Advanced Webtech Coursework 2 Report

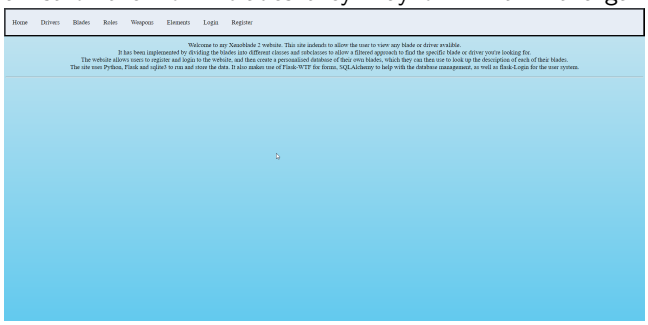
Luke Reeder
40277803@napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

1 Title

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

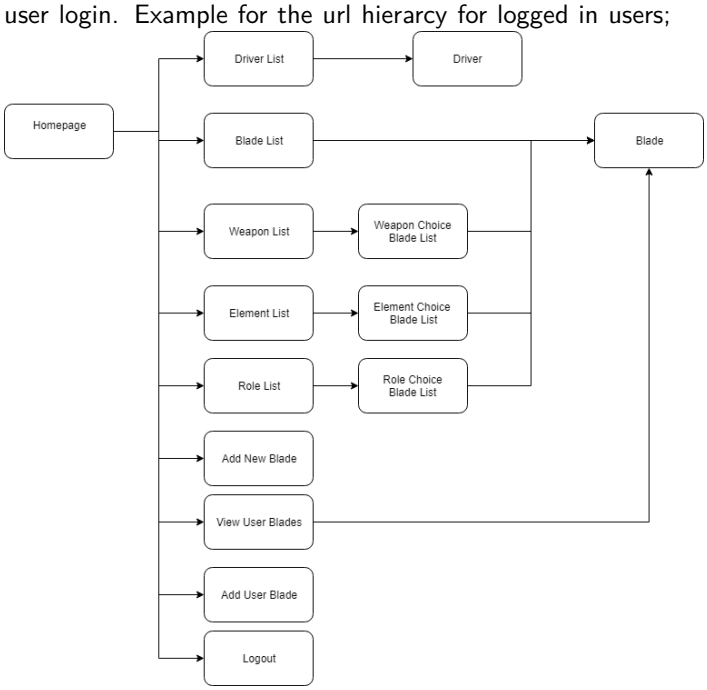
2 Introduction

The web app that I designed is based off of the nintendo switch game called Xenoblade Chronicals 2. It has a database of the blades and drivers that are used within the game, with some blades being left out to make use of the add blade feature. The website also offers the user the ability to register a personal user on the website and then create a list of their own blades they may own within the game.

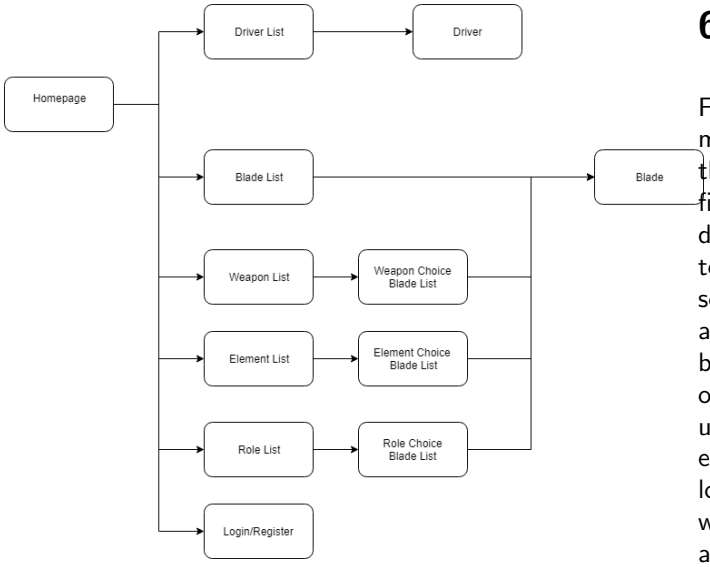


3 Design

My webapp is structured so that the base website allows anyone to be able to browse through the database of blades and drivers. This allows them to access either the full list of blades, and each individual blade itself, or use one of the links to filter through the database by using a series of server side sql searches relating to the link, eg the role link would return a list of Dps, Healer, or Tank. Clicking these would then further refine the search to then display a list of all of the blades matching that role description. This also applies for the element and also the weapon type. Since there are only a few drivers, it returns a list of all of the drivers since there is not different classifications of them. If a user registers a unique account with the website, they will then be given access to add new blades to the database, as well as also adding blades to their own personalized database. This then allows the user to have quick access to their own blades and view their description, while also supporting multiple users at once. Although the website is protected against SQL injections and CSRF attacks, it is possibly still open to a XSS attack. The user information is stored in a user model, which the password is hashed using werkzeug's generate hash method. This newly hashed password is then passed into the database, and the check hash method is called to authenticate the password. Certain parts of the site require an authenticated user to view or use, such as the add blade and view user blades. This was achieved by using the flask login manger to display the links to these parts as well as the logout link only if the user had been verified and logged in successfully. If the user had not logged in, the navigation bar would not display these links, and instead display the links to register or login to the site. The flask-wtf was utilized to generate the forms for the login and register, both validating that the user had put in data for each part before passing the data to sqlalchemy to query the database and return a user login. Example for the url hierarcy for logged in users;



And a url hierarchy for a logged out user;



4 Enhancements

One of the main enhancements I would like to add to the website is the ability to add the blades to the user database straight from the blade description. I did initially try to make this the case, but was unable to figure out how to do so. Other enhancements I would make would be to sure up the site security wise, researching into more possible attacks and how to counteract them such as the XSS attack mentioned earlier. I would also like to migrate all of the internal sql to the alchemy database rather than having a mix of alchemy for the new additions, but still using raw sql for the server connected sqlite database.

5 Critical Evaluation

For the web app, I built a site that allows a user to browse a database, register and login to the site, add new blades to the blade database and then also create their own personalised blade database. For the most part, I feel as if the site works well, although I need to implement more error handling for example if the user searches for a blade that does not exist it causes an exception. For the forms, I feel as if they work very well due to the clarity of the forms as well as the verification provided from the flask-wtf form library. This ensures that the fields have to be filled when registering, followed by sqlalchemy providing further varifacation of a unique user and password. Some of the features require the user to be logged in to view them, however, if the user knows the url for the logged in path, they would still be able to type it in manually. To counteract this, I have utilized a redirect for the site for the user blades and adding new blades to the database back to the login page to ensure they cannot access that part of the site. This is done by using the flask-login manager's currentuser.is-authenticated method. For the user's database, I again used the flask login manager to find the current user's id to feed into the sql query for the search to then display the correct table. Overall, I feel that the additional libraries were utilized well throughout the app.

6 Personal Evaluation

For this coursework, I feel I performed fairly well, but with still more room for improvement. One of the main parts I felt as though I improved on was learning how to better debug and fix my website, mainly due to the plethora of bugs that arose during implementation. Some of these bugs were caused due to import loops which led to restructuring the app's back end, some small silly mistakes such as a missed capitalization or a dot where an underscore should have been. Others were a bit more tricky to fix such as when trying to work with some of the newer flask libraries utilized, mainly making sure to use the proper syntax for each part. I believe that with more experience, these would become second nature and have a lot less errors occuring in building. Other challenges I faced was trying to come up with other features to implement into a system as I usually struggle with being creative in that sense. I tried to overcome this by searching for other popular sites and then seeing what features they have that I could feasibly implement on my own site while learning how to do so efficiently.

7 References

For the site, I used "The Flask Mega-Tutorial" by Miguel Grinberg to help learn about some of the new features i'd never used before and how to implement them. I used additional flask libraries for the app including Flask-Login, SQLAlchemy and Flask-wtf.