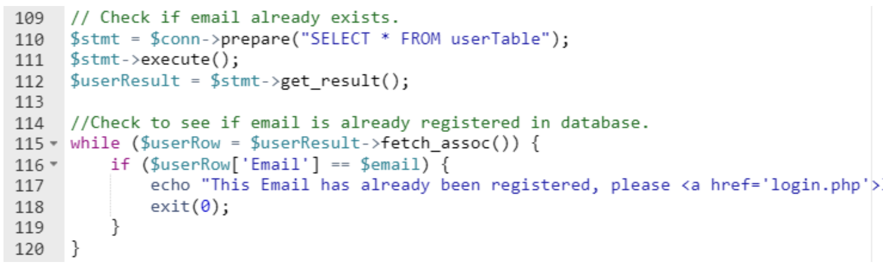
  - All emails are forced to be lowercase when being processed to prevent duplicate emails with mismatched case from being registered.



```
$email = strtolower(filter_var($_POST['email'], FILTER_SANITIZE_SPECIAL_CHARS));
```

Figure 4: Emails being forced to lowercase to prevent duplicate emails due to Case discrepencies. (registerCheck.php)

  - Registering an email which is already registered onto the system will prompt the user to log in, instead of registering. Users cannot re-register an email.

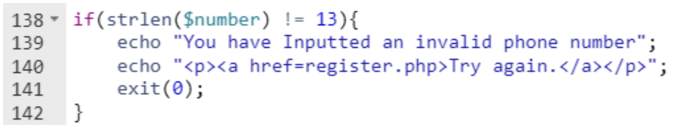


```
// Check if email already exists.
$stmt = $conn->prepare("SELECT * FROM userTable");
$stmt->execute();
$userResult = $stmt->get_result();

//Check to see if email is already registered in database.
while ($userRow = $userResult->fetch_assoc()) {
    if ($userRow['Email'] == $email) {
        echo "This Email has already been registered, please <a href='login.php'>log in</a> instead.";
        exit(0);
    }
}
```

Figure 5: duplicate email checking (registerCheck.php)

- **Passwords**
  - The registration form will not allow the user to proceed if the site's password policy (see task 3), is not followed, this increases the difficulty of guessing passwords.

- **Phone Numbers**
  - Users must provide a phone number to be contacted and for 2 factor Authentication.
  - Phone numbers which don't follow the UK International Phone number standard (starting with +44, followed by 10 digits) are rejected.
  - Entering an invalid phone number will stop registration.

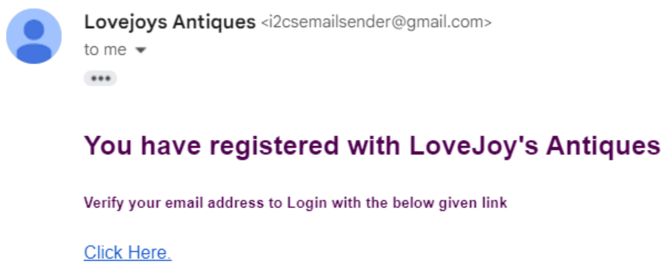


```
if(strlen($number) != 13){
    echo "You have Inputted an invalid phone number";
    echo "<p><a href=register.php>Try again.</a></p>";
    exit(0);
}
```

Figure 6: Phone number length checking. (registerCheck.php)

- **A Security Question.**
  - Users are required to select a security question and provide an answer.

- After this form has been correctly filled, the user is Registered onto the Site's database. A confirmation email is sent to the email, which provides a link for the user to verify their email address, which ensures users use legitimate emails.


Lovejoys Antiques <i2csemailsender@gmail.com>
to me

You have registered with LoveJoy's Antiques

Verify your email address to Login with the below given link

Click Here.

Figure 7: A Screenshot of the verification email sent during registration.

Once the link is clicked, the user can log in. (see Task 2)

Storage and Additional Security/Validation

- Upon registration, users are given a randomised Unique User ID rather than an incrementing ID.

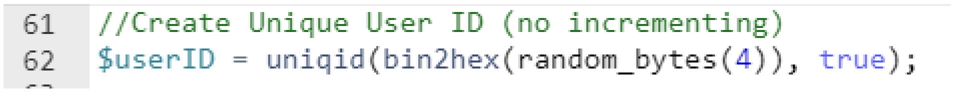

```php
//Create Unique User ID (no incrementing)
$userID = uniqid(bin2hex(random_bytes(4)), true);
```

Figure 8: Code showing how Random User IDs are generated (registerCheck.php).

- Passwords are not stored as plaintext anywhere on the system but are instead stored as a salted hash.

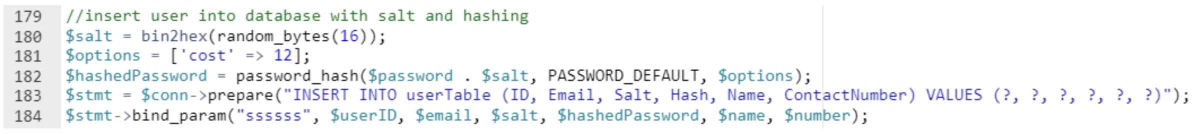

```php
//insert user into database with salt and hashing
$salt = bin2hex(random_bytes(16));
$options = ['cost' => 12];
$hashedPassword = password_hash($password . $salt, PASSWORD_DEFAULT, $options);
$stmt = $conn->prepare("INSERT INTO userTable (ID, Email, Salt, Hash, Name, ContactNumber) VALUES (?, ?, ?, ?, ?, ?)");
$stmt->bind_param("ssssss", $userID, $email, $salt, $hashedPassword, $name, $number);
```

Figure 9: Code showing Passwords being salt and hashed for Database insertion (registerCheck.php).

- The same hashing and salting are also done for the answers to security questions.

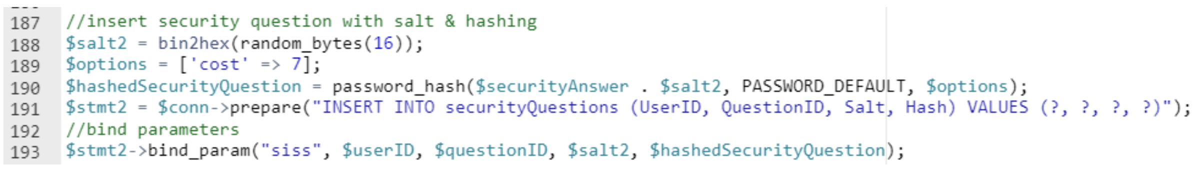
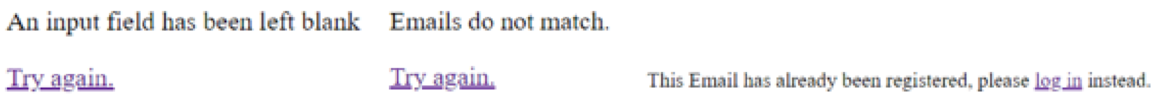

```php
//insert security question with salt & hashing
$salt2 = bin2hex(random_bytes(16));
$options = ['cost' => 7];
$hashedSecurityQuestion = password_hash($securityAnswer . $salt2, PASSWORD_DEFAULT, $options);
$stmt2 = $conn->prepare("INSERT INTO securityQuestions (UserID, QuestionID, Salt, Hash) VALUES (?, ?, ?, ?)");
//bind parameters
$stmt2->bind_param("siss", $userID, $questionID, $salt2, $hashedSecurityQuestion);
```

Figure 10: Code of how user security questions are inserted into the database. (registercheck.php).

- All input boxes in the form are required to be filled for the user to be registered to the site. Leaving a box blank will result in an error message and will stop registration. All 'confirm' boxes must also match.
  Stopping registration for invalid input stops the database from breaking when invalid user details would be added.


An input field has been left blank

Try again.

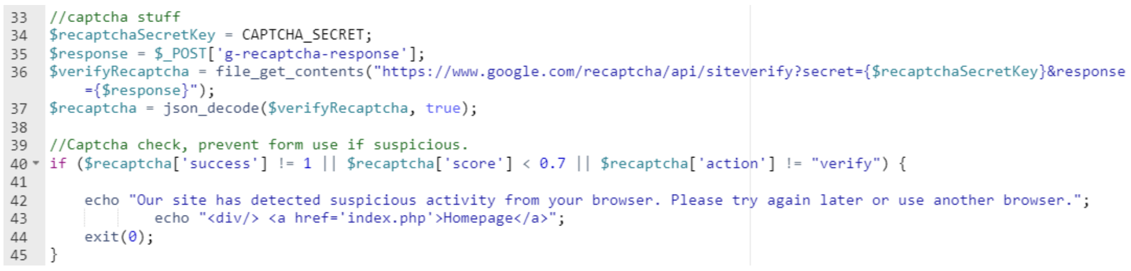
Emails do not match.

Try again.

This Email has already been registered, please log in instead.

Figure 11: Examples of error messages from invalid registration forms.

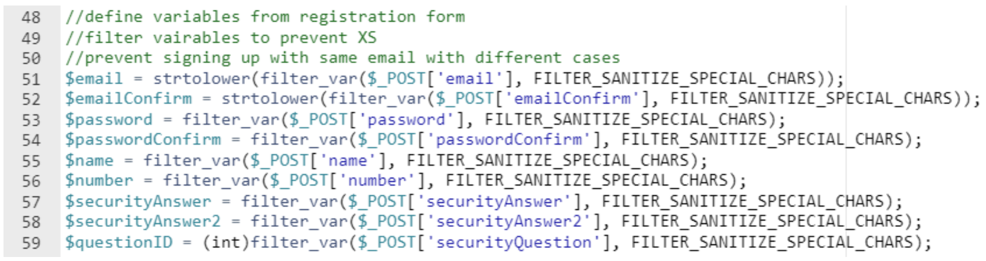
- Google's reCAPTCHA v3 is used on this registration page to prevent bots from using it. Suspicious users are prevented from accessing the site.



```php
//captcha stuff
$recaptchaSecretKey = CAPTCHA_SECRET;
$response = $_POST['g-recaptcha-response'];
$verifyRecaptcha = file_get_contents("https://www.google.com/recaptcha/api/siteverify?secret={$recaptchaSecretKey}&response
    ={$response}");
$recaptcha = json_decode($verifyRecaptcha, true);

//Captcha check, prevent form use if suspicious.
if ($recaptcha['success'] != 1 || $recaptcha['score'] < 0.7 || $recaptcha['action'] != "verify") {

    echo "Our site has detected suspicious activity from your browser. Please try again later or use another browser.";
            echo "<div/> <a href='index.php'>Homepage</a>";
    exit(0);
}
```

Figure 12: Part of the code used to prevent bots from using the site. (registerCheck.php).

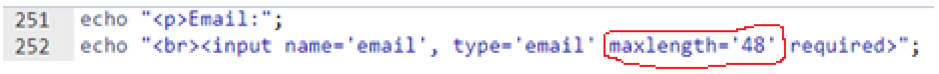
- All user Input on the page is sanitized to prevent Cross site scripting (XSS).



```php
//define variables from registration form
//filter vairables to prevent XS
//prevent signing up with same email with different cases
$email = strtolower(filter_var($_POST['email'], FILTER_SANITIZE_SPECIAL_CHARS));
$emailConfirm = strtolower(filter_var($_POST['emailConfirm'], FILTER_SANITIZE_SPECIAL_CHARS));
$password = filter_var($_POST['password'], FILTER_SANITIZE_SPECIAL_CHARS);
$passwordConfirm = filter_var($_POST['passwordConfirm'], FILTER_SANITIZE_SPECIAL_CHARS);
$name = filter_var($_POST['name'], FILTER_SANITIZE_SPECIAL_CHARS);
$number = filter_var($_POST['number'], FILTER_SANITIZE_SPECIAL_CHARS);
$securityAnswer = filter_var($_POST['securityAnswer'], FILTER_SANITIZE_SPECIAL_CHARS);
$securityAnswer2 = filter_var($_POST['securityAnswer2'], FILTER_SANITIZE_SPECIAL_CHARS);
$questionID = (int)filter_var($_POST['securityQuestion'], FILTER_SANITIZE_SPECIAL_CHARS);
```

Figure 13: Code showing Form information being sanitized to prevent XSS. (registerCheck.php)

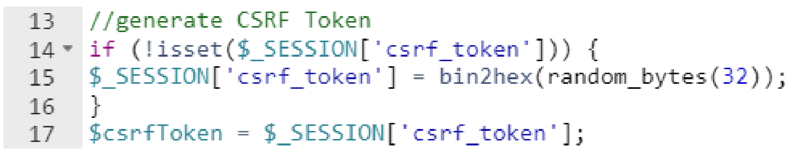
- All User input used meant for use with the database is length restricted as not to cause issues with SQL statements.



```php
echo "<p>Email:";
echo "<br><input name='email', type='email' maxlength='48' required>";
```

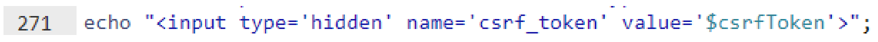
Figure 14: Code showing the input length restriction. (register.php)

- A Hidden CSRF Token is included in the form to prevent Cross-Site-Request-Forgery Attacks.



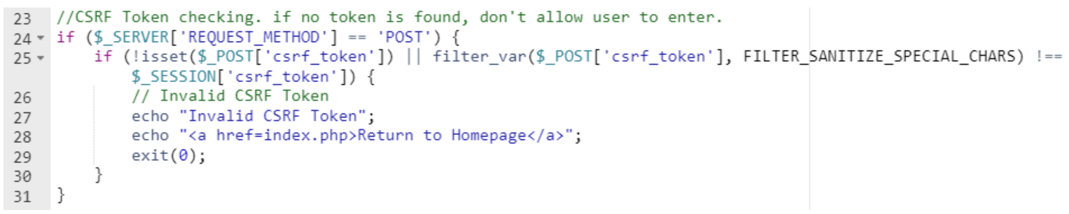
```php
//generate CSRF Token
if (!isset($_SESSION['csrf_token'])) {
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
$csrfToken = $_SESSION['csrf_token'];
```

Figure 15: Code of how CSRF Tokens are generated and stored. (register.php).



```php
echo "<input type='hidden' name='csrf_token' value='$csrfToken'>";
```

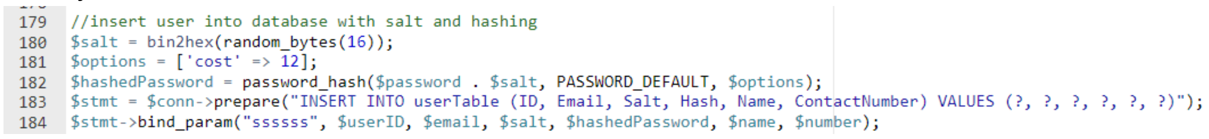
Figure 16: Code of how CSRF Tokens are inserted into forms (register.php).



```php
//CSRF Token checking. if no token is found, don't allow user to enter.
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (!isset($_POST['csrf_token']) || filter_var($_POST['csrf_token'], FILTER_SANITIZE_SPECIAL_CHARS) !==
        $_SESSION['csrf_token']) {
        // Invalid CSRF Token
        echo "Invalid CSRF Token";
        echo "<a href=index.php>Return to Homepage</a>";
        exit(0);
    }
}
```

Figure 17: Code of how CSRF Tokens are Verified (registerCheck.php).

- All database interactions are done using prepared statements and parameter binding, preventing SQL Injection.



```php
//insert user into database with salt and hashing
$salt = bin2hex(random_bytes(16));
$options = ['cost' => 12];
$hashedPassword = password_hash($password . $salt, PASSWORD_DEFAULT, $options);
$stmt = $conn->prepare("INSERT INTO userTable (ID, Email, Salt, Hash, Name, ContactNumber) VALUES (?, ?, ?, ?, ?, ?)");
$stmt->bind_param("ssssss", $userID, $email, $salt, $hashedPassword, $name, $number);
```

Figure 18: examples of prepared SQL Statements being used to prevent SQLi (registerCheck.php).

Database Table(s)

Upon registration, users are inserted into the database. There are 7 tables on the database, with 3 tables being relevant to registration.

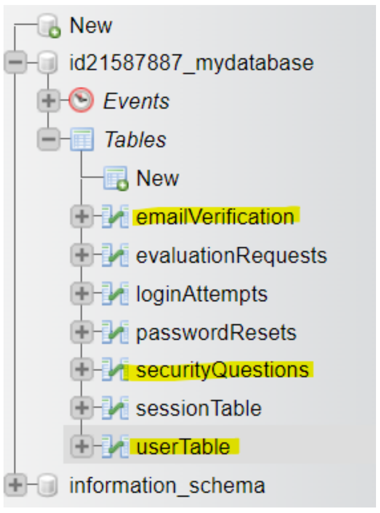
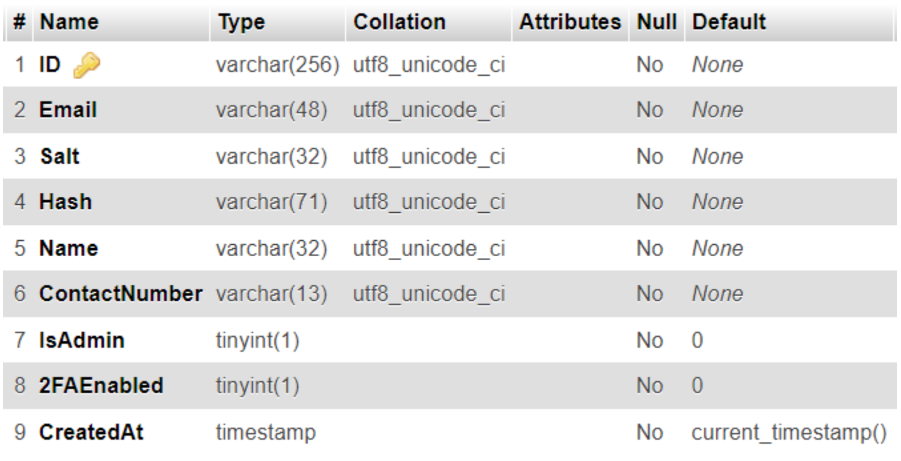


Figure 19: The structure of the site's Database, with the tables relevant to registration highlighted.

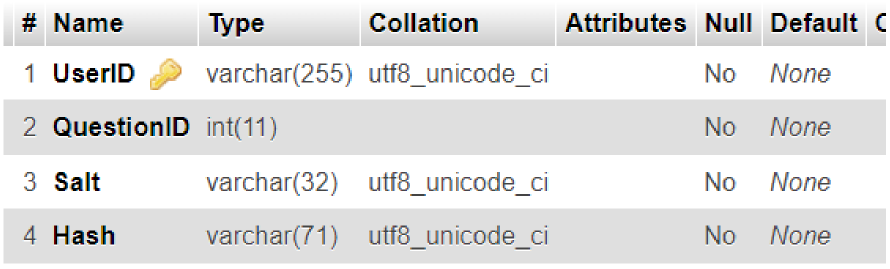
Table `userTable` is used to store the user's basic details.



#	Name	Type	Collation	Attributes	Null	Default
1	**ID** 🔑	varchar(256)	utf8_unicode_ci		No	*None*
2	**Email**	varchar(48)	utf8_unicode_ci		No	*None*
3	**Salt**	varchar(32)	utf8_unicode_ci		No	*None*
4	**Hash**	varchar(71)	utf8_unicode_ci		No	*None*
5	**Name**	varchar(32)	utf8_unicode_ci		No	*None*
6	**ContactNumber**	varchar(13)	utf8_unicode_ci		No	*None*
7	**IsAdmin**	tinyint(1)			No	0
8	**2FAEnabled**	tinyint(1)			No	0
9	**CreatedAt**	timestamp			No	current_timestamp()

Figure 20: The structure of `userTable`.

Table `securityQuestions` is used to store information about user security questions.

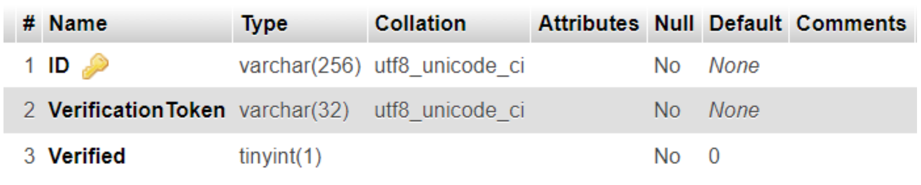


#	Name	Type	Collation	Attributes	Null	Default	C
1	UserID	varchar(255)	utf8_unicode_ci		No	*None*	
2	QuestionID	int(11)			No	*None*	
3	Salt	varchar(32)	utf8_unicode_ci		No	*None*	
4	Hash	varchar(71)	utf8_unicode_ci		No	*None*	

Figure 21: The structure of `securityQuestions`.

UserID is a foreign key referring to a user's ID in userTable.

Table `emailVerification` stores information relevant to email verification.

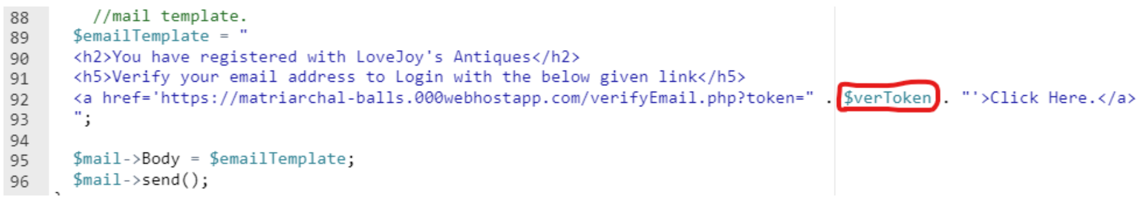


#	Name	Type	Collation	Attributes	Null	Default	Comments
1	ID	varchar(256)	utf8_unicode_ci		No	*None*	
2	VerificationToken	varchar(32)	utf8_unicode_ci		No	*None*	
3	Verified	tinyint(1)			No	0	

Figure 22: The structure of `emailVerification`.

ID is a foreign key referring to a user's ID in userTable.

VerificationToken is a one-time use token which is used for email verification. This token is contained in the link within the verification email which a user receives when they register.

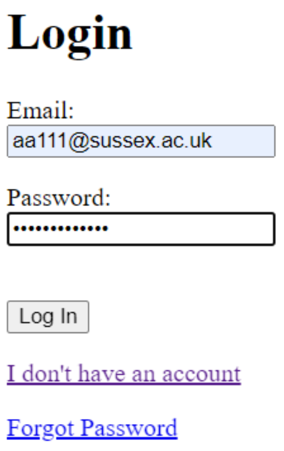


```
  //mail template.
$emailTemplate = "
<h2>You have registered with LoveJoy's Antiques</h2>
<h5>Verify your email address to Login with the below given link</h5>
<a href='https://matriarchal-balls.000webhostapp.com/verifyEmail.php?token="  . $verToken . "'>Click Here.</a>
";

$mail->Body = $emailTemplate;
$mail->send();
```

Figure 23: Part of the code for the Email which is sent to the user during registration. VerificationToken is circled in red. (registerCheck.php)

Task 2 – Secure Login Feature

Below is a screenshot of my site's login page.



Login

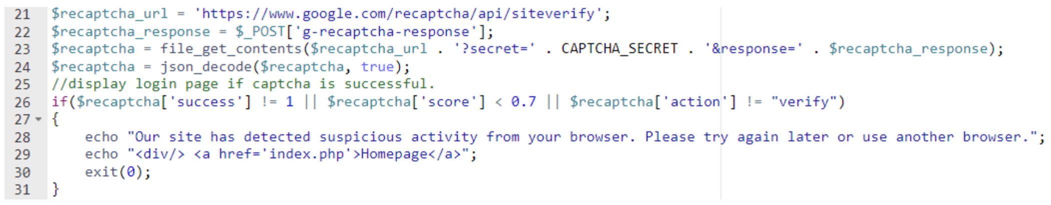
Email: aa111@sussex.ac.uk

Password: •••••••••••••

Figure 24: A screenshot of the site's Login form. (Filled)

Once a user inserts their email and password when logging in, the site does the following checks:

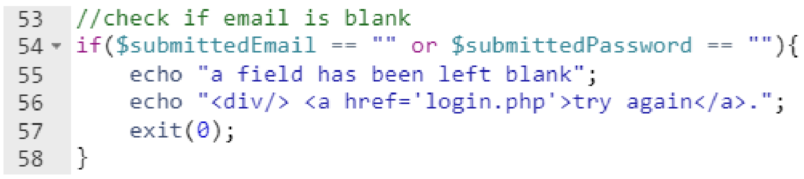
- An automatic background CAPTCHA Check using reCAPTCHA v3 to prevent bot attacks.



```php
$recaptcha_url = 'https://www.google.com/recaptcha/api/siteverify';
$recaptcha_response = $_POST['g-recaptcha-response'];
$recaptcha = file_get_contents($recaptcha_url . '?secret=' . CAPTCHA_SECRET . '&response=' . $recaptcha_response);
$recaptcha = json_decode($recaptcha, true);
//display login page if captcha is successful.
if($recaptcha['success'] != 1 || $recaptcha['score'] < 0.7 || $recaptcha['action'] != "verify")
{
    echo "Our site has detected suspicious activity from your browser. Please try again later or use another browser.";
    echo "<div/> <a href='index.php'>Homepage</a>";
    exit(0);
}
```

Figure 25: Part of the code used for CAPTCHA Verification. (loginCheck.php)

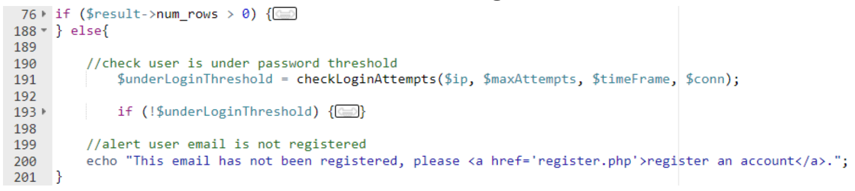
- A check to see if any input fields have been left blank.



```php
//check if email is blank
if($submittedEmail == "" or $submittedPassword == ""){
    echo "a field has been left blank";
    echo "<div/> <a href='login.php'>try again</a>.";
    exit(0);
}
```

Figure 26: Code checking for blank fields. (loginCheck.php)

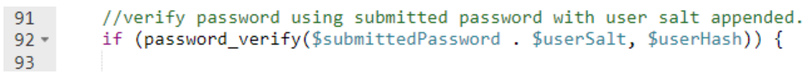
- A check to see if the user's IP is temporarily blocked from the site due to spam. (see below – Spam prevention)
    - A check to see if the user is registered to the site.



```php
if ($result->num_rows > 0) {⬚}
} else{

    //check user is under password threshold
        $underLoginThreshold = checkLoginAttempts($ip, $maxAttempts, $timeFrame, $conn);

        if (!$underLoginThreshold) {⬚}

    //alert user email is not registered
    echo "This email has not been registered, please <a href='register.php'>register an account</a>.";
}
```

Figure 27: Code checking user has been registered (loginCheck.php).

- A check to see if the user's password is correct.



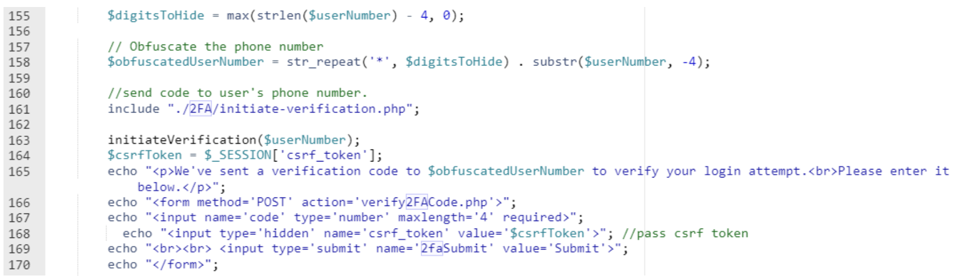
```php
    //verify password using submitted password with user salt appended.
    if (password_verify($submittedPassword . $userSalt, $userHash)) {
```

Figure 28: Code checking the user's password (loginCheck.php).

- A check to see if the user has 2 Factor Authentication activated.

Users which do not have 2FA activated are prompted to answer their security question when logging in. This is checked against the answer stored in the database. If the inputted answer matches the database's answer, the user is logged into the site.

- Those with 2FA activated are sent a text message via SMS to the number they registered with. (Using Sinch's SMS API).



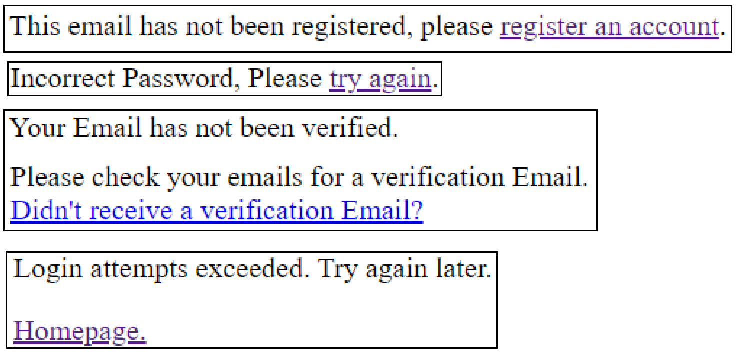
```php
	$digitsToHide = max(strlen($userNumber) - 4, 0);

	// Obfuscate the phone number
	$obfuscatedUserNumber = str_repeat('*', $digitsToHide) . substr($userNumber, -4);

	//send code to user's phone number.
	include "./2FA/initiate-verification.php";

	initiateVerification($userNumber);
	$csrfToken = $_SESSION['csrf_token'];
	echo "<p>We've sent a verification code to $obfuscatedUserNumber to verify your login attempt.<br>Please enter it
	    below.</p>";
	echo "<form method='POST' action='verify2FACode.php'>";
	echo "<input name='code' type='number' maxlength='4' required>";
	  echo "<input type='hidden' name='csrf_token' value='$csrfToken'>"; //pass csrf token
	echo "<br><br> <input type='submit' name='2faSubmit' value='Submit'>";
	echo "</form>";
```

Figure 29: Code which handles sending 2FA messages for login. (loginCheck.php)

- Entering the code from the SMS message sent will log the user into the site.

Note: This number used with 2FA is verified when 2FA is activated, so users can't easily lock themselves out of their accounts by inputting a number that isn't their own while registering.

If any of these checks are to fail, the site will not perform any other checks and display a relevant error message.



This email has not been registered, please register an account.

Incorrect Password, Please try again.

Your Email has not been verified.

Please check your emails for a verification Email.
Didn't receive a verification Email?

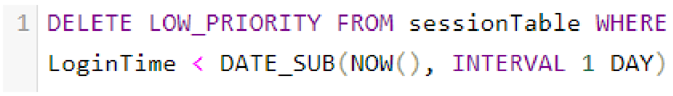
Login attempts exceeded. Try again later.

Homepage.

Figure 30: Examples of Error messages when logging in.

Accessing the site & Session Tokens

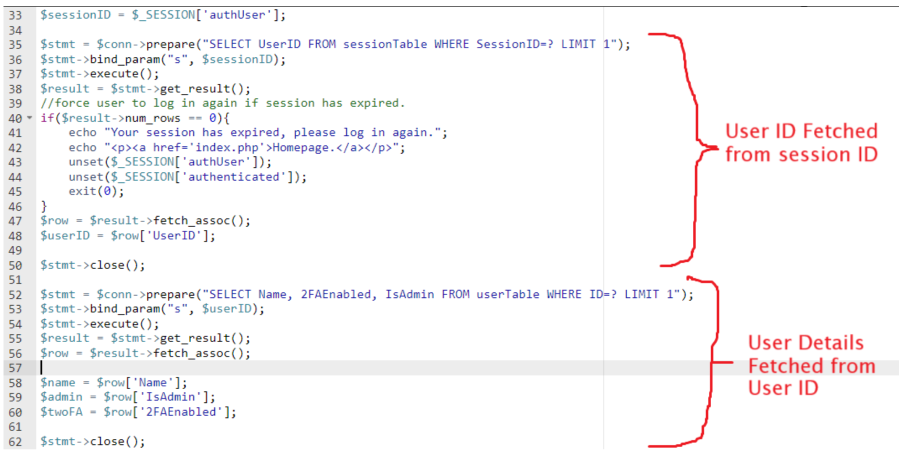
- When being granted access to the site, users are given a randomly generated session token which is valid for 1 Day after login.



```sql
DELETE LOW_PRIORITY FROM sessionTable WHERE
LoginTime < DATE_SUB(NOW(), INTERVAL 1 DAY)
```

Figure 31: The SQL Statement defining automatic session expiration Event.

- This session token is used to identify the user while using the site. All the user's details are fetched from this Session token, rather than storing the user's sensitive information in their session.

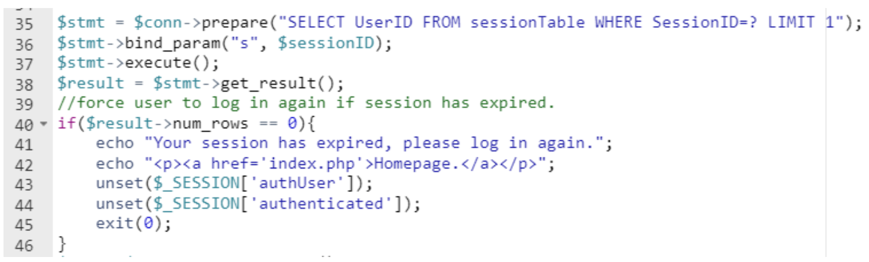


```php
$sessionID = $_SESSION['authUser'];

$stmt = $conn->prepare("SELECT UserID FROM sessionTable WHERE SessionID=? LIMIT 1");
$stmt->bind_param("s", $sessionID);
$stmt->execute();
$result = $stmt->get_result();
//force user to log in again if session has expired.
if($result->num_rows == 0){
    echo "Your session has expired, please log in again.";
    echo "<p><a href='index.php'>Homepage.</a></p>";
    unset($_SESSION['authUser']);
    unset($_SESSION['authenticated']);
    exit(0);
}
$row = $result->fetch_assoc();
$userID = $row['UserID'];

$stmt->close();

$stmt = $conn->prepare("SELECT Name, 2FAEnabled, IsAdmin FROM userTable WHERE ID=? LIMIT 1");
$stmt->bind_param("s", $userID);
$stmt->execute();
$result = $stmt->get_result();
$row = $result->fetch_assoc();

$name = $row['Name'];
$admin = $row['IsAdmin'];
$twoFA = $row['2FAEnabled'];

$stmt->close();
```

User ID Fetched from session ID
User Details Fetched from User ID

Figure 32: User information being fetched from a Session ID (dashboard.php)

- This session token is checked on every page which is accessible to logged in users. If the token does not correspond to a user, or has expired, the user will be asked to log in again.



```php
$stmt = $conn->prepare("SELECT UserID FROM sessionTable WHERE SessionID=? LIMIT 1");
$stmt->bind_param("s", $sessionID);
$stmt->execute();
$result = $stmt->get_result();
//force user to log in again if session has expired.
if($result->num_rows == 0){
    echo "Your session has expired, please log in again.";
    echo "<p><a href='index.php'>Homepage.</a></p>";
    unset($_SESSION['authUser']);
    unset($_SESSION['authenticated']);
    exit(0);
}
```

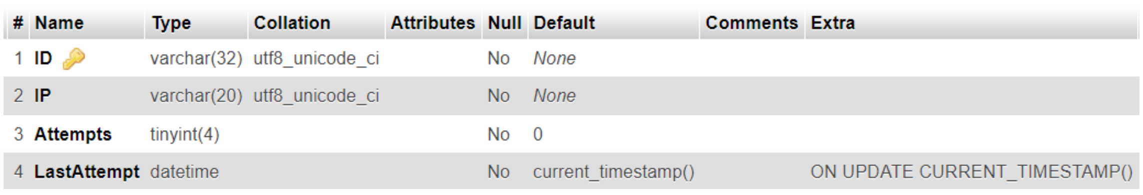
Figure 33: Code for a session token check. (dashboard.php)

Spam Prevention

When users insert an incorrect password, their IP is logged, and a counter is created to keep track of how many incorrect attempts they have made.

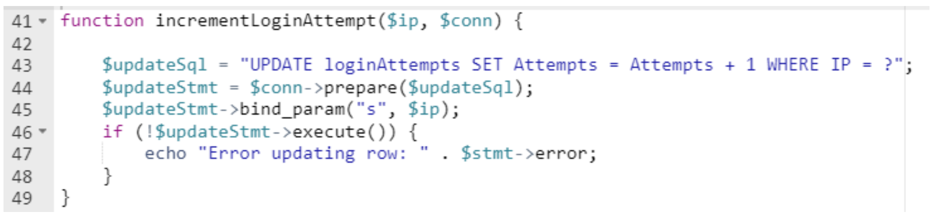
5 incorrect attempts will block a user's IP from making more login attempts for 10 minutes.

This is tracked using the `loginAttempts` Table in the database and functions found in utils.php.



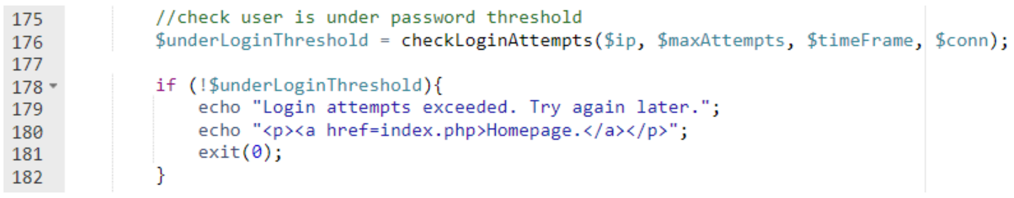
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**ID**	varchar(32)	utf8_unicode_ci		No	*None*		
2	**IP**	varchar(20)	utf8_unicode_ci		No	*None*		
3	**Attempts**	tinyint(4)			No	0		
4	**LastAttempt**	datetime			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()

Figure 34: The structure of `loginAttempts`



```php
function incrementLoginAttempt($ip, $conn) {

    $updateSql = "UPDATE loginAttempts SET Attempts = Attempts + 1 WHERE IP = ?";
    $updateStmt = $conn->prepare($updateSql);
    $updateStmt->bind_param("s", $ip);
    if (!$updateStmt->execute()) {
        echo "Error updating row: " . $stmt->error;
    }
}
```

Figure 35: Code for inserting a new login attempt (utils.php).

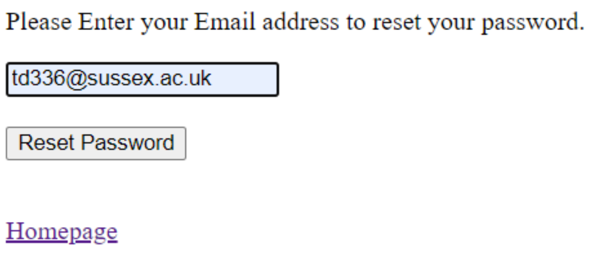


```php
    //check user is under password threshold
    $underLoginThreshold = checkLoginAttempts($ip, $maxAttempts, $timeFrame, $conn);

    if (!$underLoginThreshold){
        echo "Login attempts exceeded. Try again later.";
        echo "<p><a href=index.php>Homepage.</a></p>";
        exit(0);
    }
```

Figure 36: code showing login attempts being checked during login (loginCheck.php).

Task 3 – Password Strength and Password Recovery

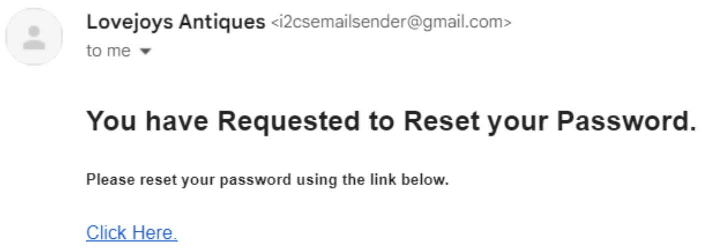
If a user has forgotten their password, they are able to reset their password by requesting a ‘password reset’ link through an email.



Please Enter your Email address to reset your password.

td336@sussex.ac.uk

Figure 37: A screenshot of the page used for sending password reset requests.



Lovejoys Antiques <i2csemailsender@gmail.com>
to me

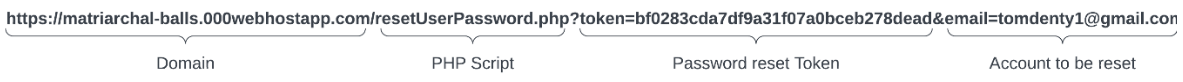
You have Requested to Reset your Password.

Please reset your password using the link below.

Click Here.

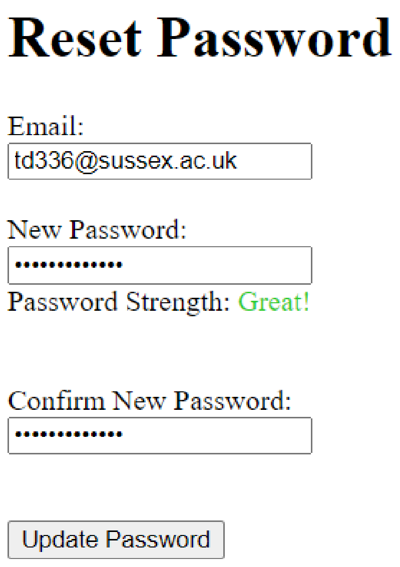
Figure 38: An email containing a password reset link.

The link contained within this email is valid for 10 minutes after being sent to prevent unauthorised access. The token is unique to that user. Clicking this link allows the user to insert a new password using a form.



https://matriarchal-balls.000webhostapp.com/resetUserPassword.php?token=bf0283cda7df9a31f07a0bceb278dead&email=tomdenty1@gmail.con

Figure 39: An example of a password reset link.



Reset Password

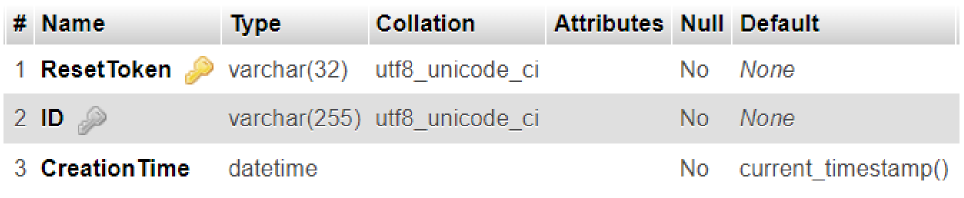
Email:
td336@sussex.ac.uk

New Password:
•••••••••••••
Password Strength: Great!

Confirm New Password:
•••••••••••••

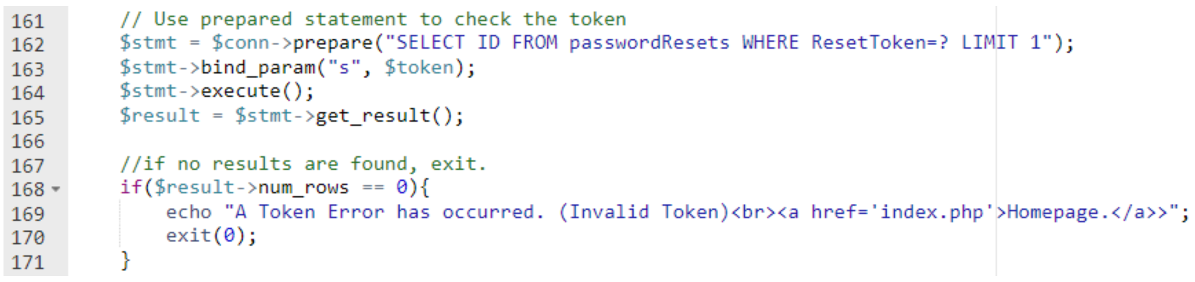
Figure 40: The password reset form. (filled)

Data about password resets are stored in the \`passwordResets\` table, where ID is a foreign key of ID in userTable.



#	Name	Type	Collation	Attributes	Null	Default
1	**ResetToken**	varchar(32)	utf8_unicode_ci		No	*None*
2	**ID**	varchar(255)	utf8_unicode_ci		No	*None*
3	**CreationTime**	datetime			No	current_timestamp()

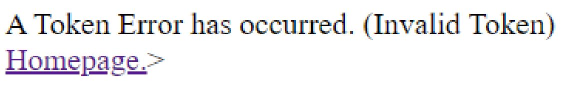
Figure 41: The structure of \`passwordResets\`.



```
// Use prepared statement to check the token
$stmt = $conn->prepare("SELECT ID FROM passwordResets WHERE ResetToken=? LIMIT 1");
$stmt->bind_param("s", $token);
$stmt->execute();
$result = $stmt->get_result();

//if no results are found, exit.
if($result->num_rows == 0){
    echo "A Token Error has occurred. (Invalid Token)<br><a href='index.php'>Homepage.</a>>";
    exit(0);
}
```

Figure 42: Code for checking if a given token is valid or not.

Altering the link to change the token being used will result in an error when the form is submitted. The ‘Email’ field in the form is purely aesthetic, and changing it will not cause any security issues.



A Token Error has occurred. (Invalid Token)
Homepage.>

Figure 43: An Error message shown if the inputted token is invalid.

Changing the user’s password generates a new Salt for their Password for additional security.

A Password strength policy has been implemented onto both the registration page and the password reset page. Users are encouraged to follow tips to make their passwords more secure and are given a rating based on how secure their password is, from 0 to 5.

The security rating is based off the following factors:

- Passwords should have a length of 12 characters or more.
- Passwords should have uppercase and lowercase characters.
- Passwords should contain a number.
- Passwords should contain a special character.

Requirements fulfilled	**Security Rating**
1	Bad
2	Bad
3	Okay
4	Good
5	Excellent

Table 1: Numbers of requirements fulfilled and their corresponding 'security rating'.

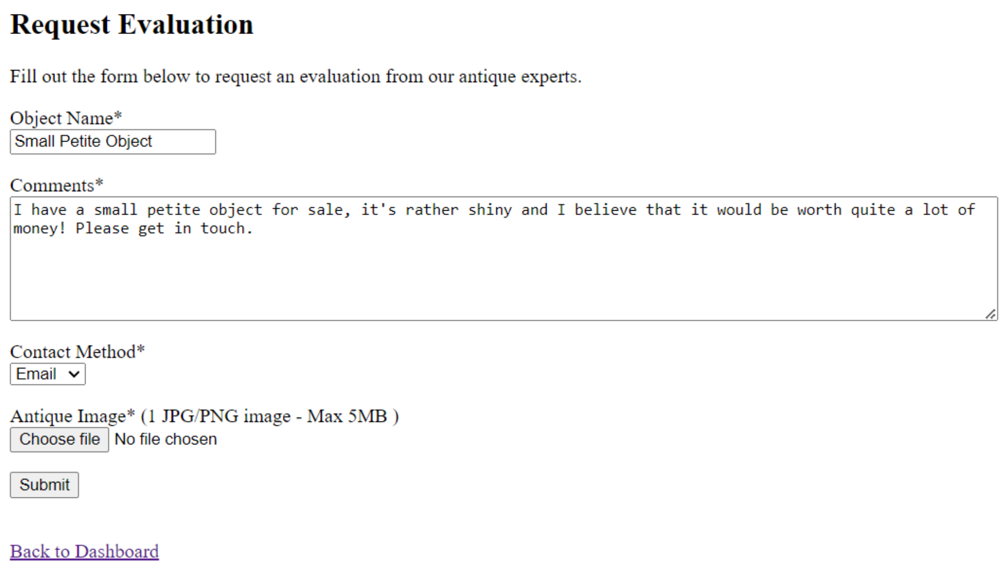
- Passwords must "Okay" or better to allow a user to register or reset their password.
- Passwords with a length of less than 6 characters are rejected entirely, regardless of the password's scoring.

The implementation of this policy can be seen on lines 45 -236 in register.php.

Task 4 – Evaluation Request page

Non-admin users can request items for evaluation using the site's 'request evaluation' page.

This form includes all the required fields from the coursework spec. Below is a screenshot of it.



Request Evaluation

Fill out the form below to request an evaluation from our antique experts.

Object Name*
Small Petite Object

Comments*
I have a small petite object for sale, it's rather shiny and I believe that it would be worth quite a lot of money! Please get in touch.

Contact Method*
Email

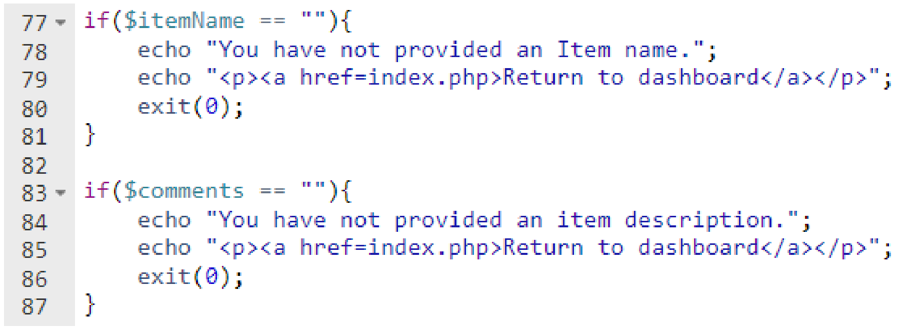
Antique Image* (1 JPG/PNG image - Max 5MB)
Choose file No file chosen

Submit

Back to Dashboard

Figure 44: A Screenshot of the site's Evaluation Request Form.

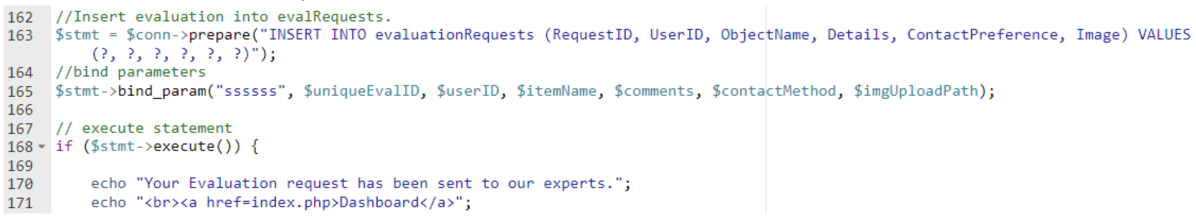
- The form is checked to make sure that users haven't left any fields empty.



```
if($itemName == ""){
    echo "You have not provided an Item name.";
    echo "<p><a href=index.php>Return to dashboard</a></p>";
    exit(0);
}

if($comments == ""){
    echo "You have not provided an item description.";
    echo "<p><a href=index.php>Return to dashboard</a></p>";
    exit(0);
}
```

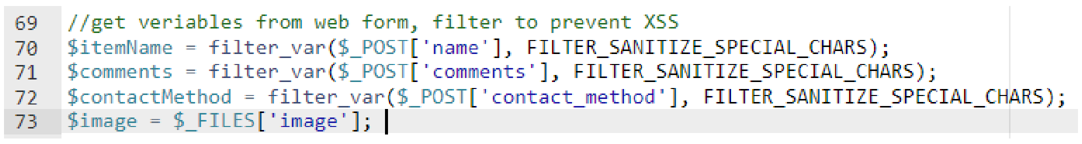
Figure 45: Code showing Fields being checked for being empty. (sendEvaluation.php).

- Like all other forms on this site, limits have been put on input fields as not to cause problems with SQL Varchar limits and all user input is Sanitized to prevent XSS, and all SQL statements are also prepared and have bound parameters to prevent SQL Injection. CSRF Tokens have also been used and are checked to prevent CSRF Attacks.



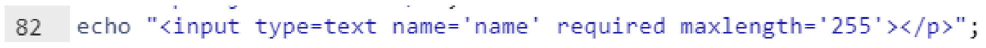
```
//Insert evaluation into evalRequests.
$stmt = $conn->prepare("INSERT INTO evaluationRequests (RequestID, UserID, ObjectName, Details, ContactPreference, Image) VALUES
    (?, ?, ?, ?, ?, ?)");
//bind parameters
$stmt->bind_param("ssssss", $uniqueEvalID, $userID, $itemName, $comments, $contactMethod, $imgUploadPath);

// execute statement
if ($stmt->execute()) {

    echo "Your Evaluation request has been sent to our experts.";
    echo "<br><a href=index.php>Dashboard</a>";
```

Figure 46: SQLi Prevention (sendEvaluation.php)



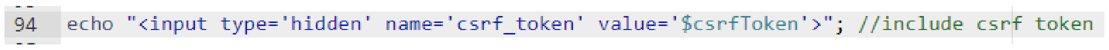
```
//get veriables from web form, filter to prevent XSS
$itemName = filter_var($_POST['name'], FILTER_SANITIZE_SPECIAL_CHARS);
$comments = filter_var($_POST['comments'], FILTER_SANITIZE_SPECIAL_CHARS);
$contactMethod = filter_var($_POST['contact_method'], FILTER_SANITIZE_SPECIAL_CHARS);
$image = $_FILES['image'];
```

Figure 47: XSS Prevention (sendEvaluation.php)



```
echo "<input type=text name='name' required maxlength='255'></p>";
```

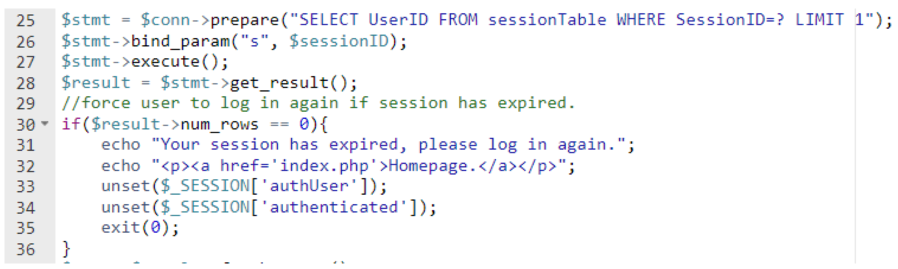
Figure 48: Input Length Limiting (requestEvaluation.php



```
echo "<input type='hidden' name='csrf_token' value='$csrfToken'>"; //include csrf token
```

Figure 49: CSRF Attack Limiting (requestEvaluation.php)

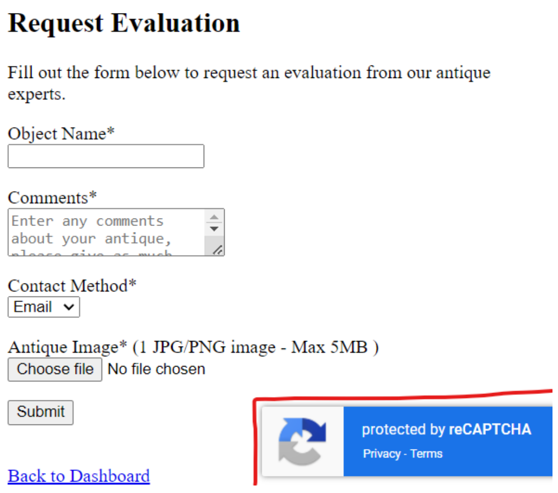
- Requests cannot be sent without a valid Session ID.



```
$stmt = $conn->prepare("SELECT UserID FROM sessionTable WHERE SessionID=? LIMIT 1");
$stmt->bind_param("s", $sessionID);
$stmt->execute();
$result = $stmt->get_result();
//force user to log in again if session has expired.
if($result->num_rows == 0){
    echo "Your session has expired, please log in again.";
    echo "<p><a href='index.php'>Homepage.</a></p>";
    unset($_SESSION['authUser']);
    unset($_SESSION['authenticated']);
    exit(0);
}
```

Figure 50: Code showing Session ID Checking. (sendEvaluation.php)

- reCAPTCHA v3 is used on this page to prevent evaluations from being attacked by bots.



Request Evaluation

Fill out the form below to request an evaluation from our antique experts.

Object Name*

Comments*

Enter any comments about your antique, please give as much

Contact Method*

Email

Antique Image* (1 JPG/PNG image - Max 5MB)

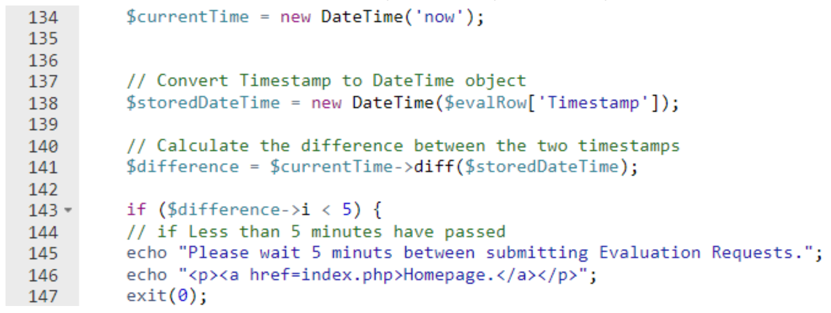
Choose file No file chosen

Submit

Back to Dashboard

Figure 51: reCAPTCHA v3's widget on the evaluation request page.

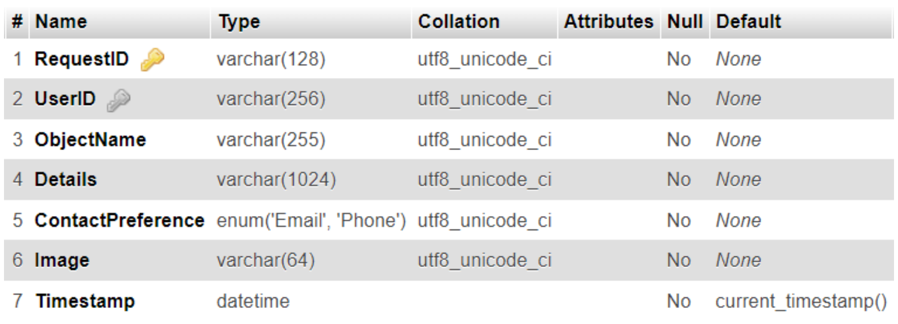
- There is a 5-minute timeout on evaluation requests to prevent spam.



```php
$currentTime = new DateTime('now');


// Convert Timestamp to DateTime object
$storedDateTime = new DateTime($evalRow['Timestamp']);

// Calculate the difference between the two timestamps
$difference = $currentTime->diff($storedDateTime);

if ($difference->i < 5) {
// if Less than 5 minutes have passed
echo "Please wait 5 minuts between submitting Evaluation Requests.";
echo "<p><a href=index.php>Homepage.</a></p>";
exit(0);
```

Figure 52: Code showing the Request Timeout System (sendEvaluation.php)

After all checks have bene performed, the evaluation is inserted into the `evaluationRequests` Table, which stores all information regarding evaluation requests.

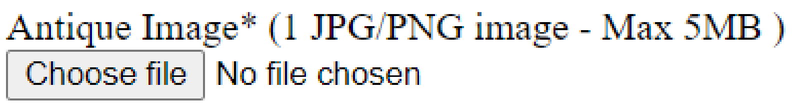


#	Name	Type	Collation	Attributes	Null	Default
1	**RequestID**	varchar(128)	utf8_unicode_ci		No	*None*
2	**UserID**	varchar(256)	utf8_unicode_ci		No	*None*
3	**ObjectName**	varchar(255)	utf8_unicode_ci		No	*None*
4	**Details**	varchar(1024)	utf8_unicode_ci		No	*None*
5	**ContactPreference**	enum('Email', 'Phone')	utf8_unicode_ci		No	*None*
6	**Image**	varchar(64)	utf8_unicode_ci		No	*None*
7	**Timestamp**	datetime			No	current_timestamp()

Figure 53: Structure of the `evaluationRequests` Table.

Task 5 – Picture submission

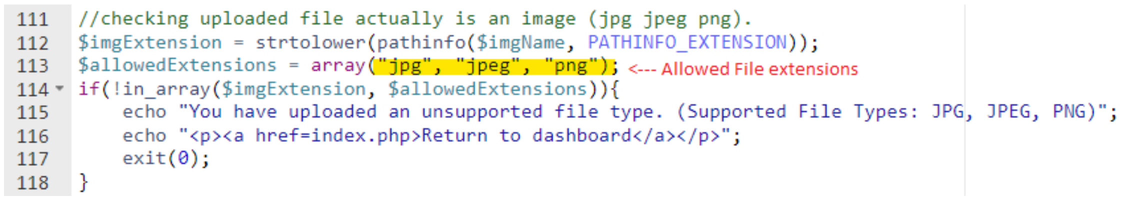
The evaluation request form has a required box for an image file of the antique.



Antique Image* (1 JPG/PNG image - Max 5MB)

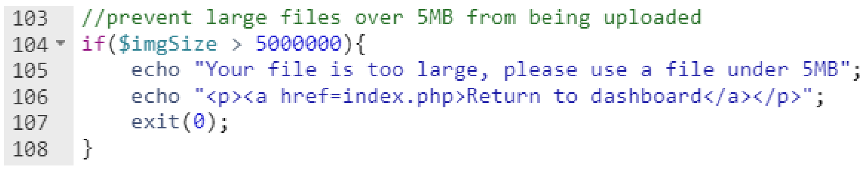
Figure 54: The image selection part of the form.

- There are both frontend and backend checks to ensure that the size and file type of the file are appropriate. There is a 5MB restriction on file size and only JPG and PNG image types are allowed.



```
//checking uploaded file actually is an image (jpg jpeg png).
$imgExtension = strtolower(pathinfo($imgName, PATHINFO_EXTENSION));
$allowedExtensions = array("jpg", "jpeg", "png"); <--- Allowed File extensions
if(!in_array($imgExtension, $allowedExtensions)){
    echo "You have uploaded an unsupported file type. (Supported File Types: JPG, JPEG, PNG)";
    echo "<p><a href=index.php>Return to dashboard</a></p>";
    exit(0);
}
```

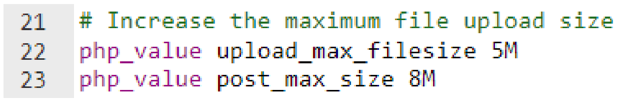
*Figure 55: Code showing Backend file **type** restrictions (sendEvaluation.php).*



```
//prevent large files over 5MB from being uploaded
if($imgSize > 5000000){
    echo "Your file is too large, please use a file under 5MB";
    echo "<p><a href=index.php>Return to dashboard</a></p>";
    exit(0);
}
```

*Figure 56: Code showing backend File **size** restrictions (sendEvaluation.php)*

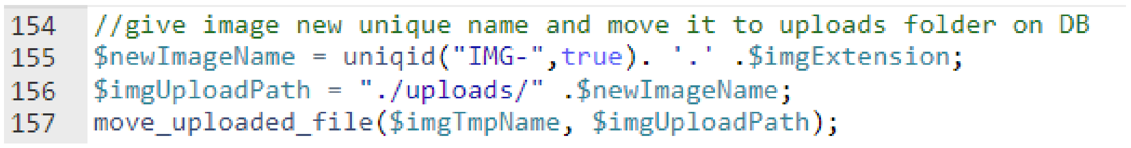
- An additional restriction in the site's configuration restricts the maximum file size and POST value to prevent any file size issues.



```
# Increase the maximum file upload size
php_value upload_max_filesize 5M
php_value post_max_size 8M
```

Figure 57: Size restrictions in the site's configuration (.htaccess)

- All uploaded images are given random, high entropy names and stored in an uploads folder. These high entropy names will make it difficult for attackers to guess the file names.



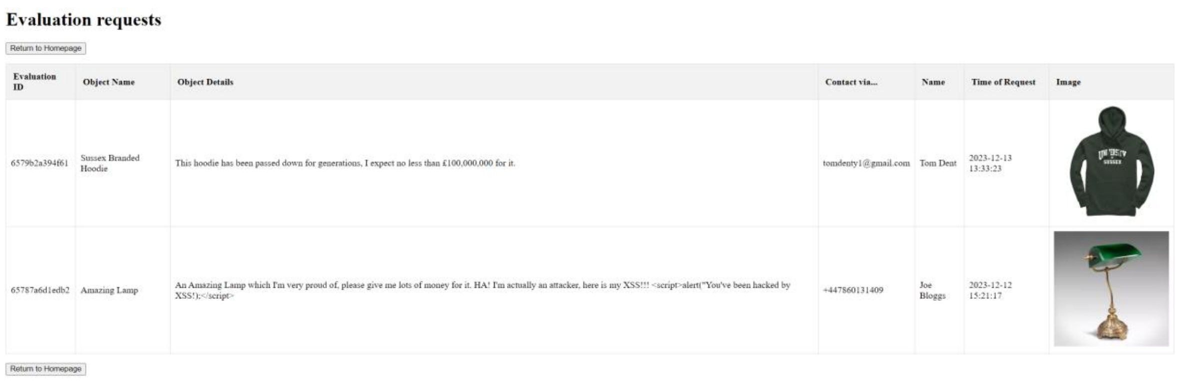
```
//give image new unique name and move it to uploads folder on DB
$newImageName = uniqid("IMG-",true). '.' .$imgExtension;
$imgUploadPath = "./uploads/" .$newImageName;
move_uploaded_file($imgTmpName, $imgUploadPath);
```

Figure 58: Code showing the random image naming system.

Rather than being stored in the database itself, images are stored in a folder on the web server and are fetched from the stored directory on the database.

Task 6 – Evaluation Request Listings page

Users who have isAdmin set to 1 on the Table `userTable` can access the list of evaluation requests.



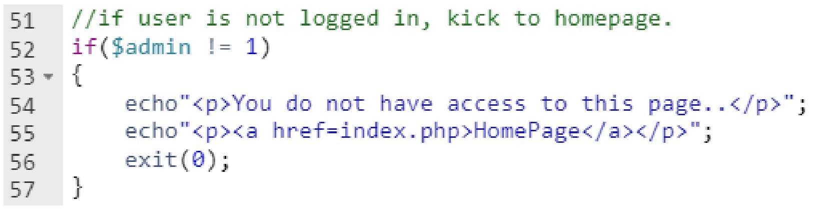
Evaluation requests

Evaluation ID	Object Name	Object Details	Contact via...	Name	Time of Request	Image
6579b2a394f61	Sussex Branded Hoodie	This hoodie has been passed down for generations, I expect no less than £100,000,000 for it.	tomdenty1@gmail.com	Tom Dent	2023-12-13 13:33:23	
65787a6d1edb2	Amazing Lamp	An Amazing Lamp which I'm very proud of, please give me lots of money for it. HA! I'm actually an attacker, here is my XSS!!! <script>alert("You've been hacked by XSS!);</script>	+447860131409	Joe Bloggs	2023-12-12 15:21:17	

Figure 59: A Screenshot of the Evaluation Listings page.

Each request contains the relevant information which an admin would need.

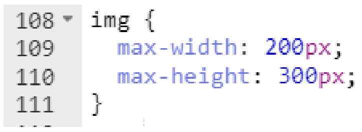
- All input to this page is already sanitized when it's inserted into the database, so no issues should arise regarding XSS.
- The SQL statement to read from the database is prepared to prevent SQLi.
- CSRF Attacks aren't an issue as no forms are used between the dashboard and Evaluations list page.
- Non-admin users will automatically be redirected back to the homepage, as will non-logged in users.



```php
//if user is not logged in, kick to homepage.
if($admin != 1)
{
    echo"<p>You do not have access to this page..</p>";
    echo"<p><a href=index.php>HomePage</a></p>";
    exit(0);
}
```

Figure 60: Code used for checking Admin Privileges (viewEvaluations.php)

- All images will be resized to have a max width and height of 200px to ensure that large images don't break the screen.



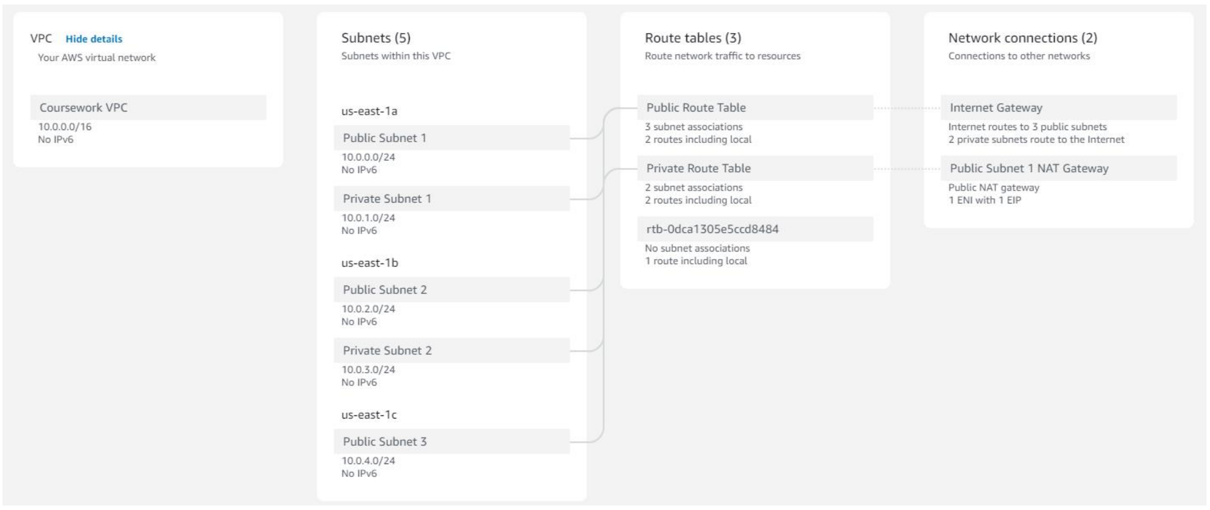
```css
img {
  max-width: 200px;
  max-height: 300px;
}
```

Figure 61: Image Width & Height Restrictions (viewEvaluations.php)

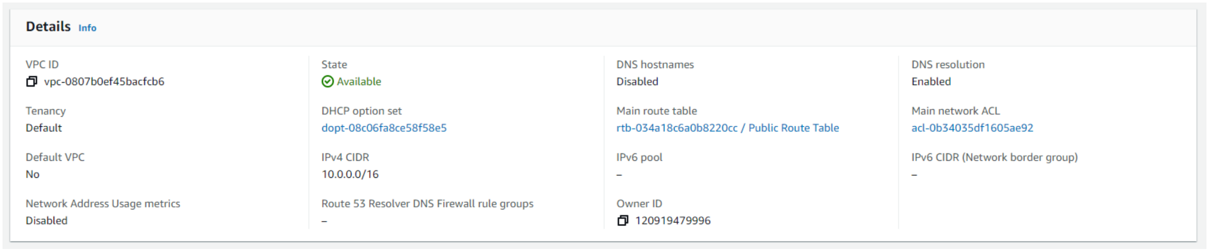
Task 7 – AWS VPC

Below are screenshots of my AWS VPC Resource map and VPC Configuration. I'm unsure if this is everything I need to include, so I have included some additional screenshots.

VPC Resource Map



VPC Details

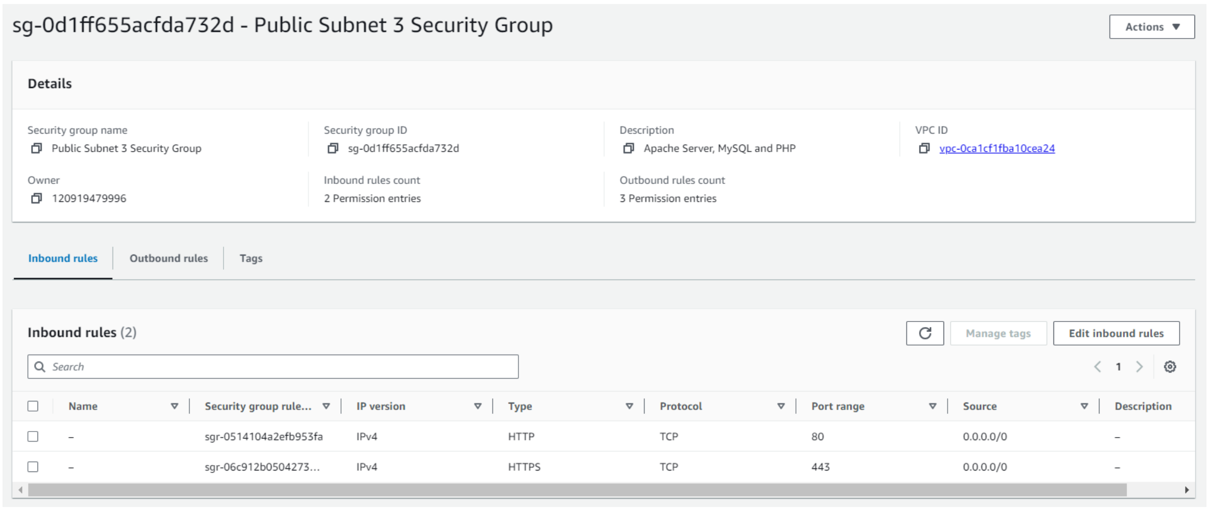


Details Info

VPC ID vpc-0807b0ef45bacfcb6	State Available	DNS hostnames Disabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-08c06fa8ce58f58e5	Main route table rtb-034a18c6a0b8220cc / Public Route Table	Main network ACL acl-0b34035df1605ae92
Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool –	IPv6 CIDR (Network border group) –
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups –	Owner ID 120919479996	

Additional Screenshots

Security Groups



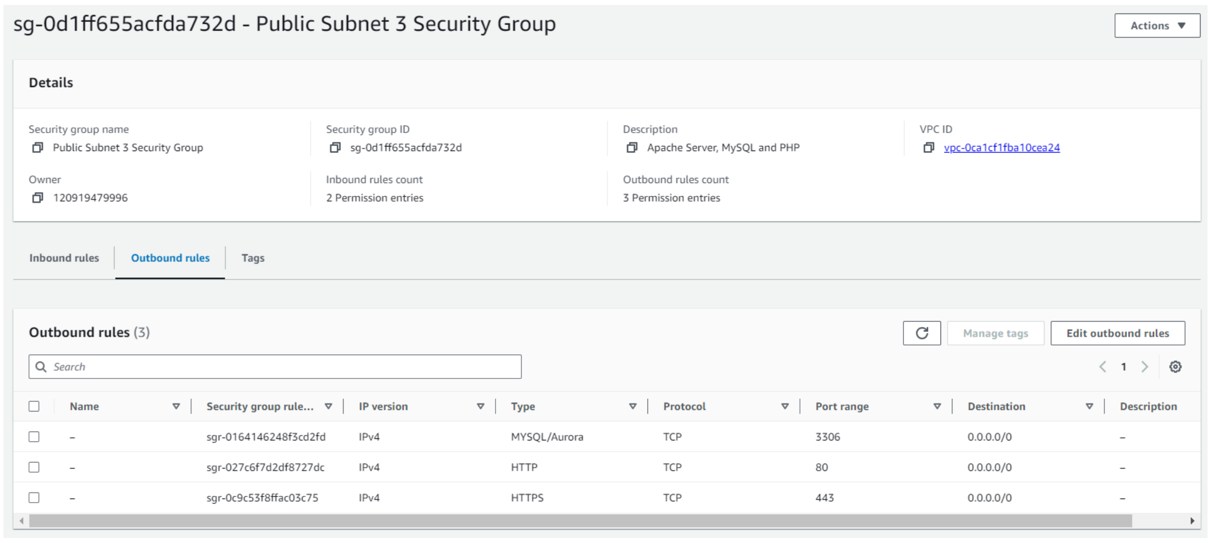
sg-0d1ff655acfda732d - Public Subnet 3 Security Group

Details

Security group name Public Subnet 3 Security Group	Security group ID sg-0d1ff655acfda732d	Description Apache Server, MySQL and PHP	VPC ID vpc-0ca1cf1fba10cea24
Owner 120919479996	Inbound rules count 2 Permission entries	Outbound rules count 3 Permission entries	

Inbound rules (2)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
–	sgr-0514104a2efb953fa	IPv4	HTTP	TCP	80	0.0.0.0/0	–
–	sgr-06c912b0504273...	IPv4	HTTPS	TCP	443	0.0.0.0/0	–

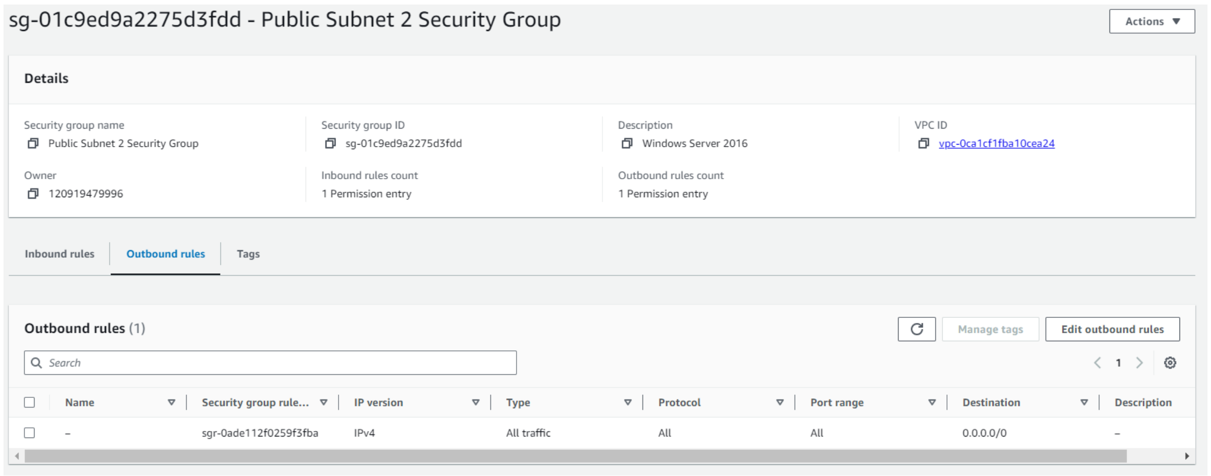

sg-0d1ff655acfda732d - Public Subnet 3 Security Group

Details

Security group name	Security group ID	Description	VPC ID
Public Subnet 3 Security Group	sg-0d1ff655acfda732d	Apache Server, MySQL and PHP	vpc-0ca1cf1fba10cea24
Owner	Inbound rules count	Outbound rules count	
120919479996	2 Permission entries	3 Permission entries	

Outbound rules (3)

Name	Security group rule...	IP version	Type	Protocol	Port range	Destination	Description
–	sgr-0164146248f3cd2fd	IPv4	MYSQL/Aurora	TCP	3306	0.0.0.0/0	–
–	sgr-027c6f7d2df8727dc	IPv4	HTTP	TCP	80	0.0.0.0/0	–
–	sgr-0c9c53f8ffac03c75	IPv4	HTTPS	TCP	443	0.0.0.0/0	–

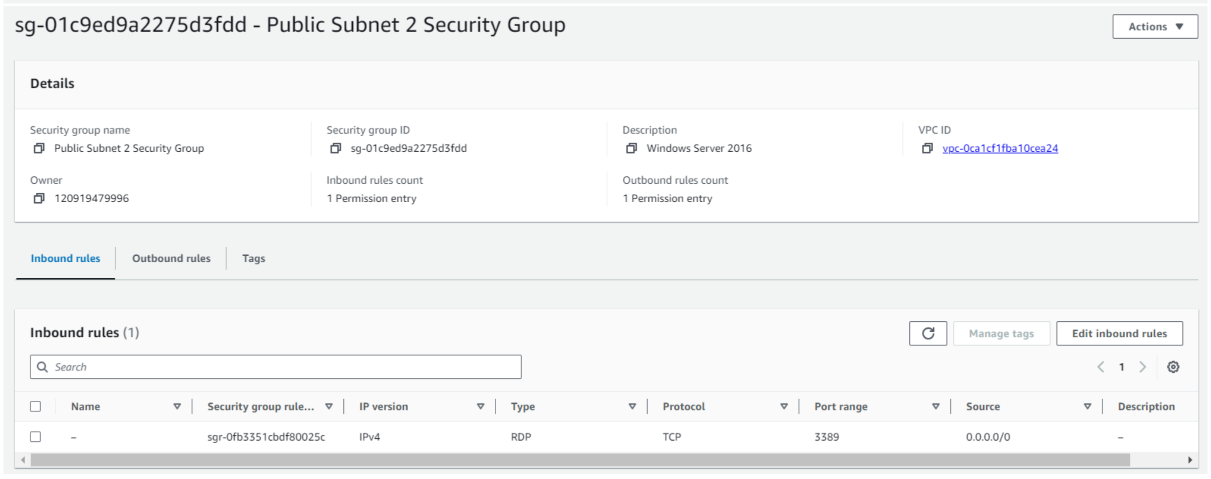

sg-01c9ed9a2275d3fdd - Public Subnet 2 Security Group

Details

Security group name	Security group ID	Description	VPC ID
Public Subnet 2 Security Group	sg-01c9ed9a2275d3fdd	Windows Server 2016	vpc-0ca1cf1fba10cea24
Owner	Inbound rules count	Outbound rules count	
120919479996	1 Permission entry	1 Permission entry	

Outbound rules (1)

Name	Security group rule...	IP version	Type	Protocol	Port range	Destination	Description
–	sgr-0ade112f0259f3fba	IPv4	All traffic	All	All	0.0.0.0/0	–


sg-01c9ed9a2275d3fdd - Public Subnet 2 Security Group

Details

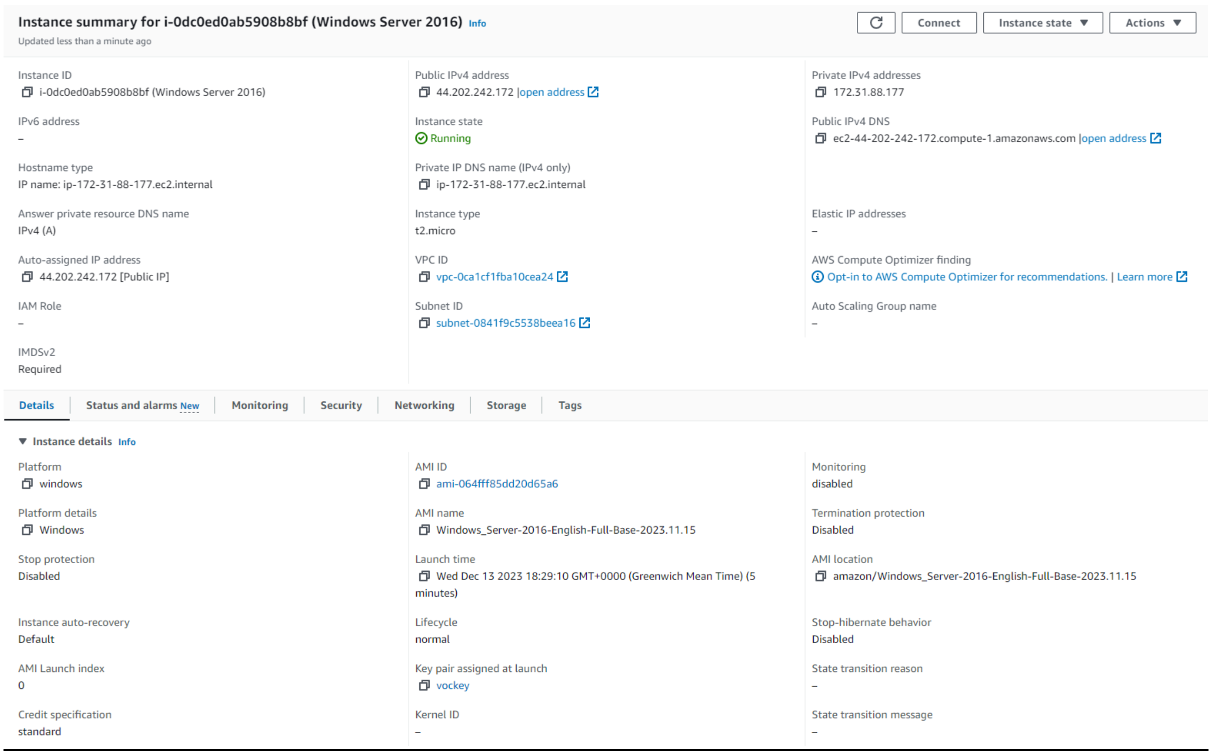
Security group name	Security group ID	Description	VPC ID
Public Subnet 2 Security Group	sg-01c9ed9a2275d3fdd	Windows Server 2016	vpc-0ca1cf1fba10cea24
Owner	Inbound rules count	Outbound rules count	
120919479996	1 Permission entry	1 Permission entry	

Inbound rules (1)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
–	sgr-0fb3351cbdf80025c	IPv4	RDP	TCP	3389	0.0.0.0/0	–

Instances

Windows Server 2016



Instance summary for i-0dc0ed0ab5908b8bf (Windows Server 2016) Info

Updated less than a minute ago

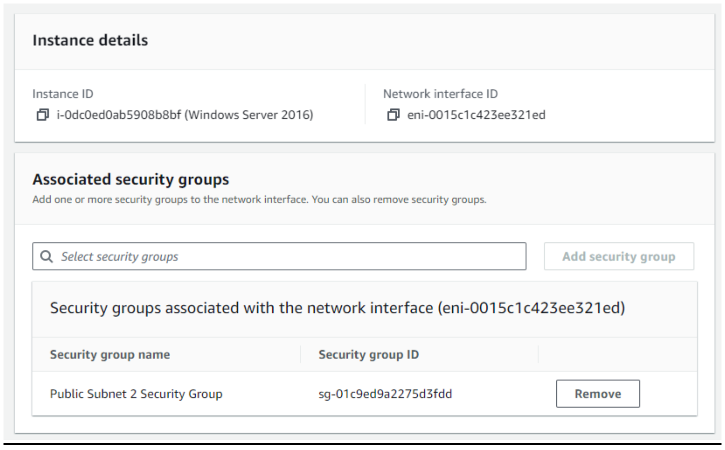
Connect | Instance state ▼ | Actions ▼

Instance ID i-0dc0ed0ab5908b8bf (Windows Server 2016)	Public IPv4 address 44.202.242.172 |open address	Private IPv4 addresses 172.31.88.177
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-44-202-242-172.compute-1.amazonaws.com |open address
Hostname type IP name: ip-172-31-88-177.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-88-177.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 44.202.242.172 [Public IP]	VPC ID vpc-0ca1cf1fba10cea24	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-0841f9c5538beea16	Auto Scaling Group name –
IMDSv2 Required		

Details | Status and alarms New | Monitoring | Security | Networking | Storage | Tags

▼ Instance details Info

Platform windows	AMI ID ami-064fff85dd20d65a6	Monitoring disabled
Platform details Windows	AMI name Windows_Server-2016-English-Full-Base-2023.11.15	Termination protection Disabled
Stop protection Disabled	Launch time Wed Dec 13 2023 18:29:10 GMT+0000 (Greenwich Mean Time) (5 minutes)	AMI location amazon/Windows_Server-2016-English-Full-Base-2023.11.15
Instance auto-recovery Default	Lifecycle normal	Stop-hibernate behavior Disabled
AMI Launch index 0	Key pair assigned at launch vockey	State transition reason –
Credit specification standard	Kernel ID –	State transition message –



Instance details

Instance ID i-0dc0ed0ab5908b8bf (Windows Server 2016)	Network interface ID eni-0015c1c423ee321ed

Associated security groups

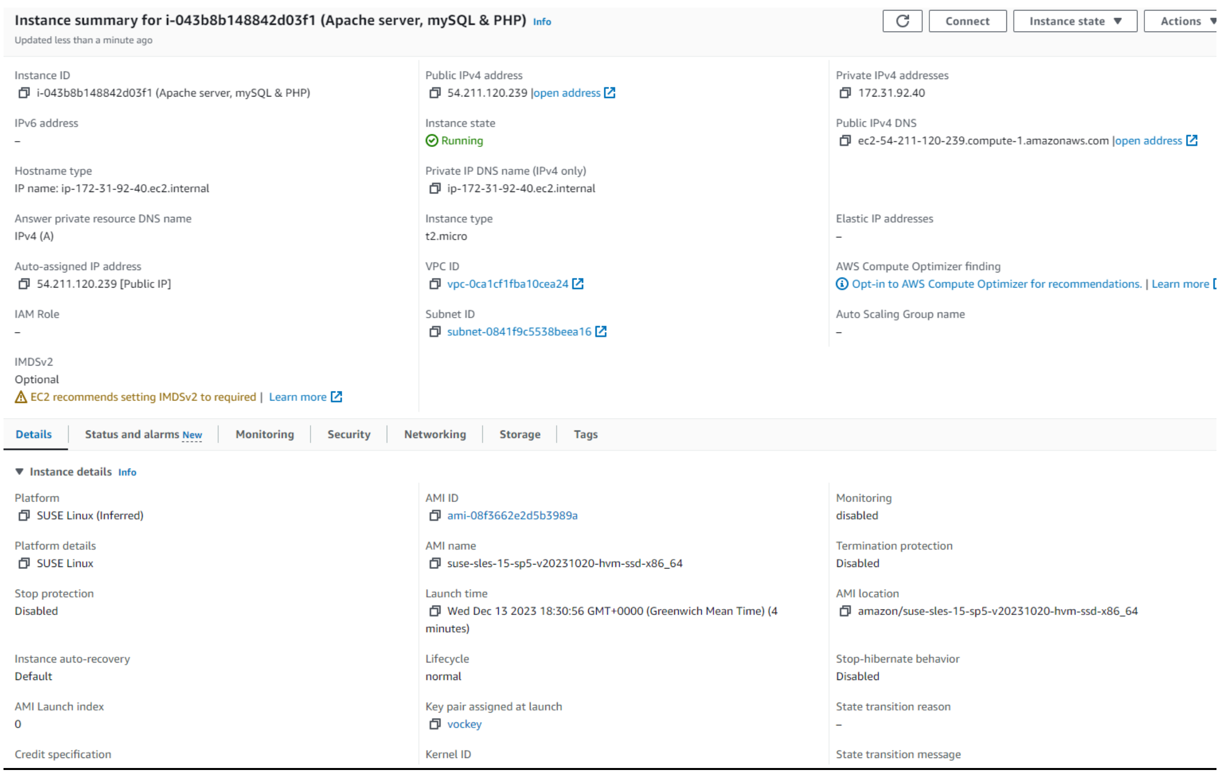
Add one or more security groups to the network interface. You can also remove security groups.

Select security groups | Add security group

Security groups associated with the network interface (eni-0015c1c423ee321ed)

Security group name	Security group ID	
Public Subnet 2 Security Group	sg-01c9ed9a2275d3fdd	Remove

Apache, SQL & PHP Server



Instance summary for i-043b8b148842d03f1 (Apache server, mySQL & PHP) Info

Updated less than a minute ago

Connect | Instance state ▼ | Actions ▼

Instance ID
i-043b8b148842d03f1 (Apache server, mySQL & PHP)

IPv6 address
–

Hostname type
IP name: ip-172-31-92-40.ec2.internal

Answer private resource DNS name
IPv4 (A)

Auto-assigned IP address
54.211.120.239 [Public IP]

IAM Role
–

IMDSv2
Optional
⚠ EC2 recommends setting IMDSv2 to required | Learn more

Public IPv4 address
54.211.120.239 |open address

Instance state
Running

Private IP DNS name (IPv4 only)
ip-172-31-92-40.ec2.internal

Instance type
t2.micro

VPC ID
vpc-0ca1cf1fba10cea24

Subnet ID
subnet-0841f9c5538beea16

Private IPv4 addresses
172.31.92.40

Public IPv4 DNS
ec2-54-211-120-239.compute-1.amazonaws.com |open address

Elastic IP addresses
–

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Auto Scaling Group name
–

Details | Status and alarms New | Monitoring | Security | Networking | Storage | Tags

▼ Instance details Info

Platform
SUSE Linux (Inferred)

Platform details
SUSE Linux

Stop protection
Disabled

Instance auto-recovery
Default

AMI Launch index
0

Credit specification

AMI ID
ami-08f3662e2d5b3989a

AMI name
suse-sles-15-sp5-v20231020-hvm-ssd-x86_64

Launch time
Wed Dec 13 2023 18:30:56 GMT+0000 (Greenwich Mean Time) (4 minutes)

Lifecycle
normal

Key pair assigned at launch
vockey

Kernel ID

Monitoring
disabled

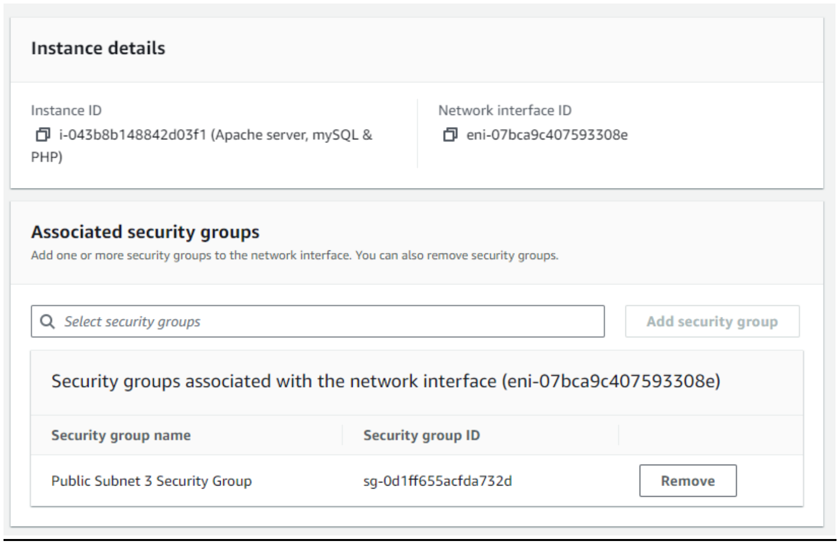
Termination protection
Disabled

AMI location
amazon/suse-sles-15-sp5-v20231020-hvm-ssd-x86_64

Stop-hibernate behavior
Disabled

State transition reason
–

State transition message



Instance details

Instance ID
i-043b8b148842d03f1 (Apache server, mySQL & PHP)

Network interface ID
eni-07bca9c407593308e

Associated security groups

Add one or more security groups to the network interface. You can also remove security groups.

Select security groups | Add security group

Security groups associated with the network interface (eni-07bca9c407593308e)

Security group name	Security group ID	
Public Subnet 3 Security Group	sg-0d1ff655acfda732d	Remove