Coursework 2 Report

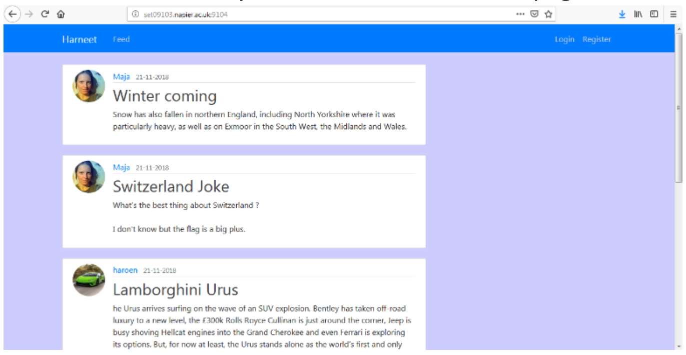
Haroen Ahmed
40403712@napier.ac.uk
Edinburgh Napier University – Advanced Web Technologies
(SET09103)

Contents

Introduction

The aim of this project was to create a web app for an online directory about a subject of my choice. This report is to demonstrate my understanding of Python and the micro-framework Flask.

Harneet blog is the web-application product I created that is running with Python within the Linux environment. The concept of the app is to allow registered users to post content. Other users can read the posts, but a user doesn't need to be registered to read posts. When the user is logged in they can post content by clicking on the new post link in the navigation bar. Then the user has to fill the in a form by entering a title for the post and the content he/she wants to display. When the post button is clicked the post is posted onto the feed page of the web app. Once the post has successfully been made the user will automatically be redirected to the feed page.



Maja 21-11-2018
Winter coming
Snow has also fallen in northern England, including North Yorkshire where it was particularly heavy, as well as on Exmoor in the South West, the Midlands and Wales.

Maja 21-11-2018
Switzerland Joke
What's the best thing about Switzerland ?

I don't know but the flag is a big plus.

haroen 21-11-2018
Lamborghini Urus
he Urus arrives surfing on the wave of an SUV explosion. Bentley has taken off-road luxury to a new level, the £300k Rolls Royce Cullinan is just around the corner, Jeep is busy shoving Hellcat engines into the Grand Cherokee and even Ferrari is exploring its options. But, for now at least, the Urus stands alone as the world's first and only

Figure 1 Home page

Design

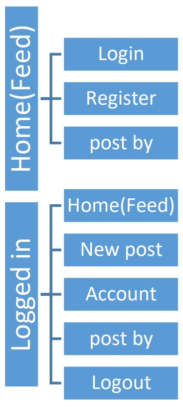


Figure 2 navigation map

The previous image shows the navigation structure used in the web app. I have structured it this way because I find it is the easiest and most logical way to navigate and run this app. The architecture of the application is composed of 6 different routes that are running through Python and flask.

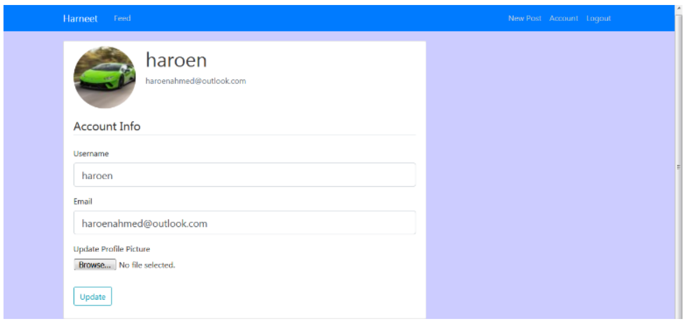
The first route (home) is the page where anyone can see all the posts made with the username, profile image, date, title and content displayed. The most recent post displays at the top of the page. The app allows 5 posts per page. This is so users don't need to scroll too far down in order to see posts, instead they can use the pagination system at the bottom of each page and view other posts on different pages.

The second route (register is the page where users can register an account. They can do this by filling in the registration form that can be found on that page. The validation code behind this registration form works well. If the user enters a username that already exists, the system will give the user feedback by displaying an error message informing the user to use a different username. If the user enters an invalid email address, an error message will be given explaining what went wrong. The same happens if the user enters un-matching password.

The third route (login) is used to allow already registered users to log in. If the user enters invalid data (wrong email or password) the app will tell the user to check the email they have entered and the password. The app gives the user that is logging in the option to choose if they want their log in details remembered. When clicked, the app will remember the users email and password so when the website is revisited they can log in straight away by just clicking on the login button. When someone that isn't registered with the web app lands on this page, they have the option to register an account by clicking the link at the bottom outside of the form.

The forth route (user posts) can be accessed by anyone. Users can access user posts by clicking on the username of a poster, this will take them to the users post page where they can see the amount of posts made by that user. All posts will also be displayed and filtered nicely, showing the latest post at the top and the oldest one at the bottom.

The fifth route (account) can only be accessed by registered users and can only be accessed when logged in. This page displays the user's information. It will display the users profile picture, username and email. The email and username can be changed on this page. If the user decides to change their username for example, they can fill in the form by changing their name and updating it by clicking on the update button. If the username is not already taken the newly entered username will be set instantly and the app will give the user feedback by giving the user a flash message saying they have successfully changed their username. The below update form is already populated with their username and email. See figure 3



haroen
haroenahmed@outlook.com

Account Info

Username: haroen

Email: haroenahmed@outlook.com

Update Profile Picture: Browse... No file selected.

Update

Figure 3 account page

The sixth route (new post) can again only be accessed by users that are registered successfully and are logged in. This page allows users to make new posts. Very little is needed to do this just a title and the content of the blog post. When the post button is clicked and the post has been made the user will automatically be transferred to the home page where they can see their post live. When on the homepage, at the top of the page the user can see that their post has been posted successfully. This is confirmed by a flash message telling that their post has successfully been posted.

Coming to the choice of the design of the web app. I have used Bootstrap4 with Jinja2. I found a nice fluid bootstrap theme on boostrap.com. It makes the web app more attractive. This makes users feel welcome and attempts to get them to stay longer on the web app to create an account and explore the apps features. The app is fully responsive and can be viewed on any device. The layout is consistent throughout the web app giving the app a professional feel.

Security

With regards to the security of this web app I don't believe there are any issues.
The user's information is stored on a database (sqllite3). Their information is safe and secure on the database. The passwords are hashed, so even if someone was somehow able to gain illegal access to the database, their efforts would be useless as they would still not be able to attain any password. Email information is also encrypted. These encryptions are secured to the extent that even the developer or admin can't see this information.

Enhancements

Due the lack of time and limited experience I was restricted to what I could create. If I had the necessary time and skills I would add a feature to the login page that will have a forgotten password link at the bottom of the page that will take the user to a request new password page, where the user can enter their email and receive their password in their email.

On the account page, a feature that would change the profile picture of the user would be nice and it is something I would like to implement if I had more time. This would work through a file upload where the user can click on a button and search on their device what image they would use. When they select an image they can submit the change by clicking on the update button.

Another feature that I would have implemented which I did not think of doing whilst development of this app is a comment section on the feed page. Where users could interact with each other by replying/commenting on each other's work.

Another thing I would have created is a search bar where users would be able to search and view their desired posts. The search bar would also give suggestions to what information is available for the user to see.

Another feature I would add would be an admin panel, where the admin would be able to log in the web app and delete and modify content and user information.

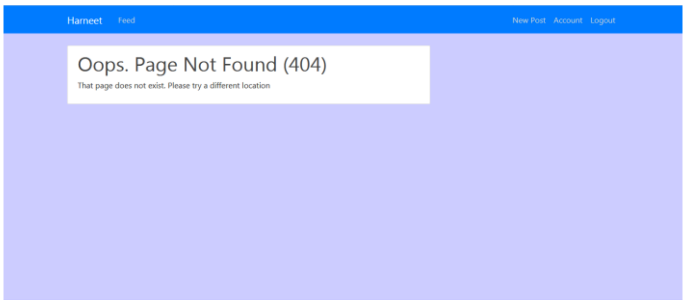
With regards to coding, all the data of the users and the posts are stored on the database. This data is all fetched and displayed on the website, the passwords are hashed meaning that they are very difficult to crack if someone was to hack the database and get access to sensitive data. Although it will still be hard to crack the password because they are hashed.

Critical Evaluation

The main aim of the web app is to allow user to interact with the system and to some extend with other people. This can be done through posting content onto the website and search the website for posts made by fellow users. Users can also update their username and email if they want to.

A feature that works well is the navigation bar, it is simple and effective and will change into a drop-down navigation bar when the screen size goes smaller. When a user logs in, a session is set which will give the user the option to logout and access more links in the navigation bar.

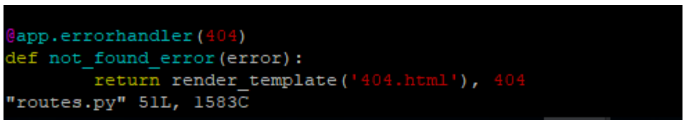
I have added custom error handling pages to the app. These pages will inform users if they have tampered with the URL. If this happens the app will inform the user what has happened and it will give the user the option to return to a previous page.



Oops. Page Not Found (404)

That page does not exist. Please try a different location

Figure 4 showing the error page



```
@app.errorhandler(404)
def not_found_error(error):
        return render_template('404.html'), 404
"routes.py" 51L, 1583C
```

Figure 5 showing the error handling code

Personal Evaluation

For the second course work with Python-Flask, I have struggled even more than the first assignment. With help of fellow class mates and a lot of studying and tutorials I managed to make a decent application. I learned a lot from this project and will use these skills for future projects.

I struggled at the beginning with setting up the databases and creating the tables, I have used databases in the past but those databases had a user interface which made using them easy. For this project it was done through the command line and it was challenging. I made a lot of errors while typing at the beginning, but after watching a few tutorials on Lynda.com I managed to do correct my mistakes.

Using the Linux command and git went a lot smoother this time as I had worked with them on the previous assignment.

While working on the project I had to use any time I had to educate myself on how to effectively use database systems with python and flask to achieve my goal. At the beginning it was difficult and frustrating working alone and not being able to understand anything, but with persistence, and continuingly studying books, online resources, watching and working through video tutorials, I managed to get a good grip on how it's done.

I have used a few flask plugins that were not installed on the dev server. I used these to help me make the application work in the best and quickest way possible. I used online video tutorials to understand these plugins and libraries.

I used flask-wtf library to help me with rendering and validating forms. Flask-wtf can define the form fields in python script and render them using html templates. It is great for applying validation to the wtf field. To install flask-wtf you can use the pip install flask-WTF command.

I also used SQLAlchemy, it is a very powerful python toolkit that gives application developers the flexibility and power of SQL. Using raw SQL in Flask web apps to perform queries on databases can be very annoying and tedious. SQlAlchemy is a library that facilitates the communication between python programs and databases such as sqllite.

SQLAlchemy can be installed using the pip command: pip install flask-sqlalchemy Then you will need to import SQLAlchemy class from the module: from flask_sqlalchemy import SQLAlchemy.

I have tried to make the user request a new password by sending the user a link to their email with a reset password link, however I was not capable of doing it in time, but I have used the Flask-Mail library and I am familiar with it. It is great for web-based applications and it is often required if you want your system to send any emails to users or clients. You can install Flask-Mail by using the pip command: pip install Flask-Mail.

Finally, I used Flask-Login to manage user session, it is great for tasks such as logging in, logging out and remembering the users over extended periods of time. I have chosen this library because it helps to store the active user's ID in the session and it allows the user to log in and out with ease restricting views to logged in user, and users that are logged out. It helps protect the user's session from being stolen by cookie thieves. This flask library can be installed by typing install flask-login in the command line.

In conclusion, I believe that I have learned and made a lot of progress since the first coursework. I know that there is still a lot of room for improvements, but I feel that I have succeeded in making a professional application.

References

The Bootstrap framework used to create the website:
https://getbootstrap.com/

Learning Resources:
Simon's Workbook https://moodle.napier.ac.uk
https://www.codecademy.com/
Vim: https://www.openvim.com/
https://www.youtube.com/playlist?list=PL6gx4Cwl9DGDi9F_sl cQK7knjtO8TUvUs (flask videos)
https://stackoverflow.com
https://www.w3schools.com
https://www.lynda.com
http://flask.pocoo.org/