Advanced Web Technologies Course Work 2 Report

Darwon Rashid
40280334@napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

Abstract

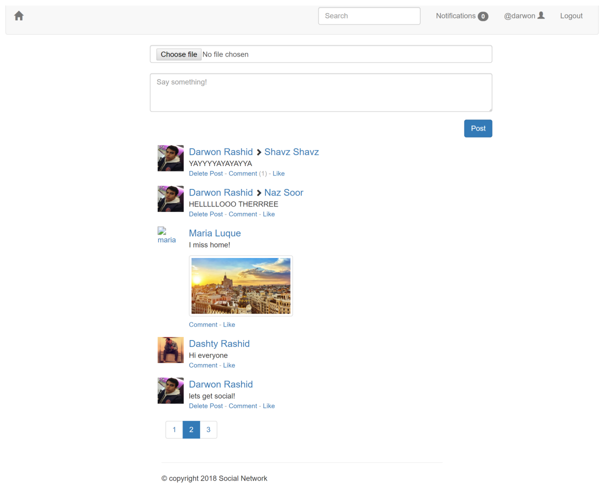
This course work's purpose is to showcase an advanced understanding of the Python Flask framework. The assignment includes an implementation of a Flask web-app that showcases advanced functionality of the framework on the server-side.

The choice for this assignment was a social networking web-app that allows users to communicate with each other. This was chosen for the structure of a social network requires dynamic functionality which was ideal for what the assignment required.

1 Introduction

The web application starts with the user registering up for an account in which they use to communicate with other users on the platform. Once the user has registered, they get a confirmation email in which they confirm their email. After email confirmation, the user then gains access to the platform and either can start posting or start looking for other users to add. When a user finally has a friend, the user can post on its friends profile or the user can send its friend a private message. Any future posts between the user and its friends will be visible on respective home pages. Users can like or comment on each others posts if they wanted to. Users can also include photos on their posts if they wanted to.

The user will mainly navigate the platform by the navigation bar, so it has everything the user needs to use the platform. The navigation bar includes a home icon which links to the home page where the user can post and view posts from either its own or its friends. There is then the profile icon which links to the profile page where the user can view or edit information about itself. The user can also visit other user's profiles, but can only interact with them if they are friends on the platform. The profile icon also has quick shortcuts to edit profile and change passwords. There is also the search box that the user can use to search the platform for users and posts. If a user gets a notification, there is a notification tray in the navigation bar that holds it and provides quick actions for dealing with the specific notification.



Darwon Rashid > Shavz Shavz
YAYYYYAYAYAYYA

Darwon Rashid > Naz Soor
HELLLLLOOO THERRREE

Maria Luque
I miss home!

Dashty Rashid
Hi everyone

Darwon Rashid
lets get social!

© copyright 2018 Social Network

Figure 1: Home Page

2 Design

Before beginning work on the web-app, the structure of how the project is set up had to be done first. Since the web-app is more complicated, there was need to structure it in a way so that it is easy to manage. Flask has guidelines to follow on their documentation on larger apps which were used for this web-app. The strategy was to separate similar logic into their own folders. Static files are saved in a folder called static. These include the CSS and JS files for the web-app. There is also an images folder in the static folder which include post and user images. The forms that are used in the web-app are located in the forms folder. There is a models folder that hold the logic for each collection on the database for the web-app. The model files include classes that are used to represent entities that are linked between the database and web-app. The templates folder holds the pages for the web-app and all other files relating to the presentation side of the web-app. There is an utilities folder that holds two files with functions that are used all over web-app. These two files act as a set of tools that the web-app uses in certain contexts. The last folder is the views folder which include all the routes for the web-app. Each views file has routes for a certain context. For example, user-related functionality lies in the "users.py" file. Doing all this separation is vital for it allows to develop certain parts of the web-app independently.

There is a requirements.txt file which includes all the modules needed for this web-app. The entry point of the web-app is the application.py file, which sets up the web-app and runs it. The configurations needed for the web-app are in the settings.py.

As for the structure of the web application, this hierarchy shows the general outline of how it is set up.

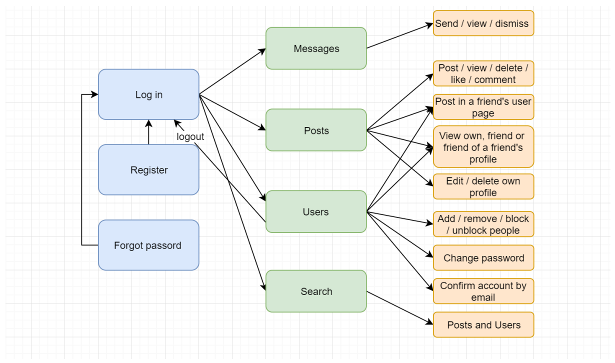

Log in
Register
Forgot passord
logout
Messages
Posts
Users
Search
Send / view / dismiss
Post / view / delete / like / comment
Post in a friend's user page
View own, friend or friend of a friend's profile
Edit / delete own profile
Add / remove / block / unblock people
Change password
Confirm account by email
Posts and Users

Figure 2: Hierarchy

While users can manually navigate to the information they want to find through the web-app, they can also navigate using the url. The structure for how the user uses the url to navigate through the website is very straightforward and is made to be as simple as possible. Users for example, can go straight to another user's profile if they wanted by doing /username. Having it setup like this allows users to cut the time it takes for them to get information they want.

The database this web-app uses is of a No SQL database called MongoDB. The data that users input all go to be stored in this database. For security purposes, no administrator can know the passwords user create for their accounts. Each password is encrypted upon creation, so the passwords are hashed when displayed in the database.

The look and feel of the website was designed by using the popular bootstrap framework that allows for easy and responsive designing. The icons that were used for the web-app all came from Bootstrap for it supports them natively. Bootstrap allows for flexibility and has an extensive set of tools to use for designing the web application. Bootstrap also allows the web application to be mobile responsive, so the look and feel of the web application adjusts to the size of the user's screen. It was very important to make a design that wasn't foreign to the user in terms of usability, so all of the choices taken on how information is displayed and grouped took this into consideration. The web-app also uses Bootstrap Modal, which is a component in Bootstrap that provides stylish and responsive pop-ups.

The Flask framework has a template engine called Jinja2 which is used in the html pages to add more dynamic functionality. There is a "base.html" page and a "navbar.html" page that all other pages inherit from. Jinja2 allows for all the dynamic rendering of data across the web-app. It is used to display the data that it fetches from the server-side dynamically depending on context.

3 Enhancements

As enhancements go, there is a couple that is needed for the web-app. For starters, the overall look and feel for the web-app could be improved. Most time was spent on the server-side and compromises were made due to that. The private message page where users send messages to each other doesn't look the best, so it could benefit from further design changes. I would also make changes to how the search results page looks like.

Since I haven't tested the web application with real users, I would like to update how the user navigates through the url after I get feedback from real users. I would also like to add more notification types. I would add functionality so that when people comment or like on the users post, it sends a notification to the user. I would like to also add a messages page, where the user can go to this page to view all conversations with all of the users friends. As it stands, a user can message a friend by going to their profile and then by clicking message.

I would also like to improve the comments system by allowing users to respond to specific comments in the post. I had no time to implement deleting of comments, so that would be another future enhancement.The error handler in the website is very basic, and for the future I would like to implement a more dynamic error handler that adjusts to the different types of errors. I Would like to improve the search functionality so that it finds posts that belong to friends of the user. As it is now, it finds any posts that is associated with the users search input. Blocking a user for now just means they can't send friendship requests to you anymore, I would like to change it so that they can't view any post or comment by the user that blocked them.

4 Critical Evaluation

4.1 Features

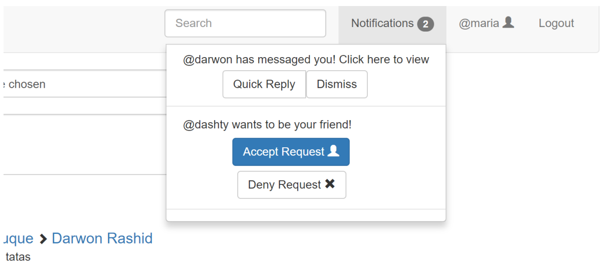
For a social networking platform, there are enough features to satisfy the user. None of the features of the web application are foreign in concept or how they were implemented. There was careful background reading done on how to implement these features.

4.1.1 User Relationships

When the user creates an account and sets up their account, it can start adding other users to be its friend on the platform. The user can accomplish this by first visiting the users profile and then by clicking the request friendship button. The user that was requested now must either accept or reject the friendship request. If the user accepts the request, then they both can now view each others posts and even begin private messaging each other if they wanted to. Users can also block other users that annoy them. The blocked user will not be able send a friendship request to the user that blocked them anymore. Users can also obviously edit their profile.

4.2 Notifications

When a user gets a friendship request or gets a private message, they also receive a notification that appears in their notification tray that lies in the navigation bar. There are only two types of notifications for this web-app. If the user gets a friendship request, they can choose to either accept or deny the request straight from the notification tray. If the user gets a private message, they can choose to either dismiss the message or quickly reply to the message by bringing up a pop-up form (using Bootstrap Modals). The notification tray also keeps track of how many notifications the user has by having a numbered indicator right next to it.



@darwon has messaged you! Click here to view

@dashty wants to be your friend!

Figure 3: Notifications

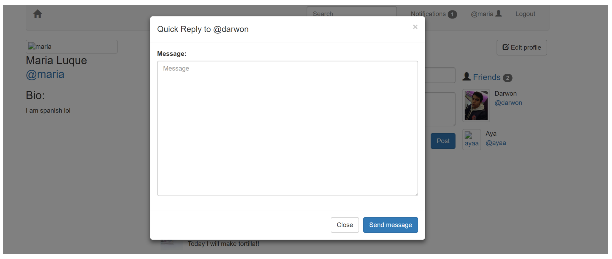


Figure 4: Quick reply to message notification

4.3 Flask Mail

Built in the web-app is a mailing system. It sends emails to users for certain situations. If a user registers up for an account, they receive an email from the web-app with instructions to confirm their email. If a user gets a friendship request, they receive an email notifying them for it. If any user forgot their password, they could go to the log-in page and click on forget password. It then prompts the user to type in their email so that they receive an email detailing further instructions on how to reset their password. If there was no mailing system, none of these important features would be possible.

4.4 Bcrypt

Bcrypt module was used in this web-app to hash passwords for protection. It is one of the most reliable modules for hashing. When a user is registering, Bcrypt takes what the user has input as password and hashes it so that it is secure. This means that only the user has access to its account. When the user tries to log-in and inputs its password, it uses what the user has input as a key to the hashed password in the database. If what the user input is the same as their password, then it is a match, and they have successfully logged-in.

4.5 Flask WTForms

Flask WTForms is a module that was used in this web-app for user inputted forms. It allows the creation of forms on the server-side. This is useful because developers can add custom validation for each field of the form and then render them in HTML. These forms were used for this web-app to make validation more efficient and dynamic. They were also used for re-usability for they save time when working on presentation.

4.6 Pagination

Since a social network platform will have a lot of data, it is important to lay out data in a concise and clean way. Pagination is built in all places that have to do with loading of data. If a user has one-hundred posts that they can view on their feed, without pagination, all one-hundred posts would show up on the user's feed in one long vertical page. What pagination does is take large amounts of data and presents it in a page format. What this means, is that a specific number of posts would show up on the user's feed instead of all one-hundred. If the user wanted to go through the posts, they would they scroll by page. This makes moving around pages faster for there is less loading of data. Pagination exists on the home page, the private messages page, the friends page , and the profile page of the web-app.

4.7 Flask Mongo Engine

Flask Mongo Engine is a module that is used in this web-app for all database related logic. MongoDB has been proven to be a very reliable database for web technologies. Flask Mongo Engine allows for all database functionality in the web-app. It creates the collections that the web-app uses. It inserts and fetches all the data for the web-app. It even has a built-in search engine that is used for search functionality in the web-app. What the user inputs in the search box of the navigation bar is taken and then indexed through the database. For example, if a user types in a first name looking for a user, they get any post or user that includes that phrase. The search engine works by indexing the documents in the database, hence the accuracy of the results.

5 Personal Evaluation

This coursework allowed me to expand my knowledge in Web Technologies, Flask, Python and other areas. When I first did the Workbook provided for the module I thought this coursework would be quite easy since the Workbook has information about all the base things I need to know to start the coursework. I kept reading online about all that Flask can do, the possibilities are infinite. After seeing all I could do I decided I would put all the effort it took to be the next Mark Zuckerberg, and create an amazing social network. I looked at many social networks to view all the features I could implement in my web-app. It was a challenging coursework, but it pushed me to make me do everything I wanted on it and now I can see that I am capable of doing anything I challenge myself to.I am very proud of my performance in this coursework. The part that I found the hardest was the design.I still struggle in the front-end side for it takes a long time and gets very tedious to get to where you want with it. Using Bootstrap helped a lot with the designing of the web-app which made me spend less time compared to previous course works on design due to this. I still feel like there is much more that web technologies can offer me than I expected.

6 References

Information about social networks came from here:

[1]
https://blog.miguelgrinberg.com/post/
the-flask-mega-tutorial-part-i-hello-world
[2]
https://www.smashingmagazine.com/2009/07/
social-network-design-examples-and-best-practices/

Information on all things Flask and Jinja2:

[3]
http://flask.pocoo.org/docs/1.0/
[4]
http://jinja.pocoo.org/docs/2.10/
[5]
http://docs.mongoengine.org/projects/
flask-mongoengine/en/latest/Flask Mongo Engine
[6]
https://pythonhosted.org/Flask-Mail/Flask Mail
[7]
https://flask-wtf.readthedocs.io/en/stable/Flask
WTForms

Information on all things design

[8]
https://www.w3schools.com/bootstrap4/bootstrap_
get_started.asp
[9]
https://getbootstrap.com/docs/4.1/
getting-started/introduction/

Where I got some images from

[10]
https://images.google.com/