# Project 1: Dynamic Web Site

Build a website with PHP and promote *something*.

## 1. Overview

Your first project is to build a small dynamic 3-4 page website about something about which you are interested or passionate for two target audiences.

### 1.1. Learning Objectives

- Learn to design for multiple target audiences.
- Implement a dynamic web site with PHP.
- Use templates to reuse web page components across a website.
- Process user input server-side.
- Improve form usability for with *sticky forms*.

### 1.2. Deadlines & Receiving Credit

| Milestone | Points | Grade | Sip Days | Deadline |
|---|---|---|---|---|
| Project 1, Milestone 1 (*p1m1*) | 15 | Feedback (*completion*) | **Not Permitted** | Tue 2/5, 4:00pm |
| Project 1, Milestone 2 (*p1m2*) | 15 | Feedback (*completion*) | **Not Permitted** | Tue 2/12, 4:00pm |
| Project 1, Final (*p1fin*) | **100** | Rubric (*p1m1* + *p1m2* + *p1fin*) | Maximum: 2 days | Tue 2/19, 4:00pm |

**No slip days allowed for milestones.**

**Milestones are graded *twice*.** First for feedback (completion grade only; no partial credit). Lastly, for points at the final submission (via rubric). **All** work is graded for points (not completion grade) at the final submission.

Completion credit will be awarded so long as you made a good faith effort to complete the milestone's requirements. Very obviously incomplete milestones will receive a 0 for the completion grade.

**Use your milestone feedback to improve your final grade; revise milestone work prior to final submission.** If you didn't submit a milestone or received negative feedback, make sure you make up or revise the work for the final submission.

We provide feedback on your milestones to help guide your work for the final submission. Our feedback is designed to help you learn more; our feedback is not a "*pre-grade*". This feedback is designed to catch large problems (which we sometimes miss). **Regardless of the feedback (or lack of feedback) that you get, you are responsible for meeting**

**all of the project's requirements for the final submission.** "My feedback said X was good enough..." or "My feedback never explicitly told me I had to..." or "My feedback told me to do X (which contradicts the project write-up and/or syllabus)" will not be accepted as valid arguments for a regrade.

### 1.3. Git Repository & Submission

Clone `git@github.coecis.cornell.edu:info2300-sp2019/YOUR_GITHUB_USERNAME-project-1.git`. Replace **YOUR_GITHUB_USERNAME** in the URL with **your actual GitHub username**. This is usually your NetID.

Submit **all** materials to your GitHub repository for this assignment. **See *README.md* in your Git repository for submission instructions for each milestone.**

**Tip**: Commit and push your changes every time you work on your project. Every time you commit and push you store your changes on the GitHub server. This acts as a back-up for your work. It also means that if you forget to submit before the deadline, there's something already on the server that the TAs can grade.

## 2. Requirements

Build a website with PHP and promote *something*.

For Project 1 you'll build a 3-4 page website to promote a topic of your choice. It's probably best to pick something you find interesting or are passionate about. It can be something that you do as a hobby, something related to your academic career, or something you just find enjoyable. Or it can be just plain silly as long as you meet the requirements. Your site should include some basic information about what you are promoting (why you're promoting it) and images to describe the thing that you are promoting. Examples include (but are not limited to) a sports team, clubs, organizations, individuals, items (technology, fashion/clothing, book), photography, music, yourself as a person, etc.

### 2.1. Base Requirements

- All code must be your own work. You may not use code from other classes, including INFO 1300.
- All content is cited according the course citation policy.
- 3-4 page website designed for two specific target audiences.
  - Thoroughly planned site. Design and plan first, then code.
- Navigation bar must be implemented as a PHP template.
  - Navigation bar must programmatically indicate the current page.
- Site must include 1 additional template besides the navigation bar.
- Site must include a sticky form with a minimum of 4 inputs (does not include submit button).
  - Form must provide error message handling via PHP (no JavaScript).
  - Form confirmation page must show the form input from the user.

## 2.2. Design & Design Process Requirements

- Design a 3-4 page website for 2 specific target audiences.
    - You may have more than 4 pages. However, we will only look at 4 pages.
    - Target audiences should be specific.
    - *Future employers* is specific while general target audiences like *everyone* are not.
- Design must be planned in the **design-journey.md** before coded.
    - Plan the design through liberal use of sketching, card sorting, etc.
    - Document your design process. **You should show us the evolution of your design.**
    - Minor changes to the design are acceptable during implementation.
    - Major changes in implementation require revising the design.
- The site should be well-designed, have a consistent "look and feel," and be aesthetically pleasing.
- Site must properly leverage visual design principles.
    - Repetition & Consistency
    - Typography
    - Color
    - Contrast
    - Hierarchy
    - Alignment
    - Proximity
    - Whitespace.
- Site content should be logically organized for both target audiences.
- Site must have clear navigation appropriate for both target audiences.
- Site must include images.
    - **All images must be cited according to the course citation policy.**
- You should design your website for desktop computers.
    - You do not need to make your website responsive. You can if you want, but it's not required.

## 2.3. Template Requirements

- All reusable content should be implemented as a PHP templates using PHP's `include()` function.
    - You are required to have a template for the site's navigation.
    - You are required to have 1 additional template besides the navigation.
    - The reusable content must be used on all pages of the site.
- The website's navigation should be implemented as a template.
    - The navigation template should programmatically indicate the current page.
- All *includes* must reside in the **includes** directory.

## 2.4. Sticky Form Requirements

- Plan your form before coding. Include your pseudocode in the design journey.
- Your site must include at least 1 HTML form with a minimum of 4 input controls.
    - You may not count the submit control as part of the 4.
    - 2 of the input controls must be required.
    - All input controls must have an associated `<label>` element for accessibility.
- You must implement server-side form validation.
    - Client-side form validation is prohibited. (i.e. No `required` HTML attributes, no JavaScript, etc.)
    - Check whether the required inputs exist in PHP ( `isset()` , `empty()` ). If not, show an error message to the user.
- Validated and submitted forms should show a confirmation page with the form input displayed on screen.
    - All input must be properly sanitized when placed on the site: `htmlspecialchars()` (do not use `htmlentities()` ).
- Non-validated forms should return to the user to the form to fix their mistakes.
    - The form must be sticky: the user should not have to re-input what they've already submitted.
    - Error messages should shown for all mistakes.
    - Error messages should be appropriate for both target audiences.

## 2.5. Coding Requirements

- Site must be coded in valid HTML, CSS, and PHP.
    - JavaScript is prohibited for this assignment.
    - All user accessible pages (viewable in the browser) must contain valid HTML.
    - Validate HTML by viewing the PHP page in the browser, then view the HTML source in the browser. Copy the source and paste it into the HTML validator: https://validator.w3.org/nu/#textarea
- Your site should display reasonably well (not necessarily identically) across Firefox and Chrome.
- You may assume that your web page will be viewed in a desktop browser. You do not need to design a web page for mobile browsers. You do not need to use media queries.

## 2.6. Best Practices

As a professional web developer, your boss will not give you a rubric telling you that your design needs to be aesthetically pleasing or that you need to name the main HTML file **index.php**. It is expected that you know the standards, conventions, and expectations of your field. I expect the same in this class. **This write-up may not explicitly state these expectations and we will deduct points if you fail to follow them.**

As a reference, here are some of these expectations that you should already know:

- **All code is your own work**, unless the assignment states otherwise.
- **Content is cited** correctly as specified in the syllabus.

- Your website is designed to **meet the needs of a specific target audience(s)**.
  - Design employs **visual design principles**.
  - Your design is **aesthetically pleasing**.
  - A multi-page site should be **well organized and include proper navigation**.
- Your code is documented with comments where appropriate.
- Main page is named **index.html** or **index.php**.
- The **HTML is well structured** for your site's content (i.e. use of `<header>`, `<main>`, `<section>`, `<aside>`, `<footer>`, `<strong>`, `<em>`).
  - **No** `<b>` or `<i>`, etc.
- **External styling via CSS**. No inline or internal styling (i.e. `<style>` elements or `style=""` attributes).
- Multiple CSS files are okay as long it's for **legitimate structural reasons**.
- Your code (HTML, CSS, JavaScript, PHP) is **well written, readable, formatted, and properly indented**.
- **No broken or dead links**. Remember that some computers use **case sensitive** file and folder names!
  - *Your instructor's computer is case sensitive.*
- **No hotlinking** of external resources. This almost always includes *embedding*.
  - Google fonts is hotlinking unless you host the fonts locally.
- **Validated HTML5 and CSS3**. You must have 0 errors; warnings are permitted.
- Error free code (PHP). Warnings are okay.
- All files are reasonably sized (~< 1MB) for the web (this includes: images, videos, PDFs, etc.)
- Well organized site files.
  - Images are located in an **images** directory.
  - Your CSS files are located in the **styles** directory.
  - Your PHP includes are located in the **includes** directory.
- You have tested your website in **Firefox** and **Chrome**.

## 3. Milestone 1: Plan your Web Site

For Milestone 1, you'll identify two target audiences and provide a sketches for each page of your website. You'll also start coding up your website.

### 3.1. Design

Complete the **Milestone 1** section of the design journey: **documents/design-journey.md**.

**A word about sketches:** Before you code anything you should design your web page first. Sketching is the best way to jump start this process. Sketches are quick and help you generate ideas that will improve your overall design. You should sketch on paper. Sketching using a computer program takes too long and misses the point of sketching.

Your sketches don't need to be polished. The lines don't need to be straight. You don't need to write out all text; squiggly lines are okay to *represent* text. You should only use one color: black and white. The idea here is to generate ideas.

Take a picture of your sketches (I recommend the Microsoft Office Lens app) and place the image files in the *documents* directory and link them in your Design Journey. **DO NOT PLACE DESIGN JOURNEY IMAGES IN THE WEB SITE'S IMAGES DIRECTORY!**

**All design/planning images must be visible in your design journey when using Markdown Preview. No credit will be provided for images in your repository that are not properly linked in Markdown.**

### 3.2. Implementation

After planning and designing your site, start implementing it. You don't need all 3-4 pages. You don't need templates yet either. However, you should have some HTML content in PHP files and some CSS done.

## 4. Milestone 2: Draft Website

For Milestone 2, you will finish your PHP templates and plan your sticky form.

### 4.1. Templates

Your templates should be fully functional and complete. At this point your website should be mostly done except for the sticky form. This means your content should be in place. Your navigation should work. Your CSS is mostly there. The website should look like it's coming together.

### 4.2. Sticky Form

In the design journey, plan out your sticky form. After planning it, start coding it. It doesn't need to be sticky yet. The error messages don't need to work yet. You don't need to validate the inputs yet. At a minimum you should `echo()` the inputted values on the confirmation page.

## 5. Final Submission: Complete & Polished Website

For the final submission, finish implementing your designed and planned website. Your final site should be complete and polished. Your form should be sticky and provide validation to missing inputs.

Tell us how your complete and polished website meets the needs of your target audiences in the design journey. Also take this time to reflect on your experience of building this website.

**All design/planning images must be visible in your design journey when using Markdown Preview. No credit will be provided for images in your repository that not properly linked in Markdown.**