Coursework Report

Jostein Dyrseth
40223535@live.napier.ac.uk
Edinburgh Napier University - Adwanced Web Technologies (SET09103)

1 Introduction

Describing your web-app.

1.1 The Task

So the task in this coursework was to make any type of webapp, as long as we demonstrate our understanding of serverside development (POST and GET) as well as showing mastery of the Python micro-framework. In other words, the app we want to build is completely up to ourselfs. We should also research similar websites as our own.

The code should be well written and of good quality. It should be well structured, well commented, efficient and show proof of version control. Use of routing, static files, requests, responses, templates, sessions, logging, testing, CSS, JavaScript, multiple users and data storage.

2 Design

Explaining how you architected your web-app.

Designing the web-app was quite fun, challenging and time consuming. Creating it I used Python, Flask json, css, html, Jinja2 and some JavaScript. First off I had to come up with what to create. First I thought of a simple chatting app, but then I had a cool idea for a Recipe App. I looked online for similar ones and found a few: bbcgoodfood.com, allrecipes.co.uk and geniuskitchen.com (see references section). I liked the structure for the bbcgoodfood.com site, so I mostly looked at this to get some brief ideas about what I liked about it, and also what I didn't like that I wanted to improve.

2.1 Header

I wanted to go for a simple app, using material design. Material design is often supposed to be very minimalistic. Using this style often gives the user a minimal amount of elements to interact with, making the whole experience overall better. I also wanted to make the layout well outlined and structured.

The header in this app is fixed, meaning that when the user scrolls, the header follows. It contains a menu-button, so when the user clicks all of the categories will show up. The user can now select depending on what food they want to make. Also there is a header-title just next to the menu letting the user know where he is in the url-hierarchy. Followed by a search-bar. This will search through the current category the user is in.

To the right is a profile-icon (only if the user is logged in), a login and signup button. If the user has no profile yet, then they can create one by signing up. Then they can login whenever they want.

2.2 Plus Button And Recipe Modal

In the lower right corner there is also a fixed button with a plus on it. Clicking this button a modal will pop up with a file-input and form. The user has to find the file (only image files allowed) on the computer, and then fill out the form as well. The form consists of eight input-values to be able to make the recipe. It consists of the title (string), description

Figure 1:

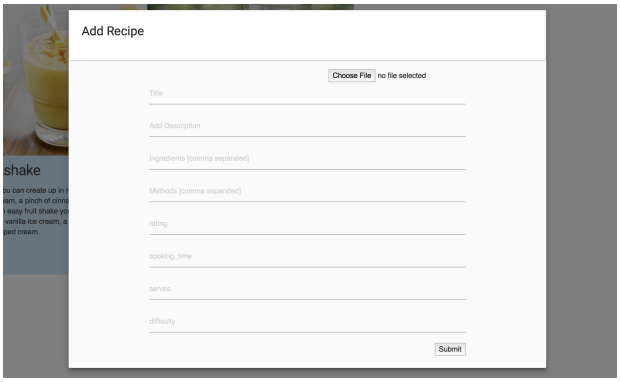

Figure 2:

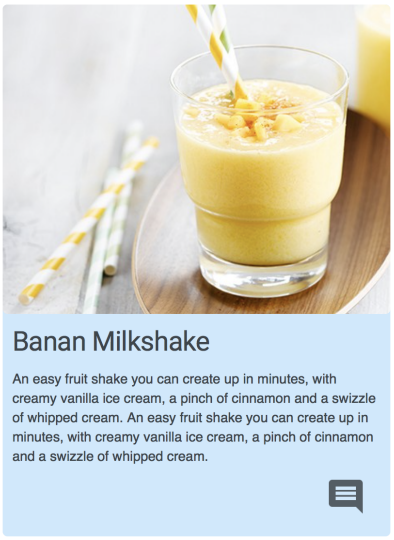

Figure 3:

(string), list of ingredients (strings), list of methods (strings), rating (int), cooking time (int), serves (int), difficulty (string) and a list of comments (strings).

```
ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'])
```

2.3 Main Recipes Content

The main content in the web app is to display all the recipes that have been made. The recipes will only show a thumbnail image, it's title, a breif description and a comment icon. When you click on it more information will show up. The comment icon will let the user make a comment on that particular recipe letting other users know what they thought of it.

When rendering the recipes, the python app will read the .json file and send the data as a dictionary to the browser, and then looping (using Jinja2) through each object. In the loop jenja2 will create a "card" for each recipe (see figure).

2.4 Signing Up

When the user is ready to sign up and clicks the 'sign up' button, they will be given a 'modal' at that same page, where the content will display a form of four input-values. The values are: user-name, full-name, email, password and retype password. These are the least values that any site normally gives the user while signing up. These were chosen because this is the most important ones when dealing with creating a user. Once filled in correctly and the user hits the submit-button, the form will be POST'ed to the server and python will request the form values and store them in the users unique user-name on the server in the .json file.

2.4.1 Password Hashing And Salting

Dealing with storing passwords in general are a big security-issue many companies and users fall into. Therefore the developers of the site must know what is going on at any given time, hence be absolute certain that no one can access anyone's credentials and passwords except themselves. Also it is important to never store the actual password on the server. This can leak somehow and end in disaster, especially when having an application with thousands and maybe millions of users.

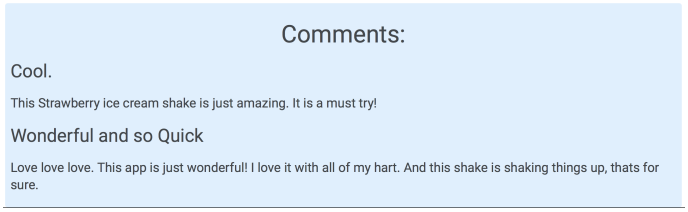

Figure 4:

In the application python will protect the users password by immediately hashing it and giving it salt as well. This is done using two modules from the 'werkzeug.security' library (see references). One to generate the hashed and salted password and another to check if the current login-input-password from the form is correct with the stored hash value in the json file. The actual password is never stored on the server making sure the user is 'safe and sound'!

2.5 Logging In

The user is now stored on the server, so whenever the user tries to log-in with the right credentials, the user will indeed be granted access hence his/her user profile will be activated and the user icon will also be added in the header. The user will now see a small round image of himself in the top right corner. The user should also be able to comment on any recipe and the comment should in theory be stored on the server so everyone can see it (public), but this is not implemented yet.

3 Adding A Recipe

In the beginning of the app, it will read all the recipes from the .json object and keep it temporarily in a dictionary. The pop-up modal as already seen has eight inputs plus the image-file. When the post is done by the user the python app will check if it is indeed a post as well as that there is a file in the requested file html-form. Now the app will write back to the .json file and store the data given to the recipe. It will keep the image in the static folder and just hold the image-filename in the .json file so that it knows where and what to display for any given recipe.

4 Enhancements

Describing the features that you would add or improve.

There are quite a few things I would indeed improve on this webapp.

4.1 Comments

This is a big one. I am very happy that I got to implement some quick and overall nice comment section. However, there are many things I would've enhanced here if I had more time. Obviously there is no add comment button yet, so the user has no way to post a comment. The current comments are just some comments that I have hardcoded into the json file. At least we can read them, but not write to the server (yet anyways - maybe in a later version). Also considering security here we need to check each comment so that people don't just write irrelevant stuff.

4.2 Categories

The categories does not really show any different data - yet anyways. This was not the point, the point was to begin from almost no recipes (just a few example-recipes). and let the users create the recipes rather. After deploying the website, more and more recipes would be added, and the different categories will show different content hopefully.

4.3 User Logged In

The user-icon only shows if the user is logged in. This is done as shown in the code underneath. Basically calling on the logged-in variable asking if the user is indeed logged in, then display the icon, if not then don't.

```
{% if user_logged_in == True %}

    <img id="user_icon" src="../static/user_icon.png" alt="user"
    height="35px" width="35px" style="float: right;
    margin: 15px;">

{% endif %}
```

4.4 Search

The search is not working at the given version, but would be very welcomed in the not so long future. It could be possible to do a simple linear search to return the exact matching search with the header for example, or just using a substring to make the search return results only for part of the header-values. Also it would be good if the search only happened within that category and not in the whole dataset.

5 Critical Evaluation

Explaining the features that you feel work well, or work poorly, and why.

5.1 Worked Well

5.1.1 Security

I feel that a lot of features worked quite well in this app. The login feature, only displaying the user icon if he/she is logged in. Also the password hashing and salting was done properly. The hashing and salting is happening on run-time and assuming we would deploy the app using the HTTPS protocol we can be certain that the password will (most likely) not be stolen by any third-party users or hackers. However the creation of a new user (only under the login-checking) is not correctly implemented, but it works so for now I will leave it, hence not being a problem yet of this scale.

5.1.2 Layout

The layout works well and the user is not distracted from the different sites as this has the same base in all occurring routes. The displaying of the recipes are shown on a grid view making the site neater and better placed.

5.2 Worked Not So Well

There are quite a few things that is not working that well. The user is not able to look at it's user-profile and user-account to change it's attributes like email, password, username and so on. The user icon shows up, but it would be great if it showed a picture of the user rather than a placeholder (but it works for now). Also the user is not able to make comments, but at least in this version they can read (the hardcoded) comments. I already know how to do a simple post so I did not unfortunately prioritize to implement that before the deadline, and lastly the cards' content at the main site must be of a given size so that the content does not display outside of the card itself. Also the rating is not supposed to be set by the user that creates the recipe, but rather by everyone else that reads the recipe. This could be implemented easily, but here too I had to prioritize.

There are a few other things as well that did not work that great, but those were the most important ones.

6 Personal Evaluation

Reflecting on what you learned, the challenges you faced, the methods you used to overcome challenges, and how you feel you performed.

I learned a lot in this coursework. Some of the things I got a grip of and something I know is important as well was the security part. I am happy that that section was handled very professionally and correctly, even though having quite a few errors thrown in my face. I also learned about how to upload files to the server, hence displaying it right afterwords. It was very important to get the image working making the site more pleasant and quicker for the user while browsing the different recipes. There is also a security part to uploading the picture, basically making sure the file is allowed indeed as an image file and nothing else. Also the name must be checked and this is done using secure-filename method (see reference).

7 References

(Optional) If you have used additional resources then these should be cited. Otherwise this section may be omitted.

Fileuploads: flask.pocoo.org/docs/0.12/patterns/fileuploads/, Hashing and Salting: flask.pocoo.org/snippets/54/, Valid Json: jsonlint.com, BBC Good Food: bbcgoodfood.com, All Recipes: allrecipes.co.uk and Genius Kitchen: geniuskitchen.com