Introduction to Computer Security

**Lovejoy’s Antique Evaluation Web Application**

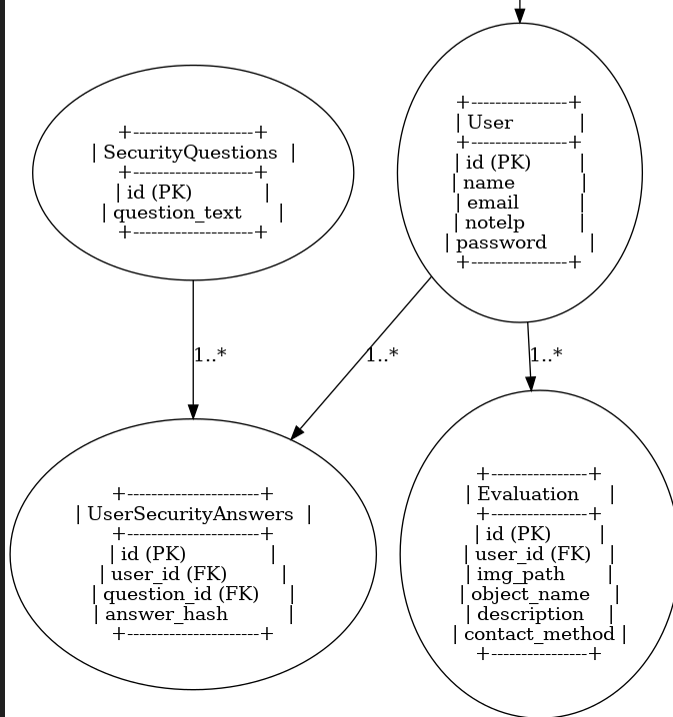
In this part of the coursework, you will develop a secure web application for a local antique dealer named Lovejoy. Lovejoy wants a minimum viable product allowing customers to register and then request evaluations of potential antique objects. Lovejoy has many rivals in the antique business who may sometimes resort to underhand tactics and so is very concerned about the security of the application.

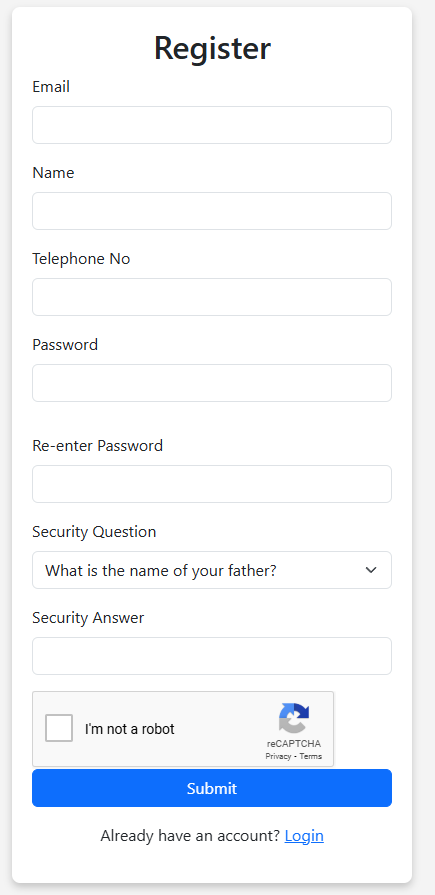
Your secure web application will need to have these features for the minimum viable product (MVP) release: user registration and login, a password policy, “request evaluation” page and then an extension of the “request evaluation” page file upload to allow upload of photos. Finally, Lovejoy needs a request listing page.

|  |
| --- |
| Task 1 - Develop a secure web form that allows customers to register in the application. They must register an email address, password, name and contact telephone number. The users’ details should be stored in a database. |
|  |
| Task 2 - Develop a secure login feature. |  |
| Task 3 – Extend the password management feature to provide password strength recommendations and password recovery. |  |
| Task 4 - Implement a “Request Evaluation” web page only accessible to logged in users. This web page should have a comment box to type in the details of the object and their request, and a dropdown box for preferred method of contact between phone or email. The evaluation page should allow for file upload of a photo of the object. |  |
| Task 5 – Implement a page that displays a list of evaluation requests. This page should only be visible to an administrator role. |  |

## Task 1 – User registration

DATA BASE DESIGN





Every input box that are implemented in the user registration will be checked in the back end thoroughly.

To preven xxs attack, I set the HTTP security headers

<?php

// Set HTTP security headers

header("Content-Security-Policy: default-src 'self'; script-src 'self'; object-src 'none';");

header("X-Content-Type-Options: nosniff");

header("X-Frame-Options: SAMEORIGIN");

?>

I also use randomize token to prevent csrf by implementing it in frontend and check at the back end

<input type="hidden" name='csrf\_token' value ='<?php echo $\_SESSION['csrf\_token']; ?>'>

//CSRF Prevention

if (!isset($\_SESSION['csrf\_token'])) {

$\_SESSION['csrf\_token'] = bin2hex(random\_bytes(32));

}

if($\_SERVER['REQUEST\_METHOD'] === 'POST'){

if(!isset($\_POST['csrf\_token']) || ($\_POST['csrf\_token'] !== $\_SESSION['csrf\_token'])){

die('Invalid CSRF token');

}

Check all the input to be not empty and sanitize all of them

// Retrieve all attributes from the user

$email = filter\_var(trim($\_POST["email"]), FILTER\_SANITIZE\_EMAIL);

$name = htmlspecialchars(trim($\_POST['name']));

// example of one of the user input

if (empty($name) ) {

$\_SESSION['error'] = "Invalid name format.";

header("Location: index.php");

exit();

}

I use reCAPTCHA from google to preven bot attack

// Check reCAPTCHA response

$recaptchaResponse = $\_POST['g-recaptcha-response'];

.....

if (intval($responseKeys["success"]) !== 1) {

$\_SESSION['error'] = 'reCAPTCHA verification failed. Please try again.';

After all user input are valid, I redirect the page to do email authentication. Unfortunately, I can only show this code below as the email authentication page is too long.

// email authentication

$\_SESSION['source'] = 'regis';

header("Location: ../email-authentication/generatepin.php");

exit();

After the user done with email authentication, now the website can store all the user credentials into the database. There are two table of databases which is ‘user’ for storing name, email, telephone number, and password. Another one is ‘user\_security\_answers’ table for storing the additional question for resetting the password. Note that the password and the answer to the question has been hashed with bcrypt before inserting to respective database.

// Hash the password and security answer

$password = password\_hash($\_SESSION['password'], PASSWORD\_BCRYPT);

$securityAnswer = password\_hash($\_SESSION['securityAnswer'], PASSWORD\_BCRYPT);

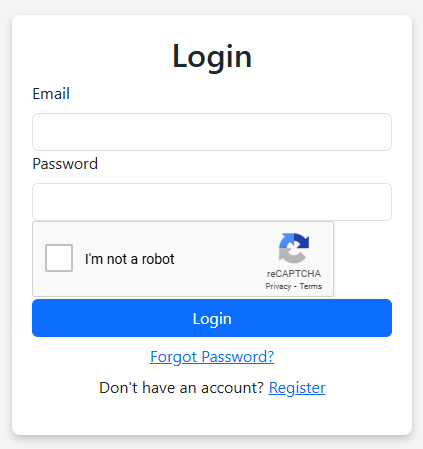
$stmt = $connection->prepare("INSERT INTO user (name, email, notelp, password) VALUES (:name, :email, :notelp, :password)");

$stmt->bindParam(':name', $\_SESSION['name'], PDO::PARAM\_STR);

$stmt = $connection->prepare("INSERT INTO user\_security\_answers (user\_id, question\_id, answer\_hash) VALUES (:user\_id, :question\_id, :answer\_hash)");

After done registering, it will go automatically to the front page.

## Task 2 - Develop a secure login feature.



Just like Task1, I implemented the same code for xss attack, csrf and captcha to make this login form secure from the attacker. In order to minimize the code, I decided not to show all the prevention in this section

All the input from the user is being sanitize to prevent sql Injection

// Validate email format

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$\_SESSION['error'] = 'Invalid email format';

..

After the email and password has been validated, the system will check into the ‘user’ database using PDO

$stmt = $connection->prepare("SELECT \* FROM user WHERE email = :email");

$stmt->execute(['email' => $email]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

..

if ($user && password\_verify($password, $user['password'])) {

..

If the email is found in database but the user input the wrong password, the system will send error messages saying ‘wrong email or password’ instead of wrong password only. However, if the user repeatedly input the wrong email/password, the system will put a pause to the user for 60 second to prevent an attack such as brute force. Note that I decided to put only 60 sec for unit testing reasons.

//if wrong, increment the flag and send error message. (intentionally )

$\_SESSION['attempts']++;

if($\_SESSION['attempts'] >= 5) {

$\_SESSION['lockedTime'] = time();

}

After the user input the right email and password, the system will redirect to 2fa have the same features as email authentication: sending 4 digit pin to the email to continue and the user only have 60 seconds to validate.

//2Fa

$\_SESSION['source'] = 'login';

$\_SESSION['otp\_email'] = $email;

header('Location: ../email-authentication/generatepin.php');

If the user typed the wrong pin for 3 times, the system would send a new pin to validate. However, if the system sends the pin more than 3 times, the system will ban the email by inserting the email into another database called ‘banned\_emails’. This logic is applied in Task1 for email authentication and the upcoming task. I decided to reuse the authentication email system as 2fa without reducing the security risk.

//check the flag

if($\_SESSION['wrong'] < 0){

$\_SESSION['tries']--;

checkFlag($connection);

$\_SESSION['error'] = "Incorrect pin, sending a new pin to your email";

//Check flag. If the server already send the pin 3 times, email banned

if($\_SESSION['tries'] < 0){

banEmail($connection, $\_SESSION['otp\_email']);

Only then, it will continue to the main page.

## Task 3 - Implement password strength and password recovery

For password strength, I implemented it both on front end of the registration page to act as a reminder and back end for ensuring the password is strength enough. The system will check for minimum length, uppercase letter, lowercase letter, number and special characters.

var strengthText = document.getElementById("password-strength-text");

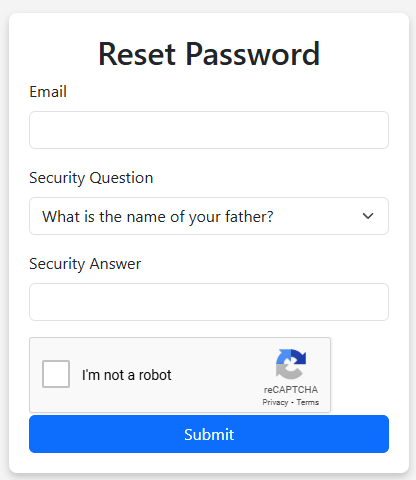
var strongPattern = /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?=.\*[\W\_]).{8,}$/; // At least 8 characters, one uppercase, one lowercase, one digit, one special character

// Check minimum length

if (strlen($password) < 8) {

$\_SESSION['error'] = "Password must be at least 8 characters long.";

...



To reset the password, first the user will need to input the email address, pick their security question and their asnwer. Just like all the task before xss, csrf, recaptha and sql injection prevention has been embedded in this page.

One thing that I want to point out is there are several questions in the security question, and it has been inserted into database alongside with the hashed security answer upon registration, making it more difficult for the attacker to make an attack.

The system will check the email from database and security question with its answer in different while using PDO for preventing sql injection.

//check the selected question matches in different table

$stmt = $connection->prepare("SELECT id FROM security\_questions WHERE id = :question\_id");

$stmt->bindParam(':question\_id', $securityQuestion, PDO::PARAM\_INT);

$stmt = $connection->prepare("SELECT answer\_hash FROM user\_security\_answers WHERE user\_id = :user\_id AND question\_id = :question\_id");

$stmt->bindParam(':user\_id', $user['id'], PDO::PARAM\_INT);

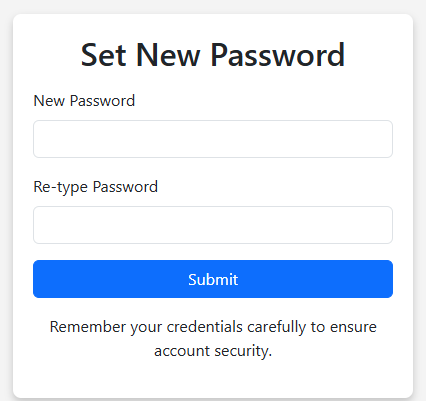
if (!password\_verify($securityAnswer, $storedAnswerHash['answer\_hash'])) {

$\_SESSION['error'] = "Invalid.";

After all input are valid, the system will redirect to another page to set a new password

$\_SESSION['reset\_password'] = True;

header("Location: ../reset-password/index.php");



This page can only be accessed if the user already pass with the security question. If the user forcing into this page, it will redirect to the previous page

//check if it already pass the backup question

if(isset($\_SESSION['reset\_password'])){

header('Location: ../forgot\_question/index.php');

exit();

}

This page has the same logic with the registration page which is check the password strength.

// Check for uppercase letter

if (!preg\_match('/[A-Z]/', $password)) {

$\_SESSION['error'] = "Password must include at least one uppercase letter.";

After the new password is valid, it will do another 2fa

//2Fa

$\_SESSION['newpassword'] = $newPassword;

$\_SESSION['source'] = 'forgot-password';

$\_SESSION['otp\_email'] = $\_SESSION['reset\_email'];

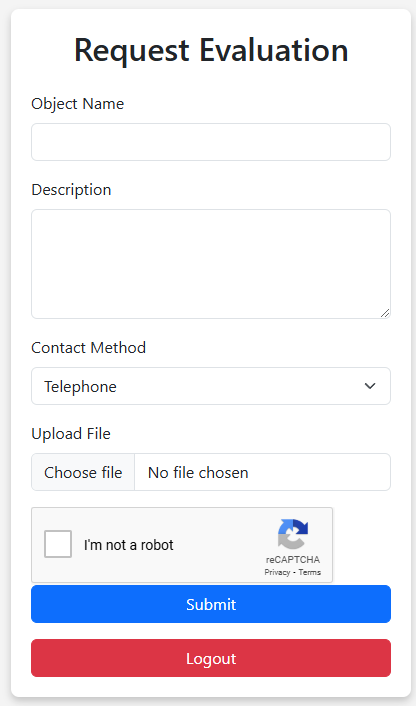
header('Location: ../email-authentication/generatepin.php');

Finally, the system will update the user password

$newPassword = password\_hash($\_SESSION['newpassword'] , PASSWORD\_BCRYPT);

$stmt = $connection->prepare("UPDATE user SET password = :password WHERE email = :email");

## Task 4 - Implement a “Evaluation Request” web page.



Xss, csrf, recapthca and sql injection has been prevented just like others. All of the user input cannot be empty and has been sanatize.

Only registered user can access this page.

// Check if the user is logged in

if (!isset($\_SESSION['user\_id'])) {

$\_SESSION['error'] = "You must be logged in to access this page.";

The most important security feature in this task in to have a secure file upload. For this, I check the size of the file

$filepath = $\_FILES['upload-file']['tmp\_name'];

$fileSize = filesize($filepath);

//limitate the size

if ($fileSize > 10000000) { //roughly 10 mb

And limitate the file type to be only jpg and png

//limitate the type of the file

$allowedTypes = ['image/png' => 'png','image/jpeg' => 'jpg'];

if(!in\_array($filetype, array\_keys($allowedTypes))) {

$\_SESSION['error'] = 'invalid file';

After that I move the file into secure file system

$newFilepath = $targetDirectory . "/" . $objectName . "." . $extension;

//check the copied file

if (!copy($filepath, $newFilepath)) {

$\_SESSION['error'] = 'invalid file';

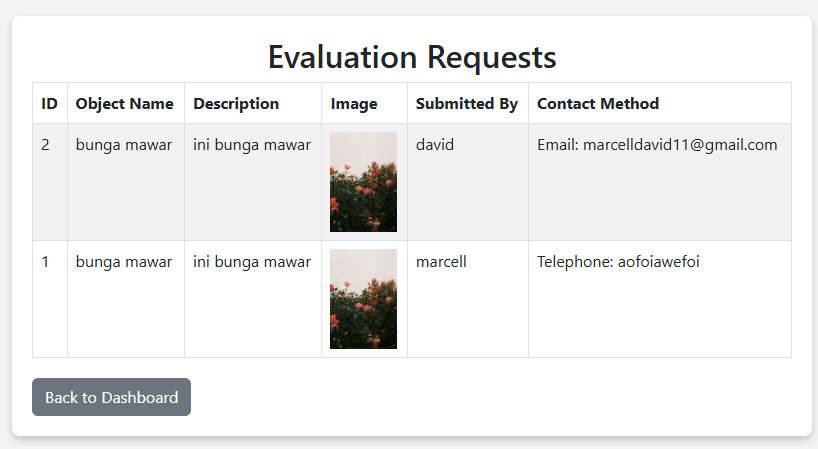
And After that I insert all of the evaluation including the filepath into new database table called ‘evaluation’

//input to database

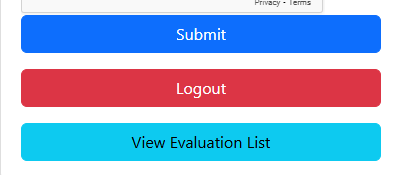
$stmt = $connection->prepare("INSERT INTO evaluation (object\_name, description, contact\_method, user\_id, img\_path) VALUES (:object\_name, :description, :contact\_method, :user\_id, :img\_path)");

$stmt->bindParam(':object\_name', $objectName, PDO::PARAM\_STR);

## Task 5 – Request Listing Page



This page can only be accessed by the administrator. In this project, I automatically decide that the first user in the database is the administrator. To access this page, the administrator will log in like others. However, in the evaluation request, there is button that only appeared for the admin to access this page.



In this page the system will extract the information from multiple database table with using PDO

//extract from database all of the information

$stmt = $connection->prepare("

SELECT e.id, e.object\_name, e.description, e.img\_path, u.name AS username, u.notelp, u.email, e.contact\_method

FROM evaluation e

JOIN user u ON e.user\_id = u.id

ORDER BY e.id DESC

");

$stmt->execute();

$evaluations = $stmt->fetchAll(PDO::FETCH\_ASSOC);

All of the data that has been shown in the table are secure

echo '<td>' . htmlspecialchars($evaluation['id']) . '</td>';

echo '<td>' . htmlspecialchars($evaluation['object\_name']) . '</td>';