Steven Lota, James Lawn, Drew Ritchie

Web platform development 2

Team Report

Contents

[**Introduction** 2](#_Toc39698309)

[**Link Design** 3](#_Toc39698310)

[**Register: “/”** 3](#_Toc39698311)

[**Login: “/Login”** 4](#_Toc39698312)

[**Coursework: “/coursework”** 4](#_Toc39698313)

[**Coursework: Create - “/coursework/create”** 5](#_Toc39698314)

[**Coursework: Update - “/coursework/update”** 5](#_Toc39698315)

[**Coursework: Delete** 5](#_Toc39698316)

[**Future Implementation** 6](#_Toc39698317)

[**Persistence** 7](#_Toc39698318)

[**Testing** 8](#_Toc39698319)

[**Acceptance Testing** 8](#_Toc39698320)

[**Acceptance Testing Evidence** 9](#_Toc39698321)

[**Compatibility Testing** 10](#_Toc39698322)

[**Security Testing** 10](#_Toc39698323)

[**Application Security** 11](#_Toc39698324)

[**References** 12](#_Toc39698325)

# **Introduction**

This report describes the process of creating a prototype software application for the Web Platform Development (WPD2) module at Glasgow Caledonian University (GCU), in session 2019-20. The application focuses on helping students cope with the stressful life of university by implementing a web-based coursework organiser application that will display coursework deliverables that the students need to submit over the life of the module. The application will require the user to define their own milestones for each coursework project and are only visible to the user that creates them. Each coursework module requires the following:

* A coursework Title
* The coursework module for which is needs to be submitted for
* Breakdown of coursework milestones
* Intended Due Date
* Completion Date

The application requires the core functionality of CRUD features that is a way of persistence in relation to the database to pull and retrieve data for the application, to specify, the application must allow Coursework’s to be Added, Updated, Read and Deleted. Another feature requires a user to have the ability to another user through a unique URL that will allow them to view a coursework even if it is not tied to their user login as long as they have the URL link. The project is implemented using the node.js runtime environment with node express.

# **Link Design**

This section describes the URL’s used within the web design application and explain if they are appropriate, are distinct and contain what they are supposed to. This program’s intended purpose is to ease the burden of university life for a student by enabling them to sort their coursework’s into manageable segments and having it all centralized within one page. The group has adapted this application for the purpose of making it easier to navigate to each page and know, always, where they are navigating to. After logging in, users will be able to easily navigate to each page using the navbar and buttons that are descriptive and located on each page. In Link Design we have taken into consideration there are three descriptive factors that go into making a good URL:

* Short - Users will appreciate a URL that is short which makes searching more efficient and easier to understand. If a URL appears longer it will require more characters in the search bar to pop up and will prove to have less frequent visitors as one with a shorter more memorable link.
* Descriptive – A URL must be descriptive so the user can tell what where they are going just by looking at the contents displayed on the search bar. It should be consistent with the website and the individual pages taking into consideration, the hierarchy. The website will feel easier to use for users if they know the location of each page on the site just by the URL description.
* Efficiency – When a URL is efficient, it will enable filtering so users can only navigate through individual pages and disallowing specific pages from being entered without certain conditions being met, for instance, logging into an existing account or loading the “purchase successful” screen without paying for a product.

## **Register: “/”**

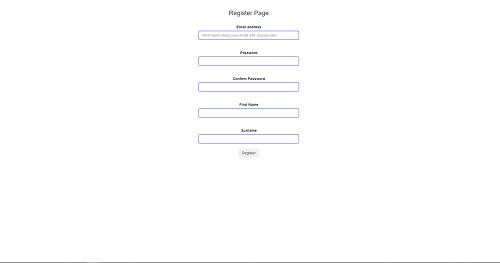
****

Figure / Register Page

For our application, the user journey begins with the register page (figure 1) that will require any user to create a user login before accessing the website. The link uses a basic slash as that is the first page that the user will be directed to when the application is started, when the application requires more features and information to be shown on site, it will usually be replaced with “/register” or “/signup” so it is more explanatory but in this application it is not needed as the user will need to sign up with an account to produce any input/output.

## **Login: “/Login”**

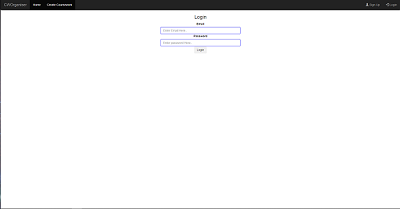
****

Figure / Login Page

Login utilizes the username and password retrieved from a user when they have registered an account through the previous Register page. This will take the information gathered from them and utilize a method to take their first name and surname variables to create a username which they can use to login.

## **Coursework: “/coursework”**

Figure / Coursework Page

This is the main/home page for the application (figure 3) and will show how a student that is logged in can view all their coursework modules in one place so they can view and organise them through the use of CRUD features (Create, Read, Update, Delete). The URL design is also kept simple here so the user will be able to identify how to navigate through the pages. When navigating through the buttons to use CRUD features it will be displayed as “Coursework/Edit:id” the id here will be the id of the specific coursework while being kept efficient and small to help them identify where they are navigating to and what coursework they are editing. At present the application will get all coursework’s from the database instead of a distinct unique users coursework and display them all on the homepage, this is due to time constraints and lack of experience when obtaining a unique user id and using that within the database method to retrieve unique entries within the database.

## **Coursework: Create - “/coursework/create”**

Figure / Coursework Create Page

This is the page that will direct the user to a form through clicking a button to create a new coursework entry (figure 4). The link shows that it is expecting an input from the user and what they can expect the output to be through coursework. This naming sense will stay consistent throughout the CRUD features, so it is easier for the user to determine the meaning for each page when they are directed towards it. For the application, the group has not been able to implement milestones for the program due to time constraints and focused on the functionality to display and create coursework’s for each student displaying the title, module, due date and completion date.

## **Coursework: Update - “/coursework/update”**

This page serves as the second CRUD feature in update that will allow the user to update each individual coursework to change the name, module description, due date, completion date and the milestones that are set for that coursework. The same consistency is set for each of the links that serve as a feature in CRUD, so it is simple for the user to navigate to each of the features by replacing “Update” with “Create” etc. This feature was omitted due to time constraints, so it is not currently implemented inside the current project.

## **Coursework: Delete**

The delete page for coursework will be displayed as a button to pass the id for each individual coursework. Ideally it will display a pop-up for the user to ask them if they wish to continue with the deletion, but due to time constraints, the implementation is focused first to ensure that the selected coursework is deleted from the database file. The link shows to the user that they will be deleting a coursework entry and again, from the id passed in the link they can determine which one will be deleted from the placement of the button above the selected coursework, making it easier to navigate and ensure that the coursework that the user wants to delete is actually subtracted from the database.

## **Future Implementation**

Our application has implemented the model-view-controller methodology which will allows us the grow the application easier in the future when we are presenting more pages with more information to display to users. This makes it easier to organise the pages and code for easier implementation for new features and ease of scaling up our project when the opportunity presents itself.

# **Persistence**

Persistence is the continuous existence of data even after the process that created it ceases or the machine running it is switched off, when data is created and needs to be persistent it’s saved in a non-volatile storage location. Nedb is a lightweight/embedded database created from JavaScript that is packaged as a node module that is used as an in-memory or persistent datastore, as this is a lightweight database it helps reduce the overall footprint of the database on the system which is suitable for a small scale application such as this project.

Basic database operations include Creating, Reading, Updating and Deleting, each of which are present in the application. Examples of where operations can be found, in the registration of a new user and in the creation of a new project, Reading can be found in the retrieval of all coursework projects to display and the login to find the users data by email.

By utilizing the model-view-controller methodology inside our application our group has been able to organise the persistence of code into designated parts where the class models that include a User and a Coursework and their variables have been created inside the models folder to manage the data and logic of the application. The view has the User Interface’s that are used throughout the application to display information to user’s in a simple and digestible format as the program is made with the intent to ease the display of information to the user and organise it and lastly the controller accepts input from the user and converts it into instructions for the program to follow when communication with the model and view to display certain information that they have selected.

To handle the data of the application in Nedb, we have the controller that will pull information from the data access layer to display information to the user, this is achieved through instantiations to share the datastore with the application to get and retrieve information through methods that the user has selected to retrieve or has entered information such as a registration form or creating a coursework that must be communicated with the database in Nedb. In the application the section that is responsible from pulling and getting information from the database resides in the controllers folder; more specifically, the routes files in “CourseworkRoutes.js” and “UserRoutes.js” that have been organised to split the instructions that are communicated within the coursework pages, or pure navigation through the website in the User Routes like logging in/out or navigating to different pages.

Using these methods in our application will make it easier to scale when the website is presented with new data to store requiring in a more robust database such as switching from remote to a local database to increase the security by having more than one server in the event of a crash or downtime, increasing the speed of information etc. The point is to make the program is portable as possible and having the code as recognisable as they are in different folders to display their purpose quicker will enable future developers to understand what each segment of code is being used for and through the use of comments to adapt it to suit future requirements and purposes that the program can fit.

# **Testing**

Testing assures the application has met the requirements that were contained in the brief and responds correctly to numerous different users' input and situations that the application might experience. Without testing the application produced in the project we would not know how the system would respond to different scenarios, which allows for adaptation and improvement. Ideally, some tests would be conducted by end users so we can fully observe a normal interaction with the system that is not biased or influenced. The testing documents produced during the project consist of Acceptance tests, compatibility tests, security tests.

## **Acceptance Testing**

This test was conducted to establish that all the requirements in the project brief had been accomplished successfully in our application *[figure 5]*. The registration component of the application was tested with various data sets to understand how the application might respond to different situations a typical user might experience. Normal input being an average user input, extreme being an input at the lower and upper limits of the application and extreme data which should not be accepted.

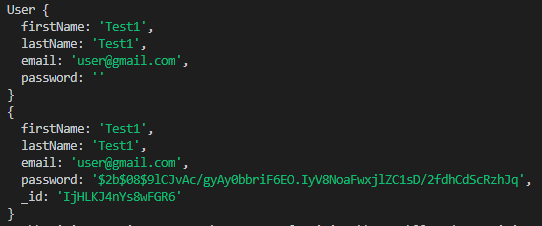
|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Data** | **Expected Response** | **Response** |
| 1.Registration | Normal  -user@gmail.com  -TN28544  Extreme  -user2@gmail.com  -Long Password    Exceptional  -”&£&£  -\*empty\* | Normal  - Account Registers  Extreme  -Account Registers because no limit on pass length or character  Exceptional  -Error Message | Normal  -Account Registers  Extreme  -Account registers    Exceptional  -Error Message |
| 2.Add Project | Add project info | Project gets added | Project got added |
| 3.Remove Project | Click remove project | Project gets removed | Project got removed |

*Figure 5 - Acceptance Test*

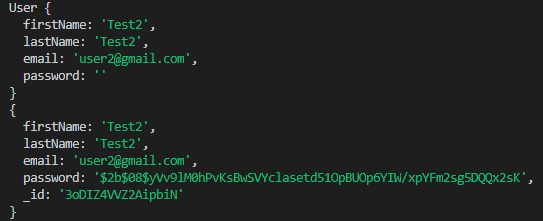
### **Acceptance Testing Evidence**

#### **Registration**

##### **Normal user**



##### **Extreme user**

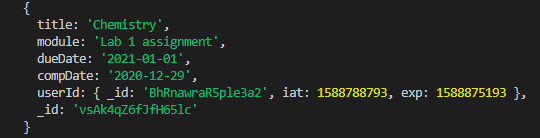


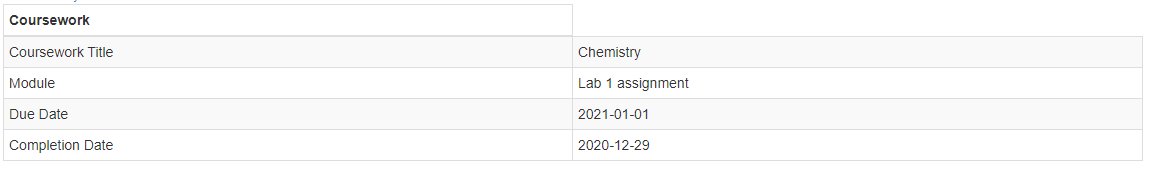
##### **Exceptional user**





#### **Adding a Coursework**





#### **Removing a Project**



## **Compatibility Testing**

The hardware compatibility test includes the testing of popular operating systems to make sure they all work correctly *[figure 6]*. Making sure the application works on all operating systems ensures the widest range of users to interact with the application.

|  |  |
| --- | --- |
| **Operating System** | **Successfully works** |
| Google Chrome | ✔ |
| Mozilla Firefox | ✔ |
| Internet Explorer | ✔ |
| Safari | ✔ |

*Figure 6 - Compatibility Test*

## **Security Testing**

Security which can be found in the form of hashing (discussed in more details in the Application Security Heading) the users passwords is an important feature and is subject to testing as it is vital that the feature works constantly in order to keep users information more secure. Testing to see if the password hashing worked consisted of trying several acceptable passwords in the registration form and checking the data storage to see if the passwords stored are different from what was originally entered. Furthermore, decoding the hashed codes from the password would bring back the original passwords entered to check if everything went correctly. *[figure 7]* Shows various passwords that have been hashed in our system, in the chance of a breach of the system this is how the passwords would be viewed.





*Figure 7 - Hash Test*

# **Application Security**

Application security is an important factor when developing an application which stores data that might be sensitive to users, such as names, passwords and in our scenario sensitive project information. Steps our application has taken to provide security to the users is hashing the user’s password when it is getting stored in the database. Our application has hashing for cryptography because it covers the original data entered by the user with a different value which means no one can see what was entered, the only way to decode the data is by searching up in the hash table which users don’t have access to.

Upon future development of the application, enhancements to the security could be implemented. This could contain two step verification to protect users with an additional layer because the users would have to verify access via email or mobile device. Also frequently asking security questions randomly when logging into the account could be a possibility that would improve user security because the user would need to entire another step of verification to proceed. Furthermore, the use of third-party apps for content should always be kept up to date especially if they interact with the data that is being pulled from the database. As with any ongoing application, security testing should always be considered and implemented to try and find flaws in the system's security to improve and advance technology to combat it.

Other types of security threats that could be problematic for the application include phishing which is the act of sending fake emails to users acting as our service in order to retrieve personal information, as with any service users should know that companies will not send emails asking for personal information in a reply email and instead all interactions with the users sensitive data will be made by the request of the user. As our site does not include any downloadable content users should not need to worry about downloading anything that will be harmful to their computers and jeopardize their personal information.

# **References**

Applicure.com. 2020. *Best Practices for Database Security*. [online] Available at: http://www.applicure.com/blog/database-security-best-practice [Accessed 6 May 2020].

Deep Crawl. 2020. *A Guide to URL Design - Deep crawl*. [online] Available at: https://www.deepcrawl.com/blog/best-practice/guide-to-url-design/ [Accessed 6 May 2020].

GitHub. 2020. *Louischatriot/Nedb*. [online] Available at: <https://github.com/louischatriot/nedb> [Accessed 6 May 2020].

Sciencedirect.com. 2020. *Persistence Mechanism - An Overview | ScienceDirect Topics*. [online] Available at: https://www.sciencedirect.com/topics/computer-science/persistence-mechanism [Accessed 6 May 2020].

Stack Abuse. 2020. *Nedb: A Lightweight JavaScript Database*. [online] Available at: https://stackabuse.com/nedb-a-lightweight-javascript-database [Accessed 6 May 2020].

W3schools.com. 2020. *HTML Tables*. [online] Available at: https://www.w3schools.com/html/html\_tables.asp [Accessed 6 May 2020].