**Software report (5%)**

**Technical Description:**

The RecycleMore webpage was created to propagate the importance of recycling and reducing waste. The web app was created using HTML, CSS, PHP, JavaScript, and Bootstrap. Upon completing the basic template for the RecycleMore webpage, the next phase of the project required the implementation of three core features. The web app offers three functions – these include, creating user accounts, generating dynamic content, and uploading images and files.

**How the code functions**

One of the main features the RecycleMore web app possesses is the ability to register user accounts. The registration page contains a HTML form, this being the catalyst in capturing user information and transmitting the information to the web server. Conditional statements have been implemented for security and integrity purposes, if a user entered an objectively weak password, the program would prevent them from using such a password.

Once the user has entered their details into the HTML registration form, a user account will be generated and stored on the web server hosted by MariaDB. All users are given a unique ID number associated with their user account. Upon logging into their user account, a request for the users’ credentials will be send to the web server and a validation check will take place and a new session ID will be created.

If the user wishes to log out and subsequently end their current session - ?

As the RecycleMore web app allows users to create user accounts linked with a session ID for improved security, the first task was to connect the website registration output to a server and subsequently a back-end database. The main technologies used to develop the backend functionalities of the web app were PHP for data transmission and MySQL for the database development.

Conditional statements have been used to verify that user input data adheres to a set of rules imposed by the development team. Conditional statements have been implemented to prevent SQL attacks – for example, if an attacker attempts to insert an SQL script into the email input field, the conditional statement will throw an error and exit the program.

The $POST method is the favourable choice of data transmission between the browser and web server. Unlike the $POST method, any data send from a HTML form using the $GET method can be seen in the browser URL, sensitive information such as passwords and emails can become compromised as a result. The $POST method stores information in the HTTPS header rather than in the URL, this improves overall security and prevents malicious users from obtaining sensitive information.

Text

Description automatically generatedThe $dbcreds variable encapsulates all the functionality provided by MySQLi technology combined with additional information pertaining to the MySQL database’s location.

Text

Description automatically generatedThe $stmt variable, in this instance, is crucial in preventing SQL-attacks as the database will be expecting an explicit format of information – rather than accepting any value.

**Secure Development Life Cycle – Security Modelling & Testing (10%)**

The following section of the report pertains to the security policies and procedures adopted by the development team. We’ve chosen to emulate a similar secure development life cycle used by Microsoft. Our SDL has been tailored to the size of our project and encapsulates the most important aspects of the complete SDL used by Microsoft.

**Development Policies:**

1.Use of comments.

2.Research common vulnerabilities associated with libraries before implementation.

3.Test new features before updating repository code.

4.Attempt to fix, mitigate or remove comprised features in a timely manner.

5.Ensure any redundant code is removed from the final software product.

**Threat Modelling:**

Research similar web application as a background procedure to prevent potential bugs in your own design.

Highlight libraries that are used/will be used in the development and their overall security risk.

Design High-level application model to provide visual representation – (lecture 8, page 26)

List high value assets that require protection – example, server, database, and source code.

List threats ranked by risk compared to what features our website has and what libraries are used.

List how these risks will be mitigated or attempted mitigation. Actions – do nothing, remove feature, turn off feature, warn user or counter technology (ie. Conditions written into code).

**Continuous Development Testing:**

This section requires a table containing all website features, tests used, results of the tests and changes made.

**Bug Bar:**

Defines what bugs should be fixed, left or mitigated.

Sets a standard at which the software is secure enough

Catorgories – Critical, Medium and low risk

**ASA (Attack Surface Analysis):**

ASR (Attack surface reduction):

Aims to reduce the possible mechanisms used by attackers- example, adding conditional statements for what character can be entering into input fields on website.

ASR Flowchart (Lecture 8, page 20)

**FSR – Final Security Review:**

Review the website once completed – talk about persistent security flaws

**Security Response Planning:**

\*Locate security vulnerability

\*Create DFD to solve the entire issue rather than fixing the individual bug caused by a common action performed by user such as enter their name in the search bar.

\*Fix the bug and update the source code

**Evidence Of Repository Usage – GitHub (5%)**

**GitHub Commands**

**SSD Work Log**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Time | TODO | Priority (1-5) |
| 23/02/2022 | 9:40 am | Build website home page | 5 |
| 23/02/2022 | 12:00 pm | Build Login/ registration page | 5 |
| 23/02/2022 | 1:00 pm | Build Portfolio & contact page | 5 |
| 23/02/2022 | 3:00 pm | Start adding CSS and dynamic content sizing | 3 |