Assignment 3

Part 1: Small Web Showroom

A local fashion store asked you to create a website where they can showcase their products.

1. Create a first route that is going to be your homepage. Use HTML to add some content that fills the page (feel free to copy text and images from your favourite online store). Add an *externally linked* CSS file and add your own styling and arrangement. You are free to choose as you like, but make sure that you use at least 20 different CSS statements with at least 10 different properties. The properties have to be substantially different e.g. margin-left and margin-top will not count as distinct.

2. Create a second version of the homepage, this time using Bootstrap. Include:
    a. A carousel (e.g. showcasing different store locations)
    b. A collapsing element (e.g. with some information about the company)
    c. A navigation bar with the logo of the company (choose any logo you like) and some placeholders for other pages that you will link to later
    d. A fourth element of your choice from the Bootstrap components page (except cards)

    Set this page as your new homepage and give your page from task 1 a different route like */old-homepage*

3. The store manager isn’t quite sure yet which products she would like to use. She would like to have a page where she can upload the data for the products when she’s ready. (In a real setting, you would hide this behind a login wall but we are omitting this for now)

    Create a route called */upload-product*. On that page create a form where the user can (1) enter the name of the product (2) a short description of the product (3) the price (just a number, don’t worry about currencies here) (4) an image of the product. Store the images in the static folder of your Flask app.

    As a replacement for our database we will use a CSV file so that data is preserved even when the app is restarted. Make sure that this CSV file is loaded as a global object when your app starts. When a new file is uploaded save it in this CSV file (add a new row and directly overwrite the local CSV file). For the images: save the location of the image in your flask application in the CSV file. You should use pandas to work with the data, but you do not have to save the csv file to your static folder, as it won' t be accessed directly by your HTML content.

When you' re done, upload a few example products (again using images and texts from your preferred online store if this saves time for you)

Notes:

a. A first version of the what the CSV should look like is contained in the assignment folder
b. If your using hydrogen to test code snippets of your flask app from the main flask file itself, then make sure that in the settings "directory to start kernel in" is set to "current directory of the file", so that you won' t have to change the file paths when you run your app.
c. Look at this Stackoverflow post for overwriting the dataframe as a global variable inside the route. This is isn' t the recommended way of doing this, but we will cover databases in the break
d. Hint: Check the contents of your CSV from time to time to make sure that everything saved correctly

4. Create a new route called */showroom*. Add a card view of the products uploaded from the */upload-product* route. The card should contain all the information collected in the form and should also display the image as well as a button, which for now does not lead anywhere.

Notes:

a. Use Jinja to dynamically iterate over the dataframe in the HTML file and render the product cards dynamically
b. Have a look at this Stackoverflow post for how to iterate over a dataframe
c. Use the Bootstrap card template. You might have to make small adjustments depending on which template you choose.

5. Add the homepage, */upload-product* and */showroom* to your navigation bar and make sure that the navigation bar is displayed on *all* pages

6. Create a new route called */product/<product_name>* which links to a single page view of the respective product. You can work with the assumption here that each product name entered to the database via the form is unique and thus you can index for the dataframe row of that item by methods known from previous lectures. The HTML file of the route should be simple but also styled with bootstrap (and also include the navigation bar). Also replace the links for the buttons in your */showroom* route.

Part 2: Using a foreign HTML Template with Flask

In the assignment folder you will find a template called “Split” that was downloaded from here.

Make this template run in Flask and reconfigure it to run your “homescreen” route from last week’s assignment. The name and Weather should be displayed on the right while the city image from Unsplash should be displayed on the left. Feel free to delete any content you don’t need from the template. If you like you can also use the example solution for last week’s assignment.