Advanced Web Technologies

Introduction:

Online Catalogue is simple website that is meant to store user's information on certain topics; such as movies, music, video games as well as other types of data. A user might use this kind of website to store their favourite collection of these topics, and ideally add and remove them. However, for this coursework, the objective is to merely display data on either of these topics or similar topics.

Design:

My architecture of this webpage followed the minimum requirements for this coursework. I created a 'homePage.html' template, which had a basic bootstrap template design, On this design was a navbar, which would also collapse for mobile user. On this navbar was the links 'Home' and 'Display'; the display link, routed you to the display page, where I had designed to display all the information that was in the database 'movies.db'. I used a loop within the 'displayPage.html' template to loop through all items within the database and display them on a bootstrap table, the display page also hads basic bootstrap template. The init_db() function within the main program 'OnlineCatalogue.py' this function is responsible for creating a table called 'movie.sql', with the attributes title, runtime, director, leading actor and release date, using sqlite3. This same function also runs another script called 'movieInsert.sql'; this script would insert predefined data I made within this script, so the only way to edit data stored in the database is changing this script. For this exercise, I didn't make a function to initialize the database on run time; so I had to run a basic .py program I made called 'init_db.py'; this imports the function from the main program 'OnlineCatalogue.py', to initialize the database before running the main program.

Enhancements:

I had several ideas for this website to be more then it was. I initially had an idea to had an input page. This page would simply display 5 input boxes, one for each of the attributes in the movies table within the movies database I created; which would be followed by a submit button which would post this data to the database from the webpage. Once it was added, the display page would display the data in the database along with the data the user added. Also, the input boxes would be validated with a script using JavaScript; the script would do 2 things; first stop users from entering nothing into the boxes, so every box had to have something in them. The other would be to stop user from entering just numbers into certain fields, like director and leading actor, as these are peoples names and most people wouldn't have numbers in there names.

Second idea I had was a delete page. As an input page, should - in my opinion - have a delete page. As for example; if a user enters the title wrong or enters other data wrong, then there's no way to delete that item, without having to start from scratch. So in my delete page you had a search feature where you had to search the database using the title of the movie as your search parameter; this would then search the database for the title of the movie and the rest of the attributes for that one argument; then there would be a submit button called 'Delete' or something similar; it would then use the title and remove all the attributes along with that title, from the database.

A key enhancement for a webpage like this would to include a login system. Since if this was an open webpage anyone would edit the database using the input and delete pages. So I would create another table in the 'movies.db' called users, with attributes like "first_name, last_name, email_address, password, user_id" etc. This table would be used to of course validate users. As well as using encryption to ensure user safety.

Critical Evaluation:

Like I said previously, I only followed the minimum requirement for my coursework, besides using a database (sqlite3). The webpage is very limited, as it only has a display and home page running on the webpage, I didn't experiment with any css/bootstrap, I didn't create a unique design to the website; I used a basic template that was included In the workbook. I attempted to add a little more functionality, by adding a basic redirect, which redirect user to the home route which was "localhost:5000/" if they entered this instead "localhost:5000/home". Furthermore, I added a custom error handler for the 404 error that consists of one paragraph, simply stating this route does not exist on this web app. Thus there is little to no ambition shown in the web app I created; Since I have only reached the minimum requirements. There is a lot of improvements that could have been made.I attempted to create a third page, which was going to be the input page I was talking about in the enhancements subsection. However, I did not overcome this challenge; I simply didn't understand the language well enough to implement this into my web app.

Personal Evaluation:

- I feel like I have a strong knowledge on the basics on python flask, which to be honest is all you can hope for at the beginning of learning a new language. I have learned how to route pages together through trial and error and some internet research, and understanding that you need to run all the app.routes in one single .py and were point to them in the template files, rather than separate files, however you can import them as well.
- I faced a challenge on how to display data from a database onto a webpage table. I used a few internet sources and the workbook to help me understand the loop needed in the template to display all data onto the webpage and the css/html needed to create a table on a webpage.
- I learned how to create a database and use it within my flask app, I also know how to run scripts within flask, rather then having to write an SQL script within flask using a variable to store it. I only had a basic understanding of this so far; but I feel like I know how it work within flask.
- I learned how to use templates, rather then writing html code stored in a variable and used in the flask app to display a webpage, also I learned how to store them in a file hierarchy used for flask web apps.
- I have a basic understanding of redirects, as I used a basic one within my web app, I also have read through the workbooks and worked through all examples shown within the workbook.
- I have a basic understanding of custom error handling, as I used an example from the workbook, to display a custom error message to show that a route that wasn't defined within my flask app, didn't exist.

I didn't face many struggling challenges, during this assignment; as it mostly to show you basic understanding of python flask, and how it works. Only real challenge I faced was attempting to create an input page for my web app. I didn't overcome this challenge as the input proved more difficult then expected; and for lack of time I didn't include this in my final product.

Resources & References:

The Workbook provided for this module was useful on getting some basic knowledge on how some techniques worked.

I used w3schools to help me with the bootstrap table code used on the display page.

I used stack overflow to help with the code for displaying the data from the database onto the table on the webpage.

[CITATION Sta \l 2057]

[CITATION W3s \l 2057]

References

Stackoverflow. (n.d.). Retrieved from http://stackoverflow.com/questions/28258703/python-displaying-sqlite3-database-records-on-a-flask-local-website

W3schools. (n.d.). Retrieved from http://www.w3schools.com/bootstrap/bootstrap_tables.as