Advanced Webtech CW1 Report

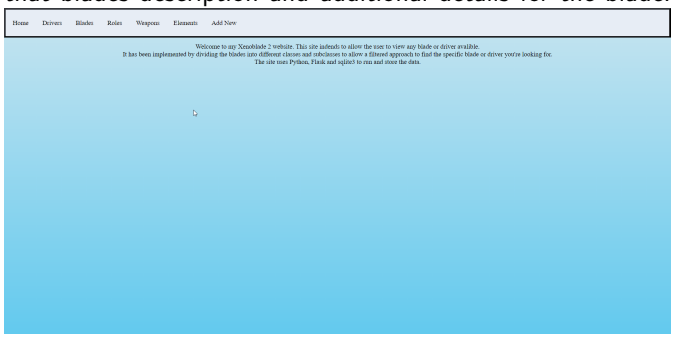
Luke Reeder
40277803@napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

1 Title

The title of my web app is "Xenoblade App".

2 Introduction

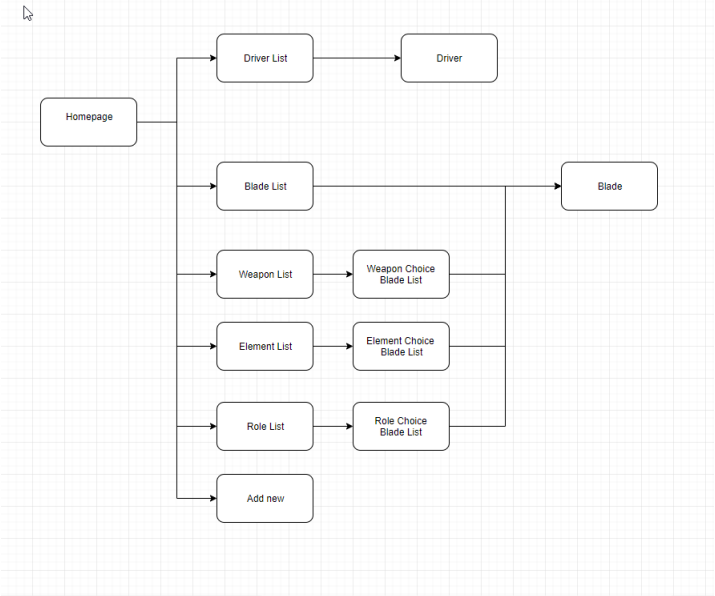
This web app is based on a Nintendo Switch game called Xenoblade Chronicals 2. Within the game there are 5 different playable characters, each of which have a variety of different weapons available to them known as blades. These blades are personified as humans as well and have a variety of different stats and attributes. The web app allows a user to browse through a database consisting of the different drivers and blades, while also allowing the user to refine their search through a hierarchy of filters, such as searching blades by their weapons followed by displaying each blade that corresponds to that result. This then provides a link to view that blades description and additional details for the blade.



3 Design

My web app is structured with having the website connect to a database that consists of two seperate tables. One for Drivers - the playable characters, and then also a table for the blades. The app allows to search through the list of drivers, but since there are so few with very little separating them aside from their personality and description, I decided to not have refined search for this section.

The blades however are a lot more in depth, allowing multiple roles with different elements and weapons. This allowed me to refine the navigation through the website for the user to find exactly what blade they are looking for. The url hierarchy for the app is as follows:



4 Enhancements

For additional utility, I would like to add a function to allow a user to refine their search over multiple parameters, eg looking for a certain blade with a select element and weapon and display any results that have both of those attributes. I feel this could be achieved by creating a new html template with the search parameters with dropboxes for each attribute for a blade, then have the results feed into an sql search. If the search returns a result, it would display the information by retrieving the blade name and importing it back into the relevant existing template.

One other change I would make would be the ability to amend blade or driver attributes in the database. This could be done from the blade description page by adding a button that returns the information in a template with input boxes, uses the blades name as a unique identifier and then updates the information given in the amended text boxes. This would also be run through the execute function to sanitise the input from an sql injection.

5 Critical Evaluation

For this app I built a python flask based web app that Incorporated an sqlite3 database to store initial data and allow the user to add additional blades to the database. The html side was taken care of by templates with the Jinja extensions. For the flask routes, I started by building the url

routes that the user would possibly use. For example, starting with the home route, then adding the ('/blades'), followed by ('/blades/roles') and so on. For each route I tried to make rough notes of what each function should have in it, and any additional resources necessary. Once all of the potential routes were mapped out, I then started working on building the html templates. Initally I made the index template for navigation across the site and used that as a base for all of the following templates. This template included links to the home page as well as all of the other routes, as well as the html framework.

After the initial navigation was sorted, I then made the base database. This was done by utilizing the sqlite3 server provided to us, and referencing the workbook as to connect to the server. I wrote a schema to drop the tables named "drivers" and "blades" if they existed, followed by creating the tables then populating them with initial data. Since the database was now up and running, I could then start to connect the flask app to the database providing the location.

The routes were then edited to search the database for each relevant section for the database by using simple sql select statements, taking in variables if necessary. Where user input was taken however, I used the initial sql statement in conjunction with a dictionary to then run through the execute function to sanitise the input to prevent the user from trying to perform an sql injection and potentially damage the database. For each search, I had the results from the database appended onto a list and then fed into an html template to display the results.

For the add new blade, I drew up a html template with input forms for each attribute for the blade with individual identifiers, and passed them into my flask app using the POST method [1]. These were stored as variables and passed into an sql insert function[2] with dictionary mapping to again sanitize the inputs and then add the data to the database.

With the routes for the flask app, I made the url have a variable input at for paths that would have multiple extensions. This allows the user to then type in the url manually if they wish and it will search for that input. This also cuts down on repetition within the coding as to not have the same code over and over for a single change in name.

6 Personal Evaluation

For this coursework, at first I did struggle a bit, mainly with the python syntax as I was not used to the differences compared to c which I am most comfortable with, for example the use of white space compared to encapsulation for loops for example. Although, once I managed to correct myself with this problem the main issues with this came from trying to extract the data from the database in the correct format. A lot of attempts were met with either returning no results, empty lists or tuples. I eventually overcame this by trying to identify the problem bit by bit to see what happened when I changed certain lines of code by having the app print out certain variable to the console at certain points so I could see what was going on.

I also had initial trouble with the template extension, but this was quickly solved due to misreading from the workbook and was corrected and utilized throughout the app. I feel like I would benefit a fair bit from trying to go over some simple python problems to help become more confident with the language and more easily identify errors as they arise, such as casting from tuples to strings. Also I would spend more time trying to understand bootstrap templates to help with my css.

Overall, I think I performed quite well with the site being fairly neat and easy to use, along with the database being robust as to not allow injections.

References

[1] S. Chawla.

[2] P. S. Foundation.

author = "Shaantam Chawla" title = "Flask Web Development Tutorial 3: Web Forms" year = "2016" url = "https://www.youtube.com/watch?v=lIi6e5oDZ68"

Author = "Python Software Foundation Title = DB-API 2.0 interface for SQLite databases url = "https://docs.python.org/2/library/sqlite3.html