WEB322 Assignment 3

Submission Deadline:

Friday, June 14th, 2024 @ 11:59 PM

Assessment Weight:

9% of your final course Grade

Objective:

Build upon Assignment 2 by adding a custom landing page with links to various sets, as well as an "about" page and custom 404 error page. Additionally, we will be updating our server.js file to support more dynamic routes, status codes and static content (css). Finally, we will publish the solution using Vercel.

If you require a *clean version* of Assignment 2 to begin this assignment, please email your professor.

NOTE: Please refer the sample: https://rg-web322.vercel.app/ when creating your solution. The UI does not have to match exactly, but this will help you determine which elements / syntax should be on each page.

Part 1: Installing / Configuring Tailwind CSS

For this assignment, we will be adding multiple pages, including a landing page with links to some of your Lego sets. To make these appealing to the end user, we will be leveraging our knowledge of Tailwind CSS. With your Assignment 2 folder open in Visual Studio Code, follow the follow the steps identified in Tailwind CSS & daisyUI to set up Tailwind CSS, ie:

- Installing the "tailwindcss" command as a "devDependency"
- Installing "@tailwindcss/typography" and "daisyui"
- Initializing tailwindcss
- Creating a "tailwind.css" file in /public/css
- Editing tailwind.config.js to ensure your .html files are watched during the build and "daisyui" / "@tailwindcss/typography" are added as plugins. Also, you may add a "theme" - the sample uses "fantasy", but you're free to use whatever you like.
- Adding a "tw:build" script to your package.json
- and finally building a main.css file.

Part 2: Adding .html Files

Now that we have our primary "main" css file in place, we can focus on creating the "views" for our application. At the moment, this is **home.html** ("/"), **about.html** ("/about") and **404.html** (no matching route). These must be created according to the following specifications:

NOTE: Before you begin, do not forget to mark the "public" folder as "static", ie: **app.use(express.static('public'));** in your server.js file

File: **views/home.html**

- Must reference "/css/main.css" (ie: compiled tailwindCSS)
- Must have a <title> property stating something like "Lego Collection"
- Include a **responsive** navbar with the following items:
    - "Lego Collection" (or something similar) as the large text (left) which links to "/"
    - Link to "/about" with text "About"
    - Dropdown with Label / Summary "Theme"
        - The items in this dropdown should be links to **3 themes** that are available in your dataset in the form: **someTheme**
- Have an element with class "container mx-auto", containing:
    - A "hero" daisyUI component featuring some text inviting users to explore the Lego collection and a link styled as a "btn" that links to "/lego/sets"
    - A responsive grid system containing **3 columns**, each containing a "card" component featuring one of the items from your lego set (hard-coded in the html). Each card must contain:
        - An image from the lego set's "img_url"
        - The set "name"
        - A small blurb about the set (**NOTE:** You can find some information about each set on the "Rebrickable" site, ie: https://rebrickable.com/sets/001-1 will give you information about set with set_num 001-1
        - A link styled as a "btn" that links to "/lego/sets/**set_num**" where **set_num** is the number of the set in the card, ie "/lego/sets/001-1" for set 001-1

File: **views/about.html**

This file should follow the same layout as **views/home.html**, ie: reference main.css, have a <title> property and identical navbar. However, the navbar must have the text "About" highlighted by using the "active" class, ie:

About

Additionally, this view should feature:

- an element with class "container mx-auto", containing:

    - A "hero" daisyUI component containing the header "About" with some additional text, ie "All about me", etc.
    - A responsive grid system containing **2 columns**:
        - The left column should show an image that you like / represents you.
        - The right column should be a short blurb about yourself (hobbies, courses you're taking, etc).

File: **views/404.html**

Once again, this file should follow the same layout as **views/home.html**, ie: reference main.css, have a <title> property and identical navbar.

Additionally, this view should feature some kind of 404 message / image for the user. The sample uses a "hero" daisyUI component

Part 3: Updating server.js

To support dynamic routes, status codes and our custom 404 page, we must make the following changes to our server.js code from Assignment 2:

- Update the "/" route to respond with the "/views/home.html" file
- Add an "/about" route that responds with "/views/about.html" file
- Update the "/lego/sets" route such that:
    - If there is a "theme" query parameter present, respond with Lego data for that theme, ie "/lego/sets?theme=technic" will respond with all sets in your collection with the "technic" theme
    - If there is not a "theme" query parameter present, respond with all of the unfiltered Lego data
    - If any errors occurred, return the error message with the "404" status code
- Update the "/lego/sets/num-demo" route such that:
    - Instead of returning the hard-coded Lego set from assignment 2, it should instead return the Lego set with a "set_num" value that matches the value after "/lego/sets/". For example, "/lego/sets/**028-1**" should return the Lego set with set_num **028-1**, etc.
    - If any errors occurred, return the error message with the "404" status code
- Delete the "/lego/sets/theme-demo" route – we no longer need this one, since "/lego/sets" now supports the "theme" query parameter
- Add support for a custom "404 error". However, instead of returning text, respond with the 404 status code and the "/views/404.html" file

Part 4: Deploying your Site

Finally, once you have tested your site locally and are happy with it, it's time to publish it online.

Check the "Vercel Guide" for more information.

Assignment Submission:

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

```
/*********************************************************************************
* WEB322 – Assignment 03
*
* I declare that this assignment is my own work in accordance with Seneca's
* Academic Integrity Policy:
*
* https://www.senecacollege.ca/about/policies/academic-integrity-policy.html
*
* Name: _____ Student ID: _____ Date: _____
*
* Published URL: _____
*
********************************************************************************/
```

- Compress (.zip) your web322-app.
- Create a screen recording of your running app. You are also required to explain your code. Use this screen recording to present the work done in the assignment. During the screen recording, you must verbally narrate/explain what you are doing while you are doing it. **Max time: 10 minutes**.

- **Submit the following:**
  - **Project zip file**
  - **Screen Recording**
  - **Published URL (put this in the submission comments)**

  Note:
  - **All items listed above must be submitted for your submission to be considered complete/valid.**
  - **Incomplete/Invalid submissions will not be accepted and will receive a grade of zero (0).**
  - **Project zip file and screen recording must be submitted separately – do not zip them together.**

- **PS**: If the recording file size is too large to be attached in submission: Upload your screen recording to **Microsoft OneDrive** and ensure that the link is set to: "Anyone with the link can view". Paste a link to the recording in the submission comments.

Important Note:

- **NO LATE SUBMISSIONS** for assignments. Late assignment submissions will not be accepted and will receive a **grade of zero (0)**.

- 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.