WEB DEVELOPMENT GROUP REPORT

Stewart Mcafferty, Kyle MacDonald, Christopher McNeil, Josh Cowan

29/04/2019

BSc Computing

Table of Contents

[Introduction 2](#_Toc7388772)

[Link Design within the Application 2](#_Toc7388773)

[Persistence 3](#_Toc7388774)

[Functionality and Testing 4](#_Toc7388775)

[Login System 4](#_Toc7388776)

[Milestone Test’s 5](#_Toc7388777)

[Application Functionality 6](#_Toc7388778)

[Application Security 7](#_Toc7388779)

[Potential attacks 7](#_Toc7388780)

# Introduction

The aim of the application is to create a milestone planner for a user project. This involves the user being able to input the description of the milestone, an intended due date of the milestone completion date, and the actual completion date of the milestone.

Each milestone must be displayed on a scrollable list on the page where the all inputted milestones may be viewed, edited or removed. The milestone list must also have an external share link that can be viewed by other users.

The website must have a secure login system that allows the user to enter a username and password to gain access to the website. All data about previous set milestone should be saved from the user’s last visit to a separate database.

This overview report will include detail on the following topics relating to this project: Link design in the application, data persistence, functionality and testing, and application security.

The link design of the application discusses how all the functions of the site connect to each other, front-end and back-end.

The data persistence is the database schemas used and how the data access layer of the website connects with the rest of the website.

Functionality and testing of the website are key to the success of the website. In this section, both the testing tables and the functionality proof is showing in tabular form with evidence.

Application security is how the website was successfully secured and protected against XSS, SQL injection and password hashing.

# Link Design within the Application

Having a webpage that links together every part of the site is the basics for ensuring it is suitable for the end user. Therefore, the website currently in development was designed to be as user-friendly as possible. With this in mind, this section will discuss how every functional part of the website links to each other from front end, to back end.

First of all, the login page. This login page allows the user to log in with a specific username and password that is defined in the database. This username is stored on said database under a table with columns “username” and “password” which allows unique accounts to be stored on these tables.

This directly connects to the back-end database schema as the tables mentioned above are included in the login system. This then allows the user to use their login credentials to access the website.

Once the user is on the main webpage and connected with the back-end, the functionality is as follows. The functional buttons on the site are add milestone, edit milestone, complete milestone. The milestones will also be displayed on a list at the bottom of the page where the user can also view, edit and complete the milestones as well.

These functional buttons link to the database as there is a separate table for each user’s milestones. The separate table links directly to the user details table via a relationship column on each of the tables, carrying data between the tables, allowing user data to be saved on both. On the webpage, the buttons then initiate the original code with the new data that it is given by the database.

The list function on the main webpage allows the milestones to be displayed to the user with the availability to edit or remove the milestone. This directly links to the buttons mentioned above as the list takes the data from the user input from the button and displays them in the milestone box.

Overall, all the functionality of the website is linked to each other as the site would be incorrect and buggy without this connection. Linking all the pages is crucial to the basic functionality of the website and improves the compatibility and smoothness of the site.

# Persistence

# Functionality and Testing

## Login System

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Number | Test Data | Reason | Expected Outcome | Actual Outcome | Pass/Fail? | Comment |
| 1 | “username  Password” | Verify user can login with valid username and valid password | Log’s in and takes user to landing index page. | Logged user in took user to landing index page | Pass | N/A |
| 2 | Username and password which isn’t stored in database | Verify user can’t login with an invalid username and password | Throws user error and doesn’t allow user to proceed to the landing page. | Threw error and refused access to the main site. | Pass | N/A |
| 3 | Blank in username and password text fields | Verify if user can login to site with username and password fields blank. | Throws error to user that they must input a username and password | Threw user error and doesn’t allow user to enter site. | Pass | N/A |
| 4 | ‘password’ | Verify if data in password field is visible as asterisk. | Data shows as asterisks. | Data showed as when asterisks surrounded it. | Fail | In order for password security it is best that the data won’t show if surrounded by asterisk’s, this will need to be implemented in further versions of the site. |

## Milestone Test’s

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Number | Test Data | Reason | Expected Outcome | Actual Outcome | Pass/Fail? | Comment |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Number | Test Data | Reason | Expected Outcome | Actual Outcome | Pass/Fail? | Comment |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Number | Test Data | Reason | Expected Outcome | Actual Outcome | Pass/Fail? | Comment |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Number | Test Data | Reason | Expected Outcome | Actual Outcome | Pass/Fail? | Comment |
| 1 | Milestone description | Verify description to milestone can be added | Milestone description appears | Milestones description is added | Pass | N/A |
| 2 | Milestone due date | Verify due date can be added to milestone | Mile stone due date can be added | Mile due date added | Pass | N/A |
| 3 | Click complete milestone | Verify when complete milestone is clicked, milestone competition date shows. | Milestone competition date shows | Milestone completion date added and visible | Pass | N/A |
| 4 | Login and Log out with milestones added | Verify that milestones exist longer than one session | Milestones stay in the state they were last login | Milestones appeared no change | Pass | N/A |
| 5 | Login as another user | Verify that each user can only see their personal milestones | User can only see their personal milestones | Can only see the user’s personal milestones | Pass | N/A |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Number | Test Data | Reason | Expected Outcome | Actual Outcome | Pass/Fail? | Comment |
| 1 | Remove milestone | Verify milestones can be removed | Milestone removed and confirmed by a popup message | Milestone removed | Pass | N/A |
| 2 | Edit milestone | Verify milestone can be edited | Milestone can be edited | Milestone is edited | Pass | N/A |
| 3 | Remove milestone | Verify milestone can be removed | Milestone is removed | Milestone is removed | Pass | N/A |
| 4 | List incomplete miles stone’s | Verify incomplete milestones are listed | Incomplete milestones are listed | Incomplete milestones are listed for the user | Pass | N/A |

# Application Functionality

# Application Security

Web applications can be easy to attack if the appropriate precautions must be taken. as out application has a need to store passwords steps must be taken to make sure that the user account is safe and secure in case of a breach.

Password Hashing; password hashing is a one-way function. They turn data in to a fixed length dataset that is irreversible. The algorithm that created these hashed passwords are so sensitive that if even on character is changed then the resulting hashed password is completely different. It is the hashed password that is stored in memory, never is a plain text version saved anywhere. When the user inputs their password it is then rehashed and if it matches the hashed password stored in the memory.

An additional method of adding security to password hashing is salt, Salt is random data that is added into a hashed password. This is different for every user, meaning that two users with the same password would have different salt. This is important because it means the users have different hashed passwords. This will be implemented into our webpage as we will be storing user passwords, we are obliged to protect that information as well as possible and this is a proven technique to save passwords.

## Potential attacks

Cross site scripting (XSS) is a type of attack that targets web applications by executing their own script in the web browser. This is done by injecting java script into the web page, usually through any input boxes the web page may have, and the website runs it like any other code. It can be sued to access other user’s session cookies. Meaning the can appear to be the user whose session data they have stolen, it also gives the attacker access to the user’s sensitive data. however a XSS attack can also be used to vandalise the webpages itself. Our Site is vulnerable to this attack as we have places where users put in input. In order to defend against this kind of attack you can implement input sanitation.

Input sanitation is where you filter the users input that remove any HTML tags such as <script>, this stops the page from running it and instead turned into a harmless sting of text. This I the simplest way to stop an XSS attack.

SQL injection, this is another injection-based attack thought the same vectors that, instead of performing scripts it performs SQL statements meant to attack a database. It can be used to again access to any SQL based data base. Like XSS the injection takes place in user inputs. They are very dangerous and can a successful one can dump an entire database. They can be prevented by using input sanitation similar to XSS but instead looking for SQL related inputs.