Conspiracy Nuts

Scott Postlethwaite
40281026@live.napier.ac.uk
Edinburgh Napier University - Module Title (SET09103)

1 Introduction

The aim of this report is to explain and justify the design decisions taken within the development of my application.
My site is an answer to the significant lack of interactive conspiracy sites online. Through intensive research I found that the only dedicated sites for conspiracy theories were wikis with no user interactivity whatsoever and as conspiracies are such an individual entity and such a talking point I felt that theorists deserve a platform to voice their opinion not only on their own conspiracies but on others as well.
The server side elements are handled by flask where the front end is generated by a series of filled in jinja2 templates. For persistence I have used a single json file containing a series of posts, users and comments.
This site has been set up to allow any user to register an account and start posting and commenting. Users are also able to follow other users and keep up to date with all of their posts. If users have made a mistake they can also edit or delete their posts. They can also update their bio and profile picture as well as changing their password. I feel this gives the user enough interactivity with the site to keep it engaging while also keeping it easy to use.

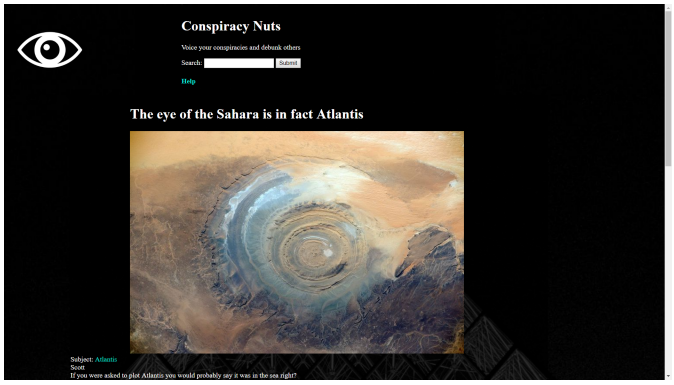

Figure 1: **Home Page** - The home page of my blog)

2 Design

2.1 Server Side

The server aspect of the blog is upheld through Flask. The HTML is generated by several templates that are dynamically populated with 'post' objects from a json file.
The entire site was coded on the class development server to ensure that there are no unwanted bugs upon assessment.

I initially used the workbook [1] and Flask Documentation [2]to supplement my learning. I followed the tutorials for routing, redirects, error handling, requests and URL variables to create a functioning search feature which allows the user to search for the name of the sighting, the year or the country of origin This then generates a page with the corresponding data in the json file using URL variables. I had previously implemented a redirect only system but that limited me to only hard coded entries so I felt it had to be changed in order for my site to scale. The URL variable method allows user uploaded sightings to be dynamically generated easily and the provided navigation tools make it easy to get where you want to go.

2.2 The HTML

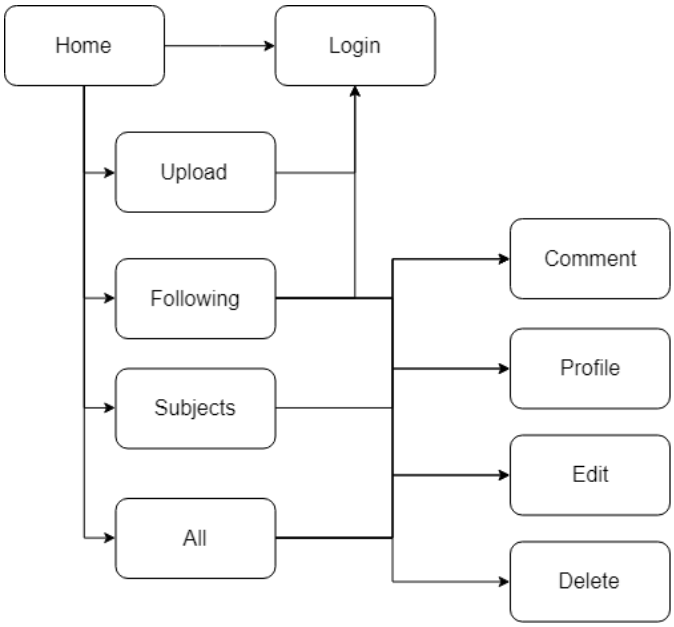

Figure 2: **HTML Navigation Map** - a navigation map representing the URL hierarchy of my site)

There are relatively few URLs in the site, this is because the majority of pages are dynamically generated through URL variables as discussed earlier. I feel that as users mainly use in built navigation as opposed to typing the URL in each time this does not make a difference to them. It also made my site scale a lot better than before.
The each page's HTML is generated by Flask templates keeping the same design but showing different information. The main page is simply a search bar with a hovering logo which acts as a drop down menu. For the search bar I followed the 6th chapter in the workbook [1] and experimented with different routing methods to find the fastest and most scale-

able option. In the end I decided to fill a template with the posts that match the search. The main drawback of this is the lack of a share-able URL. I feel the speed of the search and the variety of navigation methods make up for that so I have stuck with this method.

The alternative navigation methods are to show all subjects. This provides the user with a list of all documented subjects which contain hyperlinks that have dynamically generated links to URL variables. This allows for the user to click any subject and see all of the posts surrounding it. The reason I have used auto generated URL variables is that it allows for new entries in these lists to be added without the modification of the HTML. This list is updated as part of the upload feature. If you were to upload a post with a subject that has previously not been documented it will add the subject to the list.

You can also view the posts from users that you are following. This will append the posts to a results list if the author is in the current user's following list. When a user is followed they are added to the current user's list of followed profiles. The followers is then incremented. The unfollow method is similar where the user is removed from the list and their followers count is decremented. While I could have stored the user's followers as well as a following list I felt that that would be overkill for my site as I would like the user's list of who they are following and who follow them to be private. If I were to add this in the future I would allow users to create private profiles so people could not snoop on their activity.

For uploading, the user first must log in to the site. If they do not have an account they can register relatively easily. Once they have logged in they are greeted with a series of text boxes requesting a title, the subject and their theory. This is then appended to the json file. The user is also given the option to upload an image to support their theory. If they do, the image is saved into the static folder and the file name is added to the sighting in the json file. Images are kept optional so that any theory can be published no matter what evidence you have.

For editing the user is redirected to a url generated by the template. The url variable determines which post id will be passed through to edit. The user will be greeted with a series of text inputs which are all filled in with the post's current attributes. Upon submitting these text inputs are passed to the post object and dumped into the json file. This will only run if the author is equal to the current user. Deletion works in much the same way where if the author == the current user the post object will be removed from the post list and the list will be dumped into the json file.

All of the relevant information will be centered on the screen with a logo at the left corner. The logo will act as a hidden menu as upon hovering it will display each function available to the user in the form of links. To make use of hidden menus and text I will first inspect websites that I use myself in order to see how they have implemented similar features. I will also make use of the W3 schools tutorials[3].

2.3 The Python

The base of this sites functionality is in three lines of python. Opening a json file, reading from the file and appending to the file. Once this was figured out all I had to do was add a few if statements to allow searching through the data and I had a functioning site. To log in to the site users have to input a user name and password. The password was encrypted using Bcrypt and were checked against the stored users. If this was successful the user was added to the session and the session type was changed to logged in and the current user is stored in the session. This allowed me to have users edit their posts by checking that the current user was the author of the post. Deleting works the exact same. I have also allowed users to follow other users. This is done by checking whether the user they want to follow is in the following list of the current user. They are then appended to the list and the follower count is incremented. Unfollowing works the same where if they are in the list they will be removed and their follower count will decrement. Each post has a list of comments. This is handled in the same way as posts where if you are not logged in you will be redirected to the login page. When the user makes a comment it will show under the post along with the author which is taken from the session. These are all written to the json file using json.dumps().

In order to properly format text without making life more difficult for the user, I used the string.replace method to replace new line characters with break tags. In order for this to work in the jinja2 template I had to remove auto escaping. As soon as that was implemented I decided that the feature was ready to go live so I provided some test users the URL and let them loose on the site. Due to the custom error routing, refined user creation, editing and post system, none of them were able to break the site.

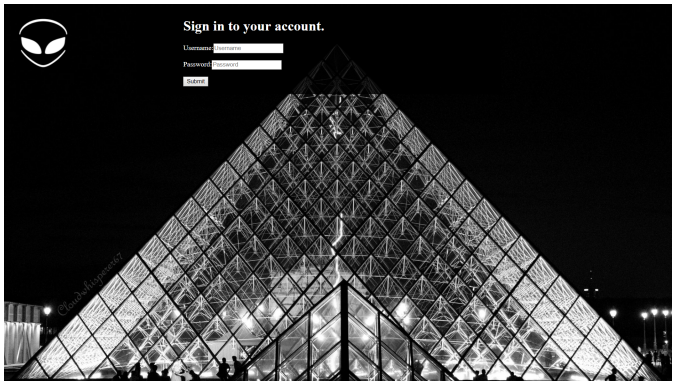

Figure 3: **The Login Page** - The login page for my site

2.4 The CSS

For the design of this site I started with white posts over an off white background similar to facebook's. When my users got a hold of this they thought that it was quite dull. I have been very against the use of colour as it can either make or break the design of a site. I feel that the user's uploaded images provide the colour that is needed for the site and that any more would be too childish and overbearing. it is for this reason that when I was looking for backgrounds I would exclusively look at monochrome images. In this search I found an incredible black and white photo of the louvre in Paris which was licensed for reuse. As this was used for the closing scene of 'the Davinci code' I thought it would be fitting for the site's background. I set about altering the css to suit this

background. The posts that once had a white background to apear as pages were now transparent black as not to obstruct the user's vision from the text while maintaining the background. I had users test both versions of the site and it was clear that they preferred the option with the background. It was said to look more professional, less like a word document and more unique. When asked which they would prefer to use they all chose the back grounded option. If is for this reason that I made the switch and have implemented this to my site. The main challenge that I had with this was that when changing the opacity of the div used in the background it would also change everything contained inside. This made text hard to read and impossible to edit. I found a solution to this in the form of rgba backgrounds where you can change the alpha level. This effectively achieves the same outcome so I persued this solution as opposed to opacity.

Figure 4: **User Profile** - A user's profile page)

2.5 Data Storage

The entire directory of posts is held in a single json file along with a list of each user and comment. I chose to use a single json file as opposed to separate files per entry as I found through initial testing that it sped up my search feature and made the site generally quicker.

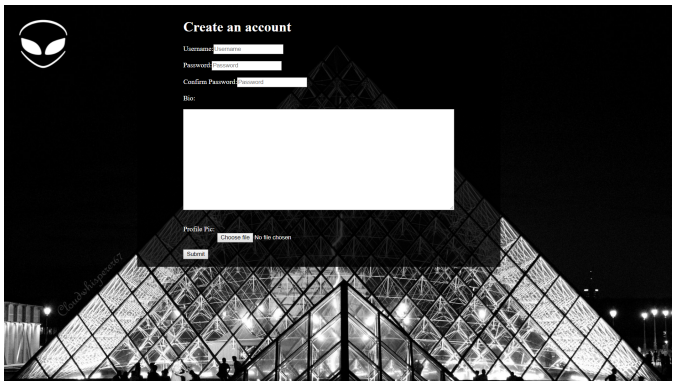

Figure 5: **Registration Page** - The registration page of my site)

I allow registered users to create posts with an image. This appends their post to the existing json file and saves their image to the static folder. Any user can then comment on this post. I have done this to allow users to get involved and interact with each other. I feel this is the main feature that sets my site apart from the plethora of conspiracy wikis online.

The user is then able to edit and delete their own posts. This is done by checking whether the user logged in matches the author of the post. I do this through the use of session markers to store the current user and when a post is made recording the author as the username stored in the session. Users are also able to follow other users so they can keep up to date with their posts. This is done by initializing an empty list inside the user class and appending to it each time the follow button is pressed. This will add the user followed to the list. The unfollow button works in the exact same way however this removes the username from the list. I feel that the following and commenting aspect of this site differs it majorly from the conspiracy wikis that are so prevalent online. It provides users with what I feel to be the correct level of interaction with each other. I did think of adding a messaging feature to the site however I felt that this would rarely be correctly used and often abused by users and, as much of the site's data is encrypted, it would be impossible for me to regulate.

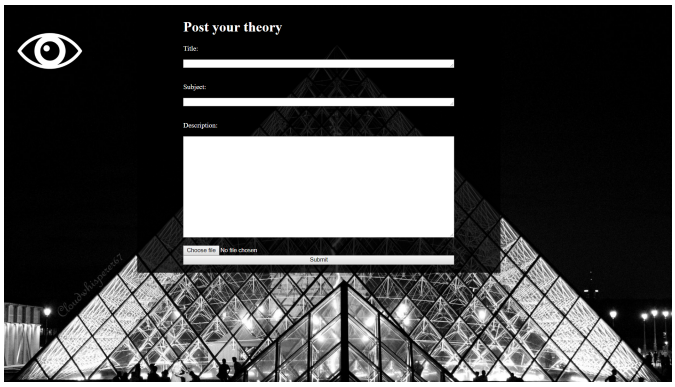

Figure 6: **Upload Page** - The writing page of my site)

I had implemented an sqlite3 database in a copy of my main python file however I ran into a few issues when adding comments to the posts. I had comments stored in a new table where post id was used as the foreign key. This linked the two tables in such a way that they would link to the post that was commented on. In practice this caused a few issues. First of all the comments were not compatible with my jinja template. Upon fixing the template it would only show a couple of the comments as opposed to them all. This was all working perfectly in the json solution and in practice the json solution loaded in faster with no issues what so ever. It is for this reason that I decided to stick with json for the purpose of this site. If I were to do this again I would start using a database and center my logic around it to reduce the likely hood of these errors cropping up. If the database did not cause errors I would have persevered with it and I feel that in the future it could make it easier to add new features such as a private messaging service however I did not feel that messaging was needed for this site which meant that there were no obvious drawbacks to using json.

3 Implementation

The implementation of this site is very simplistic. Posts can be viewed, filtered, searched, added, edited, deleted and commented on all without an SQL database. I have used a variety of methods such as hover tags, dynamic sizing to make my site look professional and pleasing to the eye. I have allowed the user several points of interaction which allow them to navigate the site easily without fear of getting lost or encountering errors. To ensure this I let a small group of target users loose on the site with no instruction or direction. The feedback and results were generally positive. They really liked the idea of being able to follow conspiracy theorists and to interact with them in the comment section just like how they would in a social media site such as Facebook. When users were actively researching and engaging in discussions the key navigation feature they used the most was filtering by subject or followed users. It is for this reason that I added these to the search feature along with a separate list. If the user could not find something using the search feature they are notified that it "could not find what they were looking for they could add it through the upload feature." This use of custom error routing means that the user is never disconnected from the site. I feel the functionality of the site is it's greatest feature. There are many improvements that can be made to the site which I will discuss below however as a beta product I feel that it meets the user's needs and is pleasing to use.

4 Implementation Evaluation

4.1 Does It Meet The Spec?

The User is able to navigate throughout my site with ease. They have several ways of doing this: either through a search bar where they can search for a given subject, title or user, a list of all subjects or all posts from followed users. This allows the user to see every reported theory and filter by the subject and the author of the post. All posts are displayed underneath the search bar on the home page. There is also a separate page in which you can view all posts. I feel that the plethora of options for the user to navigate the site make it a pleasant site to use and interact with.

Users are then able to create posts, comment on any post, edit and delete posts. This is done in such a way that only the author of the post can edit and delete their posts whereas any user can comment. This allows for a multitude of possible interaction methods with the site. The fact that users are able to upload an image to support their post and the way that the text will be formatted without the user needing to add any break tags or new line characters allows users to create posts in as much depth as they like. Users can also create a profile page where they can upload a profile picture and a bio. They can then update these at any time. These profiles will show all of the user's posts. I feel this gives the site great scale-ability and provides users a platform to share their thoughts and theories however they see fit.

The writing portion of the site is locked behind a log in system to keep trolling to a minimum. As registering accounts is an easy process, I believe a script to determine whether entries are legitimate and whether the files submitted were

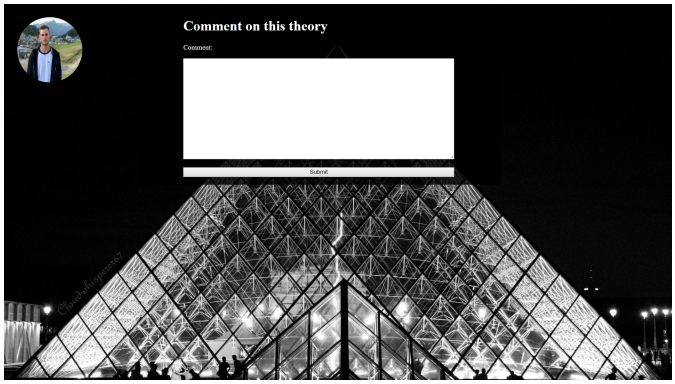

Figure 7: **Post A Comment** - The input for posting a comment)

safe for work would benefit my site greatly and I would look into adding this if users felt that it was needed. The writing portion of the site has similar styling to Microsoft word with the white text area over the slightly off-white background. It contains the same style sheet and logo for design continuity and brand recognition. Through research I have found that users respond better to web pages keeping the same design with only subtle changes. As far as advertising and brand recognition goes, a well designed, subtle, ever present logo has proved one of the most effective ways to drill your brand, site or application to your user to keep them coming back. That is why I have implemented similar ideology for my site.

Figure 8: **Edit Page** - The editing page of my site)

4.2 Potential Enhancements

If this were to go live I would add features based upon real user feedback. As it has only been tested by a select few, in a closed test environment, I was not able to see whether problems like bullying would be prevalent on the site. If this were the case I would add a block and report feature to help tackle this. If I were to go about this I would implement it in the same way that I have done the follow feature however it would exclude them from interacting with or seeing the user's posts.

While I feel it would be overkill for my site, I would like to provide users with a chat feature. This could be in the form of direct messaging or as a live chat. These messages would need to be encrypted to the highest standard as conspiracy

theorists are extremely conscious about how they communicate. It is for that reason, plus the fact that I feel it would be abused without regulation, that I have decided to exclude it from my site. If I were to add it in I would use an sqlite database to handle the storage of these messages and Bcrypt to encrypt the message. I would store the username that sent the post along with the id of the chat and the encrypted message. This would be a useful feature for some users however I feel a comment section will do much of the same thing while keeping everything interactive and engaging for the community rather than just the participants.

I could allow users to share posts. If I were to do this I would create a shared post list inside the user class and would append to this whenever a post is shared. This would then show alongside the user's posts on their profile and the following page. The reason I have excluded this from my site is that in the prototype of the feature that I had, it would show the shared posts beneath the rest of the posts rather than alongside. The only way that I was able to fix this was by adding the shared post to the user's collection of posts. When doing this though it would show the user who shared the post as it's author. Because of it's buggy state I decided to remove this feature. I would like to add a refined version in the future though.

While I do not agree with the feature myself, it was suggested to me that I could add a rating system to each post. This would be a simple implementation as all I would need to do is add a rating signature to the post and allow users to up vote or down vote the post just like reddit. This would then allow me to show users the top rated posts based on this figure. The reason I do not agree with this and have not added it however is the fact that abusing this system has a detrimental effect on young people's self confidence. Even though this is a site for sharing conspiracy theories I still feel that people would look for some kind of validation in this figure and as Twitter are trying to get away from this system I feel that it will die out and become a thing of the past. While it would be nice to see what theories are trending I feel that a like or vote system would not be the way to do this.

If I were to add a trending list I would like for it to show at the side of the page. I would like it to show the subject with the most posts in the past 24 hours along with the post with the most comments in that time as well. I feel that these would be a good indication of what is being talked about at the moment. I did not add this in to the original site as I felt that facebook's system of showing all posts would be a better fit for the site however as the site has progressed I am starting to believe that a trending tab would be of more use and will offer the user greater interactivity. I feel that out of all of these enhancements that this is the one most relevant and so should be added first.

5 Personal Evaluation

I would love for this site to go live as I am extremely proud of my work. I feel that there is a need for this site in the conspiracy community and I feel it can only develop further based on feedback from a plethora of real world users. I feel that I have performed well especially in the physical design of the site. The design in terms of user interaction is what I am most proud of. I feel that the research and experimentation payed off and have made the site look pleasing and professional and these skills will stick with me throughout future projects.

The server side aspects of the site were completely new to me. I am pleased that I have managed to develop a site that not only generates the HTML from a series of HTML templates that I didn't even know was possible beforehand but that it also manages to enable you to store and write new entries to the directory all without the use of a database with full persistence on the site. I couldn't be happier with the server side of the site.

I have faced many challenges throughout the making of this site. My initial challenge was with dynamically shaping the background. To begin with everything was fixed meaning if you have a 1080p display you would find the website pleasing to use but if the site was windowed or used on a lower resolution display, the border will extend past the screen. This makes my site almost inaccessible to anyone with different specifications to myself. As I am unaware what computers my site will run on I had to make the text's border dynamically fill the screen. This caused me a multitude of problems as initially my border would fit in the screen but would shift my text too far to one side. In order to then center my text I had to put my background on a second div two above the body so that it would not affect the layout of the text.

I spent a lot of time struggling with adding posts. The problems came after I had users test my initial upload feature. When they tried to post without adding an image the site would crash. To fix this I added a simple if statement to check that there was a file in the input. Upon adding this I noticed that sightings in my template would still have a 300px bar above the text in place of an image. To sort this I had to add an if statement to the template to check that the sighting had an image and if it did that another div should be added with the img tag inside. There are many other ways that this could be achieved however, I feel this method is the most consistent. If I were to improve this in the future I would have a query that checks the type of file being uploaded as to only allow supported image types.

Adding comments was another challenge as I was working on 2 separate forms of the site at once. While in the json solution it was as simple as appending the comment to a list, in sqlite3 I had to connect the two tables through the use of a foreign key, in this case post id, and add them to a comment table. In practice this would add to the database perfectly fine. The problems came when I displayed these comments to the user. There was a problem with my jinja template where the comments would not display. This was working perfectly in the json solution which seemed to load the page faster as well. For those reasons I stuck with the json and stuck with it while adding new features.

Following users was another problem for me as when a user was followed it would append to the json list fine however when you went to view them on the site it would still show that you were not following them. I eventually discovered that this was due to the session not being updated. I then updated the user in the session to equal the user with the updated following list. This fixed all of my issues and taught

me how to use sessions properly.

Along with this I also struggled with the formatting of text. I had initially tried to add ¡br¿ tags into the text however they were outputted as strings in the html. I then tried to replace all new line characters with break tags using the string.replace function but the output was the same. Eventually I found that the problem was in the way that jinja was handling the variable. The solution was to turn off jinja's auto escape feature which would close the tags where it saw appropriate. As soon as I had done this the text was formatted perfectly and led to a professional looking site.

My greatest challenge has been finding the best data storage solution for my needs. While I had started development using json files I decided that it would not be a viable solution when it came to updating or editing posts. Because SQLite3 was so new to me I struggled with it's implementation. Eventually I had two working versions of the site: one using Json and one using SQLite3. The json solution allowed my users more functionality and did not have any negative side affects for the user so I decided to stick with the json solution. My problem with the database solution was that the user could not make comments or follow users. I eventually handled following by having a separate json file to hold a list of who the user was following. For comments I had used the post id as the foreign key in order to link the two. The problem with this was in displaying the comments to the user. There was a bug in the template where it would not show any of the comments despite the fact that they were all in the correct format. As it worked perfectly in the json solution and in practice was slightly faster I decided to stick with it. If I were to go about adding any new features that I felt required a database in order to function properly, such as a chat feature, I would consider reintroducing it again in the future but for now I feel the json solution is all the site needs.

References

[1] S. Wells, *Advanced Web Tech Workbook*. 2018.

[2] Pallets, "Flask documentation," 2018.

[3] w3schools, "The world's largest web developer site," 2018.