



University for the Common Good

**School of Engineering and Built  
Environment**

**Dept. of Computer, Communications  
& Interactive Systems**

**Web Platform Development 2**

**M3I322955**

**Trimester B**

**Session 2019/2020**

**Coursework 2**

**Group Report**

**Group H**

**<https://webdev2-coursework.herokuapp.com/>**

**Paul Harbison-Smith – S1712745 – pharbi200@caledonian.ac.uk**

**Jay Malley – S1703629 – jmalle200@caledonian.ac.uk**

**Jordan McCormack – S0933311 – jmccor204@caledonian.ac.uk**

**Isaac Waldron – S1715300 – iwaldr200@caledonian.ac.uk**

## Table of Contents

<b>1. Link Design</b>	<b>1</b>
1.1. URLs	1
1.2. Page Access and Functionality	1
1.3. Justification	2
<b>2. Persistence</b>	<b>3</b>
<b>3. Testing</b>	<b>4</b>
3.1. Acceptance Testing	4
3.2. Unit Testing Test Cases	6
<b>4. Security</b>	<b>9</b>
4.1. Implemented Security Measures	9
4.2. Security Considerations (Not Implemented)	9
4.2.1. Input Validation	9
4.2.2. Passwords	10
4.2.3. Shareable Link (Coursework)	10
4.2.4. Sessions	10
4.2.5. AJAX	10
4.2.6. HTTP Protocols	10
4.2.7. Cross-site Scripting (XSS)	11
<b>5. References</b>	<b>12</b>

## 1. Link Design

### 1.1 URLs

The URLs used for this project are as such:

- <https://webdev2-coursework.herokuapp.com/>
  - Uses a singular /, if you have session data then you are immediately taken to the Home page otherwise it renders the Login page
- <https://webdev2-coursework.herokuapp.com/register>
  - Renders the Register page
- <https://webdev2-coursework.herokuapp.com/home>
  - Retrieves session data, ensures it exists and if it does then the user is taken to the Home page and the NeDB database is searched using the SearchById() function and retrieves the username and course, which is used to render the Home page and display coursework details
  - We were originally posting user ID to the Home page and rendered the Home page using the POST method instead of the session details, which meant that these links could not be safely and reliably bookmarked because you cannot bookmark POSTs since information has to be inputted and processed by a POST method in order for it to be used, which is not the case with session details where we can use a SearchById() function and GET user details (i.e. we can simply GET details we have stored rather than having a user send data every time they access the application)
- [https://webdev2-coursework.herokuapp.com/view?view=\[COURSEWORKID\]](https://webdev2-coursework.herokuapp.com/view?view=[COURSEWORKID])
  - Used to redirect users to the View page for a particular piece of a coursework
  - **[COURSEWORKID]** used as the query string meaning that the user is redirected to the View page for the chosen coursework (coursework ID is shown as a random variable, e.g. \$2b\$10\$5p6tZ7PjC7oHSbtZpvllfe)

### 1.2 Page Access and Functionality

Beginning with the 'entry' page (using the URL, singular /, with no further path), a user is redirected to either the Login page or the Home page depending on whether session data is set or not. Assuming a user starts with the Login page, they can either enter their login details and sign into the application or, if they do not have login details, they can register by clicking the "Register" button and registering an account which can then be used to sign in. Following registration, the new User ID is generated using `bcrypt.gensalt()`, which is cryptographically random. The Register page can be accessed using either the aforementioned "Register" button on the Login page or using the `/register` URL path following the domain.

As discussed above, the Home page can be accessed using the default URL if there is set session data. Otherwise, the Home page is accessed after logging in using the "Login" button or using the `/home` URL path following the domain. The Home page shows the list of coursework set by the user and the user can add coursework, modify coursework, delete coursework, and view coursework (mostly CRUD functionality). Besides this, users may also filter coursework using set parameters (e.g. show only completed coursework, sort by alphabetical order, etc.) When deleting coursework, users can either delete individual coursework items, delete all completed work, or delete all coursework regardless of whether it has been completed or not.

The View page is accessed using the "View" button on the Home page, which is next to each coursework entry. The URL can be used however it needs to be provided in full in advance, using

view?view=[**COURSEWORKID**], since the coursework ID is a randomised variable which, in all likelihood, isn't going to be memorised. With no query string (i.e. just the /view path), a blank page will be returned notifying the user that the wrong coursework ID has been used since there would not be any coursework ID to query. This was done primarily for security reasons since the application would be left vulnerable should any user be able to guess the coursework ID of another user (although this link *can* be shared to other users, as mentioned above). This ensures that each coursework ID is unique for each user. On the View page, users can view details for a piece of coursework and check off completed milestones for said coursework. From the View page, users can return to the Home page using the provided back button.

### **1.3 Justification**

This schema was chosen as a simple, readable representation of application functionality, with each path being named after its respective page wherein there is virtually no need to change the names of these paths, ensuring the URLs are bookmarkable. These links are relatively consistent and come from a single domain (webdev2-coursework.herokuapp.com). Furthermore, these links are concise and secure: only relevant data is shown for redirection purposes (e.g. the use of the coursework ID for the View page, which itself is randomised as discussed previously) and no private or irrelevant data is shown (e.g. no useless search terms are used nor are session IDs shown). Clear, consistent, and concise links are important when developing a user-focused experience where users can intuitively understand how the webapp is structured and therefore how to get from page to page.

The application and, by association, the URLs are extensible and future-proof due to the use of a consistent and uniform hierarchical naming structure. Should further functionality or new pages be added to the application, the namespace can simply be changed using a new URL path. Also, should any changes be made to the naming of an existing page users can simply be redirected to the updated page using an updated namespace, keeping changes to a minimum and ensuring a fluid, unaffected user experience.

## 2. Persistence

In this webapp, a database is required to both save user coursework projects as well as store individual user accounts.

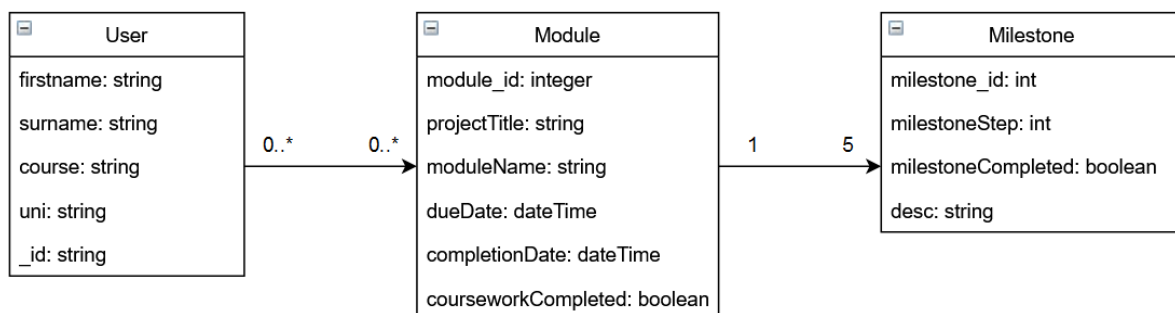
To persist this data, we used a NeDB with NoSQL database with the data saved in JSON format, which integrates perfectly with Node.js as it is written in JavaScript. The NoSQL format allows us to change the way in which we store data without having to make massive changes to the database itself, as well as allowing us to input new information if we require it without having to interact with the database itself.

The file structure of the database starts with a user which encompasses everything else and this has been illustrated in Figure 1.1. The User object has in itself: firstname, surname, course, uni\_id and module. All of which are stored as strings except for the module which is an array of module type objects.

The Module object has in itself: module\_id, which is an integer, projectTitle and moduleName, which are both strings, dueDate and completionDate is dateTime, courseworkCompleted which is boolean and milestones which is an array of objects of type "Milestones".

The final object type in the database is the Milestone object. This has milestone\_id and milestoneStep both of which are integers, milestoneCompleted is a Boolean and finally desc which is a string.

**Figure 1.1** Class diagram of data persistence structure



To manipulate the data in the database, we have a JavaScript file dedicated to doing so that itself is controlled using the controller. The controller has endpoint URLs which we call using either a simple form POST or most often we call the endpoint using an AJAX function in the client-side JavaScript. We decided to use AJAX calls instead of using forms in the vast majority of cases to interact with the endpoints, due to the asynchronous nature of AJAX this allows us to manipulate the database without having to reload the page which provides for a more fluid experience for the user improving user experience.

The model used to access the database itself has a variety of functions that allows it to manipulate and get all of the objects in ways which we require. For example, we can insert new User objects and delete that record with everything else in it deleted and we can search for these users in two ways:

- by id which is randomly generated
- by name, of which we do not allow duplicates

All of the functions written to manipulate the database were designed in a way in which it will allow us in the future to write new database actions on top of them to support extensibility and scalability.

### 3. Testing

#### 3.2.1 Acceptance Testing

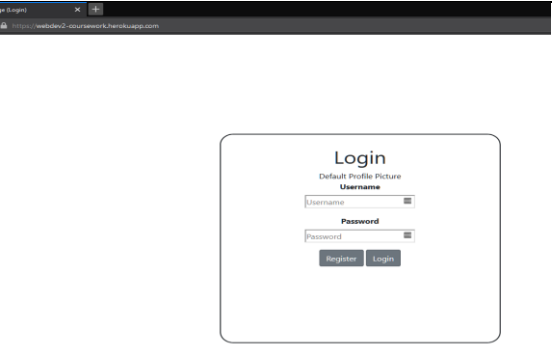
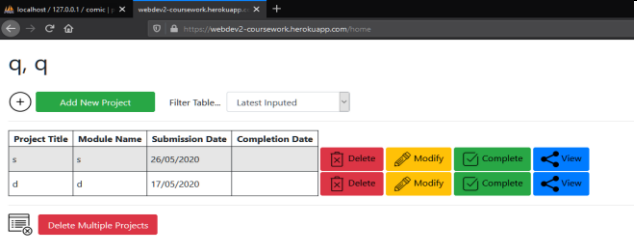
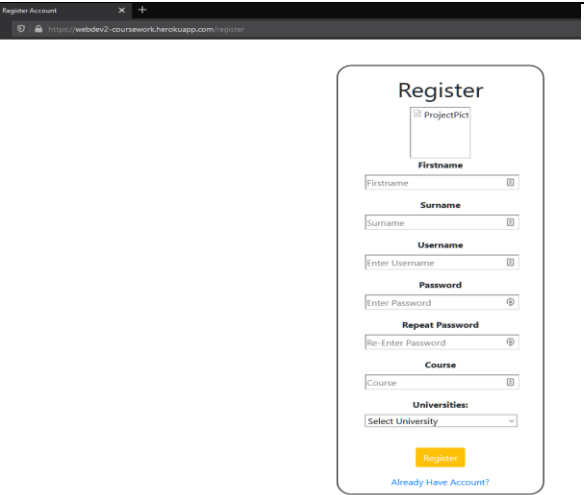
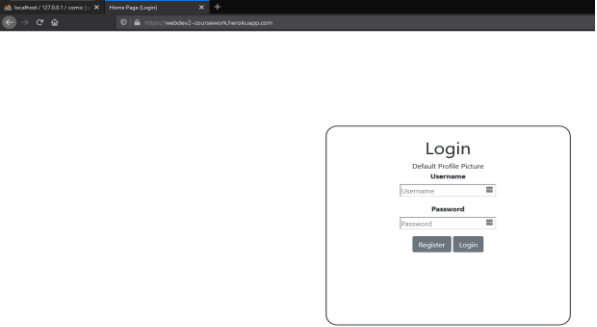
The main form of testing that was used to test our webapp was acceptance testing. Acceptance testing is a test conducted to determine if the functionality that has been laid out in the functionality documentation has been met.

The results of this testing will conclude whether our website was successful in hitting all of our laid-out functionality or not.

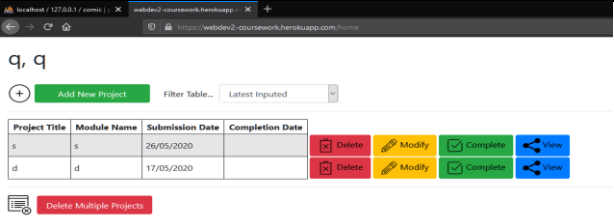
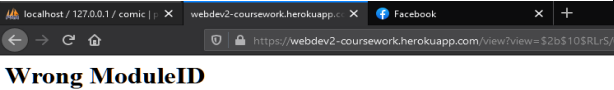
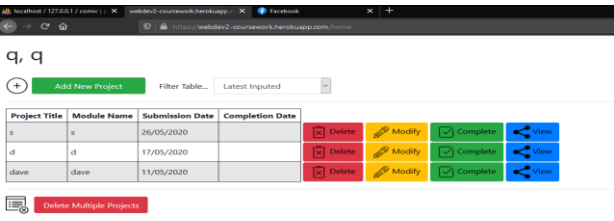
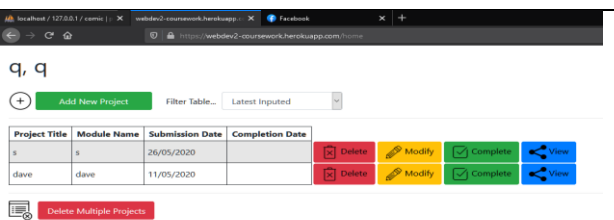
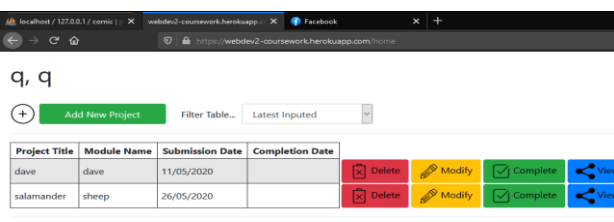
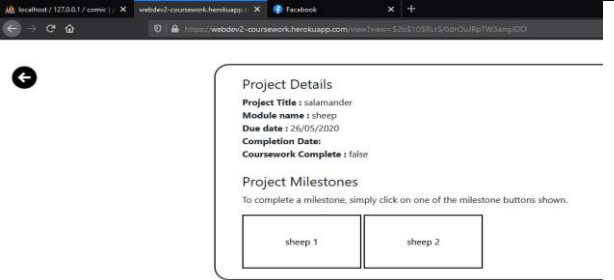
Sr.	Functionality	Typical Components	Detailed Description	Delivered
1)	Registration	Form	On screen load user should be promoted with a form that they are then able to fill in with their details and press submit	✓
		Registration Event	Once the form has been filled in and pressed submit the user's details should then be validated	✓
		Error Message	If there is an error in registering the user an error message should appear telling the user what is wrong i.e. Please insert your email address	✓
		Details are stored	Details should then be successful stored in our databases	✓
2)	Login	Form	On screen load user should be promoted with a login form where they can fill in their username and password	✓
		Login Event	Once the user has filled in the form and pressed submit the user's details should then be validated	✓
		Error Message	If there is a problem logging in an error message should be displayed letting the user know what is wrong, i.e. Incorrect details	✓
		Access	Once details have been successful validated the website should create a session with the user logging them in and giving them access to the main content of the site	✓
3)	Add new coursework	Form	Be promoted with a add new coursework form which can be fill in with the details of their coursework	✓
		Store coursework	Once the details have been validated, they should then be successful stored in the database	✓
		Update	The table should update asynchronously to show the new coursework	✓

4)	Remove coursework	Button	Have a button beside each coursework in the table that can be pressed on to remove the coursework in that row	✓
		Mass deletion	Be able to press on a button where they can select if they wish to mass delete all project or all complete projects	✓
		Remove coursework	The coursework(s) in question should then be removed from the database	✓
		Update	Since we are making use of AJAX the table should update asynchronously removing the coursework from it	✓
5)	Modify coursework	Form	The user should be given a modify button at the end of each row, once this has been pressed a form should pop up with the field already filled in with the coursework in questions details	✓
		Update Database	The system should then validate the details that the user has or has not changed and update only the details that have been changed	✓
		Update table	The table should then update asynchronously with the modified fields in it	✓
6)	Filter	Uncompleted	Update table to show uncompleted projects	✓
		Completed	Update table to show completed projects	✓
		Ascending alphabetical order	Update table in ascending alphabetical order of project titles	✓
		Descending alphabetical order	Update table in descending alphabetical order of project titles	✓
		Earliest Due Date	Update table to show earliest due dates first	✓
		Latest Due Date	Update table to show latest due dates first	✗
7)	View page	Button	Be able to press a button at the end of the coursework row to be taking to that coursework's page	✓
		URL	A link will be generated which can then be shared with other users to see the coursework in question	✓
		Milestones Checker	Milestones should be able to be checked off once complete which will update the database and set the milestones background to green to show that it has been updated successfully	✓

### 3.2.2 Unit Testing Test Cases

Test ID	Action	Expected Outcome	Status	Evidence
1.1	localhost:3000 (no session data)	Login page loaded	OK	
1.2	localhost:3000 (with session data)	Redirect to /home	OK	
1.3	localhost:3000/register	Register page loaded	OK	
1.4	localhost:3000/home (no session data)	Redirect to /	OK	



1.5	localhost:3000/home (with session data)	Home page loaded	OK	
1.6	localhost:3000/view (Wrong ID)	Wrong module ID prompt shown	OK	
2.1	Add Project	Project added	OK	
2.2	Delete project	Project deleted	OK	
2.3	Modify project	Project modified	OK	
2.4	View project	/view with correct project details	OK	

2.5	Delete completed projects	All completed projects deleted	OK	<div><div>DeVenema, Computing</div><div><div><div><div></div></div><div>Add New Project</div><div>Filter Table...</div><div>Latest Inputed</div></div></div><div><table><tr><th>Project Title</th><th>Module Name</th><th>Submission Date</th><th>Completion Date</th><th></th><th></th><th></th><th></th></tr><tr><td>Coursework 1</td><td>WPD2</td><td>18/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr><tr><td>Coursework 2</td><td>WPD2</td><td>16/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr><tr><td>Group Report</td><td>IP3</td><td>31/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr><tr><td>Poster</td><td>RSP1</td><td>21/05/2020</td><td>05/05/2020</td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr><tr><td>Project Proposal</td><td>RSP1</td><td>31/05/2020</td><td>05/05/2020</td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr></table></div><div><div><div></div></div> Delete Multiple Projects</div></div> <div><div>DeVenema, Computing</div><div><div><div><div></div></div><div>Add New Project</div><div>Filter Table...</div><div>Latest Inputed</div></div></div><div><table><tr><th>Project Title</th><th>Module Name</th><th>Submission Date</th><th>Completion Date</th><th></th><th></th><th></th><th></th></tr><tr><td>Coursework 1</td><td>WPD2</td><td>18/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr><tr><td>Coursework 2</td><td>WPD2</td><td>16/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr><tr><td>Group Report</td><td>IP3</td><td>31/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr></table></div><div><div><div></div></div> Delete Multiple Projects</div></div> <div><div>DeVenema, Computing</div><div><div><div><div></div></div><div>Add New Project</div><div>Filter Table...</div><div>Latest Inputed</div></div></div><div><table><tr><th>Project Title</th><th>Module Name</th><th>Submission Date</th><th>Completion Date</th></tr><tr><td>Coursework 1</td><td>WPD2</td><td>18/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr><tr><td>Coursework 2</td><td>WPD2</td><td>16/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr><tr><td>Group Report</td><td>IP3</td><td>31/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr></table></div><div><div><div></div></div> Delete Multiple Projects</div></div> <div><div>DeVenema, Computing</div><div><div><div><div></div></div><div>Add New Project</div><div>Filter Table...</div><div>Latest Inputed</div></div></div><div><table><tr><th>Project Title</th><th>Module Name</th><th>Submission Date</th><th>Completion Date</th></tr></table></div><div><div><div></div></div> Delete Multiple Projects</div></div>	Project Title	Module Name	Submission Date	Completion Date					Coursework 1	WPD2	18/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Coursework 2	WPD2	16/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Group Report	IP3	31/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Poster	RSP1	21/05/2020	05/05/2020	<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Project Proposal	RSP1	31/05/2020	05/05/2020	<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Project Title	Module Name	Submission Date	Completion Date					Coursework 1	WPD2	18/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Coursework 2	WPD2	16/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Group Report	IP3	31/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Project Title	Module Name	Submission Date	Completion Date	Coursework 1	WPD2	18/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Coursework 2	WPD2	16/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Group Report	IP3	31/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Project Title	Module Name	Submission Date	Completion Date
Project Title	Module Name	Submission Date	Completion Date																																																																																																																	
Coursework 1	WPD2	18/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Coursework 2	WPD2	16/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Group Report	IP3	31/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Poster	RSP1	21/05/2020	05/05/2020	<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Project Proposal	RSP1	31/05/2020	05/05/2020	<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Project Title	Module Name	Submission Date	Completion Date																																																																																																																	
Coursework 1	WPD2	18/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Coursework 2	WPD2	16/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Group Report	IP3	31/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Project Title	Module Name	Submission Date	Completion Date																																																																																																																	
Coursework 1	WPD2	18/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Coursework 2	WPD2	16/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Group Report	IP3	31/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Project Title	Module Name	Submission Date	Completion Date																																																																																																																	
2.6	Delete all projects	All projects deleted	OK	<div><div>DeVenema, Computing</div><div><div><div><div></div></div><div>Add New Project</div><div>Filter Table...</div><div>Latest Inputed</div></div></div><div><table><tr><th>Project Title</th><th>Module Name</th><th>Submission Date</th><th>Completion Date</th><th></th><th></th><th></th><th></th></tr><tr><td>Coursework 1</td><td>WPD2</td><td>18/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr><tr><td>Coursework 2</td><td>WPD2</td><td>16/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr><tr><td>Group Report</td><td>IP3</td><td>31/05/2020</td><td></td><td><div><div></div></div> Delete</td><td><div><div></div></div> Modify</td><td><div><div></div></div> Complete</td><td><div><div></div></div> View</td></tr></table></div><div><div><div></div></div> Delete Multiple Projects</div></div> <div><div>DeVenema, Computing</div><div><div><div><div></div></div><div>Add New Project</div><div>Filter Table...</div><div>Latest Inputed</div></div></div><div><table><tr><th>Project Title</th><th>Module Name</th><th>Submission Date</th><th>Completion Date</th></tr></table></div><div><div><div></div></div> Delete Multiple Projects</div></div>	Project Title	Module Name	Submission Date	Completion Date					Coursework 1	WPD2	18/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Coursework 2	WPD2	16/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Group Report	IP3	31/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View	Project Title	Module Name	Submission Date	Completion Date																																																																												
Project Title	Module Name	Submission Date	Completion Date																																																																																																																	
Coursework 1	WPD2	18/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Coursework 2	WPD2	16/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Group Report	IP3	31/05/2020		<div><div></div></div> Delete	<div><div></div></div> Modify	<div><div></div></div> Complete	<div><div></div></div> View																																																																																																													
Project Title	Module Name	Submission Date	Completion Date																																																																																																																	
2.7	Check off milestone	Milestone can be successfully marked as completed and uncompleted	OK	<div><div><div>Project Details</div><div><div>Project Title : Poster</div><div>Module name : RSP1</div><div>Due date : 14/05/2020</div><div>Completion Date:</div><div>Coursework Complete : false</div></div><div><div>Project Milestones</div><div>To complete a milestone, simply click on one of the milestone buttons shown.</div><div><div><div>Problem Statement</div><div>Project Outline</div><div>Ethical/Legal Issues</div></div></div></div></div><div><div><div>Project Details</div><div><div>Project Title : Poster</div><div>Module name : RSP1</div><div>Due date : 14/05/2020</div><div>Completion Date:</div><div>Coursework Complete : false</div></div><div><div>Project Milestones</div><div>To complete a milestone, simply click on one of the milestone buttons shown.</div><div><div><div>Problem Statement</div><div>Project Outline</div><div>Ethical/Legal Issues</div></div></div></div></div></div></div>																																																																																																																

## 4. Security

Web security is a necessary consideration when developing any online service designed to have user interaction, as such we have taken the initiative to deploy security features for the website in this coursework. This section will discuss the implemented security features to provide the website with baseline security measures, as well as displaying an awareness of additional security risks and solutions.

### 4.1 Implemented Security Measures

Initial security measures begin with standard user input validation. Input forms on the website are validated on both the client-side via JavaScript, as well as the server-side before being used within code blocks to confirm the input is suitable. The reason for this is as follows:

- To enforce input correctness. E.g. values are correct type or within range.
- To ensure the server can perform required task.
- To check input syntax / input format is correct, e.g. Date.

(Open Web Application Security Project, 2020)

The next logical step in security from standard input validation was to protect against code injection, a process where an attacker constructs a query to be executed by the server-side when posted. This was done in two ways:

- Writing robust enough code which will not blindly accepting inputs.
- Preparing the input by sterilising it e.g. formatting input as a string.

(Belmer, 2019)

Following input validation, the second security concern was to securely store data in the database, namely account details to protect user's personal data. To ensure the safety of registered users even in the event of a database breach, passwords should not be stored as plain-text, but instead can be encrypted by using a 'hash' (a one-way function to turn plain-text into a fixed-length 'fingerprint'). However, this alone is not enough, as if two users share the same password, they will have the same hash, to prevent this password were given a 'salt' (a randomised string attached to the end of the password). Both hash and salt generation were provided by the Node.js framework. Passwords were also case sensitive for a slight improvement in security.

Furthermore, thanks to comprehensive planning and close examination, the website only stores necessary data that is required for proper functionality, thus preventing the risk of user's personal information being stolen.

### 4.2 Security Considerations (Not Implemented)

#### 4.2.1 Input Validation

Outside of simply validating inputs are correct in type and range, it is also important to write vigilant checks against potentially malicious inputs. While it is common for developers to utilise input blacklists to detect possible dangerous characters this does not entirely stop malicious users which may be able to find alternative inputs, for this reason a whitelist may be better suited as they instead check for input to meet a specified criteria determined by the website developer, especially for well-structured but variable data types like date, SSN, zip codes and e-mail addresses. However, Blacklists are particularly useful for detecting and stripping well-known payload inputs, e.g. 'img.src'.

Moreover, when logging login failures, ensure enough meaningful fields exist to identify suspicious accounts and is logged long enough to allow delayed forensic analysis. Similarly, have an alert system in place to detect suspicious activity such as brute force attacks.

#### 4.2.2 Passwords

When a user is registering an account, we could have prompted them to enter a 'secure' password which meets certain criteria, e.g.

- At least 10 characters long
- Requires 1 Uppercase/Lowercase
- Requires 1 Number
- Requires 1 special character (! " £ \$ % ^ & \*)

Additionally, to prevent brute force attacks attempting to guess passwords, simple solutions would be:

- Limited Login Attempts.
- Incorporate Login captcha or Two Factor Authentication checks.

(Rehman, 2018)

#### 4.2.3 Sharable Link (Coursework)

Another security risk arose in the form of the shareable link functionality. Currently, the generated link utilised a GET request to the server, which pulls data from the database with a matching module\_ID to be displayed. This is an inherent problem as the URL is easily editable by the user and can result in users viewing ANY module data if they happen to find a match.

A potential work around we considered was to implement an additional field for the link, which would store an array of "authorised emails" set by the sharing user upon 'generating' the link, however there is still no feature in place from stopping users brute forcing an email match.

#### 4.2.4 Sessions

Sessions are used to maintain a persistent connection between the user and the website, even when moving between pages. Cookies are a popular way of implementing sessions, however they are stored within the browser and are therefore vulnerable not only to 'session hijacking', but also user manipulation. To protect against these attacks, we can use middleware services such as passport.js to serialize and authenticate sessions individually, as well as binding the session ID to other user or client properties such as their IP address to guarantee it is the same real-world user (Open Web Application Security Project, 2020). These changes also make it easier to prevent simultaneous logins of the same account, simply by locking a session username to the initial IP address.

#### 4.2.5 AJAX

The Open Web Application Security Project (OWSAP) lists numerous ways to help secure AJAX calls, including:

- (Client-Side) Use .innerText instead of .innerHTML.
- (Client-Side) Do not rely on client side (business) logic for security of functionality.
- (Server-Side) Use Cross-Site Request Forgery (CSRF) Protection.

(Open Web Application Security Project, 2020)

#### 4.2.6 HTTP Protocols

- Utilise the HTTPS protocol to securely encrypt and transmit usernames, passwords, or tokens.
- Use the HttpOnly flag in the http response header to prevent cookies from being accessed via client-side scripting.

#### **4.2.7 Cross-site Scripting (XSS)**

XSS attacks allow attackers to steal other user's session data through an otherwise benign website by injecting malicious code into the client-side webpage and having users access it. The simplest way to prevent these attacks, besides not using untrusted data, is to implement an HTML mark-up sanitisation library. These libraries validate that HTML is 'clean' (Open Web Application Security Project, 2020). As a bonus, sanitization libraries prevent 'tag poisoning' where invalid tags of HTML can corrupt the whole document causing faulty layouts (Ganss, 2020).

## 5. References

1. Open Web Application Security Project, 2020. *Input Validation* [online]. OWASP. [viewed 28 April 2020]. Available from: [https://cheatsheetseries.owasp.org/cheatsheets/Input\\_Validation\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html)
2. BELMER, C., 2019. *A NoSQL Injection Primer (With Mongo)* [online]. Nullsweep. [viewed 28 April 2020]. Available from: <https://nullsweep.com/a-nosql-injection-primer-with-mongo/>
3. REHMAN, I.U., 2018. *What is a Brute Force Attack?* [online]. Cloudways. [viewed 28 April 2020]. Available from: <https://www.cloudways.com/blog/what-is-brute-force-attack/>
4. OPEN WEB APPLICATION SECURITY PROJECT, 2020. *Session Management* [online]. OWASP. [viewed 28 April 2020]. Available from: [https://cheatsheetseries.owasp.org/cheatsheets/Session\\_Management\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html)
5. OPEN WEB APPLICATION SECURITY PROJECT, 2020. *AJAX Security* [online]. OWASP. [viewed 28 April 2020]. Available from: [https://cheatsheetseries.owasp.org/cheatsheets/AJAX\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/AJAX_Security_Cheat_Sheet.html)
6. OPEN WEB APPLICATION SECURITY PROJECT, 2020. *Cross Site Scripting Prevention* [online]. OWASP. [viewed 28 April 2020]. Available from: [https://cheatsheetseries.owasp.org/cheatsheets/Cross\\_Site\\_Scripting\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)
7. GANSS, M., 2020. *HtmlSanitizer* [online]. GitHub. [viewed 28 April 2020]. Available from: <https://github.com/mganss/HtmlSanitizer>