Web Platform Development 2

Group Report

Module Code: M3I322955

Team ID: M

Team Members:

|  |  |
| --- | --- |
| **Name** | **Student ID** |
| Lewis Bohme | S1623730 |
| Calum McDonald | S1625124 |
| Robert Wildman | S1509388 |
| Ross McGregor | S1510552 |

I declare that all work submitted for this coursework is the work of Robert Wildman, Calum McDonald, Ross McGregor & Lewis Bohme alone unless stated otherwise.

Contents

Contents

[Introduction 3](#_Toc7474579)

[Communication – Discord/Facebook Messenger 3](#_Toc7474580)

[Document Sharing – OneDrive 3](#_Toc7474581)

[Code Repository – GitHub 3](#_Toc7474582)

[Link Design 3](#_Toc7474583)

[Language Choice – NodeJS 3](#_Toc7474584)

[Database Choice – Azure Database 4](#_Toc7474585)

[Persistence Mechanism 4](#_Toc7474586)

[Test Reports – Acceptance Tests 5](#_Toc7474587)

[Login Functionality Testing 5](#_Toc7474588)

[Project Functionality Testing 5](#_Toc7474589)

[Milestone Functionality Testing 6](#_Toc7474590)

[Security Methods 7](#_Toc7474591)

[Conclusion 8](#_Toc7474592)

# Introduction

The purpose of this report is to cover Group M’s design and development process of the Milestone Planner Application. The Milestone planner was to include a list of milestones for a user defined project. The application was also requested to have a login system to make the planner have persistence for its milestones. This report will cover the functionality of the program, the chosen persistence mechanism, the security of the application and testing for the project.

## Communication – Discord/Facebook Messenger

For our inter group communications we have decided to use Discord alongside Facebook messenger. We are using Discord to stay connected by using it’s screensharing a voice communication channels to work with each other even when not physically next to one another. We are also using Facebook messenger to keep in touch for faster communication if we want to do a quick check up with each other or get some input.

## Document Sharing – OneDrive

For document sharing such as this group report and our example use case and class diagrams, we have decided to use OneDrive which is easily accessible by each team member as it is a resource provided by the university. It is a cloud-based document sharing space provided by Microsoft.

## Code Repository – GitHub

For our choice of Code Repository, we settled on using GitHub instead of BitBucket. One of the reasons we decided to use GitHub is because it has a large user base of over 20,000 users so that if there were any errors on the repository side of the project, we would have a large support network to fall back on. We also already have a private repository set up to accommodate the project and since we all have experience working with each other on prior projects it was very easy to connect ourselves on the repository. Also, each team member has prior experience using GitHub, so we are cutting out the time it would take to learn and become familiar with its features.

# Link Design

## Language Choice – NodeJS

To write the application we have decided to use NodeJS as our runtime environment to build the application up using Java. We have decided to use this instead of MustacheJs because as a team we have more experience working with NodeJS and did not need to spend extra time learning how to fully utilize the features that it provides. NodeJs also performs well in small scale environments such as the one we are working on. For our system we added lots of frameworks to allow our system to flow better when coding, these frameworks were EJS, Express, async and Tedious. EJS allowed us to create easy HTML templates using basic JavaScript coding which made for simple creation of webpages to add to our system. Express was for creating our server to run our website on and create API’s for use in said website. Async allowed for easy flow of work when dealing with functions in the JavaScript coding as we were working asynchronously and finally Tedious which allowed us to implement the Azure database into the system and for us to send SQL queries to the database. NodeJS was also chosen because it works better with web frameworks which as Azure is our database choice, a cloud based data store, as a team we decided it would integrate better.

## Database Choice – Azure Database

The link between each front-end feature and the backend was done through the use of the REST API that Azure uses, it was connected to the system by code implemented into the Javascript files that allowed the system to connect with our specific azure database. Using our set login for security we were able to send connection requests to the database easily and it also gave us the option to encrypt our database for added security on the back end of the system.

API Design:

/ - Returns the Index Page as HTML

/ - Returns a page to create a website

/api/account/ - Get – Returns all Accounts

/api/account/id/:id – Returns the account with the inputted ID

/api/account/create - Creates an Account with the required inputs

/api/account/login - Calls the login function and checks inputs with the databases.

This is repeated for Project and Milestone.

# Persistence Mechanism

For the data management software, we need during this project we have elected to use Microsoft Azure which is a cloud computing service maintained by Microsoft which allows the testing and development through Microsoft’s servers.

Creating our database, we needed 3 separate tables to hold all of the data required for the system these tables consist of Accounts, Milestone and Projects. The Accounts table is set up to use an ID that is automatically set by the system when a new account is created on the site, this Account ID serves as the primary key for the Accounts table and is stored as an integer. The other fields consist of Username which is stored as a varchar, this holds the username that the account creator decided on in the login page. The Password column stored the Password that the user input on the login screen, this is also stored as a varchar as it is a string. Finally there is a text column that contains the Projects list, this stores all the projects associated with that specific account.

The Projects table holds the details for all of the projects created by Logged in users. The fields are ProjectID which is also generated by the database whenever a project is created, Project Name holding the name of the created project in a string. Saved as text is the Project description column and using the date format are the two fields of Project due and Project Completed, finally the Project holds a fields called ProjectUserID which will directly link to the Users generated ID so that all of the Projects are linked to a specific user and if a user has multiple projects then the value of the field will be the same.

The last table is the Milestone table, which is very similar to the Projects table but with some slight differences. It still holds the similar fields of Milestone Name, Description, Due and Completion these are stored in the Varchar, text, date and date formats, respectively. The final field is the Milestone project ID which will directly link one to many milestones with a specific project as they will be the same value as the Project ID of the linked project.

Overall the Azure server is a solid system for our Project as it allows us to read and write to the database easily, edit the rows and column if any maintenance needs to be done and easily read the contents of the Database to test if our code was working as intended.

# Test Reports – Acceptance Tests

This section is dedicated to a description of the features of the application and the test cases used to evaluate the project.

## Login Functionality Testing

|  |  |  |
| --- | --- | --- |
| **Title** | **User Story** | **Acceptance**  **Working (Yes/No/Part)** |
| Login | The user is a returning user and wants to access their projects. | 1. Enter User Name - Yes 2. Enter Password - Yes 3. Click Login - Yes 4. Login Successful - Yes |

|  |  |  |
| --- | --- | --- |
| **Title** | **User Story** | **Acceptance**  **Working (Yes/No/Part)** |
| Create Account | This is a new user and they would like to create an account, so they can keep track of their Projects and Milestones. | 1. Click Create Account – Yes 2. Enter User Name – Yes 3. Enter Password - Yes 4. Account Creation successful - Yes |

## Project Functionality Testing

|  |  |  |
| --- | --- | --- |
| **Title** | **User Story** | **Acceptance**  **Working (Yes/No/Part)** |
| Create Project | This user wants to create a new project to keep track of milestones during their project. | 1. Click Create Project - Yes 2. Enter Project Name – Yes 3. Enter Project Description - Yes 4. Enter Project Due Date – Yes 5. Project Creation successful - Yes |

|  |  |  |
| --- | --- | --- |
| **Title** | **User Story** | **Acceptance**  **Working (Yes/No/Part)** |
| Delete Project | This user wants to delete a project because it is no longer needed. | 1. Click Delete Project – No 2. Project Deletion successful - No |

|  |  |  |
| --- | --- | --- |
| **Title** | **User Story** | **Acceptance**  **Working (Yes/No/Part)** |
| Edit Project | This user would like to edit a project because they made a mistake when creating it. | 1. Click Edit Project - No 2. Edit Name Field – No 3. Edit Description Field – No 4. Edit Due Date Field – No 5. Edit Completion Field – No 6. Click Finish Editing – No 7. Edit successful - No |

|  |  |  |
| --- | --- | --- |
| **Title** | **User Story** | **Acceptance**  **Working (Yes/No/Part)** |
| Create Link | This user would like to share their project to the rest of their team so that they can also take part in it. | 1. Click Create Link – No 2. Link creation successful - No |

## Milestone Functionality Testing

|  |  |  |
| --- | --- | --- |
| **Title** | **User Story** | **Acceptance**  **Working (Yes/No/Part)** |
| Create New Milestone | This user would like to create a new milestone and add it to the project that they are undertaking. | 1. Click Create Milestone – No 2. Enter Milestone name – No 3. Enter Milestone Description – No 4. Enter Due Date – No 5. Milestone creation successful - No |

|  |  |  |
| --- | --- | --- |
| **Title** | **User Story** | **Acceptance**  **Working (Yes/No/Part)** |
| Edit Milestone | The User would like to edit the milestone’s details because they made a mistake while creating it. | 1. Click Edit Milestone – No 2. Edit Milestone Name – No 3. Edit Milestone Description – No 4. Edit Due Date – No 5. Edit Completion date – No 6. Milestone edit successful - No |

|  |  |  |
| --- | --- | --- |
| **Title** | **User Story** | **Acceptance**  **Working (Yes/No/Part)** |
| Remove Milestone | The User would like to delete this Milestone because it was added to the wrong project by accident. | 1. Click Delete Milestone – No 2. Milestone deletion successful - No |

# Security Methods

There are several ways that an attacker could attack our website. The following examples are of possible types of attacks and defenses that were not implemented into the website. Other ways that an attacker could attack our website is by using SQL injections, these would interrupt other quires being made to the server and could reveal information that wasn’t meant to be shown on the website. Or could even cause the server crash which could interrupt in users being able to use the project scheduling service provided on the website. There are different ways that an attacker could gain access to use SQL injections, such as seeing error messages, being displayed on the webpage or even the console of their browser.

Alternatively, if the attacker knows information about the database, such as table names. Using an sql injection they could use the union statement to work out how many different columns there are in the table and which of those columns will return a string they can use. A defensive solution to this could be to blacklist certain characters from user input. Some characters that could potentially be used in queries or in code. Such as ‘ or ;. Certain characters could be whitelisted such as \_ as these are sometimes used in usernames and passwords.

A security method that we were thinking about implementing into the website was ensuring that the user details are stored securely in the database. The user's data is stored using a hash encryption. This will take the users input such as a username and password and then convert it using a hashing algorithm into a string of numbers and letters. The way that this works, is by comparing the hashed password the user has entered to the hashed password stored in the database. If they match, then the user is granted access to the website. The only possible threats to this method of storing passwords is if a potential attacker is able to make educated guesses about the account, they’re attempting to gain access to. However, as there is not a limit to the amount of login attempts that a user can make, the attacker could continue to keep guessing until the they guess the correct password. However, the account it protected if the attacker found some way to view the database tables. Alternatively, the attacker could use a look up table. This would allow for the attacker to look up what the hash of a password is.

# Conclusion

The final product created lacked many of the features required, however there was a significant amount of code that was added to the repository that would have made these features possible. Sadly, due to time constraint these features could not be tied together. One of the other major changes that could have been made to the website was more security features. Such as having passwords hashed to prevent unauthorized attempts to log in to other accounts.