Idea and features

I chose to implement a blogging platform for my website and contains many features such as:

- A user can sign up for an account and login
- A user can change his password
- A user can logout
- A user can create a blog with preferred name
- A user can post many times to one of his blogs that he created
- Posting include title, content and an image which is optional to add.
- A user can view and like or unlike other users' posts
- A user can view how many likes a post has
- A user can search for a blog by name
- A user can follow or unfollow blogs of other users.
- A user can update a previous posts title and/or content if need
- A user can delete a post
- A user can view his statistics relating to followers, likes, number of posts and blogs

Website details

Website link: http://hashish.pythonanywhere.com/

Username: Blog

Password: Project123

Analysis

Web forms:

Flask WTF-forms were used for client-side validation. It was used in the signup form and it made sure that all fields were not empty. In the Sign-up form, the input entered in email was checked if it is in a form of an email with a maximum length required of 50. In the username field, the minimum length required is 4 and the maximum was set to 15. In the password field, the minimum length was set to 8 and maximum to 80. The Sign-up form also validated that the password and confirm were the same by checkout if there are equal to make sure the user did not type the password incorrectly. If any of the validators is not met, an error with a message relating to the validation is displayed to the user.

Another form was used for the login, all fields were checked that they are not empty. The email and password were used to login to the website. While the username was used for users to identify each other. A remember me checkbox was added in the form.

Another form was used for the Posts, it validated that both title and body were not empty. The same was in the Blog form where it validated that the blog name field was not empty.

Resetting or changing the password had a form that confirmed the user’s new entered password.

In server-side validation, for registering, the server checks that both email and password are unique by comparing them to what is stored in the database using queries. Also, server-side validation checks that the username and email are unique when registering. For login, the server validates by using queries to compare email and password with the ones stored in the database. If the entered data does not equal to what is stored in the database, an error is displayed. For posting, the server checks that the user selects a blog to post to otherwise an error is displayed.

Database:

3 models were used to store information:

1. Users
2. Posts
3. Blog

All models had their id as the primary key.

There are 2 many-to-many relationships and 2 one-to-many relationships.

The first one-to-many relationship is between User and blog as the user can create many blogs, but a blog can only have one user.

The second one-to-many relationship is between blog and posts as each blog can have many posts posted to them, but each post can only have one blog.

The first many-to-many relationships was a table called “followers” that linked user with blogs as users can follow many blogs and a blog can have many users.

The second many-to-many relationships was a table called likers that linked users with posts as a user can like many posts and a post can have many likes.

In the user models, the id, username, email, and password were stored to be able store the user credentials. The user models also stored the user’s blog, blogs the user followed and posts the user liked.

The blog model stored the blog name and it’s authors id and posts posted to each blog.

The posts model stored title ,date posted ,content ,author ,image name and its blog id.

Sessions /cookies:

When a user enters their credentials in the log in page, they are redirected to their account. The user can view posts from his followed blogs. Also, the user can change their password and edit or delete previous posts as much as they want. The user can view their statistics and post to their blogs. When the user logs out, the user’s page can’t be accessed unless the user logs in again. A remember me checkbox can be checked by the user while logging in. Checking remember me will store a cookie with the user’s credentials so that if the user closes the browser accidentally for example, the user will be logged in his account when opening the website again.

Authentication:

Flask-login was used to authenticate users and log them in and out. It protects accessing pages when not logged in. When the user logs in, the user’s credentials are compared to the data in the database and if they are equal, the user is logged in using flask-login.

Styling:

Bootstrap and CSS were used to style the website and make it responsive for the user. In all screen sizes, the website resizes its elements to a suitable size. At small sizes, the navbar collapses due to lack of space for the navbar. A 3 (white, blue, black) color scheme were used in website.

Logging:

Different severities were used in the logging.

Info was used to output messages for actions such as:

1. Logging In successfully
2. Creating an account
3. Creating a blog
4. Following a blog
5. Posting to a blog
6. Liking a blog
7. Changing password
8. Logging out

Warning severity was used to log if user:

1. Entered a wrong password
2. Failed to register due to a meeting all validators
3. Did not select a blog when posting
4. Failing to search for a blog
5. Failed to change password

Error severity was used to log if user:

1. Post with an image of unsupported format or submitting a pdf file etc.
2. Tries to update another user post
3. Ties to update a non-existing post
4. Tries to view a non-existing post

Error severity was used to log if user tries to access forbidden routes such as :

1. Like
2. Delete post
3. Follow

Error severity was used to output error messages of view functions if an unexpected error or bug occurred

All the logs were written to a log file where the admin can access it whenever needed.

Advanced features:

jQuery was used so that the user can delete posts without the page being refreshed and when a post was deleted, it fades out and gets deleted. jQuery also outputs a confirmation message for user to confirm the deletion of a post to avoid deleting posts accidentally. Moreover, when a user likes a post or follows a blog, jQuery was used to append them to their respective table and the button clicked changes text when clicked without the page being refreshed so not to ruin user experience.

Bootstrap was used in all html to make the website responsive. It was used to make posts and sections at appropriate sizes. Also, it was used to place icons, buttons, and navigation bar links in appropriate places. Moreover, it was used to pick different types of buttons in the website.

Evaluation and user experience

The blogging platform was designed so that it is minimalistic and does not overwhelm the user. For evaluation, 5 user accounts were signed up and errors relating to validation were displayed. Signing up, logging in, creating blog, posting, deleting posts, updating posts, following blog, changing password, viewing post, liking posts, and searching for a blog were tested and data for each section were tested and behavior was observed. Fortunately, the website acted as expected and errors were handled and written to the log file. Links for sign up and login page were added to navigation bar, so the user knows what that he needs to sign up and then login first. After signing up, a success message is displayed so that user is informed that their account has been created. Both sign up and login page have titles to show the user which page they are in.

When logging in, the user is directed to the home page. In the home page, all the blogs' posts the user has followed will be displayed. Each post has a follow/unfollow button so that the user can follow/unfollow a blog as he wants. Also, each post has its blog name so that the user knows which blog this post belongs to, post title to know what the subject of the post is, date to know when the post was posted and number of likes to know if the post is popular or not.

If the user decides to view a post, the title of post is colored with a different color, so the user knows it is clickable. When the user clicks on the title, he is redirected to the post page containing an image relating to the title and content, title, date, like/unlike button, and content. If the user likes the post, he can click like to show appreciation for the post. The like and follow change to unlike and unfollow when clicked to show the user that the website indeed registered their click.

The user may want to explore posts from blogs they did not follow of any subject, so an explore page has been added that contains all posts from blogs posted from the website. The posts in the explore page have the same elements as in the home page.

Sometimes the user might want to search for a blog with certain names relating to subjects of interest. That is why a “search blog “link was added to the blog platform. There, there is a search input field where the user can type in a blog name and click the search button. If no blog contains that name a message is displayed, but if a blog/blogs is/are found, they are displayed to the user with the number of followers for that blog. In addition, there is the follow button if the user wants to follow that blog.

The user might want to create a blog then post to it. There is a link to create a blog, where the user can type in a blog name and create it. The user is displayed with a success message so that he knows the blog has been created and gets redirected to the post page.

In the post page, the user can select any of his created blogs and type in a title and body. Also, if needed, the user can add an image/gif that will be displayed in the post page so that other users who view the page will not bored of only plain text. The user is displayed with a success message so that he knows that his post has been posted.

The user might want to change his password due to security concerns or other issues. If the user clicks on “my account” in the navigation bar and then “profile”, a page is displayed with a reset password link where the user can click and change his current password. The user is displayed with a success message that he is informed that the password has been changed successfully.

The user might want to view his statistics such as number likes and number of followers etc. A table has been created in the user’s profile page with some of these statistics

Most importantly, the user might need to edit or delete previous posts. To do that, the user can open “my posts” that contains all of his previous posts and the user can edit or delete any post. If the user clicks “update” button, he is redirected to a new page with the title of post and content filled in. The user can append or remove from them as much as the user likes. After clicking update, the user is presented with a success message. If the user clicks delete, he will be presented with a confirmation message to avoid accidental deletion. If the user confirms the deletion, the post fades away and is deleted from the database.

After finishing using the website, if the user is using a shared computer in his home or for other reasons, the user might want to logout to enable other users to login and to prevent his data from being accessed by other people. The user can click the “logout” link that logs him out of his account.

Potential security issues

Potential security concerns were with storing passwords as plain text and if accessed can be stolen easily. So, the passwords were hashed to protect them from potential theft. To protect sessions and attackers from hijacking authenticated sessions flask-security was used to authenticate and login user with correct credentials and log them out when needed. To protect against Insecure Direct object reference, all pages are protected using flask-login “@login-required” and check that the current user is authorized to access their pages. Https has been enabled on the deployed app to provide secure storage and transit security to protect sensitive data.

References

1.My blogging platform was inspired by CoreyMSchafer web app

https://github.com/CoreyMSchafer/code_snippets/tree/master/Python/Flask_Blog

2. As I am using python 3.6 and I wanted to use flask-whooshalchemy, I had to use another GitHub that maintains it flask-whooshalchemy to be usable in python 3.6

https://github.com/miguelgrinberg/flask-whooshalchemy

3.Like button icon

https://www.iconfinder.com/icons/1814076/approve_like_thumb_icon