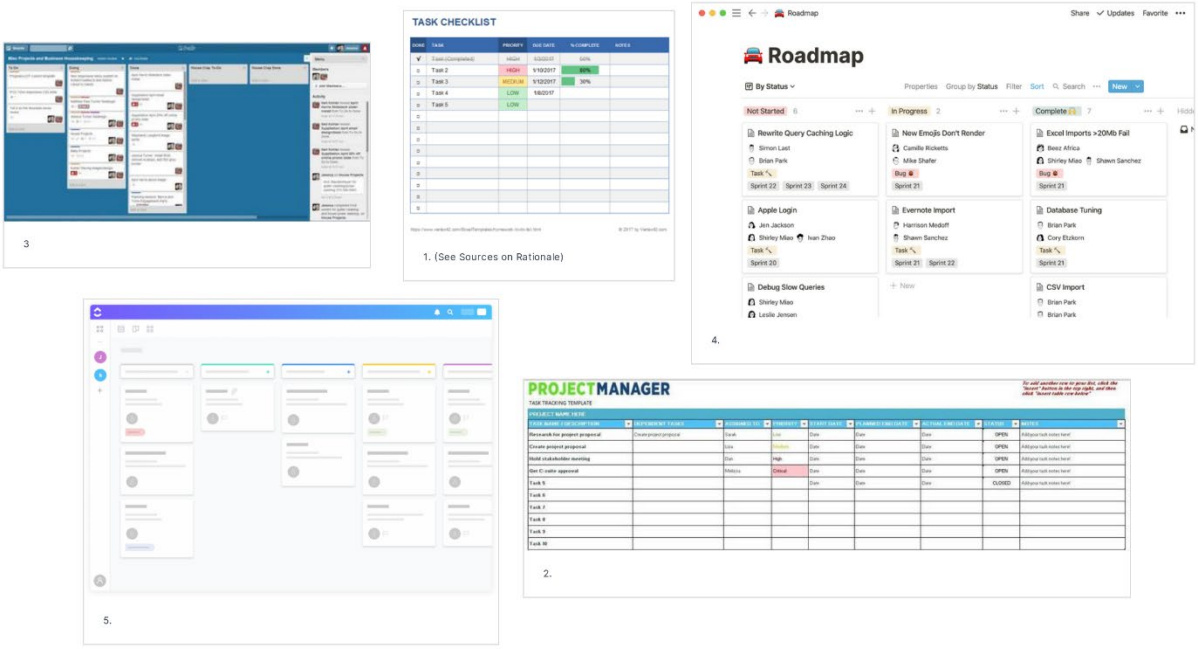
11058 Project 1 – Basic Web App

U3202988

Rationale / Process Diary

I started this project by research, sketching and identifying what I wanted to solve and what data I needed to store in the database. I decided that I wanted to create a task/assessment tracker similar to Notion or Trello.

Although the front end is not the main focus for this assessment, I created a moodboard to summarise my research. Data to add to the database included Taskname, Due Date, Assignee, Priority, Status (how much is already done) and any Notes.



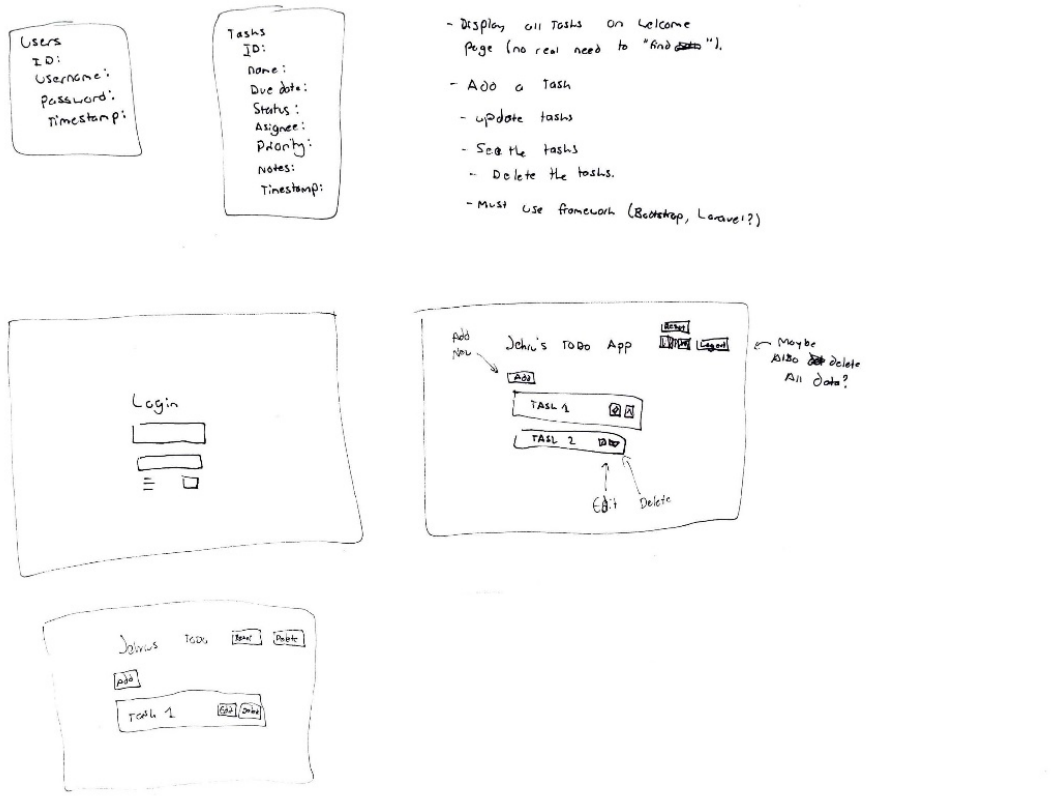
(see link to higher res version https://app.milanote.com/1KbNJq1Ql0fy3k?p=VP4nbzslqSV)

I also completed a rough sketch of what I wanted the database to look like, this helped me visualise what functions the application may need. Such as is there any account functions like logout, does clicking 'create' open a new page, etc.

Programming goals were:

1. Create a functional web app that can create, read, update and delete tasks.
2. Make it appear that the functionality was all happening on one (or two) page(s).
3. Give the user the option for showing the priority in their tasks.

4. Extra functionality such as the ability to search for tasks.
5. Give the user the options to move their tasks around on boards similar to Trello or Notion.

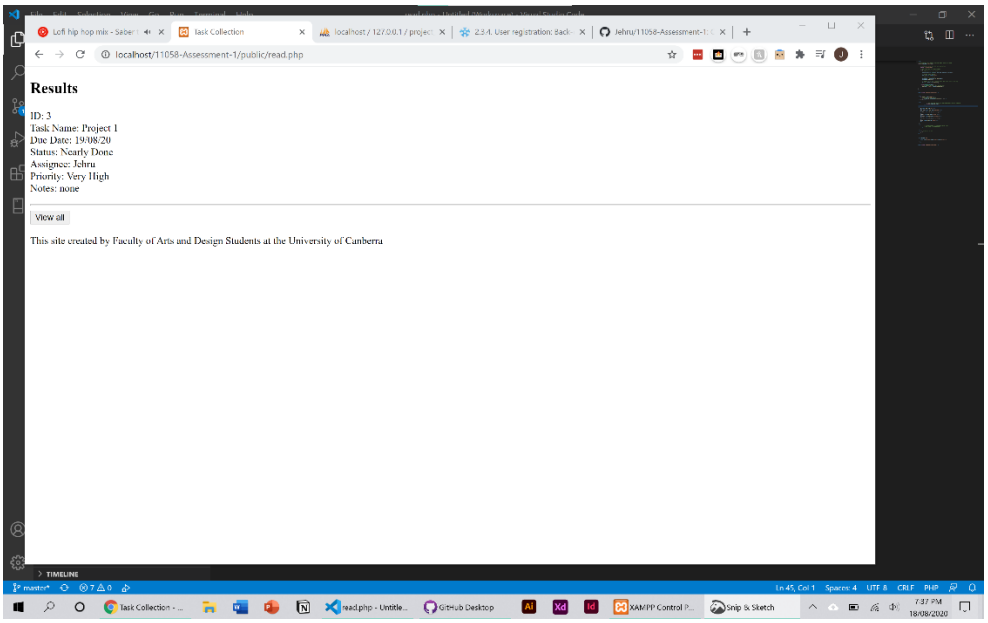


I then recreated the basic authentication functionality following the tutorial on canvas.

After I proceeded to add the basic Create, Read, Update and Delete functionality from the tutorial on canvas. I created two databases, one called tasks which holds the task information and one called users which holds the user accounts.

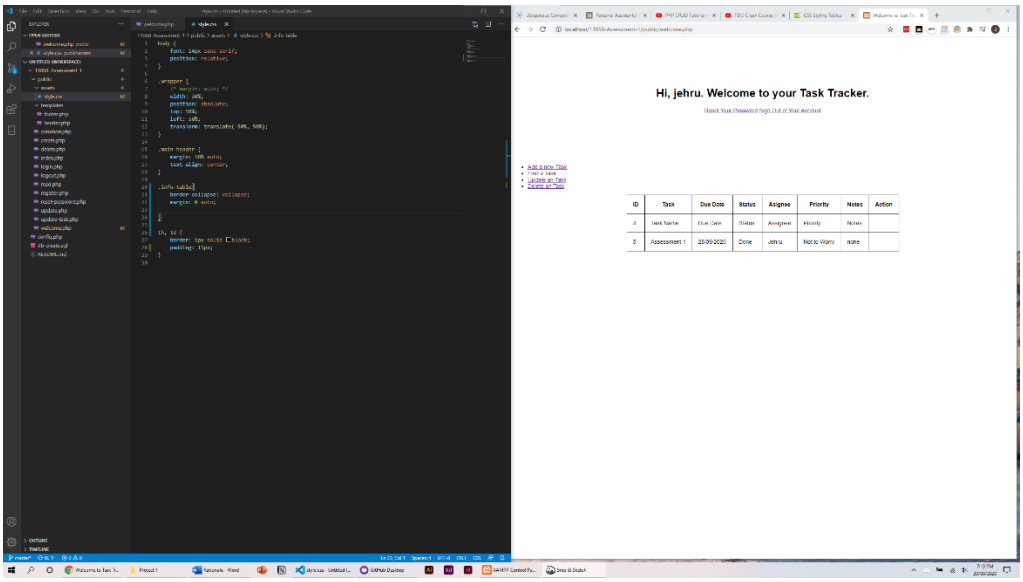
The first problem I encountered was that I wanted a table based interface display on the main welcome page so that the information is present for the user and they didn't need to click 'view all' every time they wanted to see their information. I wanted that information to always be in view.

I also disliked how the user would have to go edit > edit work to be able to edit their information. A simpler process would be one click to edit their information in the table mentioned above.



When just starting to create the above. I watched this crash course (Traversy Media, 2017) into php and PDO I found it very helpful in reaffirming the basics of PDO and how it works.

I used this tutorial (Clever Techie, 2018) to create my table and although it uses MySQLi some of the php code and concepts were transferable to PDO and was used to create this:



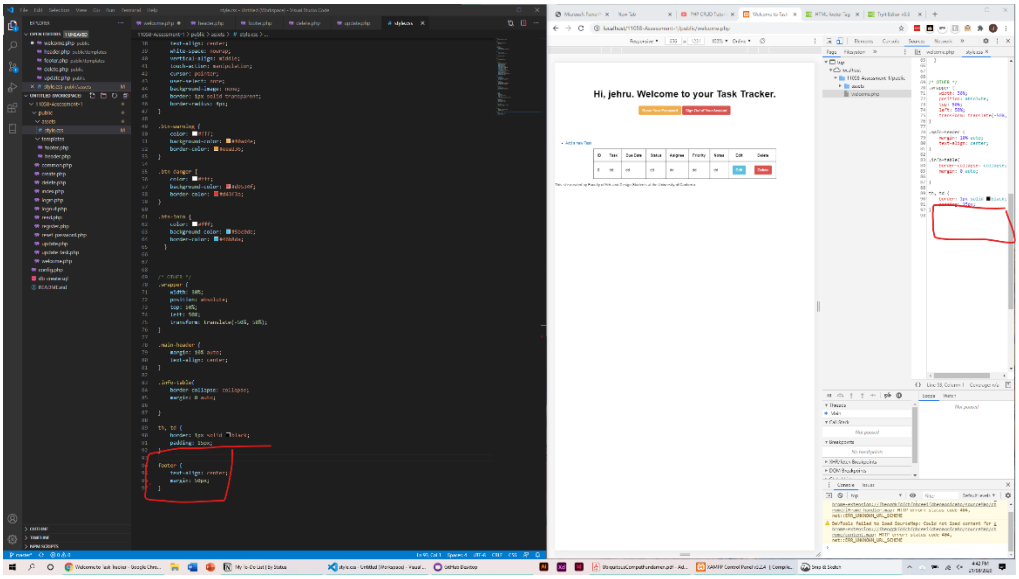
I also started to add CSS to make it look better. This made the ‘find a task’ link redundant.

Following the tutorial mentioned above (Clever Techie, 2018) I was able to add in an edit button which would access the ID in the row. This rendered the update.php file useless as it would use the id and go straight to edit-tasks.php, where you can adjust the information in that row.

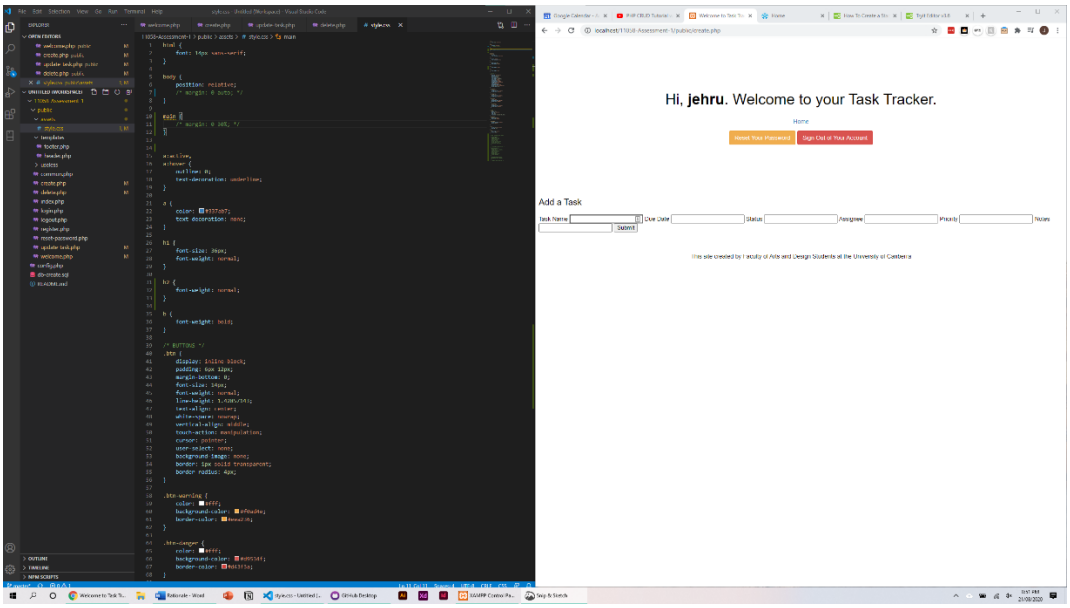
I continued through this tutorial and also added in the delete button.

At this stage I noticed the lack of navigation as I would have to type welcome.php back into the url bar to go to the main page. So, I added a button which links back to the page in the header.

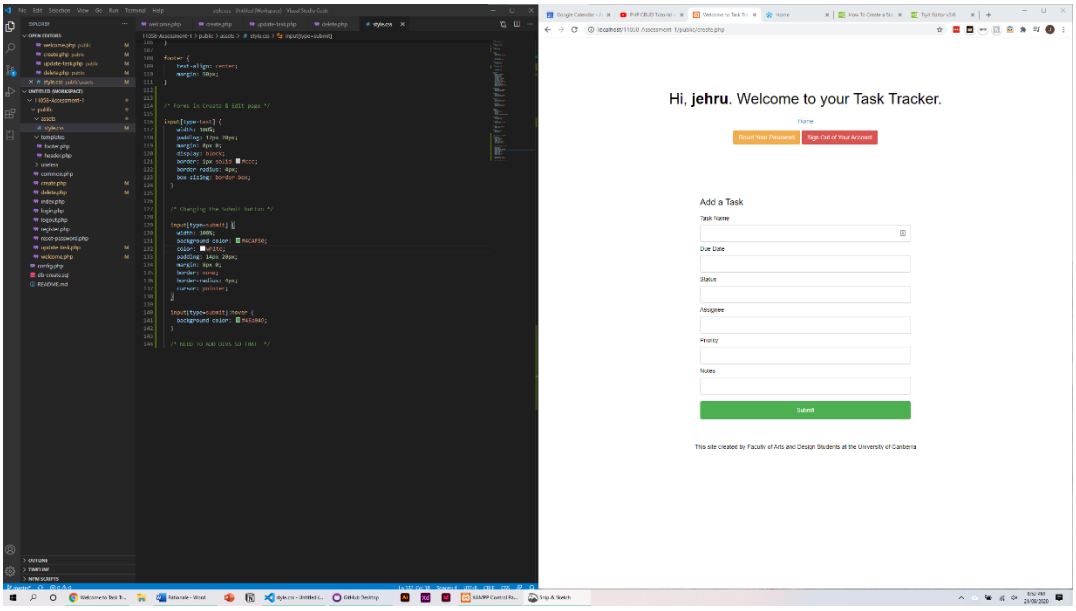
I noticed that when editing CSS code, the site wouldn't upload it properly, this made it difficult to edit it locally. After lots of experimenting and testing, I resorted to using a hard refresh in chrome (Control/Command + Shift + R) ("How to Force Reload any Cached CSS File (Local, Remote, or on a CDN)", 2018). I later found out that it was because CSS can cache itself in chrome.



(Image showcasing CSS not being updated)



Before

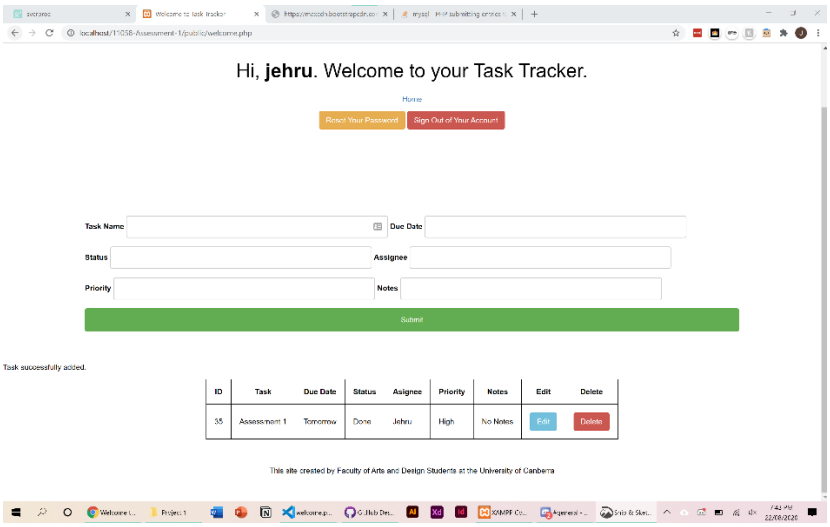


After

To achieve my goals for this assessment the edit and delete page should route back to the welcome.php page after they have deleted or edited their tasks. I noticed that this was already being used within the website where the index.php page was routing to the welcome.php page. I used similar code header (“location: welcome.php”) to route these pages back to the welcome.php page.

Another goal was to have the create functionality within the main welcome.php page. I transferred my create.php page code into the welcome.php page.

One issue that arose was that no data would appear after the user had pressed submit. The data would upload (and this was checked as when I reloaded a page) however it did not dynamically adjust into the table.

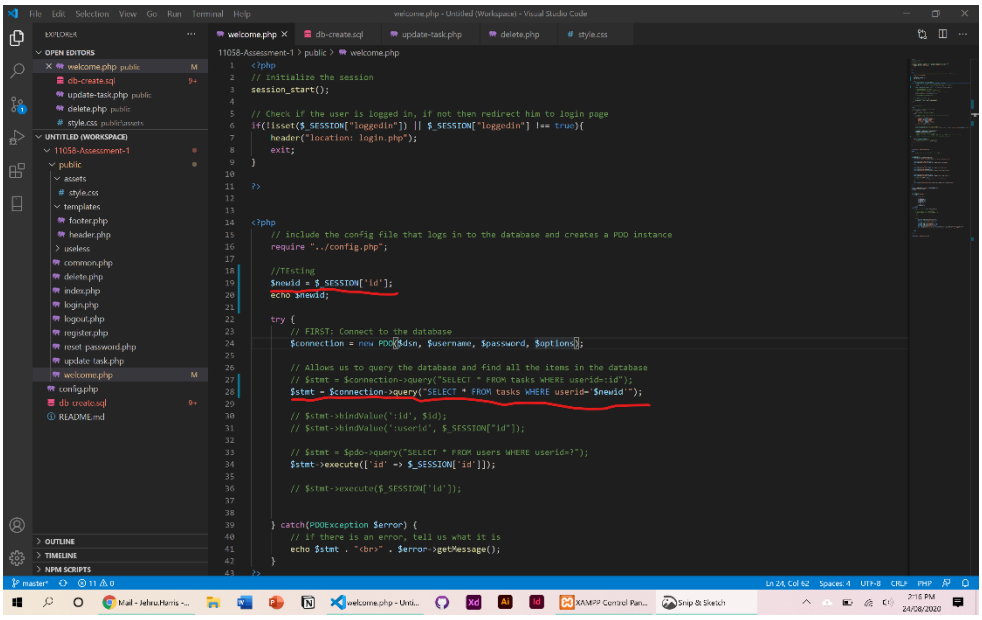


To solve this I added in the same code that ‘reads’ the data at the end of the ‘submit new task’ process. The following code:

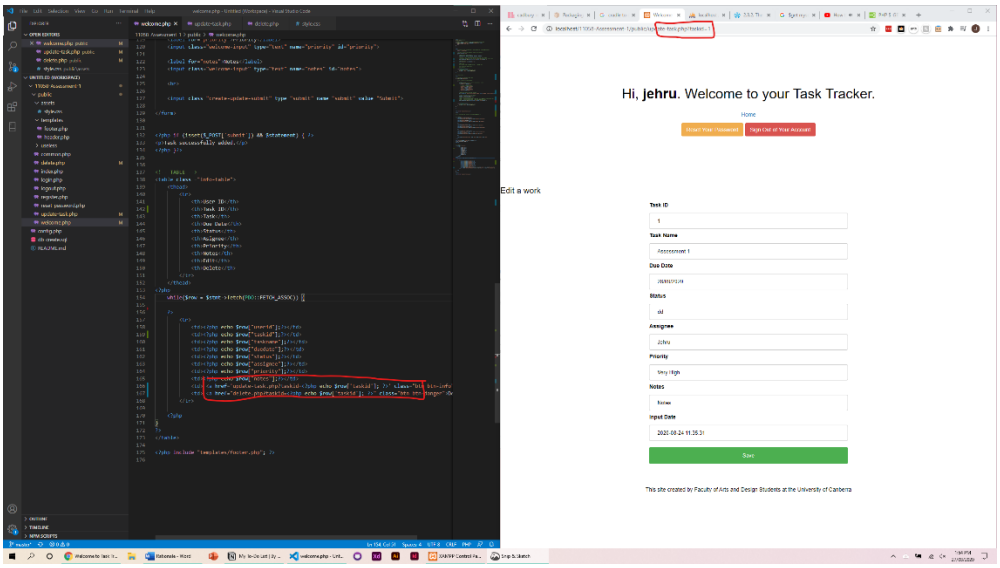
```
$stmt = $connection->query("SELECT * FROM tasks");
```

This fixed the issue as the code would refresh the new data after it was submitted.

One significant problem with my website was that any user that logged in could view all the data on the database. This meant that I could view a task created by someone else. At this stage of the process I got assistance from the Tutor. The solution was to create a new column in the database called ‘userid’ which linked to the unique ‘id’ given to the user when they sign up to the app. I continued to troubleshoot with the issue of the code not querying the logged in users information. I found a tutorial online which helped solve this. ("Filtering Records with MySQL WHERE Clause in PHP - Tutorial Republic", n.d.)



After completing this the default ‘id’ in the database was changed to ‘taskid’. This broke most of the functionality. I solved this by changing all of the ‘id’s to ‘taskid’ in the code.



At this point most of functionality of the site was working as expected, however one issue was when I refreshed the main welcome.php page the form would submit the previously inserted information. I was not able to find a solution for this issue.

I also removed the assignee column from the database as there would be no cross-user interaction. I decided within the timeframe and my coding knowledge that implementation this was not likely.

I also added in better form design such as using the date form and adding in a ‘low’, ‘medium’, and ‘high’ selections for the priority form.

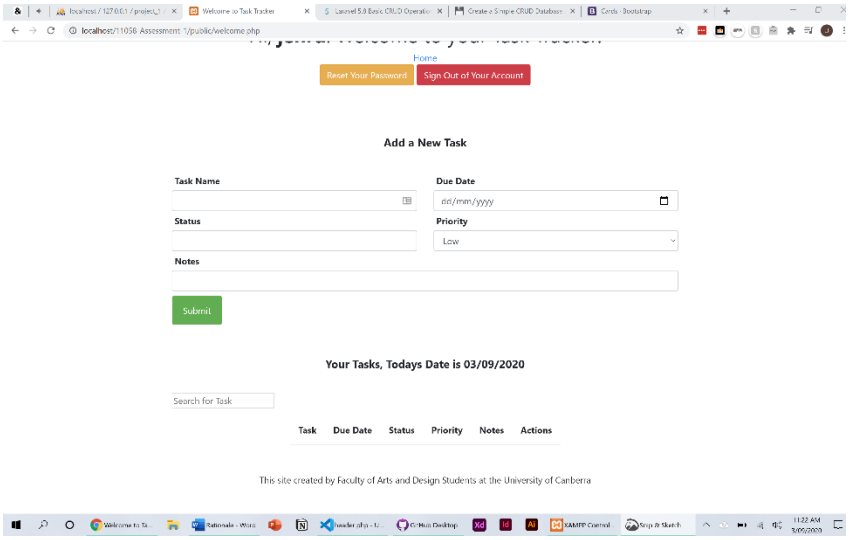
In a class tutorial I asked for assistance of what features I could add to the web app. We decided that a functionality to order the tasks by their priority was important. (High priority tasks need to be seen and get done first). One way to do this was to link the priority name (high, medium, low) to a number variable (1, 2, 3), this would allow me to order the table ascending by these numbers.

This was easy to understand but complex to implement. High priority = 1, Medium = 2 and Low = 3. Then the numbers would be shown on the table in ascending order. An if/else statement was used to so whatever the chosen level of priority was it represented a number. Then the database would the read command by an ORDER BY statement which solves this problem.

I added in a small indicator that data has been sent using Alertify JS which is a Javascript framework which makes notifications and popups simpler. I used the notifier for success and error messages. (Younes, n.d.)

I also added in a live search function, this is not the best implementation as it doesn’t work with the table on the website as it creates its own new table which is confusing. ("MySQL Database Live Search with PHP and AJAX - Tutorial Republic", n.d.)

At this stage I added bootstrap classes. I used ("Forms", n.d.) to fix up the forms, the table and I added some card style tables.



I also wanted to include some error prevention. One problem was if there was no data in a task item the delete.php page would not route back to the welcome.php page. I solved this by using an if statement that would check if there was empty information in the forms the site would not upload it.

I found this (Nabbefeld, 2013) forum to be helpful to also solve this problem. What solved the problem was that I had not set variables for $taskname, $duedate, $status, $notes before the IF statement. This meant that the variables were always empty, and the IF statement was never valid.

For the last stage of this assessment I added comments, cleaned up the code, removed duplicate code and functionality.

Conclusion

While I was able to create basic functionality, I was not able to create a board styled application like Trello or Notion. Some functions could be improved further such as the search. I also was not able to solve the problem when the user refreshed the page duplicate data would be inserted into the database.

Link to Github and Online Site

https://github.com/Jehru/11058-Assessment-1

http://jehruh.sgedu.site

References

Clever Techie. (2018). *PHP CRUD Tutorial with MySQL & Bootstrap 4 (Create, Read, Update, Delete)* [Video]. Retrieved from https://www.youtube.com/watch?v=3xRMUDC74Cw

Filtering Records with MySQL WHERE Clause in PHP - Tutorial Republic. Retrieved 3 September 2020, from https://www.tutorialrepublic.com/php-tutorial/php-mysql-where-clause.php

Forms. Retrieved 3 September 2020, from https://getbootstrap.com/docs/4.0/components/forms/

How to Force Reload any Cached CSS File (Local, Remote, or on a CDN). (2018). Retrieved 3 September 2020, from https://wpreset.com/force-reload-cached-css/

MySQL Database Live Search with PHP and AJAX - Tutorial Republic. Retrieved 3 September 2020, from https://www.tutorialrepublic.com/php-tutorial/php-mysql-ajax-live-search.php

Nabbefeld, T. (2013). How to check if any fields in a form are empty in php. Retrieved 3 September 2020, from https://stackoverflow.com/questions/14331525/how-to-check-if-any-fields-in-a-form-are-empty-in-php

Traversy Media. (2017). *PDO Crash Course (PHP)* [Video]. Retrieved from https://www.youtube.com/watch?v=kEW6f7Pilc4

Younes, M. AlertifyJS. Retrieved 3 September 2020, from https://alertifyjs.com/notifier.html

Younes, M. AlertifyJS. Retrieved 3 September 2020, from https://alertifyjs.com/

Mood board Image References

1 *Task Checklist*. (2019). [Image]. Retrieved from https://www.vertex42.com/ExcelTemplates/task-list-template.html

2 Westland, J. (2019). *Project Manager Template* [Image]. Retrieved from https://www.projectmanager.com/templates/project-task-tracking-template

3 Welker, B. (2019). *Trello Board* [Image]. Retrieved from https://project-management.com/trello-organization-using-simplicity-minimalism-to-achieve-great-results/

4 *Notion*. [Image]. Retrieved from https://www.notion.so

5 *Clickup - Task Management Software*. [Image]. Retrieved from https://clickup.com/blog/task-management-software/