Michael Fleming

coursework-2

Table of Contents

[Introduction to the web app 3](#_Toc467862796)

[Design Stage 3](#_Toc467862797)

[Typography 3](#_Toc467862798)

[Color Theory 3](#_Toc467862799)

[Layout 4](#_Toc467862800)

[Routing 4](#_Toc467862801)

[Error Handling 4](#_Toc467862802)

[Flask and Jinja 4](#_Toc467862803)

[Generator Function 5](#_Toc467862804)

[Enhancements 5](#_Toc467862805)

[Critical Evaluation 6](#_Toc467862806)

[Evaluation 7](#_Toc467862807)

[References and Resources 7](#_Toc467862808)

# **Introduction to the web app**

The Web-app I have I designed is a simple blog website containing posts mainly about web related security issues and other tech related problems. The web-app has a blogs page where users will be to view the specific blog they wish. Once they have selected their specific blog the app will display the relevant information about the blog such as title, author, date posted and the blog itself.

# Design Stage

The web-app I have designed has one main file webapp.py which contains all the main functions and routes for the website. Other folders include templates and static. The static folder contains CSS and JavaScript files, while the templates folder contains all the .html files for the website. Maintaining a neat and tidy folder and keeping on top of folder management was something I aimed to do throughout the project. I felt by doing this I could locate files and images etc. a lot easier. This resulted in very few errors arising from incorrect file path names.

The following tools and languages were used during development

* Twitters Bootstrap Framework
* Flask
* Json
* Python
* Jinja

Tools

* Levinux
* PyChamr
* DB Browser

## Typography

The main font type I plan to use on the web app is Helvetica because it is one of the most famous and popular fonts in the world, but also very easy for users to read on the web due to its dpi (dots per inch), compared to that of sans-serif. Although serif fonts are more formal and professional than sans-serif Helvetica, I still feel it gives a sophisticated and professional feel to the web-app. Helvetica is also a very easy font to read so users will have an enjoyable reading experience on the web-app. Due to these reasons I intend to use Helvetica as the main font choice for the web-app.

## Color Theory

The color scheme I plan to use on the website is a basic one which is overshadowed by the dominant background image which is more prominent. The color scheme I used was a monochromatic scheme. The color I intend to use is #EEEEEE which will be used as the background color for jumbotrons and thumbnails on the site. The navigation bar will be transparent so it fits in with the background image and will have white colored text. By using a monochromatic color scheme, it allowed me to use different shades of #EEEEEE on the admin pages to make it clearer for the user to see.



## Layout

The layout of the website will consist of a welcoming message which users will see as they first enter the website. The welcoming message will be displayed in a jumbotron and will feature a view more button which will lead users to the all blogs page. The layout of the site will be kept consistent throughout with the layout and design of blogs kept the same on all pages. The position of the navigation bar will also be kept in the same position and will be easy identifiable with the white color of the text standing out to the user. The background image will also be used on every page visible to users other than the admin area of the website.

## Routing

All routing in the app will take place in the webapp.py file. The main routes in the app include /blog-post and /blog-all. The blog-post/id will load the blog template and populate it with the content the user requested. This works by using the unique ID number given to each blog in the content table in the database. The blog-all route is the page where users can view all blogs available on the website.

## Error Handling

Error handling on the website will consist of a 404 page which users will be redirected to if they request a URL or page that does not exist. The 404-error page will display a brief message stating that the user has requested a page that does not exist and will be given the option to return home and view other blogs on the site.



## Flask and Jinja

Flask is the micro framework for python that I chose to help build my blog website. Using this framework, allowed me to use routing, render templates and response objects. As well as using flask I used the well-known template engine jinja 2. Using Jinja made it a lot easier to create a database driven website. This was particularly helpful when using Urlfor function which easily allowed to create specific pages for each blog. During this project, I used skills learned in the last project such as parsing data from my database on to the page rather than hard coding certain parts of the website.

The image below shows the jinja 2 tags in use on my index page where the for loop is used to parse data on to the page. By using the for loop in jinja it would create as many html thumbnails as there are blogs in the database. This helped to save a great amount time during development that allowed me to spend on other aspects of the website. Using a SQL lite 3 statement I could limit the number of blogs to 6 on the home page, while showing all of them in the blogs page. This is shown in the second image.





### HTTP Authenitcation

During the development of the website I implemented HTTP authentication to authenticate users trying to access the admin area of the site. Using HTTP authentication, I could simply add the @requires\_auth to all my admin routes. By doing this if a user was to request and admin page on the website they would be prompted to login with admin details. If the user enters the incorrect login details they would be redirected to an error page. The error message would tell the user the details they had entered were incorrect. To increase levels of security the error would not state weather it was the password or username that was entered incorrectly. This way it would leave a hacker or someone trying to access the admin page unaware if they got the password or username correct. The reason I did this was to give out as little information as possible as to why the login failed for security reasons. Below is the message that will be displayed.



# 

# Enhancements

Below are the enhancements I would like to make to improve the app.

One enhancement I would like to make to the app is allowing admin users not only the ability to view current posts that are in the database but also the ability to edit and delete certain posts if they wish. At the moment, the only way to add or edit a post is to do it manually in DB Browser. This way is not practically certainly if someone with limited technical experience wanted to add a new post. The way I intend to do this is add an update details on the edit\_blog\_post.html page. This would allow users to edit details of any blog very easily and quickly. The page would also contain a delete button where users could easily delete the blog if they wish to.

Another feature I wish to implement into the app is the ability for users easily add a new blog to the site using the admin page. The page would have very similar design as the edit\_blog\_post.html page. This page would include an image upload function where users would be able to upload images from their local machine to the database used to display images.

Another improvement I would like to make is the ability for users to register to the site so they could upload their own blogs. Allowing users to register on the site instead of myself adding them manually into the database would be a much more practical and functional way of adding users to the site so they could make changes and add new content.

# Critical Evaluation

During the build of the webapp I feel that I have built on skills learned during the first to help build a functional webapp. Overall I feel this project has been successful and I have improved my knowledge of flask and jinja. However, I feel that there are areas of the app I can work and improve on. The biggest negative to the app I can see is the lack of functionality in the admin area. At present changes to any of the blogs content would have to be done manually using DB Browser to make the change. This is one downside to the app which I feel greatly limits functionality. If the app did offer this kind of functionality my next step would have been to implement a simple image upload feature where users could upload images directly to the database. This feature would also include a function that manually edited the file path for the image so the user did not have to.

A positive feature the app offers is dynamically created pages and individual page for each blog. At the start of the project I wanted to create one base template for blogs which would then populated with content specific their id number. This was something I was unable to do in the first project, so it was something I was really keen to get working on this app.

Another positive of the webapp is its navigation and clear design. Once of my aims was to keep the blog very easy to use and view especially on mobile and desktop. Using Bootstrap allowed to easily design the website to be fully responsive on the app. I kept the design consistent throughout with the transparent navigation bar kept in the same position. The only change to this was in the admin area of the site where the color of the text changed to better suit the design in this section.

# 

# Evaluation

Throughout this project I feel my skills have been tested and as a result my knowledge and understanding of python has improved greatly. I also feel I have built on skills learned during the initial coursework which has helped extend my technical experience. Although I feel my python flask and jinja skills have improved, I still feel I have areas of weakness, which became apparent at some points during this project. One area I struggled with, was adding functionality into the admin area of the site. At the start of the build I intended to allow admin users to edit and delete content in the database. However, when trying to implement this functionality to the app it became apparent that with my current skill set I would not be able to achieve in the timescale set.

Throughout the build a lot of languages were used including HTML, CSS, JavaScript, Python Flaks, Jinja and SQL Lite 3. As during the first project I already an understanding of HTML, CSS and JavaScript as I had used them in previous builds. Using SQL Lite 3 for the first time during this project was also something I found challenging. Progress was slow using SQL Lite 3 epically at the start when I was trying to create the database. Once I had finally created the database I felt my SQL skills slowly improved and in time I could do more. An example of this was limiting the number of results shown from the database on the index page. I found trying to get working SQL Lite statements one of the most challenging aspects and I found myself searching the internet and books to find examples. However, I feel now that I have a deeper understanding of SQL Lite and that if I was to use it in future projects I would be able to create more complex apps.

After completing this build, I feel that I have built on skills built on skills and knowledge gathered from the first project. This has allowed me to add a much more functional app than the one I had built before. During the development of this project I have spent a lot more time adding functionality rather than reading over error messages which is what happened during the first build. This allowed me to create a much more functional and modern app. The skills and knowledge I have gained here will also help me in the future to help create an app with even more functionality.

# References and Resources

Bootstrap - <https://getbootstrap.com/>

Flask Documentation - <http://flask.pocoo.org/docs/0.11/>

Python Documentation - <http://docs.python-guide.org/en/latest/>

Background Image (../bg.jpg) <https://startbootstrap.com/template-overviews/business-casual/>

All blog posts images and content taken from their post on the <http://www.theregister.co.uk/>

<https://www.tutorialspoint.com/flask/flask_sqlite.htm>