Task 6: Create a demo app in Django

Introduction

Previously in Task 5, we saw what Flask is. Similar to Flask, Django is also a Python web framework that allows you to create web pages. Django has more built-in features than Flask. For example, Django uses and creates a SQLite database to store data such as the Django settings, any data about users that may be created like admin users, etc. It also provides a quick and easy administration page by default.

Description of the Task

Similar to task 5, my task is to create a demo app using Flask. My definition of a demo app remains the same, to create an app that is useful, i.e. not just "Hello World", but is not too complex and is not production ready. It should demonstrate the basics of the technology being used.

I shall be developing the same app I developed in task 5, but now using Django, which is, a web page similar to a login page, that users can input their name and address into.

I shall be demonstrating the use of HTTP requests that can be used within Django, such as GET and POST.

We shall be storing the data within the SQLite database that gets created with the Django app, instead of using a plain JSON file.

And finally, the web pages will be simple web pages in HTML with some styling to render a User Interface for all these operations.

Requirements

```
Django
```

Instructions:

The only prerequisites are to install the requirements, and to make sure you are working from the directory that the app is located in.

As the app has already been built, there is no need to run the typical Django commands of running migrations, etc.

As such, we will just run the application by using the command `python manage.py runserver`

Next, we will open up a web browser, and navigate to the address that will be printed in the terminal, the address will be `http://localhost:8000` or `http://127.0.0.1:8000`, both will work, as localhost is an alias for 127.0.0.1.

Enter the details that the app requires, and press the submit button that will initiate the `POST` request.

To view the data that has been input, press the button `View Data`, which will automatically refer you to the `/data/` endpoint, which will `GET` the data and display it.

To make use of the admin page functionality, you need to run the command `python manage.py createsuperuser` and create a username, password and input an email for the admin user.

To view the admin page that has been created, navigate to `http://localhost:8000/admin`, and log in to the admin account using the credentials you created with the above command.

Work done by:

Aseem Aniket Athale

athaleaseem@gmail.com