

**Web Platform Development 2**

**Group Report**

BSc Computing, Year 3, Group B

Module Code: M3I322955

Group Members:

Emma Haughey, Student ID: S1712887

Callum Barraclough, Student ID: S1715166

Srujana Chamrolia, Student ID: S1737892

Xiaobin Ma, Student ID: S1803078

The statement: We declare that all work submitted for this coursework is the work of Emma Haughey, Callum Barraclough, Srujana Chamrolia and Xiaobin Ma alone unless stated otherwise.

Contents

[**1.** **Introduction** 3](#_Toc39593144)

[**2.** **Link design** 4](#_Toc39593145)

[**3.** **Persistence** 6](#_Toc39593146)

[**4.** **Test reports** 7](#_Toc39593147)

[**4.1.** **Unit tests** 7](#_Toc39593148)

[**4.2.** **System Test** 7](#_Toc39593149)

[**5.** **Application Security** 9](#_Toc39593150)

[**5.1. Sessions** 9](#_Toc39593151)

[**5.2. User Input** 10](#_Toc39593152)

[**5.3. Passwords** 10](#_Toc39593153)

[**6.** **Conclusion** 11](#_Toc39593154)

1. **Introduction**

For our second coursework in the Web Platform Development 2 module we were tasked with creating a web based coursework scheduling application, which was to include the ability for the user to list all of their current coursework projects, define their own milestones for each of their own coursework projects as well as link these projects to an account system, with which they can use to view all their current projects on their other devices. Our final deliverable can carry out these functions, so in this report we will explore what we accomplished with our final deliverable, detailing our link design and how we’ve managed to keep it consistent, data persistence in our application and how it works, reports from our final tests of the application, as well as a review of the security of the application.

1. **Link design**

For our application, it was important for our website URLs to be readable by the user, so that they can determine where they are in the website at that current moment, as well as allowing easy access for future pages and functionality to be added to the application. It is also important for these URLs to not contain personal information on the user or contain other information that users can then possibly use to gain access to a user’s personal information. In the case of this application, the user will be inputting the details of projects they are working on, so the database for the application shouldn’t contain much personal information on the user, but it still remains important to protect their data, as the user may not want their projects to be visible by other users. In the title bar for each page, we display the name of each page, so that the user is made aware of what page they’re currently on, whilst in the URL we give the name of the page they are on within the part of the URL being displayed after a forward slash. To give an example we can use the registration page, which contains the normal URL for the application, but it also includes ‘/register’ at the end of the URL; this helps define what page the application is currently on, as well as describe to the user what page they are currently on. These sorts of URLs also set us up well for the ability to add additional pages with additional features in the future, as we can create the page, link it to the application and then set up a URL the application can navigate. If, for example, we wanted to add a page to allow the user to edit different settings, including notification and email settings, we would be able to add it using ‘/settings’ at the end of the URL.

The application itself is straightforward to use and navigate; starting with the login page you can see user entry fields for their username and their password, after entering their credentials into these fields they can then progress to their home page. If the user hasn’t already registered for the application, they can go through a registration process, with them being asked to put in their name, password, program of study and their age. These will help determine what modules their projects will belong to, so that results when adding a project can account for this and show relevant results. It also helps with preserving data given to the system, which will be discussed further in the next section. Once the registration process has been completed, the user will be directed to their home page, from where they can start adding projects. From here the user will navigate to a page where they will enter the details of the project they are working on, with the page for adding a project including fields for the title of the project, the milestones the project will adhere to, its due date, the date it was completed (if it has been completed) and whether it has been completed or not. After the user has completed all the fields on the page, they will be redirected back to their home page, where it will list all their current projects. The user can then go on to add more projects, modify existing projects, delete added projects and share projects with friends. Overall, we feel our URL design allows us the greatest amount of freedom we can have to add in additional pages and features at a later stage if needed, as well as ensuring the user can properly distinguish where they are in the application.

1. **Persistence**

Persistence involves data that is preserved between different user sessions, preserving the previous version of itself for later, whether that is for an online shop and orders made by a user, or for a social network and all posts that have been made by users. Data in our application is persisted between different user sessions with the use of a database, with the data created being saved to the database for later use, either for displaying created user projects on their home page when the user logs back into the system, or for holding any settings made by the user within the application. The data persisted between sessions in our application include:

* User’s list of projects
* Project titles
* The project’s module
* The project’s milestones
* The project’s due date
* The project’s completion date
* The completion status of projects

It is important for all this data to be persisted between user sessions as the main purpose of the app is for the user to add all their projects to the application’s database for safe keeping, as well as a list that the user can refer to later on to remind them of their outstanding work. If the system was to wipe this data between visits, the user would not be able to remember all their current projects, leading to possible future problems for them, or it would to lead to frustration at having to re-enter projects they had entered into the application previously, pushing them to seek alternative solutions to tracking all their projects. The way persistence is done in our database is through assigning inputted data to different variables; these data sets then get given an identification string to make each one of them unique, whilst they get stored alongside the other data sets in the database. Overall, we feel this is the most successful way of ensuring data persistence and doesn’t currently require further additions.

1. **Test reports**

Testing is especially important for a project. Because it can ensure that the final product meets the specifications. In the development process of the application, testing usually have four different steps and in this project will mainly use unit testing and system testing.

* 1. **Unit tests**

In this application, unit tests are used to test whether the user login function meets requirements. There are four main scenarios:

* The username and password match the information in the database;
* The username matches but the password does not match;
* The username does not match (include the case sensitive) and the password matches;
* The account does not exist.

The test can only pass if the username and password match the information in the database. The other three cases should fail the test. Because the test was conducted before development, the first step was to create a new array and set some values to simulate the database to store user information. The jest method is used to do the test. It simulates user input, and then compare the input value with the information in the array for testing. Only when the input data and the data in the array completely match can pass.

* 1. **System Test**

Before completing the unit test and pushing the product, a system test will be conducted on the entire project. This will ensure that the project meets the requirements and can be pushed. This chapter will use the login interface as an example to explain how to run System Test.

System will test the user's input, buttons, and page links in the login interface. The table and test results used in the test are as follows:

Table Test table for login page

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Action** | **Expected outcome** | **Status**  **(Pass / Fail)** | **Prove** |
| 1 | Username Box | Display ‘username’ prompts the user to enter username | Pass |  |
| 2 | Password Box | Display ‘password’ prompts the user to enter password | Pass | A screenshot of a social media post  Description automatically generated |
| 3 | User input Username | Cursor appears, the prompt text disappears when the user enters information, and the user input information is displayed | Pass | A screenshot of a social media post  Description automatically generated |
| 4 | User input Password | A cursor appears, the prompt text disappears when the user enters information, and \* replaces the information entered by the user | Pass | A screenshot of a social media post  Description automatically generated |
| 5 | Login Button  (The user enters the correct username and password) | Page Jump to Home Page | Pass | A screenshot of a cell phone  Description automatically generated |
| 5.2 | Login Button (Username or password mistake include account do not exist) | Display Username/Password incorrect! | Pass | A close up of a logo  Description automatically generated |
| 6 | URL Design | Display ‘http://localhost:8000/’ | Pass | A screenshot of a cell phone  Description automatically generated |
| 7 | Create an account button | Page jump to registered | Pass | A screenshot of a cell phone  Description automatically generated |
| 8 | Access modify coursework page | Show the modify coursework page, show all coursework for the user | Pass |  |
| 9 | Click the edit button for the coursework “Coursework Scheduling App” | Show the modify coursework page for the selected coursework | Pass |  |
| 10 | Modify the selected coursework and submit | The coursework successfully modified message is shown and the data successfully changed | Pass |  |
| 11 | Access the share coursework page | Show the share coursework page | Pass |  |

As you can see, our application managed to pass all the tests we gave it to complete, showing we were able to implement the functionality required in the coursework brief successfully.

1. **Application Security**

Since our application will have to deal with multiple user’s passwords and personal details, relating to their course, projects they have for their course, amount of work done, it is a good idea to make sure the application we’ve built is secure, so that their privacy is protected and respected, as well as to ensure that users keep using our program, instead of using alternate ones to keep track of their projects, which would make our application not worth keeping up to date and maintained. This section will run through a few of the possible security concerns had in relation to the system, starting with the way the application handles sessions, then with the way it handles user input and finally with the way it handles passwords.

## **5.1. Sessions**

Sessions, in the context of an application, are defined as the series of user interactions between the user and the application they are using, with these sets of interactions being tracked by the server. They are mainly responsible for containing the current state of the user, involving storing objects that are essential and beneficial to the running of the application, as well as successful user verifications (like when the user logs in using a username/email address and password). The session will be contained in the server for later use, for which the ID will be contained within a cookie on the client side for then retrieving the data stored with that ID. The cookie and the session can last for any length of time; for many systems (like social media) this cookie lasts forever. In our application, the user logs in via the login page, or creates their account, then they are redirected to their home page, where they will be able to see all their projects. Whilst they are in the application, they can do any of the functions of the application for any period of time, including adding projects, modifying projects and deleting projects. However, in the current version of the application, once the user closes the application, they will be logged out of the application. In most modern applications they default to a system where the user is kept logged in, until either the user logs themselves out, or the system does it automatically after a period of time, if the user chooses not to be kept logged in and the application includes this option. Whilst this may lead to user frustration, as they will have to login each time they want to access the application, this way is more secure, as the information won’t be accessible by another user using the same computer and the same application, preventing them from accessing anything contained within the original user’s application and maintaining the user’s privacy. With this said, a way to make the sessions more secure would be to protect the session with encryption, like https, so that any people outside the system have direct access to the users’ data blocked.

## **5.2. User Input**

User input can be highly variable depending on their skill level with computers, as well as be the thing that can lead to security breaches if certain vulnerabilities aren’t accounted for. Whilst many ways in which users can exploit system are difficult to account for, there are some ways user input which goes outside acceptable bounds can be removed. If, for example, a user was not to use the system for its intended purpose, creating projects which are false and contain data considered unacceptable, like swearing, racism or other content similar to it, the system can be set up to detect this content, present the user with warnings for the first two times they do it, with it escalating to suspension if it continues. However, an important thing to consider in this is that other users’ privacy could be breached if the system monitoring is carried out by people, so a better way to do this, at least for most cases, would be to have this system be automated, avoiding any breaches of user security and trust. User generated input which uses code to take over the user’s program, like when a tweet on Twitter contained code which made the user’s application retweet that tweet, would have to be dealt with on a case by case basis, although following existing security guidelines could help avoid some possible attacks.

## **5.3. Passwords**

Passwords are used a form of authentication against the user, done when the user goes to log into an application, to ensure that the user logging in is the person they say they are in the username/email field. A way to protect passwords from being exposed, even if the system’s security is breached and the attacker sees the passwords, is to encrypt existing passwords, like through a process similar to hashing and salting, when they are created and added to the system, lengthening the time taken for the real password to be discovered, or make the real password unfindable by the attacker. Our system unfortunately doesn’t currently encrypt user’s passwords, storing it as plain text to our database, which would allow easy access to all users’ passwords if the system was broken into. The main ways to prevent this would be to, as mentioned before, use encryption to change the existing password into a version which is unreadable to a normal user, which is what hashing does, along with salting, which randomizes each created hash to make sure the hashing algorithm isn’t easily predictable.

1. **Conclusion**

In conclusion, the application manages to have solid URL design by having readable URLs for users, as well as allowing easy access for additional web pages with additional functionality in the future of the application, has good data persistence, as it maintains all users’ projects by backing them up to a database for later retrieval, working functionality, as shown in our tests earlier in the report, as well as an acceptable level of security, as whilst it doesn’t encrypt user passwords and no auto moderation system for dealing with offensive content, it has a system for signing the user out to prevent access by other users on the same machine. We feel overall that the application we have created successfully delivers on what was defined in the coursework brief, as it satisfies most of the criteria given to us, whilst delivering an application which would genuinely be useful to any user who wishes to use it.