Candidate 234558

Introduction to Computer Security Report

Table of Contents

Application URL: https://danbat.es/cs/uni-compsec/

Candidate 234558

OneDrive Code Link: https://universityofsussex-my.sharepoint.com/:f:/g/personal/db524_sussex_ac_uk/EimWh_nx3m9CgqAwG_RXjVoB8lkWxMkeSvy_myAmX7dL9A?e=cVfXZ6

Test Users:

These both have 2FA disabled for your use in marking, you can test 2FA by default on the site.

Admin Privileges (can see listings)

Username: admin Password: admin1

Basic User

Username: user1 Password: user

Site-wide Protections

Overall my project may be considered more secure as it is self-hosted rather than using a service like 000webhost, so the only people able to view and edit systems are myself and my hosting provider Dreamhost.

The following are code excerpts from checkSession() a function run before everything on each page, from helper.php

Session Hijacking

```
//if IP Address doesn't match, or User Agent doesn't match -> Same Origin Policy
if ($_SERVER['REMOTE_ADDR'] != $_SESSION['ipaddress'] || $_SERVER['HTTP_USER_AGENT']
!= $_SESSION['useragent']) {
    quitSession();
}

//check the last time the session was used and quit it if over an hour ago
if (time() > ($_SESSION['lastaccess'] + 3600)) {
    quitSession();
} else {
    $_SESSION['lastaccess'] = time();
}
```

Cross-Site-Scripting

```
header("X-XSS-Protection: 1; mode=block"); //avoid XSS by blocking the page if it is detected
//set content security policy for local content & hcaptcha. Building a string as header() doesn't handle
newlines
$cspHeader = "content-security-policy: default-src 'self'; img-src 'self'; child-src https://hcaptcha.com
https://*.hcaptcha.com;";
$cspHeader .= "script-src 'self' https://hcaptcha.com https://*.hcaptcha.com frame-src
https://hcaptcha.com https://*.hcaptcha.com ";
$cspHeader .= "style-src 'self' https://hcaptcha.com https://*.hcaptcha.com connect-src
https://hcaptcha.com https://*.hcaptcha.com";
header($cspHeader); //Content security policy, avoid XSS attacks. But also allow hcaptcha to function
```

Cross Site Request Forgery

My project has just one external reference, HCaptcha (who should be reasonably trustworthy given the service they provide), and all packages (CSS & JS courtesy of Bootstrap) are stored locally to avoid CSRF from bad actors posing as content delivery networks.

Task 0: Self-Reflection

See following landscape pages

	234558			
Excellent (10-9 marks) **Student must have gone beyond**	**Good (8-6 marks)**	**Average (5-3 marks)**	**Poor (2-0 marks)**	**Criteria**
Policy has no flaw, and its implementation is excellent. Various mechanisms implemented to ensure password policy is secure.	Policy has no flaws, but implementation of policy is simple.	Password policy has very few flaws. However, different sections of policy are implemented and working.	Policy has many flaws for example password is not encrypted, and no salt applied. Password forgot policy has security flaws.	**Password policy 10marks** Password entropy, encrypted storage, security questions and recovery of password Implemented all the required features
Several countermeasures are implemented, and the quality of countermeasures are excellent.	Countermeasures are implemented in all the pages however quality of implementation is simple.	Implemented countermeasures only in some parts of the application.	Very little effort to implement countermeasures to avoid these vulnerabilities.	**Vulnerabilities 10 marks** SQL injection, XSS, CSRF, File Upload and any other obvious vulnerability. Implemented countermeasures for all the vulnerabilities but did not go beyond module resources
All the requirements are implemented to authenticate users. Implementation quality is excellent.	All requirements are implemented to authenticate the user. However, quality of implementation is simple.	Only some obvious requirements are not implemented.	Lots of obvious authentication's requirements are not implemented.	**Authentication 10 marks** User identity management (registration and login etc), Email verification for registration, 2 factor authentications (PIN and or email), Implemented all the required features
Excellent implementation of countermeasures against these attacks.	No flaws in countermeasures however quality of implementation is simple.	Some flaws in countermeasures	Very little effort against these attacks.	**Obfuscation/Common attacks 10 marks** Brute force attack – Number of attempts Botnet attack – Captcha Dictionary attack/Rainbow table attack Implemented all the common attacks
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.	**Other security features like confidentiality of important information 10 marks** For example, identify information that needs to be stored as encrypted. Attempted but only encrypted passwords
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.	**Deeper understanding, two extra web security 10 marks** Carry out your investigation and implement two more security features. These need to be complex and challenging one. Attempted one-time recovery codes and a recovery email address

Candidate 234558

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

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

Task 1: User Registration

Registration Feature Code (registration.php)

```php
<?php
set_include_path('/home/danbates/uni-compsec-back/');
include('helper.php');
checkSession();
//Registration

function registrationForm(): string
{
    return '
    <form class="mb-4 ml-5 mr-5" action="register.php" method="post">
    <div class="form-group p-1">
        <label for="displayName">Preferred Name (What should we call you?) <small>(Required)
Maximum 30 characters</small></label>
        <input class="form-control" name="displayName" required id="displayName" type="text"
placeholder="John Smith" maxlength="30" size="10">
    </div>
    <div class="form-group p-1">
        <label for="username">Username <small>(Required) Maximum 30 characters</small></label>
        <input class="form-control" name="username" required id="username" type="text"
placeholder="Username" maxlength="30" size="30">
    </div>
    <div class="form-group p-1">
        <label for="password">Password <small>(Required)<br>MUST contain at least 16 characters
including uppercase, lowercase, numerical, and special characters. Maximum 30
characters</small></label>
        <input class="form-control" name="password" required id="password" type="text"
placeholder="Password" pattern="'.getPasswordRegexJS().'" maxlength="30" size="30">
    </div>
    <div class="form-group p-1">
        <label for="emailAddress">Contact Email Address <small>(Required)</small></label>
        <input class="form-control" name="emailAddress" required id="emailAddress" type="email"
placeholder="name@provider.domain" maxlength="254" size="30">
    </div>
    <div class="form-group p-1">
        <label for="phoneNumber">Contact Phone Number <small>(Required) 8-14
Characters</small></label>
        <input class="form-control" name="phoneNumber" required id="phoneNumber" type="tel"
placeholder="+44 7123456789" pattern="'.getPhoneRegexJS().'" maxlength="14" size="14">
    </div>
    <div class="form-group p-1">
        <div class="h-captcha" data-sitekey="22353c5a-05f5-4f2f-9b3d-a644670de95c"></div>
    </div>
    <div class="d-flex justify-content-center">
        <button class="btn btn-primary m-1" type="submit" name="submit">Register</button>
        <a class="btn btn-secondary m-1" type="button" href="signin.php">Cancel</a>
    </div>
    </form>
    ';
}

//variable to store the html we're going to output all at once to avoid outputting before we can send
different headers (for redirects)
$htmlOut = '';

if (isset($_POST['submit'])) { //if submitted
```

```
    if (verifyHCaptcha()) {
        $displayName = sanitise($_POST['displayName']);
        $username = sanitise($_POST['username']);
        $password = sanitise($_POST['password']);
        $emailAddress = sanitise($_POST['emailAddress']);
        $phoneNumber = sanitise($_POST['phoneNumber']);

        $passwordHash = password_hash($password, getHashingAlgo());

        //attributes that reasonably should be unique. excludes display name for obvious reason,
        //and password as with random salts there could be a conflict
        $uniqueAttributes = ["Username" => $username, "EmailAddress" => $emailAddress,
"PhoneNumber" => $phoneNumber];
        $db = getDatabase();
        $allUnique = True;
        foreach ($uniqueAttributes as $name=>$value) {
            $query = $db->prepare("SELECT * FROM 'Users' WHERE '$name' = ?");
            $query->bindValue(1, $value);
            $queryResult = $query->execute();
            if ($queryResult->fetchArray(SQLITE3_ASSOC) === False) { //Query returns False if it fails.
                unset($uniqueAttributes[$name]); //remove attribute from the array for the case where
another conflicts
            } else { //If not False, then there is a conflicting attribute
                $allUnique = False;
            }
        }

        //boolean that is only true if all entered attributes are in the correct specified format
        $correctFormat = strlen($username) <= 30 &&
            strlen($password) <= 30 &&
            preg_match(passwordRegex, $password) && //checks if password is >16, so we don't need to
do that a second time
            filter_var($emailAddress, FILTER_VALIDATE_EMAIL) &&
            strlen($displayName) > 0 && strlen($displayName) <= 30 &&
            preg_match(phoneRegex, $phoneNumber)
        ;

        if (count($uniqueAttributes) !== 0) {
            $numberOfThings = count($uniqueAttributes) > 1 ? 'a few things': 'one thing';
            $htmlOut .= '<h3 id="alert alert-warning">Almost there! Just '.$numberOfThings.' to fix:';
            $htmlOut .= '<ul>';
            foreach ($uniqueAttributes as $name=>$value) {
                switch ($name) {
                    case 'Username':
                        $htmlOut .= "<li>The Username: $value is already in use.</li>";
                        break;
                    case 'EmailAddress':
                        $htmlOut .= "<li>The Email Address: $value is attached to another account on our
system.</li>";
                        break;
                    case 'PhoneNumber':
                        $htmlOut .= "<li>The Phone Number: $value is already attached to an account on our
system.</li>";
                        break;
                }
            }
            $htmlOut .= '</ul>';
            $htmlOut .= '</h3>';
        } else if (!$correctFormat) {
            $htmlOut .= '<h3 class="alert alert-warning">Please ensure all fields are the correct length and
format.</h3>';
        } else { //if everything that must be unique is unique -> register them
            $activationKey = generateRandomKey();
            $activationURL = 'https://danbat.es/cs/uni-
compsec/activate.php?activationKey='.$activationKey;
            $_SESSION['activationKey'] = $activationKey;
            $_SESSION['username'] = $username; //to identify the user on activation
            $_SESSION['displayName'] = $displayName; // for activation email
            $_SESSION['emailAddress'] = $emailAddress; // for activation email
            $activated = 0; //Activation Boolean

            $registrationQuery = $db->prepare("INSERT INTO 'Users' ('Username', 'Password',
'EmailAddress', 'PhoneNumber', 'DisplayName', 'Activated')
        VALUES (:un, :pw, :em, :pn, :dn, :ac)");
            $registrationQuery->bindValue(':un', $username);
            $registrationQuery->bindValue(':pw', $passwordHash);
            $registrationQuery->bindValue(':em', $emailAddress);
            $registrationQuery->bindValue(':pn', $phoneNumber);
            $registrationQuery->bindValue(':dn', $displayName);
            $registrationQuery->bindValue(':ac', $activated);

            $registrationQueryResult = $registrationQuery->execute();
            if ($registrationQueryResult !== False) { //if didn't fail

                $htmlOut .= '<h1>You\'re registered!</h1>';
                $htmlOut .= sendActivationEmail($displayName, $activationURL, $emailAddress);
                $htmlOut .= "<h2><a href='home.php'>Redirecting in 5 seconds...</a></h2>";
                header("refresh:5;url=home.php"); //redirect user back home in 5 seconds
            } else {
                $htmlOut .= genericErrorMessage();
            }
        }
    } else { //if hcaptcha verification fails
        $htmlOut .= "<h1>Sorry, we couldn't verify that you're a human, please try again later</h1>";
    }

    unset($_POST['submit']); //make sure resubmissions cannot occur
} else {
    $htmlOut .= registrationForm();
}
echo pageTop();
echo $htmlOut;
echo hCaptchaJS();
echo pageBottom();
```

Account Activation Code (activate.php)

```
<?php
set_include_path('/home/danbates/uni-compsec-back/');
include('helper.php');
checkSession();

function requestNewKey(): string
{
    return '<h3>Looks like your activation key has expired! <a
href="activate.php?requestNewKey">Click here to request a new one to your saved email
address!</a></h3>';
}
$htmlOut = ''; //string to hold html we'll output at the end so that we can still modify headers
beforehand
```

```php
if (isset($_GET['activationKey']) && isset($_SESSION['activationKey']) && sanitise($_GET['activationKey']) === $_SESSION['activationKey']) {
    $registrationQuery = getDatabase()->prepare("UPDATE Users SET Activated = 1 WHERE UserID == ?");
    $registrationQuery->bindValue(1, getUserID());

    $registrationQueryResult = $registrationQuery->execute();
    if ($registrationQueryResult) { //if didn't fail
        $_SESSION['activated'] = True;
        $htmlOut .= '<h3>Your account is now activated - you can now <a href="signin.php">sign in</a></h3>';
        header("refresh:10;location:home.php"); //redirect to home 10s after activation
    } else {
        $_SESSION['activated'] = False;
        $htmlOut .= "<h3>We couldn't activate your account right now, please try again later.</h3>";
    }
} else if (isset($_GET['requestNewKey'])) {
    $activationKey = generateRandomKey();
    $activationURL = 'https://danbat.es/cs/uni-compsec/activate.php?activationKey='.$activationKey;
    $_SESSION['activationKey'] = $activationKey;
    sendActivationEmail($_SESSION['displayName'], $activationURL, $_SESSION['emailAddress']);
    $htmlOut .= '<h3>New activation key sent! You should receive a link via email to activate your account soon.</h3>';
} else {
    $htmlOut .= requestNewKey();
}

echo pageTop();
echo $htmlOut;
echo pageBottom();
```

Database Tables

Users
CREATE TABLE Users (UserID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL ON CONFLICT ROLLBACK DEFAULT (-1) UNIQUE, Username STRING UNIQUE ON CONFLICT ROLLBACK NOT NULL, Password NOT NULL, EmailAddress STRING UNIQUE NOT NULL, PhoneNumber TEXT UNIQUE NOT NULL, DisplayName STRING NOT NULL, Created DATETIME DEFAULT (CURRENT_TIMESTAMP) NOT NULL, Activated BOOLEAN DEFAULT (0) NOT NULL, TwoFactor BOOLEAN NOT NULL DEFAULT (1), RecoveryEmail STRING,

```
    RecoveryCodes STRING,
    Q1          INTEGER,
    Q2          INTEGER,
    Q3          INTEGER,
    A1          STRING,
    A2          STRING,
    A3          STRING
);
```

Listings

```
CREATE TABLE Listings (
    ListingID    INTEGER PRIMARY KEY AUTOINCREMENT
                    NOT NULL,
    UserID       INTEGER REFERENCES Users (UserID)
                    NOT NULL,
    Comments     TEXT    NOT NULL,
    PhoneOrEmail BOOLEAN DEFAULT (1)
                    NOT NULL,
    ImageName    STRING
);
```

Listings stores all the listing data

Admins

```
CREATE TABLE Admins (
    UserID INTEGER REFERENCES Users (UserID) MATCH SIMPLE
              NOT NULL
);
```

Admins is a dumb table that just stores UserIDs of Users who are allowed admin privileges in the application – Users can be added and removed from here at the behest of the Sysadmin.

Why I think it’s secure

- Inputs validated both on the html form and in php after posting for the correct format (SQL Injection)
- All inputs are sanitised (SQL Injection)
- SQL statements are prepared and not directly written (SQL Injection)
- Uses a HCaptcha to protect against bots
- Any unexpected behaviour results in failure and no information is divulged
- User has to activate the account via the email provided, and then has to sign in doing a HCaptcha to access anything on the site

Task 2: Develop a secure login feature

Login code (signin.php)

```php
<?php
set_include_path('/home/danbates/uni-compsec-back/');
include('helper.php');

```

```php
checkSession();

function signInForm(): string
{
    return '
<form class="mb-4 ml-5 mr-5" action="signin.php" method="post">
    <div class="form-group p-1">
        <label for="username">Username</label>
        <input class="form-control" name="username" required id="username" type="text"
placeholder="Your Username">
    </div>
    <div class="form-group p-1">
        <label for="password">Password</label>
        <input class="form-control" name="password" required id="password" type="text"
placeholder="Your Password">
    </div>
    <span>First time? <a href="register.php">Register Now</a>.</span>
    <br>
    <span><a href="forgotPassword.php">Forgot your password?</a></span>
    <div class="d-flex justify-content-center">
        <button class="btn btn-primary m-1" type="submit" name="submit">Sign In</button>
        <a class="btn btn-secondary m-1" type="button" href="home.php">Cancel</a>
    </div>
</form>
';
}

function incorrectCredentials(): string
{
    return '<h1>Incorrect Username or Password!</h1>';
}

function attemptsLimit(): string
{
    return '
    <h1>The failed sign-in attempts limit has been hit for this user. Try again in an hour.</h1>
    ';
}

$htmlOut = ''; //variable to store the html we're going to output all at once to avoid outputting before we
can send different headers (for redirects)

if (isset($_SESSION['signFail']) && $_SESSION['signFail']) {
    $htmlOut .= incorrectCredentials();
    if (isset($_SESSION['signAttempts'])) {
        if ($_SESSION['signAttempts'] > 5) {
            $htmlOut .= attemptsLimit(); //if user fails 5x, lock them out until their session expires (in an
hour)
        } else {
            $_SESSION['signAttempts']++;
            $htmlOut .= signInForm();
        }
    } else {
        $_SESSION['signAttempts'] = 1; //define
        $htmlOut .= signInForm();
    }
    unset($_SESSION['signFail']);
} else if (isset($_POST['submit'])) { //if submitted
    if ($_POST['username'] <> '' && $_POST['password'] <> '') {
        //local variables for inserting into db
        $username = htmlspecialchars($_POST['username'], ENT_HTML5);
        $password = htmlspecialchars($_POST['password'], ENT_HTML5);
        //sanitized special characters to html
        // -> allows use of special characters in username and password while avoiding escape
characters

        $passwordHash = password_hash($password, PASSWORD_BCRYPT);

        $db = getDatabase();

        $attributesQuery = $db->prepare("SELECT * FROM Users where Username = ?");
        $attributesQuery->bindValue(1, $username);
        $attributesQueryResult = $attributesQuery->execute();
        $userAttributesArray = $attributesQueryResult->fetchArray(SQLITE3_ASSOC);

        $storedPasswordHash = $userAttributesArray['Password'];

        if (password_verify($password, $storedPasswordHash)) { //if correct password
            //Store relevant variables in the session
            $_SESSION['username'] = $username;
            $_SESSION['passwordHash'] = $passwordHash;

            //additional attributes to be used on other pages
            $_SESSION['userid'] = $userAttributesArray['UserID'];
            $_SESSION['displayName'] = $userAttributesArray['DisplayName'];
            $_SESSION['emailAddress'] = $userAttributesArray['EmailAddress'];
            $_SESSION['phoneNumber'] = $userAttributesArray['PhoneNumber'];
            $_SESSION['activated'] = (bool)$userAttributesArray['Activated'];

            if ((bool)$userAttributesArray['TwoFactor']) {
                $code = generateOTP();
                $_SESSION['twoFactorCode'] = $code;
                $htmlOut .= sendTwoFactorEmail($userAttributesArray['DisplayName'], $code,
$userAttributesArray['EmailAddress']);
                header("location:verify.php"); //divert to verify 2fa
            } else {
                $_SESSION['signedIn'] = True;
            }

            //Immediately redirect back to the home page
            header("refresh:0;url=home.php", True, 302);
        } else {
            //refresh with failure message
            $_SESSION['signFail'] = True;
            header("location:signin.php");
        }
    }
} else if (isUserSignedIn()) {
    header('location:home.php'); //redirect already signed-in users
} else {
    $htmlOut .= signInForm();
}

echo pageTop();
echo $htmlOut;
echo pageBottom();
```

2FA code (verify.php)

```php
<?php

set_include_path('/home/danbates/uni-compsec-back/');
include('helper.php');
checkSession();

function twoFactorForm(): string
{
    return '
    <form action="verify.php" method="post">
        <div class="form-group p-1">
            <label for="otp">Please enter the One-Time Password you\'ve been emailed - this may take a few minutes.</label>
            <input class="form-control" name="otp" required id="otp" type="text" placeholder="e.g. ABCDEF">
            <button class="btn btn-primary m-1" type="submit" name="submit">Submit</button>
        </div>
    </form>
    <form action="verify.php" method="post">
        <div class="form-group p-1">

        </div>
    </form>
    ';
}
$htmlOut = ''; //string to hold html we'll output at the end so that we can still modify headers beforehand

if (isset($_SESSION['activated']) && $_SESSION['activated']) {
    if (isset($_POST['submit'])) { //if code submitted
        $otp = sanitise($_POST['otp']);
        if ($otp === $_SESSION['twoFactorCode']) {
            //code correct -> complete sign-in process
            $_SESSION['signedIn'] = True;
            header('location:home.php');
        } else if (!(ctype_alnum($otp) && strlen($otp) !== 6)) {
            //code in wrong format
            $htmlOut .= '<h3>One-Time Password in the wrong format, please check it again</h3>';
            $htmlOut .= twoFactorForm();
        } else {
            //code incorrect
            $htmlOut .= '<h3>One-Time Password Incorrect, please try again</h3>';
            $htmlOut .= twoFactorForm();
        }

    } else {
        $htmlOut .= twoFactorForm();
    }
} else if (isset($_SESSION['userid'])) {
    $htmlOut .= '<h3>Please <a href="activate.php">activate your account</a> to use one-time passwords and sign in.</h3>';
} else {
    header('location:signin.php');
}

echo pageTop();
echo $htmlOut;
echo pageBottom();
```

Why I think it’s secure

- Inputs validated both on the html form and in php after posting for the correct format (SQL Injection)
- All inputs are sanitised (SQL Injection)
- SQL statements are prepared and not directly written (SQL Injection)
- Any unexpected behaviour results in failure and no information is divulged
- Two Factor authentication is turned on by default

Task 3: Implement password strength and password recovery

Validation and suggestions on client side

```
<div class="form-group p-1">
    <label for="password">Password <small>(Required)<br>MUST contain at least 16 characters including uppercase, lowercase, numerical, and special characters. Maximum 30 characters</small></label>
    <input class="form-control" name="password" required id="password" type="text" placeholder="Password" pattern="'.getPasswordRegexJS().'" maxlength="30" size="30">
</div>
```

Validation on server side

```
$password = sanitise($_POST['password']);
```

```
preg_match(passwordRegex, $password) && //checks if password is >16, so we don't need to do that a second time
```

Recovery (Lines 99-133 of recovery.php)

```
function doPasswordChange($old, $new): String
{
    if ($old === $new) {
        return 'Old and New Passwords are the same';
    }

    $database = getDatabase();

    $getStoredPasswordQuery = $database->prepare('SELECT Password FROM Users Where UserID = ?');
    $getStoredPasswordQuery->bindValue(1, getUserID());
    $getStoredPasswordQueryResult = $getStoredPasswordQuery->execute();
    if (!$getStoredPasswordQueryResult) return "We couldn't retrieve your password, please try again later";
    $storedPassword = $getStoredPasswordQueryResult->fetchArray(SQLITE3_ASSOC)['Password'];

    if (password_verify($old, $storedPassword)) {
        //old is what they say it is
        //then update it and let them know
        $newHash = password_hash($new, getHashingAlgo());

        $setNewPasswordQuery = $database->prepare('UPDATE Users SET Password = :pw WHERE UserID = :uid');
        $setNewPasswordQuery->bindValue(':pw', $newHash);
        $setNewPasswordQuery->bindValue(':uid', getUserID());
        $setNewPasswordQueryResult = $setNewPasswordQuery->execute();
        if ($setNewPasswordQueryResult) {
            if (isset($_SESSION['passwordHash'])) $_SESSION['passwordHash'] = $newHash;
            return "Password updated successfully, don't forget your new one!";
        } else {
            return "We couldn't set your new password, please try again later";
        }
    } else if (password_verify($new, $storedPassword)) {
        //new is what's stored
        return "Password already set to new password";
    }
    return "Sorry, we had an issue when trying to update your password and it has not been updated.
Please try again later";
}
```

Task 4: Implement an ‘Evaluation Request’ web page

Request Evaluation Code (requestevaluation.php)

```
<?php
set_include_path('/home/danbates/uni-compsec-back/');
include('helper.php');

checkSession();
enableImageUpload();

function requestEvalForm(): string
{
    return '
    <form class="mb-4 ml-5 mr-5" action="requestevaluation.php" method="post"
enctype="multipart/form-data">
    <h4>New Listing</h4>
        <div class="form-group p-1">
            <label for="comments">Comments  <small>(Required) Maximum 500
Characters</small></label>
            <textarea class="form-control" name="comments" required id="comments" rows="4"
placeholder="Description of my item, size, weight, colour, origin, etc."></textarea>
        </div>
        <div class="form-group p-1">
            <label for="phoneOrEmail">Would you prefer to be contacted via Phone or Email?</label>
            <div class="form-check">
                <input class="form-check-input" type="radio" name="phoneOrEmail" id="phone"
value="phone">
                <label class="form-check-label" for="phone">Phone</label>
            </div>
            <div class="form-check">
                <input class="form-check-input" type="radio" name="phoneOrEmail" id="email"
value="email" checked>
                <label class="form-check-label" for="email">Email</label>
            </div>
        </div>
        <div class="form-group p-1">
            <input type="file" name="image" accept="image/png, image/jpeg, image/jpg, image/gif"/>
        </div>
        <button class="btn btn-primary m-1" type="submit" name="submit">Submit Listing for
Evaluation</button>
    </form>
    ';
}

$htmlOut = ''; //variable to store the html we're going to output all at once to avoid outputting before we
can send different headers (for redirects)

if (isUserSignedIn()) {
    if (isset($_POST['submit'])) {
        //form has been submitted
        $comments = sanitise($_POST['comments']);
        $phoneOrEmail = sanitise($_POST['phoneOrEmail']);

        if (strlen($comments) > 500) {
            $comments = substr($comments, 0, 500); //truncate comments to 500 chars if too large
        }

        //set radios to boolean
        if ($phoneOrEmail === 'phone') {
            $phoneOrEmail = 0;
        } else if ($phoneOrEmail === 'email') {
            $phoneOrEmail = 1;
        } else {
            //this shouldn't be possible - default to email
            $phoneOrEmail = 1;
        }

        //image handling
        $imageFileName = '';
        if (isset($_FILES['image'])) { //if user uploaded an image
            $fileName = $_FILES['image']['name'];
            $fileSize = $_FILES['image']['size'];
            $fileTempName = $_FILES['image']['tmp_name'];
            $fileType = $_FILES['image']['type'];
            $fileNameSplit = explode('.',$_FILES['image']['name']);
            $fileExtension = strtolower(end($fileNameSplit));

            $permittedExtensions = array('jpeg','jpg','png', 'gif');

            $correctExtensionBool = in_array($fileExtension, $permittedExtensions);
            $correctSizeBool = $fileSize <= 10485760; //if less than or equal to 10MB (in binary)
            if ($correctExtensionBool && $correctSizeBool) {
                //on success, save it
                $imageFileName = uniqid().'.'.$fileExtension;
                move_uploaded_file($fileTempName, __DIR__."/images/".$imageFileName); //give it a
unique name based on the time
                $htmlOut .= "<h4>Image Uploaded Successfully</h4>";
            } else {//else ignore it and let the temporary file get automatically deleted, and tell the user
                $htmlOut .= "<h4>Notice: File must be a jpeg, png or gif, and less than 10MB!</h4>";
            }
        }

        $db = getDatabase();
        $newListingQuery = $db->prepare("INSERT INTO 'Listings' ('UserID', 'Comments',
'PhoneOrEmail', 'ImageName')
        VALUES (:uid, :cmts, :poe, :img)");
        $newListingQuery->bindValue(':uid', getUserID());
        $newListingQuery->bindValue(':cmts', $comments);
        $newListingQuery->bindValue(':poe', $phoneOrEmail);
        $newListingQuery->bindValue('img', $imageFileName);

        $newListingQueryResult = $newListingQuery->execute();
        if ($newListingQueryResult) {
            $htmlOut .= "<h3>Listing completed successfully.<br><a href='home.php'>Homepage</a>
Redirecting in 5 seconds...</h3>";
            header("refresh:5;url=home.php"); //redirect user back home in 5 seconds
        } else {
            $htmlOut .= "<h2>Error: Failed to complete listing, please try again later.</h2>";
        }
        unset($_POST['submit']); //avoid resubmissions
```

```
    } else {
        $htmlOut .= requestEvalForm();
    }
} else {
    $htmlOut .= "<h1>Please make sure you're signed in to view this page.</h1>";
}

echo pageTop();
echo $htmlOut;
echo pageBottom();
```

Why I think it’s secure

- Inputs validated both on the html form and in php after posting for the correct format (SQL Injection)
- All inputs are sanitised (SQL Injection)
- SQL statements are prepared and not directly written (SQL Injection)
- Any unexpected behaviour results in failure and no information is divulged
- Only accessible to logged in and activated users, who should be reasonably trustworthy

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

Image upload code (Lines 58-80 of requestevaluation.php)

```
//image handling
$imageFileName = '';
if (isset($_FILES['image'])) { //if user uploaded an image
    $fileName = $_FILES['image']['name'];
    $fileSize = $_FILES['image']['size'];
    $fileTempName = $_FILES['image']['tmp_name'];
    $fileType = $_FILES['image']['type'];
    $fileNameSplit = explode('.',$_FILES['image']['name']);
    $fileExtension = strtolower(end($fileNameSplit));

    $permittedExtensions = array('jpeg','jpg','png', 'gif');

    $correctExtensionBool = in_array($fileExtension, $permittedExtensions);
    $correctSizeBool = $fileSize <= 10485760; //if less than or equal to 10MB (in binary)
    if ($correctExtensionBool && $correctSizeBool) {
        //on success, save it
        $imageFileName = uniqid().'.'.$fileExtension;
        move_uploaded_file($fileTempName, __DIR__."/images/".$imageFileName); //give it a unique
name based on the time
        $htmlOut .= "<h4>Image Uploaded Successfully</h4>";
    } else {//else ignore it and let the temporary file get automatically deleted, and tell the user
        $htmlOut .= "<h4>Notice: File must be a jpeg, png or gif, and less than 10MB!</h4>";
    }
}
```

Image Upload Enabler (Lines 300-304 of helper.php)

```
function enableImageUpload(): void {
    ini_set('file_uploads', 1);
    ini_set('upload_max_filesize', '10M');
    ini_set('post_max_size', '10M');
}
```

Why I think it’s secure

- Filetypes are validated both on the page and on the server
- Files are renamed with unique IDs to avoid giving away information
- Only accessible to logged in and activated users, who should be reasonably trustworthy
- Content security policy only allows images to load from this site alone
- If an uploaded file is not of the correct type, it is deleted automatically by php

Task 6: Request Listing Page

List of Requests Code (requestlist.php)

```
<?php
//All Listings, Admin Only

set_include_path(‘/home/danbates/uni-compsec-back/’);
include(‘helper.php’);
checkSession();

echo pageTop();

function errorMessage() {
    return “<h1>Please make sure you’re <a href=’signin.php’>signed in</a> to view this page.</h1>”;
}

function SQLite3ResultToArray(SQLite3Result $result): array
{
    $rows = [];
    while (($currentRow = $result->fetchArray(SQLITE3_ASSOC)) !== False) {
        $rows[] = $currentRow; //loop over each row in the result and add it to an array of rows
    }
    return $rows;
}

function makeTable(Array $data): string
{
    $table = ‘<div class=”wrapper p-1 m-3”><table class=”order-table table table-
bordered”><thead><tr>’;
    $row0 = $data[0]; //headers will be constant
    $headers = array_keys($row0);
    foreach ($headers as $header) {
        if ($header !== ‘EmailAddress’ && $header !== ‘PhoneNumber’) {
            $table.= ‘<th>’.$header.’</th>’;
        }
    }
    $table.= ‘</tr></thead><tbody>’;
    foreach ($data as $row) {
        $table.= ‘<tr>’;
        $email = ‘’;
        $phone = ‘’;
        //foreach ($row as $item) { //values
        foreach ($row as $key=>$value) {
            if ($key == ‘EmailAddress’) {
                $email = $value;
            } else if ($key == ‘PhoneNumber’) {
                $phone = $value;
            } else if ($key == ‘Contact’) {
```

```
            if ($value === 1) {
               $table .= ‘<td>’ . $phone . ‘</td>’;
            } else {
               //$value == 0
               $table .= ‘<td>’ . $email . ‘</td>’;
            }
         } else if ($key == ‘Image’) {
            $table .= ‘<td><img src=”images/’ . $value . ‘” alt=”Not provided” width=”400”/></td>’;
         } else {
            $table .= ‘<td>’.$value.’</td>’;
         }

      }
      $table.= ‘</tr>’;
   }
   $table.= ‘</tbody></table></div>’;
   return $table;
}

$db = getDatabase();
if (isUserAdmin()) {
   $getListingsQuery = $db->prepare(“
      SELECT
      ListingID as ‘Listing #’,
      UserID as ‘User ID’,
      (SELECT Username FROM Users WHERE Users.UserID = Listings.UserID) as ‘Username’,
      (SELECT DisplayName FROM Users WHERE Users.UserID = Listings.UserID) as ‘Display
Name’,
      (SELECT EmailAddress FROM Users WHERE Users.UserID = Listings.UserID) as
‘EmailAddress’,
      (SELECT PhoneNumber FROM Users WHERE Users.UserID = Listings.UserID) as
‘PhoneNumber’,
      PhoneOrEmail as ‘Contact’,
      Comments,
      ImageName as ‘Image’
      FROM Listings”);
   $getListingsQueryResult = $getListingsQuery->execute();
   if ($getListingsQueryResult) {
      echo makeTable(SQLite3ResultToArray($getListingsQueryResult));
   } else {
      echo ‘<h1>No Listings found</h1>’;
   }
} else {
   echo errorMessage();
}

echo pageBottom();
```

Why I think it’s secure

- Limited access to only those on the Admin Table, which can only be modified by a developer editing the database
- Doesn’t give away that it’s an admin page - tells users they must be logged in to access
- Only provides limited access to the database, even the php only grabs the columns necessary