Name: Daniel Ems
Date: 06/10/2017
Current Module: SQL Development Using Python
Project Name: Safetydome

Project Goals:

The goal of the project was to provide users the relevant information from our sql database. Our server was to host HTML pages and cgi scripts that would allow users to click through and access the information provided.

Considerations:

The project required multiple forms of planning. First, you needed to plan the layout of your web pages. Which web pages were going to lead where? Second, you needed to plan what you were going to display on each web page. Was one web page going to display multiple sql queries, or only one? Third, you needed to plan your sql queries. Finally, you need to plan how you are going to pass information between web pages or cgi scripts. If you wanted the results of one cgi script to reflect the input/output of a different page, how would you do that?

Initial Design:

The initial design of the project included creating the different pieces that would ultimately be put together. This included identifying the HTML documents and where they were going to lead users. Also, developing the cgi layout that would organize the traversal of the website through different pages.

Data Flow:

The program begins by presenting an HTML page. The HTML page provides links to three separate cgi scripts; battle.py, combatant.py and ranking.py. Each of the previous cgi's access the database and perform a unique query that is presented to the user in HTML. The battle.py presents the combatants in a fight. The combatant.py presents all of the combatants. Finally, the ranking.py presents the combatants who have fought in a battle. Each of these cgi scripts have click able links that can take users to respective secondary cgi scripts. Battle.py to battle_details.py, combatant.py to combatant_details.py and ranking.py to combatant_details.py. The detail pages have unique query's that offer more information about the selection the user chooses on the cgi script prior.

Potential Pitfalls:

There are numerous pitfalls one can run into throughout this project, for example; How do you keep the information organized so as not to confuse the user, as well as other developers? Another; What happens if the database is down, how will your script respond? Finally, are any of your cgi scripts vulnerable to SQL injection? The pitfalls steer you in particular directions throughout the life cycle of your project. One way to sidestep these issues is to plan out accordingly. However, not everything can be planned for.

Test Plan:

User Test:

Each user test started with a very basic, "can I do this?". At the most primitive levels one should be able to accomplish the most simplistic task. For

example; if one wished to print the results of an sql query to the screen in HTML, One must first know if they can query the data. This can be tested by opening mysql and trying to query the information you wish. If you have done so successfully, you have just completed a step in your desired output. Then you would address the issue of printing the information in HTML. This can be done by hard coding your desired output and attempting to print. Next you would need to attempt to query the information from inside the cgi script. This can be done by hard coding the data and outputting the information to either HTML or to a terminal. Finally, you piece it all together to try and accomplish your original goal. This is nothing more than taking a complex issue and breaking it down into bite sized pieces. This is necessary in ensuring you consistently make progress.

Test Cases:

Throughout this project, numerous test cases arose to successfully complete requirements. Several of these test cases included breaking information down to their smallest bits of information. In regards to any sql related queries, I always began by seeing if I could access the information with a query in mysql. If this was accomplished successfully, that sql script was copied over to the cgi script where it would be used. In regards to the combatants_details.py, I made sure I could access as much information from the tables as necessary to insert into the cgi script. The next case would be to see if I can get a handle on the information. This was accomplished by running the script in the command line. If the information I wished to see was present, I proceeded forward. In regards to combatant_details.py I ensured I saw the information I expected in the terminal before moving on to HTML. Finally, The HTML was added last. This is necessary because it required the most amount of tinkering. Adding tags and tables, for loops and if statements in order to see what was going to work.

Conclusion:

The project offered a great deal of new material while also tying in topics we learned last week. Understanding how to communicate between the database, cgi script, and HTML. was crucial in meeting just the minimum requirements. It also offered a unique perspective on security risks with databases. Working to prevent sql injections was not only interesting, but is also a real world application. Finally, HTML was interesting to learn, and difficult to implement considering the short amount of time dedicated to working in HTML. Mainly when used inside of cgi scripts.