The University of Mississippi
CSci 354 Web Programming
Project 3

Due Date: Friday September 30 by 11:59pm

For this stage of the course project, you will use HTML and CSS to create an initial static version of your web app. You can think of this as design mock-up—it will include all of the appearance and content decisions, but no real interactivity (you will add that in the next stage). It will provide a "snapshot" of your app as it is being used, but which shows off the design and styling of the project.

Think about the design of your page **before** you start coding it! I recommend you using a pen-and-paper or PowerPoint to draft a mock-up of your page (or you could use a design tool such as Figma but there is extra work learning that software and is only free for the first 3 projects). This will let you decide what it should look like (colors, fonts, layout, etc..), and give you a nice reference to work against when you actually start coding. This design work can be done even before we go over the technical skills needed to produce the page (i.e., before reading the chapters).

We are happy to look at and give feedback on concept, design, or implementation at any point of any stage of your project!

Objectives

By completing this assignment you will practice and master the following skills:

- Creating new web pages from scratch
- Writing semantically rich and accessible HTML
- Using CSS to gives pages complex formatting and layouts
- Harnessing media queries and CSS frameworks to produce responsive web pages
- Exploring advanced CSS and formatting features

Site Requirements

While the exact subject, content, and design of your site is up to you, it will need to meet a few specific requirements to make sure you demonstrate your web development skills:

Content

Your website will need to include an **index.html** file to act as the "main page" of your application. Think of this as the "home page" of the app—the page that the user first sees and interacts with. If your app will have multiple different pages or views, you will want to focus on the "primary" one that the user will spend the most time interacting with.

- Your app will eventually support "user login" features, so design the page as if the user were logged in.

- Your app can include multiple .html pages if you wish—indeed, you will need to create at least two additional page to provide "mock interactivity" (see below).
  - Any .html pages you have should share a unified "theme" and structure—the same CSS rules must be applied to both. Later in the course we will discuss ways to "template" HTML to avoid duplicating shared elements such as navigation (or you can explore a templating engine such as Handlebars on your own).
  - Make sure each HTML page has appropriate metadata, the main page should include the *author* (you), a *description*, and a set of *keywords*.

- Additionally, your pages should include a favicon (an icon for the page tab); you can find lots of free icons at sites such as Icon Finder or The Noun Project or you could even make your own at favicon.cc or favicon.io

- Think about what content your page will be showing, and what buttons or forms it will have to enable the user to interact with it! Your web app needs to have sufficient content to make its layout and styling necessary (and to ensure it is complex enough to fulfill the project requirements)! A good minimum is about 2 to 3 *printed pages* worth of content.
  - Include enough graphics and media to make the content exciting and engaging! Your page should include both text content and at least 3 notable images or other media content (videos, etc..). If you use music, make sure it isn't set to auto play.
  - The text and images on your page should be actual (sample) content, not just *lorem ipsum* placeholder text. We should be able to look at the page and believe that it is a real snapshot of your app! Text and images can be borrowed from other sources, though be sure to include appropriate attribution (citations see HTML element <cite>) *Make sure you have the rights to any images you use*! unsplash.com, stocksnap.io, morguefile.com, and pixabay.com are all good places to start looking for free images.
  - Similarly, if your app includes a data table, be sure to give it realistic values!
  - If your app will include interactive graphics (such as a chart or a map), you should instead include a static placeholder image that is a sample of what that chart/map may be showing at a particular moment in time. You could create such a placeholder using your favorite graphics editor.

- Your page content can be structured however you wish, but should include a header (opening/top container element) with appropriate content (e.g., the title of the page) and a footer (closing/bottom container element) with appropriate content (e.g., copyright and contact info).

Mocking Interactivity

While your website is primarily a "snapshot" of the app's content and design, you will want to show off how that content might change as the user interacts

with the app (the two steps). Thus your site will include other .html pages that will show what the app will look like after the user performs some interaction (e.g., each "step" from your proposed functionality walkthrough).

To do this, you should first identify the *primary and secondary points of interaction* of your pages. Think about the main piece of functionality in your app, and the most significant interaction that the user makes as part of that functionality. For example, if you were developing an app that has the user select a choice, the main "feature" might be the ability to click a button. That button would be the primary point of interaction for your app. Similarly, for the second point of interaction.

Take these points of interaction, and implement them as a *hyperlink* to the next .html page (name the pages something appropriate). The next pages should be a version of the index.html file, changed as if the user had interacted with the page (e.g., with the new post included). Thus by following that link, the user will be able to get a sense for how the page may be interacted with.

- If the points of interaction are a <button>, you should implement them as an <a> element that is styled to look like a button! If the points of interaction are a <form> the user submits, you can have the form's action attribute refer to the next .html file. Make sure to include an appropriate ARIA role as well!

Style and Structure

Your project will need to use CSS to specify the design and layout of your app's content. The design of the app is up to you: I encourage you to get creative with how information might be presented.

> * It is **not acceptable** to build your page using an pre-defined template (e.g., in which you just fill in the content) or similar "pre-built" work. The site you submit should be entirely implemented by you. You are welcome to get *design* ideas from other sources or look up how to achieve specific effects, but the entire page needs to be coded by you, not by someone else. If you include more than 1 line of code from another source, you need to include an appropriate citation.

It is acceptable to use a CSS framework such as Bootstrap, Foundation, or Materialize to help structure and style your page, though you must use the most recent version if you do so. Moreover, even if you use a framework to help style your app, you will need to include some additional CSS rules of your own. The expectation is that you will write CSS code yourself to demonstrate and apply those skills.

- I expect you to have a few dozen different CSS rules for this project; you will need to include that much complexity in order to receive full credit on this project.

- You should use CSS to style the page with appropriate colors, typography (fonts), layout, spacing (margin/padding), etc.. You can use a tool such as

Coolors.co, Adobe Color CC, or the Material Design Color Tool to determine a color palette.

- You are welcome and encouraged to go all-out on the artistry—just make sure that the page's content remains navigable, readable, and usable!

- But in order to demonstrate your development skills, your app should not entirely follow the default "flow" of content. You must also include styling that provides substantial "non-standard" layout structures, such as columns (e.g., via flexbox) or grids. A single positioned or floated element will be worth partial credit; a multi-column layout is sufficient for full credit (and see *Responsiveness* below for related requirements). Note that simply using a flexbox to make elements horizontally "inline" will not be sufficient.

Accessibility

As with everything, your web app should be **accessible** to all users, including those utilizing screen readers. You can ensure your content is accessible by semantically using HTML, applying appropriate headings and sections to make it *navigable*, and including text equivalents for the visual content (images) to make it *perceivable*.

- You should utilize captions and ARIA attributes when appropriate, particularly if you utilize components of a CSS framework.

- It needs to be possible to navigate your page with a keyboard. If your layout is *really* complex, you might even need to specify a tab order (tabindex).

- Run your web page through an accessibility checker such as WAVE or AChecker to confirm it is accessible. Or better yet, open up a screen reader such as Voice Over and try listening to your work!

Responsive Design

Your web app must be **responsive** and usable on both large desktop displays and small mobile devices. This means using adaptive layouts and multiple media sizes (e.g., different resolution images for different devices).

- You must follow “mobile-first design principles” from the textbook when styling your page. In fact, I encourage you to take a "mobile-first" approach to both your design *and* to your implementation: make it work on mobile, and then expand to work on desktops.

  Remember to specify the viewport meta so that your page will actual be responsive!

- Using a CSS Reset file such as normalize.css or Bootstrap Reboot is a good idea for making sure your site looks the same across browsers.

* Note that just using a CSS framework such as Bootstrap is not sufficient for this requirement; your CSS **must** include some media queries to receive full credit.

Your site needs to be styled appropriately for each category of screen size (small mobile screens, medium screens, large desktop screens). This includes aspects such as like layout, white space, font sizing, media presence, and more. For example, your site could switch to multiple columns as the screen gets large, size the size of cards, add additional images or content, etc. Note that using a responsive column-based layout (such as a Bootstrap grid) is one of the most effective ways to achieving this.

- Different sized screens have different affordances. Your site almost certainly needs to look different on different screens to accommodate this.
  Just having content "wrap" on smaller screens—even if that wrapping is performed with a flexbox—is not sufficient. We want to see that you've used responsive CSS to purposefully and intentionally customize the page for both small and large displays.

Submitting the Project to Turing and blackboard:

To submit your project, do the following:

1. Confirm that the project meets all the requirements. It should work on both large and small screens. Check your code against https://wave.webaim.org/ , https://validator.w3.org/ , and http://jigsaw.w3.org/css-validator/
2. Once you are sure your work is correct. Create a Project 3 folder in your CS354 folder in Turing. Submit a link to your published web site in Turing to Blackboard using the comment section of the submit. Double check your code in Turing against https://wave.webaim.org/ , https://validator.w3.org/ , and http://jigsaw.w3.org/css-validator/ to ensure it still validates.
3. You will not be submitting any code for this project to blackboard.

Grading Rubric:

Site Content Your site presents a static mock-up of your app. The page contains sufficient content, including metadata, text, and media. The content is organized semantically, including header and footer information. Your site includes a link that will “mock” interactivity via navigation.	25.0 pts
Site Style and Structure Your web site includes basic styling and design elements, including color, typography, and (non-standard) layout. Your design makes the content navigable and readable.	20.0 pts
Accessibility Your site is accessible and can be utilized through a screen reader and keyboard. The site is both navigable (via structured and sectioning elements) and perceivable (via alt text and appropriate aria attributes).	15.0 pts
Responsive Design Your site is responsive and usable on both small and large screens. It follows mobile design principles, and includes noticeable changes based on the screen size.	20.0 pts
Code Style, Publish, and Submit Your code is valid, clean, readable, and commented when appropriate. You've avoided redundant HTML elements and CSS rules, and use properties appropriately. CSS class names are informative and consistent. You've published/uploaded your page to Turing and submitted the URL link to blackboard.	20.0 pts

Total Points: 100.0