WEB322 Assignment 4

Assessment Weight:

9% of your final course Grade

Objective:

Build upon the foundation established in Assignment 3 by providing new routes / views to support modern CSS techniques, html and publishing to a Cloud.

NOTE: You may work on Assignment 3 if you wish (not necessary though), to apply the requirements for this assignment.

Specification:

Part 1: Creating a "theme.css" file within a "public" folder (public folder is located in the same level as server.js file).

Step 1: Creating a "theme.css" file and linking to it from our HTML files.

- Within the root of your application, create a new folder called "public"
- Within the "public" folder, create a "css" folder (this is where we will place our CSS file(s)
- Finally, within the "css" folder, create a "theme.css" file.
- Now that we actually have a "theme.css" file (albeit an empty one), we must include it in all of our "views" (.html files):

  This will involve using the appropriate <link> element in the <head> of your files .
- With this complete, we must use the built in express "static" middleware within our server.js file to identify our newly created "public" folder as a source for static files.

Step 2: Updating "theme.css" to provide a unique look / feel for your application

Now that we have a "theme.css" file and its correctly linked in our html files, we can start to think about personalizing our web app by adding some CSS. There are plenty of resources online to help you pick colours and find (as well as generate) interesting styles to apply to selected elements. Some quality resources to get you started include:

- This "Colour Wheel", used to pick complimentary colours and get their "hex" values: https://www.canva.com/colors/color-wheel/
- A "Box Shadow" Generator, used to provide the complete CSS for adding a "box shadow" to an element: https://www.cssmatic.com/box-shadow
- Sample CSS for generating a "full page gradient" (**Note:** for this example to work for us, we must change the selector from "html" to "body, html" and instead of "#red" and "#blue", simply use "red" and "blue" – or whatever other colours you like): https://coderwall.com/p/ape0jg/full-page-gradient-background

Additionally, to ensure that your gradient covers the entire page, regardless of how far you scroll, you must add "**overflow: auto;"** as well.

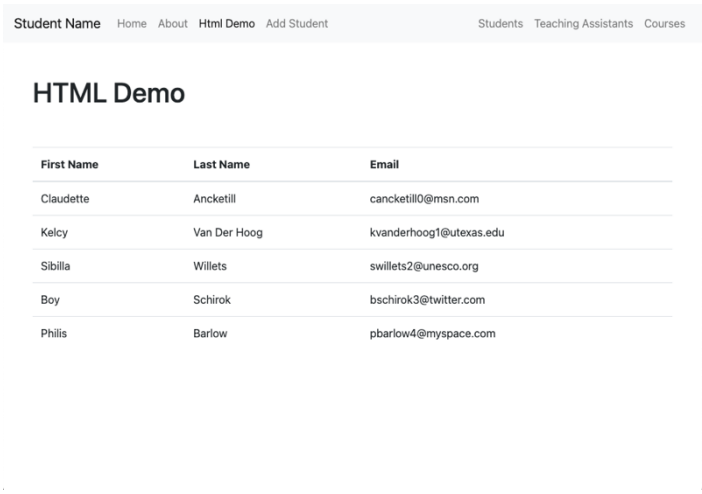
- Google Fonts, used to generate an @import statement to add to our "theme.css" file that will enable us to use 3rd party "web fonts" (**Note:** Add the @import statement at the top of your theme.css file and add the "font-family" style to whatever element you wish to style the font): https://fonts.google.com/

Regarding our own app, we must style to have an impact on the layout display (see the table in Step 3: Elements and Styles as examples). **Note:** Both step 2 and step 3 are for **reference only** - Please feel free to create/add **styles and html** if you have a specific vision for your app that goes beyond the requirements.

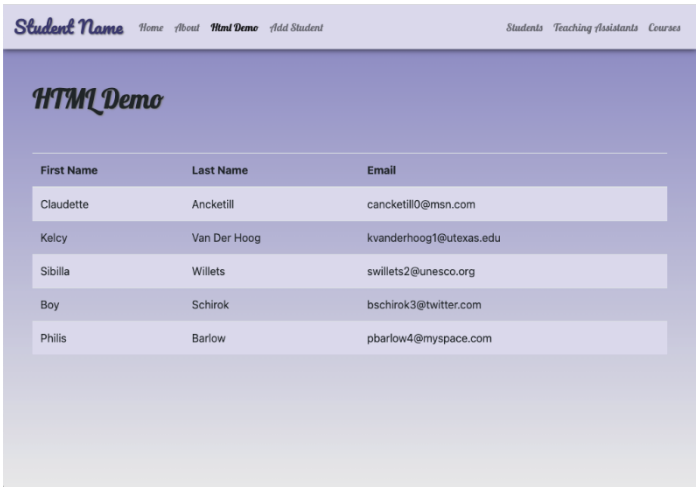
Step 3: Elements and Styles

The below table can be used as a guide to help you style your application. To get a sense of what your application could look like after applying you own styles to the following elements, please see the following "before" and "after" images (**NOTE**: Consider adding class="table" to your <table> tags – this was done in the sample):

Before



After



Element	**Style Suggestions**
Navigation Bar - Element with class "bg-light"	- Background colour or gradient - Box shadow (outline or inset)
Navigation "Link", ie "Home", "About", etc. etc. - Anchor element with class "nav-link" - Anchor element with class "nav-link" when "hovered"	- Cool font from "Google Fonts" (do not forget the @import statement) - New Text Colour / hover colour, etc.

App Background - Both body and html elements	- Full page gradient - Solid background colour - Image (consider using "cover" - https://www.w3schools.com/cssref/css3_pr_background-size.asp)
Buttons - Element with class "btn-primary" - Element with class "btn-primary" when "hovered"	- Solid Background Colour - Gradient Background - Box shadow (outline or inset) - Different colour for hover

Table cells inside "odd" numbered table rows - "td" elements, inside all (odd) "tr" elements (see :nth-child(odd) pseudo-class), inside a "table" element	- Solid Background Colour - Gradient Background
All "Heading" elements - All elements from "h1" to "h6"	- Cool font from "Google Fonts" (do not forget the @import statement) - New Text Colour - Text shadow
All "Anchor" elements (HTML Links) - "a" elements - "a" elements when hovered	- New Text Colour

If you're enjoying tinkering with the look and feel, please feel free to style additional elements or update the existing html structure of the layouts. Again, the above steps are reference only.

Part 2 : Adding new View, Routes to server.js

You are allowed to work on the existing html files of assignment 3. You are also required to create a **storefront.html** page in the views directory. The storefront page will display images of at least 10 similar products with related information such as title, description, price etc. Be aware of copyright for the images you use. Refrain from using abusive text or images. You may refer to amazon, ebay, Walmart for design guidance. You will be graded based on creativity. You may make use of online tools such as those available on css3gen.com.

This storefront.html page must have 2 applications – (1) use of Bootstrap and (2) the page has to be mobile responsive. You are allowed to include/embed css in the html page. The theme.css should have some common applications on this storefronmt.html.

If you did not complete assignment 3, then you may create/add other html pages of your choice to make the website a multipage web application. Remember the pages should have consistent look for user friendliness such as navigation, header, footer etc. – these elements should be consistently designed in all pages. The common styles need to be defined in theme.css.

Adding "Get" routes in server.js

- Adding body-parser: Add the express.urlencoded({ extended: true }) middleware (using app.use())
- Inside your server.js file, add routes for any html file(s) created such as the route "/storefront", which will simply send the newly created "storefront.html" etc.
- You must have a GET/ route defined for the home.html or to start the web application. All pages must have consistent navigation option to all other pages/routes.

Part 3: Pushing to Heroku

Once you are satisfied with your application, deploy it to Heroku (This will require having an account on Heroku as well as having the Heroku CLI Installed):

- Issue the following command from the integrated terminal: **git init** - this will initialize a local git repository in your working folder. You will notice that an icon in the left bar now has a blue icon showing a number. This represents all of the files that must be added to our local git repo. Click the button and type “first commit” for the message in the “Message” box. Once this is done, click the checkmark above the message box to commit your changes.
- **NOTE:** If, at this point, you receive the error: “Git: Failed to execute git”, try executing the following commands in the integrated terminal:
    - git config --global user.email "you@example.com"
    - git config --global user.name "Your Name"
- Once this is complete, attempt to commit your code again.
- Now that our local git repo is ready to go, we have to create an application within Heroku to send our code to. This is done by issuing the following commands in the integrated terminal:
- **heroku login** - this command will prompt you to enter your Email and Password. Once you have done this successfully (by providing the correct email and password for your Heroku account) you will see a message “Logged in as …” where … is your email address.
- **heroku create** - this command will create our new app within Heroku! The name that it gives our app is random (you can change it later). You will know the name it has given your app by looking at the text next to creating app… done, [*app name here*]. You will also see a url in the form **https://[*app name here*].herokuapp.com**
- We’re getting close, but not quite done yet - we need to issue one more command: **git push heroku master** - this command pushes the content of our local git repo to our new app on Heroku

- **IMPORTANT NOTE:** Since we are using an "**unverified" free** account on Heroku, we are limited to only **5 apps**, so if you have been experimenting on Heroku and have created 5 apps already, you must delete one (or verify your account with a credit card). To delete an app, login to the Heroku website, click on your app and then click the **Delete app…** button under "**Settings**".

Assignment Submission:

- Add the following declaration at the top of your **server.js** file:

```
/*********************************************************************************
* WEB322 – Assignment 04
* I declare that this assignment is my own work in accordance with Seneca  Academic Policy.  No part
* of this assignment has been copied manually or electronically from any other source
* (including 3rd party web sites) or distributed to other students.
*
* Name: _____ Student ID: _____ Date: _____
*
* Online (Heroku) Link: _____
*
*********************************************************************************/
```

- Compress (.zip) your app folder and submit the .zip file to My.Seneca under
  Assignments -> **Assignment** 4
  When you upload the zipped assignment, paste your Heroku link in the comment box. Failure to do will result in zero marks for this assignment.

Important Note:

- **NO LATE SUBMISSIONS** for assignments. Late assignment submissions will not be accepted and will receive a **grade of zero (0)**.
- After the end (11:30PM) of the due date, the assignment submission link on My.Seneca will no longer be available.
- Submitted assignments must run locally, ie: start up errors causing the assignment/app to fail on startup will result in a **grade of zero (0)** for the assignment.