Stewart Horne, Emil Grzanka, Norbert Bednarski

6th May 2020

“I declare that all work submitted for this coursework is the work of Group D (Stewart Horne, Emil Grzanka & Norbert Bednarski) alone unless stated otherwise.”

Web platform development 2

Group D

Contents

[Link Design 1](#_Toc39664247)

[Introduction 1](#_Toc39664248)

[URL Design - Decisions 3](#_Toc39664249)

[URL Design – Avoidance 3](#_Toc39664250)

[Persistence 4](#_Toc39664251)

[Database 4](#_Toc39664252)

[Cloud 5](#_Toc39664253)

[MongoDB 5](#_Toc39664254)

[Data 5](#_Toc39664255)

[Session Persistence 5](#_Toc39664256)

[Testing 6](#_Toc39664257)

[Unit Testing 6](#_Toc39664258)

[Integration Testing 6](#_Toc39664259)

[Functionality Testing 7](#_Toc39664260)

[Summary 12](#_Toc39664261)

[Application Security 13](#_Toc39664262)

[Hashing 13](#_Toc39664263)

[Salting 13](#_Toc39664264)

# Link Design

## Introduction

Our application was created in a way to allow all links to be easy to understand when it comes to pathing, as well as for a user with little experience in link design to be able to understand the directories they are passing. All links used on the website are listed in the table below:

|  |  |  |
| --- | --- | --- |
| **No.** | **Link** | **Description** |
| 1 | <https://web-coursework-browser.herokuapp.com/> | Default homepage of the website, allows the user to direct to either the login or register page. |
| 2 | <https://web-coursework-browser.herokuapp.com/users/register> | Register page of the website. Allows the user to either register or change to login window once registered. |
| 3 | <https://web-coursework-browser.herokuapp.com/users/login> | Login page of the website. Allows the user to login or register if the user has not made an account. |
| 4 | <https://web-coursework-browser.herokuapp.com/coursework/dashboard> | The main page of the website, can redirect the user to many areas, including adding or viewing coursework, showing incomplete and complete courseworks, as well as creating, editing and viewing milestones. |
| 5 | <https://web-coursework-browser.herokuapp.com/coursework/addCourseWork> | Page that allows the user to add courseworks to their account. Can redirect the user to the main page, or to the show courseworks page |
| 6 | <https://web-coursework-browser.herokuapp.com/coursework/showComplete> | Show complete page of the site. Shows the user courseworks that have passed their deadlines. Can redirect the user to the main page, or to the add courseworks page |
| 7 | <https://web-coursework-browser.herokuapp.com/coursework/showIncomplete> | Show incomplete page of the site. Shows the user courseworks that have not yet been completed. Can redirect the user to the main page, or to the add courseworks page |
| 8 | <https://web-coursework-browser.herokuapp.com/coursework/shareCoursework> | Share coursework page of the site. Allows the user to create a shareable link to their coursework, details on this further down. |
| 9 | <https://web-coursework-browser.herokuapp.com/coursework/editCoursework> | Edit coursework page of the site. Allows the user to edit the name, deadline or description of the coursework. |
| 10 | <https://web-coursework-browser.herokuapp.com/milestone/newMilestone> | New milestone page of the site. Allows the user to create milestone for an already created coursework and give them separate deadlines. |
| 11 | <https://web-coursework-browser.herokuapp.com/milestone/milestones> | Milestone page of the site. Allows the user to view currently created milestones for the coursework. |
| 12 | <https://web-coursework-browser.herokuapp.com/milestone/editMilestone> | Edit milestone page of the site. Allows the user to edit previously created milestones, including the name, date due and description. |

## URL Design - Decisions

Each link has the domain “web-coursework-browser.herokuapp.com”, as this is the platform the application was hosted on. The domain remained consistent with each webpage of the application as this could confuse the user, as well as confuse search engines when people are trying to search for the application through google, safari etc.

The URL design throughout the application is also consistent, as each URL follows the design intended for the application, shown below:

“<https://domain.com/repo/feature>”

The URLs were also designed in a way which allows them to be readable by humans, by attempting to remove as many session ID’s as possible in the URL, as well as naming the directories in a way which users can understand, and possibly direct the website through editing the URL. Some URLs, such as the editMilestone and editCoursework sites have an ID attached to the end of the URL when viewing an already existing coursework or milestone.

The URLs are additionally not too long, so that they are easier for the user to comprehend, as well as to allow for easier testing once the application is finished with implementation. Shorter URLs can also entice users to share the links more, as well as general convenience and professionalism. Furthermore, when using a shorter URL, you can track how many times each individual link has been clicked, which can be used for storing data.

The URLs were also designed in a way to allow for future expansions of the website application during further implementations, such as adding another feature onto one of the repos or adding another repo altogether. Further pages implemented will use the same naming scheme defined above, to remain consistent throughout.

## URL Design – Avoidance

The URL design eventually used in the application were also void of many aspects which could influence security or make the URLs potentially unreadable. These include the following:

* Avoiding the inclusion of session ID’s inside the URL
* Avoiding including private data inside the URL, such as passwords, bank details etc.
* Including foreign, or non-ascii characters in the URL.
* Avoiding using many subdirectories, which would result in a longer URL
* Avoiding loaded URLS, containing many different parameters, such as ID, tasks etc.

Avoiding these points allowed the application to have more readable, consistent and professional URL design.

# Persistence

## Database

Persistence involves collecting data over a period of time, such as any data generated by the user, or data generated from running the web application. The team maintained persistence throughout the web application was through the creation of a database. A database was chosen for persistence, as it is a good way to maintain consistency throughout the application’s data, as well as allowing all data to be stored in one place, compared to many separate files. Additionally, data security is increased, and there are recovery facilities, such as backups, if something within the database fails. Having a database also allows you to use CRUD (create, read, update, delete) operations, which you can use to obtain an input for the user as to what courseworks and milestones they want to add, view, edit or remove, as well as adding and editing login and register data.

## Cloud

Cloud persistence is also present in our application, as a way to backup data if the database in the application fails or is compromised. Cloud persistence additionally has flexibility in what database is chosen and is globally distributed. The system used for both database and cloud persistence is MongoDB.

## MongoDB

MongoDB was chosen as the selected database, as the information stored on MongoDB is on the cloud, which allows for easy backups if the database corrupts in the application, as well as storing information away from the system. This means that if hackers gain access to the backend of the application, they will be unable to modify data held in the database.

## Data

A data mapper and a data access object are used as structural patterns which separates the application layer from the persistence layer, in a way in which both layers can perform independently of each other. The data mapper’s primary use is to encapsulate what is required for the functionality of the application and stores it inside a datastore.

## Session Persistence

Session persistence is present in the web application, which ensures that each user request is sent to the same backend application database, for the duration of the session the user requires. This is implemented to increase the performance of the application, as otherwise the user will have to access data from multiple backend servers, which will increase the traffic of the website, as well as decrease efficiency.

# Testing

The web application was tested in a few different ways, to ensure that the application has the least number of undiagnosed bugs possible, as well as to ensure the application runs smoothly. The main areas of testing done on the application involved unit testing, integration testing and functionality testing, where unit testing would test each single aspect of the application, integration testing would test how different modules of the application interact with each other, and functionality testing to test if the applications functions as expected before being deployed.

## Unit Testing

Unit Testing was undertaken during the development of the application, where methods within the programs code were individually tested, to ensure they work prior to pushing the finished code onto GitHub. This mainly involved testing methods to ensure that add functions, such as adding milestones or adding courseworks would add the former to the database used for storing the courseworks and milestones, as well as testing if names were successfully edited. Limited time was spent on unit testing, due to delays caused by COVID-19.

## Integration Testing

After unit testing, integration testing was undertaken to check if the application worked when combining one or more modules together, this was mainly done by testing to see if certain form submissions would add the completed submission to the database. This test was done for both coursework and milestones, and involved some test logs, showcased in the table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Action** | **Expected Outcome** | **Status** | **Evidence** |
| 1.1 | Add a coursework to the list | Coursework is added to the list, user redirected to homepage | OK |  |
| 1.2 | Add a milestone to a previously created coursework | Milestone is added to the coursework, user is redirected | OK |  |
| 1.3 | Editing a milestones name | Milestone is edited, user is redirected | OK |  |
| 1.4 | Editing a milestones date | Milestone is edited, user is redirected | OK |  |
| 1.5 | Editing a milestones description | Milestone is edited, user is redirected | OK |  |
| 1.6 | Delete a milestone | Milestone is deleted, database is updated, user is redirected | Fail |  |

The application passed 5 out of 6 tests when the coursework and milestone modules were integrated with the main code. The delete feature will be programming in a later implementation.

## Functionality Testing

Functionality Testing was undertaken at the end of the development life cycle and involved testing the application when everything was fully integrated and ready to be deployed. A variety of testing was done on each page of the website to ensure the user experience was as bug free as possible. These tests have been detailed inside the table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Action** | **Expected Outcome** | **Status** | **Evidence** |
| 1.1 | Register clicked on welcome page | User is redirected to register page | OK |  |
| 1.2 | Login clicked on welcome page | User is redirected to login page | OK |  |
| 2.1 | Login button pressed with empty fields | User is unsuccessfully logged in, page refreshes | OK |  |
| 2.2 | User attempts to login without registering beforehand | User is unsuccessfully logged in, page refreshes | OK |  |
| 2.3 | User attempts to log in with invalid password | User is unsuccessfully logged in, page refreshes | OK |  |
| 2.4 | “No Account? Register” is clicked | User is redirected to register page | OK |  |
| 2.5 | Login is clicked when email and password are valid | User is redirected to coursework page | OK |  |
| 3.1 | Register is pressed with empty fields | User does not register, page is refreshed | OK |  |
| 3.2 | Register is pressed when user tries to register with existing email | User does not register, page is refreshed | OK |  |
| 3.3 | Register is pressed when email field does not have “@” sign | Prompt warns user email is invalid | OK |  |
| 3.4 | Register is pressed when email field does not contain any text after “@” sign” | Prompt warns user email is invalid | OK |  |
| 3.5 | “Have An Account? Login” is pressed on register page | User is redirected to login page | OK |  |
| 3.6 | User inputs valid fields, clicks register | User is successfully registered, redirected to coursework dashboard | OK |  |
| 4.1 | Add Coursework is clicked | User is redirected to add coursework page | OK |  |
| 4.2 | Coursework planner is clicked | User is redirected back to coursework page | OK |  |
| 4.3 | Add button is clicked with empty fields | Prompt warns user fields are missing | Fail |  |
| 4.4 | Add is clicked with invalid date entry | Prompt warns user date is invalid | OK |  |
| 4.5 | Add is clicked with a date already passed | Prompt warns user date has already passed | Fail |  |
| 4.6 | Add is clicked with valid field entries | Coursework is added, user is redirected to coursework page | OK |  |
| 5.1 | Add Milestones is clicked, located in the dropdown menu under milestones heading | User is redirected to add milestone page | OK |  |
| 5.2 | Add is clicked with empty fields | Prompt warns user of empty fields | Fail |  |
| 5.3 | Add is clicked with invalid date | Prompt warns user of invalid date | OK |  |
| 5.4 | Add is clicked with date already passed | Milestone is added, user is redirected to coursework page | OK |  |
| 5.5 | Add is clicked with valid fields | Milestone is added, user is redirected to coursework page | OK |  |
| 6.1 | View Milestones is clicked, located in the dropdown menu under milestones heading | User is redirected to edit milestones page | OK |  |
| 6.2 | Edit Milestone is clicked | User is redirected to edit milestones page | OK |  |
| 6.3 | Milestone is edited to have empty fields | Prompt warns user of empty fields | Fail |  |
| 6.4 | Milestone is edited to have invalid date field | Prompt warns user of invalid date | OK |  |
| 6.5 | Milestone is edited with valid field entries | User is redirected to coursework page | OK |  |
| 6.6 | Delete is pressed | Milestone is deleted | Fail |  |
| 7.1 | Show Complete is clicked, located in the dropdown menu besides “Show” in the navbar | User is redirected to completed courseworks page | OK |  |
| 7.2 | Show incomplete is clicked, located in the dropdown menu besides “Show” in the navbar | User is redirected to incomplete courseworks page | OK |  |
| 7.3 | Show All is pressed | User is redirected to coursework page | OK |  |
| 8.1 | Share is pressed, located in the dropdown menu in the more column of the table | User is redirected to share coursework page | OK |  |
| 8.2 | Go Back is pressed | User is redirected to coursework page | OK |  |
| 8.3 | Delete is pressed | Coursework and accompanied milestones are deleted, coursework page reloaded | OK |  |
| 8.4 | Edit is pressed | User is redirected to edit coursework page | OK |  |
| 9.1 | Confirm is pressed on edit coursework page with empty fields | Prompt warns user of empty fields | Fail |  |
| 9.2 | Confirm is pressed with invalid due date | Prompt warns user of invalid date entry | OK |  |
| 9.3 | Confirm is pressed with invalid completion date | Prompt warns user of invalid date entry | OK |  |
| 9.4 | Confirm is pressed with valid field entries | User is redirected to coursework page | OK |  |

## Summary

40 of 46 tests passed, with 6 tests failing their objectives. Tests 4.3, 5.2, 6.3 & 9.1 all failed due to the application having a lack of verification. Due to time constraints, the developers were unable to implement verification before the deadline, and this will be implemented in a future patch.

Test 4.5 failed as the application accepted a date due which had already passed in real time. After this test was ran, a discussion was held and was ultimately decided that entering a passed date is considered OK, as some users may be tracking work already done when first using the application, especially when milestones have been completed, but the overall coursework is still in progress.

Test 6.6 failed as the milestone was not deleted when the button was pressed. Currently, this issue has not been fixed, and the team are still looking into the issue. A workaround for this is to delete and remake the entire coursework, however this is a very long-winded work around, and can also affect the user if the coursework has many milestones already implemented.

The testing undertaken shown in the table above was done with the user “[test@test.com](mailto:test@test.com)” and the password “123456”.

# Application Security

During the first implementation of the application, security was the primary focus, as a user’s email and password are considered sensitive data, which could become an issue for the team if the sensitive data were hacked into. Specific additions to security were done in this application, with the thought that as web applications are always online, the whole application, including the client-facing and back-end sections, are suitable for an attack by hackers. If either section is hacked, the application could go offline, and sensitive data could be stolen.

To ensure that the password chosen by the user was decently secure, the website requires the password to be at least 6 characters long. This was implemented to increase user security, and potentially influence the user to use a different password than they normally use, if their normal password is less than 6 characters long.

Additionally, a system was implemented which would deny users access to any other page on the website other than login and register, if they are not already logged in. This was done, as it would be impossible to monitor a user creating courseworks or milestones if they were not logged in. If a coursework could be created without requiring an account, all other unregistered users would be able to access the information stored on the courseworks and milestones by editing the URL to redirect them to the specific pages.

## Hashing

To ensure that hackers would not be able to immediately access a user’s password, the team has implemented hashing, specifically Bcrypt’s hashing function. Hashing a password includes using a hash function within the code, which scrambles the password using a set algorithm within the code. When a user attempts to login, the password they use to login is hashed and then compared to the password already stored, as this is more secure. A session secret is also used in the application to compute the hash, as without a session secret, the access to the session would be denied. Hackers, however, can still decrypt a hashed password through a brute-force attack, meaning they constantly attempt to enter a password by continually generating hashes until it matches the user’s password. To counteract this, we have implemented salting.

## Salting

Salting a password includes adding a string of text at the end of a user inputted password before scrambling it using a hash function. This greatly increases the strength of any user inputted password, as it adds a select amount of alphanumerical characters at the end of the password string before encrypting. This makes it significantly harder for hackers to brute-force, as the password now has a higher number of characters, as well as the password not becoming a often used password, such as “abcdef” or “123456”.

The team believes that these security measures will be sufficient for now, but as more users attempt to use the application, further measures would be considered, such as monitoring the use of cookies.