Coursework Report

Dominic Mcluskey
40277061@napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies cw 2 (SET09103)

1 Title of web app:

The VOID

2 Introduction

My web app is a type of anonymous social media for sharing only insults. You can navigate the pages using a navigation bar or "navbar" located at the top of the site. The navbar will change appearance based on what page you are currently on to show you what page you are on. You can browse various insults based on what has the most likes, the newest insults. You can even use a tag based system but that is hard coded meaning new tags added don't have pages generated unfortunately. The home page has all the jokes in the database. There is an initial 10 insults and 9 users that populate the database on initialisation. This is to allow users to see and understand what is expected of them when they eventually make an account and also shows off the formatting I used for all the posts. You can register a new account using a unique username and a personal password. If you try and set your username to a username that is already in use or any other type of registration error, you will be given an error screen berating you for trying to troll the system. If you manage to successfully register then you are directed to then login. The theme of the entire site is petty and mean spirited. The login form will also call you an idiot if you forgot your details and mocks you for losing them. When you have logged in, the Home page will change to allow the user to enter in an insult and set the tags of this insult. Up to 6 tags can be added. Posting this insult will have it on the home page and permanently be stored there. On the newest page your new post will also show up. Both versions of Home, Newest and Hottest are all dynamically generated based on what is in the database.

(Figure 1 is a screenshot of my web app home screen, Figure 2 is a screenshot of my web app home screen zoomed out and Figure 3 is a screenshot of my web app home screen when logged in)

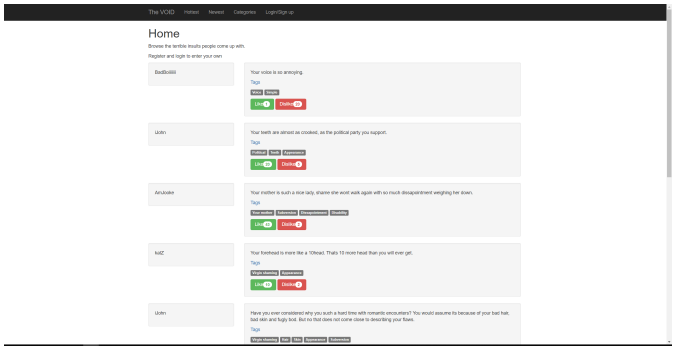

Figure 1: **Web app home screen**

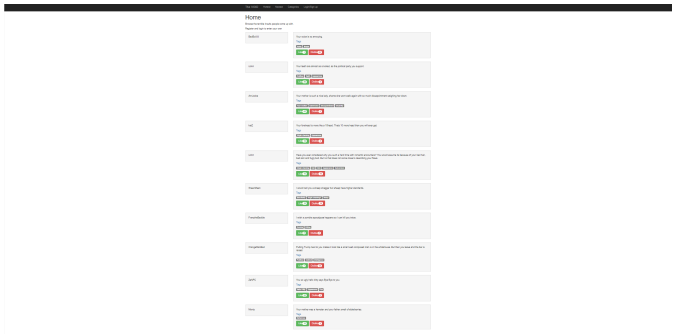

Figure 2: **Web app home screen zoomed out**

3 Design

My web app is structured around the homepage. Since the pages retrieved change based on if a user is logged in or not, I did not make that many URLS to navigate around. In terms of privacy the only thing kept that is secure, is real user passwords. Aside from that the usernames, insult posts and post values are all kept unencrypted. These values are all visable to everyone who accessed the site meaning there is no need to encrypt them on the system. There is a serious flaw in that certain inputs into the database could be insecure. Mainly posted insults and tags are of key concern. No commas should be added into the insults entered and tags. A higher level of database input validation should be considered if this site were ever to be made public. Nothing of value or intimacy would be lost. It would just potentially break the site. A conscious and serious effort was made to make this site not keep anything important. I decided to make accounts easily lost and easily remade by design. The lack of entering an email ensures two things. Firstly it means I don't have to encrypt something else besides a password. Secondly it means if accounts are lost or if there is a data breach, there is no method of tracking people back to the site. In the current authoritarian online climate, avoiding the destroying of lives by a user posting improper jokes and it getting found out, is a primary concern.

The debug page is also unreachable now as the final version of the website should not have the debug as it is a minor security

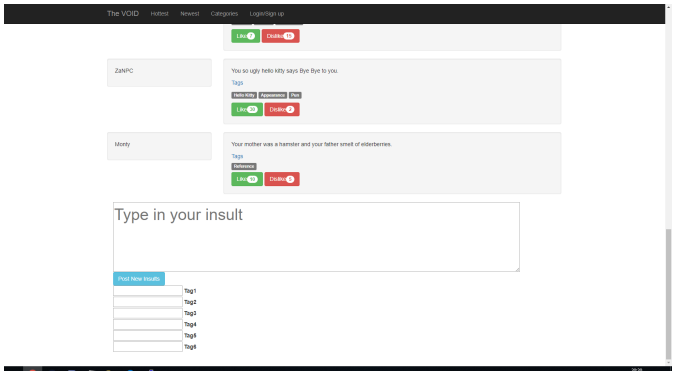

Figure 3: **Web app home screen input bar shown**

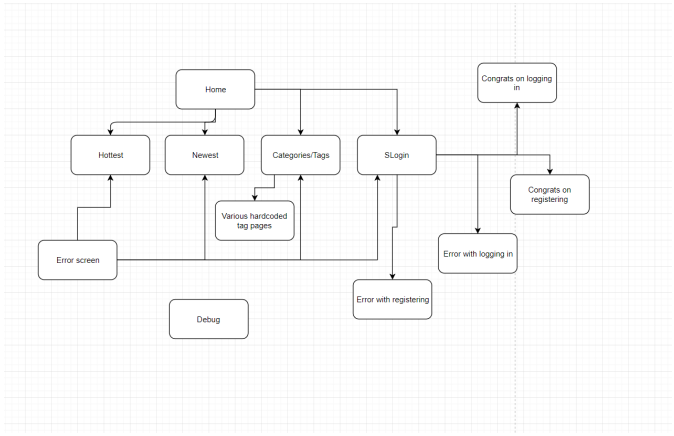

Figure 4: **URL hierarchy**

risk with people being able to see their hashed passwords, perhaps someone could crack the system by de-encrypting their own password.

3.1 Enhancements

I would have liked to get the tag system working dynamically. I had a good idea how to do it but my only problem was working out how to pass data from an html page, back to a python file and back to an html page again. I am sure there was some method I could have found and used if I had more time to do it. It was a nice idea but I just did not have the experience to code it. The like and dislike system was also not finished. I had the same problem I did with the tag system. I did not know how I could cause a number to increase, then go to the database, updating its value and then going back to the page without reloading the page at the top and disorientating users. The like and dislike system would have been unique, in that a user could simultaneously leave a like and a dislike on an insult. This is because a good insult is not one that has a lot of up-votes only. A truly great insult would have around an equal number of likes and dislikes because it is so insulting and effective at getting a response from people. Another feature I envisioned when starting this project is that users would have a little badge if one of their insults became the most upvoted or downvoted insult. It was perfectly within my capacity, all I would need to do is implement different versions of the Home, Hottest and Newest screens that have a little badge telling the user they did a good job. However as I never managed to fully implement the like and dislike system it would be a feature users of the site would never get to use and therefore a waste of effort to make it before the like dislike system. If I had more time I would like to make the database a lot more secure by using input validation to prevent users from entering data that will insert SQL code to cause problems with my database. Also I think making all values stored in the database encrypted would make more sense considering my site's stated goal of preserving privacy. Another feature I would add is more user feedback. When a user accidentally enters in the wrong password a little notification should pop up telling a user they entered in the wrong password. Same with registering and logging into the site, a part of the screen should be dedicated to showing the user's username and or allowing them to choose an avatar that is also shown in their posts. Allowing users to upload images opens a whole new can of worms with security and stopping illegal/immoral content from being exposed to unsuspecting users but I like the idea of giving users more freedom and self expression on the site.

3.2 Critical Evaluation

I am proud of the login and registration system not because it is perfect but because of how many problems I ran into to get it running. Most of the code is inspired by the workbook chapter 10 and 11. I struggled greatly getting anything done at first. I am pretty sure I managed to make it so when entering the values of a new user registering, the SQL insert statements are done in such a way that an SQL injection attack cannot occur. So that works quite well for my site. The navigation system works within a greater session control system that checks a user if they are currently logged in or not, to change what pages are generated. This I feel was done quite effectively and lowers the need for coding many numerous similar html pages. There is a ginger for loop that also dynamically populates both Home pages, the Newest and Hottest pages with the insults that are currently in the database respective of what they wish to return. I am actually quite proud of how I managed to figure out how to do this. Firstly in python I got all the values from the database, put them into a variable, then looped through that variable for each row in the database and for each value the database stores a new array is created and each instance of a value is added to their corresponding arrays. I also have another list that will act as a index in the html loop generated later being generated in python. These values are all then passed. In the html a new ginger for loop is made with the post format specified and the database values are passed through their individual lists. This allows my Home pages to be dynamic and function for millions of insults posted in theory. The insult posting and tag system was done within my ability. I think a character limit, or some method of generating an infinite list of tags would be preferable, perhaps you could filter posts by pages with a system similar to Reddit's system because scrolling down the whole page while making users observe other users' insults will only become more tedious as more insults are posted.

3.3 Personal Evaluation

Setting up the database by itself was kinda a pain to do but I managed to get that all done relatively quickly. The registration system was perhaps the easiest to do and it was easy to

change how the registration works so usernames are unique. I felt that the SQL was perhaps one of the languages I was strongest in when working on this site. The login system took me the longest to figure out. Not the actual logging in part, the making sure the details were correct part took me the longest. I had a hard time wrapping my head around the concept and step by step of how the login system should work, so using the workbook ch 11.3 I got a lot of the code I needed, I just needed to sit down and figure out how it would all go together to get the result I wanted. This was a lot of trial and error. To get past this problem I physically wrote down the steps my program should take so the logic made sense and started over again doing it that way. My plan worked and I managed to get it working. The next problem I had was figuring out how to dynamically generate pages. At first I tried using page appends with for loops in python but this method, while "working" made it so my bootstrap file did not load meaning my lovely posts and format, I spent hours making look good, would go to waste. I decided against using this idea as a friend of mine suggested I look up the programming language "ginger". With the extra functionality ginger allowed me, I had to plan how I would enter in all the data returned by the python file. Once this was figured out, through the method described above in my critical evaluation, every other part of the site was copy paste with minor alterations to have pages dynamically generate. I feel like I performed quite well, all things considered. I had some major roadblocks but once I had an understanding of what I wanted to do and how to do it, I got everything done quite quickly and without incident. A small hiccup did occur later in that I needed to change how my database stored some values from text to INTs but that was a simple fix that took me 2 minutes at most.

3.4 References

I used various parts of W3Schools tutorials to find out how to format and create certain forms in html. https://www.w3schools.com/html/ I used stack overflow to answer some of my questions about how python stored arrays. https://stackoverflow.com/questions/1514553/how-to-declare-an-array-in-python I looked around for methods to write if statements in html and found documentation for a complimentary language called ginger http://hackage.haskell.org/package/ginger-0.8.1.0/docs/Text-Ginger.html I also used a W3schools page to find out how to do forms for my login/signup page using basic javascript.