

**DESIGNING A COOKBOOK WEBSITE WITH THE USE OF PYTHON FLASK ALONG WITH AN EXTERNAL API (PART 2)**

**Author** – Fahim Hoque

**Matriculation number** – 40535445

**Module number** – SET09103

1. **Introduction**

This report is a part 2 of the first report where I was speaking about the initial ideas and concept of creating a potential cookbook website with the use of an API, Flask and PyCharm; this report speaks about the differences between my initial plan as outlined in part #1 and my final implementation, as well as describing any features that I have added or improved - not mentioned from the first report, and reflecting upon the challenges that I have faced and achievements I have also made during this assignment. Lastly, there will be scrapped ideas that I will speak about that were from the first report but not implemented on the final design, and new features implemented instead that is again, not mentioned on the first initial report.

1. **URL and GitHub Repository**

There is no deployment used (VM), but the files have been submitted to Git Repo ([ACG6th/SET09103: Advanced Web Tech Module](https://github.com/ACG6th/SET09103/tree/main)) with .env and venv files not shown due to sensitive data. The host from the application would be on ‘http://127.0.0.1:5000’.

1. **Differences between Initial Plan and Final Implementation**

To start off, the final design has the cookbook page with an API implemented called ‘Spoonacular API’, as well as the login and registration pages in which I did promise to implement those from the first report. When you launch the website, you will be greeted with other recipes you can investigate, as well as searching any food, such as cake, pancakes, strawberry, banana, and so on. The login page should show up first and if you don’t have an account, there is a link below it to create a new one, and when you create your new account with your username and your password, your information gets saved to a database called ‘users.db’ found in the instance folder. This shows the saved information from the user, with its username shown and its hashed password, so anyone who is viewing the database cannot see the password. I had to download SQLite DB Browser in order to easily look through the database. From the first report, I mentioned about an email implemented but was scrapped on the final design. I also did not mention about hashed passwords from the first report as originally, I just wanted the login and registration pages to work, and I could care less about passwords being shown. And lastly, I mentioned that it would take time to implement a web-app meeting application with the use of Flask, SQLAlchemy and Flask Login from the first report, in which, ironically, I have used for both the cookbook design and the login pages. I also mentioned about deploying the project with my VM from the first report, but I did not utilize it due to its constant problems with the Napier VM; the module leader has also mentioned that you will not get penalised for not using the VM, so I didn’t bother, as long as I had my application working well with PyCharm and Flask. One new thing I did not mention from the first report was having a secluded API Key and secret key as I found out that you are not allowed to submit it to your GitHub Repository due to sensitive data. For this case, I had to create a .env file to put sensitive info (e.g. secret key and API Key) and installing and importing ‘Python-dotenv’ to load from the .env file. Lastly, I created another .gitignore and mentioned both the .env and venv files to not be shown when submitted to GitHub.

1. **Unused Features to Improve or Enhance the Project**

There are some features that I wish I added had I was given time, such as having the website be more colourful and vibrant, and not just a plain white background; the website should’ve been more detailed compared to the ‘Delici-Awesome’ website that I created from Web Technologies, mentioned from the first report. I also thought I could have a UI template as part of a placeholder for the background to make the website stand out. Another thing that was mentioned from the first report but was not implemented was to save a recipe that a user found and have that user access to a folder or a tab of saved recipes. This could’ve saved the user’s time of searching the recipe again, but I did not implement this. Another annoying thing about the login page is that sometimes, once you launch the application, it shows the cookbook recipe page instead of the login. The login exists if you type ‘\login’ at the end of the URL of the host, but I am unsure to fix this as once you launch the application for the first time, the login *does* show up first And lastly, I could have styled the application better with the login and registration pages from the ‘base.html’, as whilst it was extended, it ran into problems, so I left it out but kept the ‘base.html’ file for references.

1. **Reflection and Achievements**

There were difficulties whilst attempting to create the login/register pages and the cookbook pages, such as running into HTTP errors, examples are Error 405 (Methods not allowed) and Internal Server Error whilst attempting to create the login page. It got to the point where since I had a working cookbook website with the API, that I thought about scrapping the login and registration pages altogether. There are few things I was pleased about such as the API, and how it functions well on the cookbook, along with how chuffed I am about it. Although the website may not look pretty compared to the first cookbook website from Web Technologies, I am happy that I created a functioning website. Second thing was an API KEY, as you are not supposed to upload it on GitHub due to its sensitive data, so I had a bit of trouble with Git and GitHub, to the point where I had to delete the repo to backtrack.