This document is meant to help you to be sure that you have included all of the required parts in your final project for GIT417. Please make a copy of this document (under File > Make a copy) so that you can edit it and check off the checkboxes as you complete the work. You do not need to submit this, it's just meant to help you as you work on your project in case you aren't sure whether you're remembering the requirements.

GIT417 Final Project Code Checklist & Tips

HTML:

- ☐ Semantic
- ☐ Valid
- ☐ Single page/file (if yours is not, make sure that this is approved by your professor)
- ☐ All code was written from scratch by you (if not, make sure that this was approved by your professor)
- ☐ No placeholder content
- ☐ Navigation works (within the page)
- ☐ There is a realistic amount of content for the page
- ☐ There is some explanation of the site/company/page purpose
- ☐ Page includes header/main/footer
- ☐ Does not contain any inline CSS or JS
- ☐ Properly indented
- ☐ Not causing any errors in the console when open in the browser

CSS:

- ☐ Valid
- ☐ Fixed at 1280px
- ☐ CSS is not inline (all CSS is in the CSS file)
- ☐ Properly Indented
- ☐ Not causing any errors in the console when open in the browser
- ☐ Well commented

JS:

- ☐ **Basics:**
  - ☐ File is properly linked to from HTML (in the head)
  - ☐ async/defer attribute is included
  - ☐ Uses strict mode globally
  - ☐ Well commented (show me that you know how your code works by commenting it well)
  - ☐ Properly Indented
  - ☐ Well organized
  - ☐ Not causing any errors in the console while open in the browser or while running

- ☐ All code was written from scratch by you (although you are welcome to reference code from the book and activities to help you with the project)

☐ Required Features:

- ☐ **Light/Dark Mode:**
  - ☐ Page loads on either light or dark mode
  - ☐ light/dark mode control is visible to the user, easy to use/spot, its purpose is clear
  - ☐ light /dark mode control functions properly to change the site styles between light/dark mode
- ☐ **Form Validation:**
  - ☐ First Name/Last Name OR Full Name is always required
  - ☐ Radios are included and properly coded, allowing user to choose their preferred method of contact - choosing preferred method of contact is required
  - ☐ Comments are always required
  - ☐ Input for email address is included, and is only required if the user chooses email as their preferred method of contact with the radio buttons
  - ☐ Input for telephone number is included, and is only required if the user chooses phone as their preferred method of contact with the radio buttons
  - ☐ Submit button is included
  - ☐ Form validates on submit
  - ☐ Error messages/styles display near to/on the inputs where there are errors in the form
  - ☐ No error messages display by inputs that do not have an error
  - ☐ No output displays/the form does not submit unless all form data is valid
  - ☐ On successful submit, form data is added to an object which is then displayed to the user (either in an alert or on the page)

☐ Feature Options (choose either option 1 OR option 2):

- ☐ **Option 1:**
  - ☐ **Guessing Game:**
    - ☐ Form includes a label, number input and submit button
    - ☐ Input only accepts number values between 1 and 10
    - ☐ Form can only be submitted when the user has entered a number between 1 and 10 (an error message displays to the user otherwise)
    - ☐ The game section makes it clear what this section of the page is for and how the game works/what happens if the user wins
    - ☐ On successful submit, the user's guess and the random number generated by your JS are both displayed on the screen with labels, along with a message telling the user whether they won or not
    - ☐ **Hint**: Snake eyes shows how to compare two numbers to see if they match, it also shows how to generate a random number between and including two values. We also did many activities where we allowed the user to enter a numerical value into a form and displayed it to the screen. Combining those things sounds a lot like what you're being asked to do here. Remember, all input read in from a form is string input.
  - ☐ **Product Switcher:**
    - ☐ Includes at least three products

- [ ] Each product should include a name, image and description/list of ingredients/etc. at a minimum
- [ ] Each product's info should display in the exact same place on the page (the content should swap out of the same location on the page)
- [ ] Only one product should be visible on page load
- [ ] There should be a control that is associated with each product (likely a button)
- [ ] This feature is <u>not</u> a slideshow
- [ ] When the user clicks on the control for a product, that product's image, name, and other details are displayed on the screen and no other product's details/image/name is visible
- [ ] All product controls are visible on the screen at all times
- [ ] It is clear that the user can click on/interact with the controls (hover styles, focus styles, using correct interactive elements, etc.)
- [ ] The control for each product includes the name of that product on it
- [ ] **Hint**: If you choose photos that are the same size, your display won't shift all over the place when you swap content out on each change. You can edit your photos to make them the same size or choose photos that are already the same size. It's a best practice to keep your HTML/CSS and JS separate, so using the JS to add and remove classes is the best way to change the style of content on your site.

- [ ] **Option 2:**
    - [ ] **Shopping Section and Cart:**
        - [ ] Includes at least three products
            - Each product includes: an image, a price, a name, and a control the user can click to add/remove it to/from the cart
        - [ ] The cart area shows:
            - A place for added products to be listed, a place to display the subtotal, a place to display shipping, a place to display tax, and a place to display the total, plus a button that allows the user to check out
        - [ ] The checkout button clears out the items in the cart as well as the totals, thanks the user for their order, and shows them their total (this can be displayed in an alert)
        - [ ] The subtotal is calculated by adding the total of all of the prices of the products in the cart together
        - [ ] The total is calculated by multiplying the subtotal by the tax percentage, then adding that and shipping to the subtotal
        - [ ] There is not a requirement that you allow the user to choose a quantity of any of the items, only that you allow them to add items to the cart, calculate the subtotal/total/tax/shipping and allow them to checkout/clear the cart
        - [ ] As an item is added to the cart, the subtotal, tax, shipping, and total are all updated with the new amounts based on calculations made from the new cart items
        - [ ] **Hint**: Each product can be represented as an object, the cart items can be represented as an array. Add to cart = push to array. Display cart = iterate through array and add to empty string, then add string to the page.