Coursework 2 Report - Recipe Sharing Web Application

Christopher Johnson
40275286@live.napier.ac.uk
Edinburgh Napier University - Module Title (SET09103)

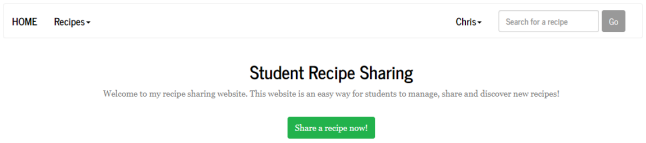
1 Introduction

The aim of this coursework was to demonstrate an understanding of server-side web development and the Python Flask micro-framework. This would be completed by completing a project in which I would design, and implement a web application on a topic of my choice.

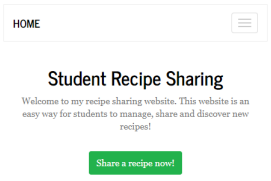
For my web application I chose to create a recipe sharing website. The website allows users to share recipes by creating an account and logging in. Users can view recipes shared by other users and favourite recipes they want to keep/have access to later.

User's have profiles where they're recipes are displayed and can visit other user's profile to see more recipes they have posted. When a user is visiting their own profile they can remove recipes and view/edit their favourites.

The main page of the website provides information about how to use the website as well as links to help get a new user get set up or for an existing user to quickly navigate to sharing a new recipe or visiting their account. The image below shows the main page, whilst logged in:

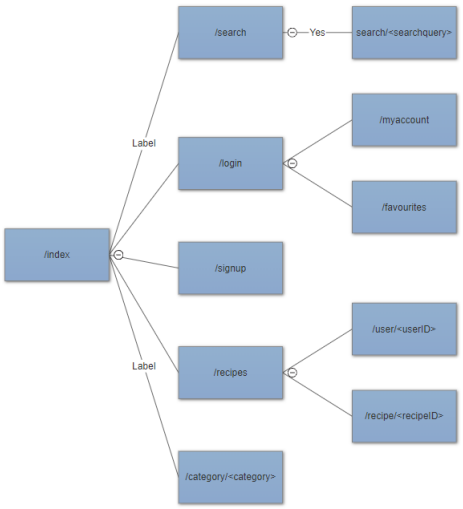


and on mobile:



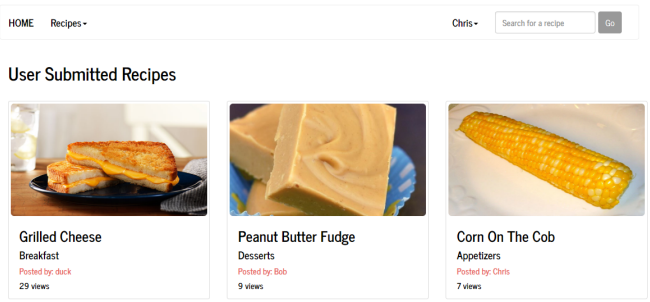
2 Design

I used the Python Flask framework, with use of tools such as PyMongo and bcrypt, to create the web application. I used multiple routes, shown in the diagram below, to structure the website with each route returning a html template.



In order for my design to be provide a satisfying and simple user experience my website needed to be easy to navigate and all options and information should be clear to the user. Upon visiting the site the user is greeted and given the option to create an account to make the most out of what my web application has to offer.

Visually, my website is simple and consistent. The navbar is consistent between each page as well as how recipes are displayed as collections or individually. Recipes are displayed as thumbnails in rows and columns which will change dimensions depending on the page size. I used bootstrap to style and build the website and then used a free bootswatch template to add a theme with custom fonts and a colour scheme.



Clicking on a recipe takes you the recipes page where you have the option to favourite the recipe if logged in. Recipes

also have a view count. Every time a recipe is viewed the view count is increased by one.

A user creates an account using a user name and password. This the only information the site requests from the user, I didn't see a need to collect user's names or email information (although this does mean if a user forgets their password they cannot gain access to their account). The user's password is encrypted and a new user item is inserted into the database. A user's database entry looks like:

```
{
"_id": {
    "$oid": "5bfae544d7a3ee5a0100fcf6"
},
"favourites": [],
"password":
"$2b$12$aXTpTSwCRFGgnyC4MI0AbeFEkZtMYCpxK39 ↩
  rkKn6X46MiPqYY5dpO",
"user": "Chris"
}
```

Encryption means that I am not storing a plain text version of the user's password. The web app only stores the user's user name and hashed password. If there were a data breach of any sort none of the site user's could lose any personal information or passwords.

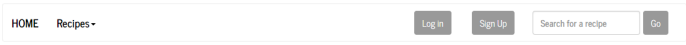
When this is completed a user can sign in and they have full access to all of the websites features. When a user signs in the password supplied is encrypted and compared against the encrypted password stored in the database. If they match the user logs in successfully.

To store user and recipe information my web application uses mongo collections. The database is hosted using mLab, a free mongo database hosting site. When a user creates an account, shares or favourites a recipe the database is updated with the new or changed data. A recipe entry looks like this:

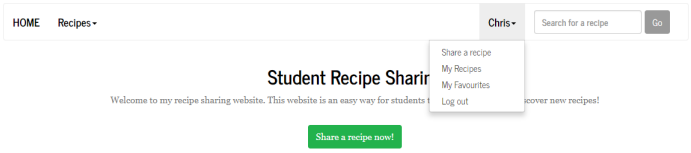
```
{
"_id": {
    "$oid": "5bf9ce5fd7a3ee00238fd343"
},
"category": "Breakfast",
"user": "Chris",
"img": "https://assets.kraftfoods.com/recipe_images/↩
  opendeploy/183819_640x428.jpg",
"views": 29,
"ingredients": "Bread\r\nCheese\r\nButter",
"title": "Grilled Cheese",
"method": "1. Toast bread in butter on pan\r\n\r\n2. Add ↩
  cheese\r\n\r\n3. Add slice of bread\r\n\r\n4. Flip\r\n\r\↩
  n5. Enjoy\r\n",
"description": "Classic grilled cheese"
}
```

Using "session[]" I am able to have my navigation bar change depending on if the user is logged in or not. When a user is visiting the website and not signed in the navbar displays sign up and log in buttons. When the user is logged in, these are no longer necessary and instead a drop down menu showing the user's name is displayed with quick links to the user's account, favourites and a log out button:

Logged in:

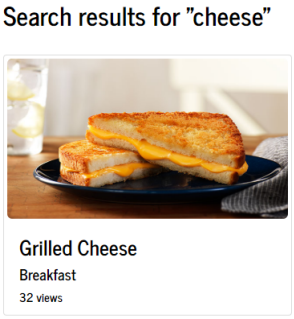


Logged out:



I have also used "session[]" to give users additional options when visiting their account which would otherwise be hidden from users, such as a button to delete a recipe. This is by getting the user name for the logged in user and checking to see if the page being visited is the user's profile. When a user logs out the navbar returns to it's original state.

The navigation bar also includes a search box, typing a word or phrase into the search box and hitting go will redirect the user to '/search/¡searchquery¿' displaying all the recipes matching the search query:



3 Enhancements

If I had more time to complete this project there are a few enhancements and features I would have liked to include. A voting system for each recipe would have been a good addition. Users could vote a recipe up or down and the recipe would have a combined score. This would have also provided another attribute to sort the recipes by with the top voted recipes being given more exposure on the website.

Another feature that would have been useful for users is a comment section on each recipe. If user's could make comments or ask questions directed towards the recipe poster it would provide communication between the viewer and the original poster.

4 Critical Evaluation

I think that my web application, overall, full-fills it's role to provide a platform for users to share cooking recipes with one another. Every implemented feature works, with no bugs discovered so far.

When building the website I used feedback from friends to ensure that using the website as a new user was intuitive and simple. Friends that I had test the site were able to post a recipe and view other user's recipes without problems with the only criticism being the inability to upload an image.

Currently the only way to upload an image for a recipe is by using an image url. User's with a photo can easily upload

an image to an image hosting site such as imgur to get a working url but it'd be much easier to be able to upload an image straight to the server and display it that way.

Overall I am pleased with how my application functions and looks. Although it does not have many advanced features, it is incredibly easy to use and there are no major problems I'm aware of.

5 Personal Evaluation

With this being the second coursework for this module, I had previous experience of Python and Flask going into it. I still really enjoy using these tools, far more than any I've used in the past (e.g. node).

One of the most important skills I have come away from this coursework with is the ability to create a website with functioning user accounts. This will be useful for upcoming projects.

I definitely could have added some more features if I'd had the time or managed my time better, I had to scrap a few features that weren't working correctly by the time it was time for submission. I think that overall, however, I am happy with how much was web technology knowledge has improved and I am for more confident using Python.

6 References

Additional Libraries Used (Licenses are provided in the git repository for my work):

- Flask : http://flask.pocoo.org/

- bootstrap : https://getbootstrap.com/

- bootswatch : https://bootswatch.com/3/journal/

- PyMongo : https://api.mongodb.com/python/current/

- bcrypt : https://pypi.org/project/bcrypt/