

Napier Demonstrator System

Nick Taylor
40203073@live.napier.ac.uk
Edinburgh Napier University - Module Title (SET09103)

1 Introduction

1.1 The Task

Moving on from the first coursework, which I thoroughly enjoyed, I was now required to create a personal project within the Python Flask micro-framework. This was a good way for me to showcase what I have learned throughout the practicals and lectures of Advanced Web Technologies. The topic was my choice and I had to be able to design, implement and evaluate the chosen topic. One of the main points of this task was to be able to carefully create a URL hierarchy that is appropriate for navigating through the web application. I was excited at the challenge ahead to create my own personal project.

1.2 The App

I created a system known as the Napier Demonstrator system. This is a multi-page interactive system where a user from the university who demonstrates practicals for other classes can access and add in what days they have worked, how many hours and what class it was. I decided to create this system as I am a demonstrator at the university and don't keep track of the hours I teach very well. I thought this would be a good way for me to be able to use something like this in future which made the system a lot more fun to create. Below is first page the user sees when launching the web app.

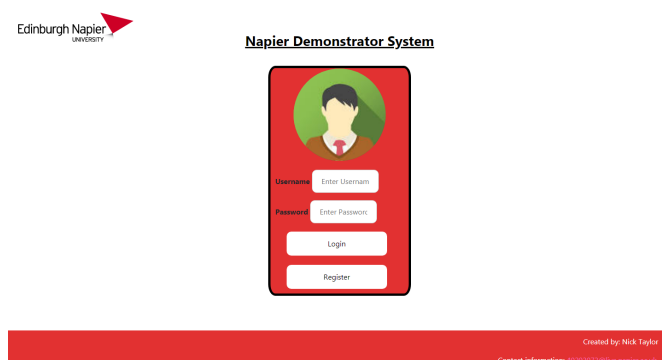


Figure 1: Login Page

2 Design

For the design, I wanted to keep it as simple as possible so that all kinds of users would be able to interact efficiently and quick with the web app. Making it simple makes it a lot easier for a demonstrator to be able to quickly go onto the system, input some classes and the system completes it all for you.

Login Page As shown in Figure 1 the user is greeted with a simple login page which contains the Napier logo, a small title and an interface for logging onto the system. If this is the first time the user has used to system they are able to register with just a username and password then from here they are able to log in. I have also added a footer to the page which contains the email address of the creator myself if the user has any difficulties accessing the system. Within the interface to log in I have added a small CSS hover animation to show the image changes from male to female when you hover over the image.

Main Page

Once the user has registered or logged in successfully, they are greeted with the interface as shown in Figure 2. This page is the main page and displays a basic calendar with a small summary block. This is where all of the users interaction occurs. The calendar is displayed by making use of some simple bootstrap which accesses the columns to make it aesthetically pleasing. I also imported multiple libraries within flask. I was able to access a calendar library to retrieve the month range and I could access a date time library. This helped to make the calendar look as nice as possible but also easy for the user to have functionality. From here the user has the chance to be able to click on any day of their choice. When this occurs they are taken to the class input page.

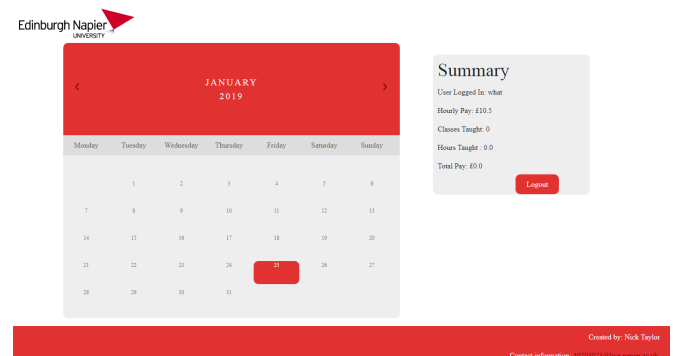



Figure 2: Main Page

Class Input Page

When on the class input page the user is now able to input data into that specific date chosen. They are able to enter the module ID, module name, lecturer, the time the class started and the time the class ended. The system will not allow for any of the text boxes to be empty when clicking the save button. Once all components are entered the user can then save the data which then the user is redirected back to the main page. Once returned, the user will then see that the date they entered data appears highlighted on the calendar with the times they entered as shown in Figure 4.

On the right hand side in the summary block the user can now notice that there is now a short summary of the data entered next to each section of the block as shown in Figure 5. This contains things like how many hours in total they have worked, how many classes taught, the user logged in on that specific session and many more. If the user has inputted a part incorrectly they are able to either amend the issue and re save or they are able to fully delete the class inputted and start again.



Class Information

Module ID:

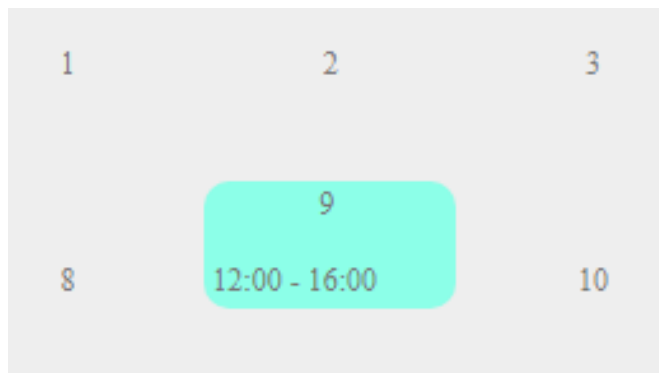
Module Name:

Lecturer Name:

Class Start:

Class End:

Figure 3: Class Input Page



1 2 3

8 9 10

12:00 - 16:00

Figure 4: Highlighted Date

Summary

User Logged In: Nick_Taylor

Hourly Pay: £10.5

Classes Taught: 1

Hours Taught : 4.0

Total Pay: £41.9

Figure 5: Summary

2.1 Data Handling

For all of the above to take place, data handling must be required for it to happen. I followed through the workbook steps learning about SQLite to enable me to create my own database called demonstratordb.db. Within this database

there are two tables known as classes and users. For the first page the user has contact with, I use the users table which stores the users username and password and also a set hourly rate of pay. This means that if the user tries to login without registering an error will be thrown and they will be asked to register before logging in. Once they have logged in successfully, the database stores the details of the user and they can now move into the system.

Moving into the data handling on the main page, when the save button has been clicked, all the values are inserted into the classes table for them to be stored. The user is also pulled from the user table so that the system is able to identify what classes relate to each user in the system.

2.2 Routing

Interacting with the application, I decided it would be best to use a collection of URL's that are routed throughout the code. The '/login' route provides an opportunity to log on to the system. If you try and type this manually when you are logged in the system bounces it back with a redirect to the page you are already on. The '/' route is used to present the current month if no variables have been passed through. When the user then picks a date the URL then updates to show the year chosen then the month then the date. For example '/2020/11/2' would represent the 2nd of November 2020. I have included arrow hyperlinks at either side of the month header where you can go to previous or following months therefore being able to watch the URL hierarchy. If a route is entered manually and incorrectly I have a custom 404 page with gives the user the option to redirect back to the main page. The 404 page can be found in Figure 6.

[Back to the Home Page](#) →



Figure 6: 404 Error Page

3 Enhancements

Login Page Within the login page, It has all that it needs, I could maybe have a full link to another page when register is chosen, where the user is then asked to input their username, password, date of birth and email address.

Main Menu A good enhancement that I could add to my main menu page is if the user had a profile tab which would direct you to another page which shows all of the information about the user and what classes they have taught.

Class Input Page In the class input page you are only able to add one class to a specific day. I would have liked to have added a way where you are able to add multiple classes taught

within the one day as this is possible to happen. This would make the system even more helpful for the users.

Tutorial Point, "https://www.tutorialspoint.com/flask/flask_redirect.html"

4 Critical Evaluation

Login The login page of my system works very well. I believe if any user was to interact and try register they would have no problems. The system benefits from the simplicity of it as there is no need to have a register page for the user to add more fields.

GET/POST Methods Throughout the system GET and POST methods are used to get or retrieve data. This works throughout the system with no issues. There is not any problems with caching sessions also when the user tries to login.

Data Handling All data is handled in the same way so that it can always be inserted or retrieved from the database. Without the database I would not be able to store logins or even save or delete any of the classes the user has taught.

Responsive Some parts of the application are not responsive which would make this a bit more challenging to make the system tablet and mobile friendly. This is definitely something I will be looking into for self study when complete.

5 Personal Evaluation

The hardest part for me within the coursework was understanding the data handling and how you are able to interpret data from the database. SQLite, I feel, was very tedious but I believe I got the hang of it the more the coursework went on. My time management was also very good when creating the system as I was able to get a login page and simple user interface working quickly so that I was then able to build further on this. I really like the design of my system which was helped by the use of bootstrap and CSS to make the system look as nice as possible.

6 Conclusion

Overall, I am very pleased with the work I created for this coursework. Due to me being so pleased with this, I have decided I am going to work on this further over the festive period to advance the system to enable many of the features I included in my enhancements and in my critical evaluation.

References

S.Wells, "Coursework Specification 2"

S.Wells, "Workbook- SQLite 3"

Flask Documentation, "<http://flask.pocoo.org/docs/1.0/tutorial/database/>"