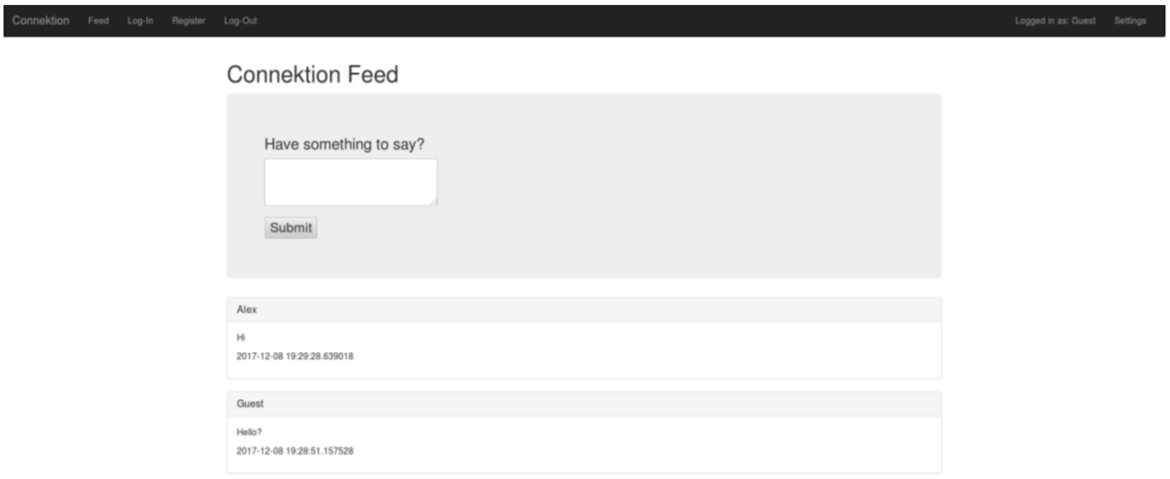
Connektion Web App Development

Introduction

Connektion is a web application that combines blogging, anonymity and social media. It has a feed where users can post any short messages they wish to be displayed. Connektion is unique because it offers the choice between anonymity and profile. If the user wishes to be associated with their messages they can register and log in, if not they can use the Guest username.

Features

The homepage of Connektion is the feed, which has a textbox form for the user to type in a short message and then submit it. The feed is sorted so that the most recently posted messages appear first. The user can choose to post on the feed as either a guest or with a username. At the top of the homepage (and all the other pages), there is a navigation bar with options to direct the user to the feed, login/logout page, registration or settings. The navigation remains consistent across all pages, which was a decision made during the design phase to improve usability.



Alex

Hi

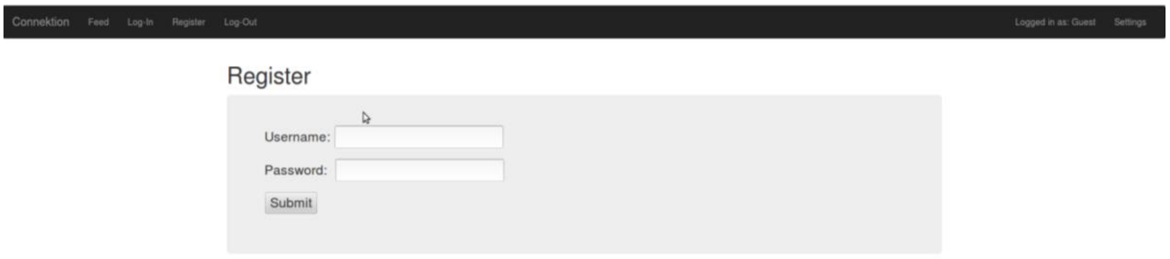
2017-12-08 19:29:28.639018

Guest

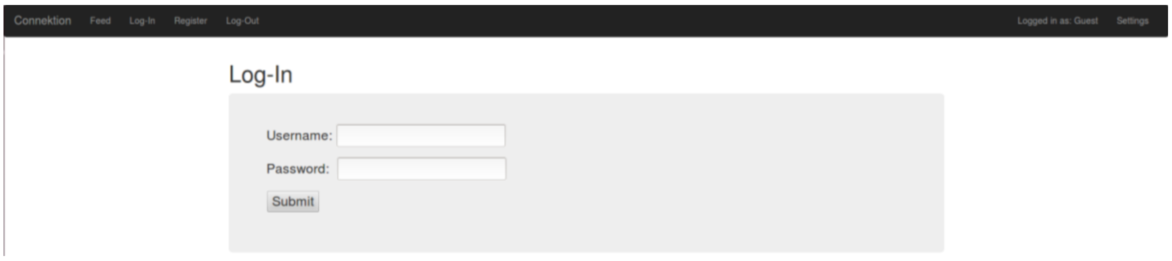
Hello?

2017-12-08 19:28:51.157528

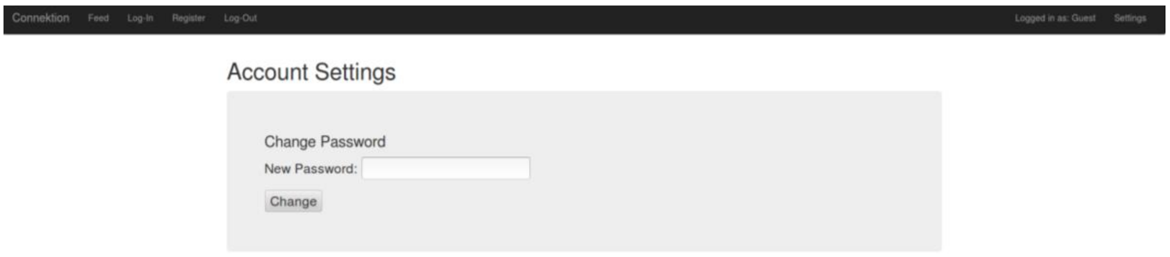
The user can register on Connektion by navigating to the registration page. On the page, the user can use the registration form to create a unique username and a password. The username the user creates is checked against the application database for a pre-existing username to ensure the username is truly unique. If a pre-existing username is found, the new username is not created and an error message is displayed.



If the user has already registered an account with Connektion, they may sign in to the profile using the log-in page. The log-in page is similar to the registration page and uses the same form to receive a username and password. The log-in form checks the username entered against the data for a pre-existing username and if found it then checks the passwords match. If the username is not found then an error message is displayed, similarly, if the password is incorrect for the user an error message is displayed also. Connektion also has a sign-out feature which sets the username back to 'Guest' and redirects them to the homepage. So that the username is maintained when the user changes pages or leaves the web application; sessions were used. Sessions allow Connektion to store data for a specific device on the server-side so that when the user returns, their username should still be available.



The user can change their password on Connektion by clicking on either their profile name or settings in the navigation. They will then be presented with a form asking for the new password. The new password form works by checking the session username against the matching username in the database. When the matching username is found then the associated password for that username is changed to the new password.



Logging is used by Connektion to record useful data on user usage of the web application. Timestamps are recorded of each event and the type of event is also recorded. Registering, logging in, posting and changing password are all recorded in the web application text-file log. The severity of each event is also recorded with low significance events being recorded as "low" severity and high severity events being "high" severity.

Evaluation

In all, I believe I succeeded in creating a cross-over between a social media blogging platform. The application is very simplistic with a lot of room for improvement but it does provide an interface for both anonymous and personal interactions. The web application is fully responsive using Bootstrap styling; this makes the web application very flexible and accessible with the option to use it on a range of devices such as mobile phones or tablets. The colour scheme is minimalistic using a monochromatic theme choice that could be upgraded for user personalisation to be monochromatic on a different colour hue than black-white.

The web application feed provides timestamps for when the user posted and the users username on each post but I believe further improvements could be made to make the feed more interactive. The option to allow users to post videos and photos would make the feed more applicable to the modern social media user. The user could also be provided with a profile (if they have logged in) where they could give themselves a profile picture and perhaps some information about themselves.

In this project, I mostly struggled with modelling the databases. I attempted to make the databases relational but I had no success. In future, I believe I could further research database relationships and implement this in my next web application. Python was not difficult for me on this project and was not much harder than any Python coding I had done in previous projects but I found the Flask framework confusing. Flask does not seem to offer a large amount of online documentation and I found configuring Flask consumed most of my development time. If I was to work on a future project with a choice of web application framework I would definitely consider using a more widely documented and supported framework such as Ruby on Rails.

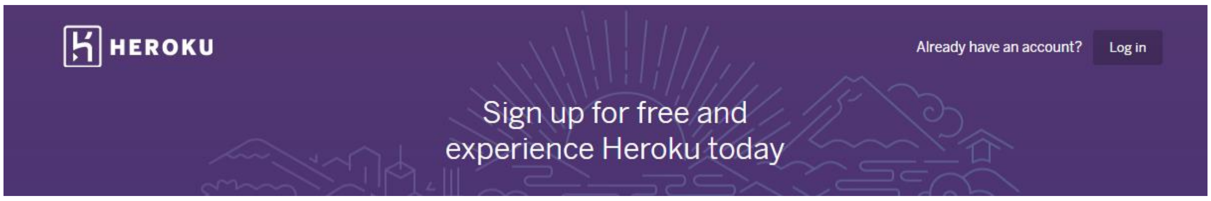
When testing experimental features, the use of Git was very beneficial. On many occasions I broke my application and couldn't reverse the changes so the ability to roll back the changes or switch branches was invaluable. Using Git by choice in this project was a great decision and I have plans to do the same in future projects.

Web Application Architecture

Web application architecture defines the connection between the front end application, the middleware and the databases (*Stackify, 2017*). The front end for my web application (what the user can see) works using templates, views and forms. The navigation for the web application uses the 'base.html' template so that it is rendered consistently throughout the webpages without repetition. A template is also used for the feed, log-in, log-out, register and settings pages. These templates mainly consist of forms, basic HTML and links to the Bootstrap stylesheet. The views python file is responsible for rendering each of these pages and bridging between the models and the front-end. Views handles majority of operations for the web application such as database manipulation and interaction with the models. The middleware in the web application would be considered as the modelling for the database. The models create fields from the forms that are suitable to then be put into the database. Finally the back-end database in my web application stores all the usernames, passwords and posts on the server so they can be accessed and saved at any time as long as the server is operational.

Deployment Plan

If I was to deploy my web application I would use the Heroku service because of its scalability for projects as well as providing support for the Flask framework and databases. Having used Heroku before, I found it very easy to use in conjunction with Git (due to it being compatible with Git commands). This would make it so able to quickly push my changes to my repository and also the Heroku platform. Heroku also starts off as a free service so unless the application was receiving large amounts of traffic the hosting would remain cost effective.



Security Issues

My web application made little consideration for security issues that could occur. Sessions were protected using a secret key but a further effort could be used to randomly generate this key. The web application also has no defence against malicious attacks such as SQL Injection. If a malicious hacker was to use such an attack they could gain access to all usernames, passwords or they could even drop all the database tables. Another security issue to be considered is a lack of password protection. The password on the application is not hidden on the form and the stream of data not encrypted. A use of hashing on the password would ensure the future security of user's passwords before the application could be deployed.

References

Stackify. 2017. *What is Web Application Architecture? Best Practices, Tutorials*. [ONLINE] Available at: https://stackify.com/web-application-architecture/. [Accessed 08 December 2017].

Coderwall. 2017. *Deploying a Flask App at Heroku (Example)*. [ONLINE] Available at: https://coderwall.com/p/pstm1w/deploying-a-flask-app-at-heroku. [Accessed 08 December 2017].

Heroku Dev Center. 2017. *Heroku Dev Center*. [ONLINE] Available at: https://devcenter.heroku.com/. [Accessed 08 December 2017].