INFO/CS 1300: Project 3 Requirements and Milestones

The purpose of Project 3 is to take a step into the more technical and programmatic side of web development. By the end of Project 3, you should be able to understand some of the basics of programming, as well as how to use them to make websites that react to users' behavior, particularly with the use of forms. For Project 3 you will be building on your Project 1 or Project 2 website by allowing users to have more meaningful interactions through the use of forms and server logic (PHP).

You will receive guidance and more information about forms, as well as useful code, during both lecture and labs. Attending these labs will help you build up your skills in this area. In addition, if you need more help or clarification, we try to have office hours at various days and times so that you can come visit and resolve any issues you may have.

We will be testing your websites on our class "approved" browsers, Firefox and Chrome. Your website should be functional at various screen sizes, however it does not need to be fully responsive. (i.e. it's okay if we have to scroll horizontally for a mobile screen size as long as everything lays out properly) This means we're not going to go out of our way to use an old browser version or try to break your website on small screens.

For this project you will need to follow the standards, conventions, and exceptions for websites of this class. See Project 2 for a reminder of some of them. This also means that if you use web fonts, you must not hotlink them! You are required to follow the procedure outlined in the Lab 7 for using fonts that are not the generic font family names!

For Project 3, all code must be your own work!

Milestone 1 (10 points): due Tuesday, October 24th @ 17:00
Milestone 2 (30 points): due Tuesday, November 7th @ 17:00
Final Milestone (100 points): due Tuesday, November 21st @ 17:00

Milestone 1: Templating and Planning your Web Form

Due: Tuesday, October 24th @ 17:00

For many of you, this may be the very first time you've attempted any kind of programming, so this milestone will be a little light in that area while Milestone 2 will focus more on programming.

This milestone is meant to show how a minimal amount of programming can be used to make your website a lot easier to modify. In addition, you will also be thinking about and implementing a basic form in preparation for Milestone 2.

Important!

Milestone 1 will be based off of your Project 1 or Project 2. Depending on how you did on the final version, you may want to consider improving it. We also have office hours where we can help you. Otherwise, it may be difficult to incorporate additional functionality, provide justifications, etc.

You will also need to reduce the size of you web page to 20MB for Project 3.

Requirements

You have three main tasks for this milestone:

1. Convert your site to use PHP **includes** for repetitive code,
2. Design and add the form to the client side, and
3. Write a brief rationale.

1. PHP Includes

Take your project website and convert it to use PHP **includes** for the part(s) of your website which are the same on every page. This will allow you to edit the reused pieces of code in one place and have it be the same on all pages instead of having to update it in multiple places.

This might include headers, footers, menus, or other parts of the structure where you have copies of code on every page. Note that you will need to rename files that use PHP to .php instead of .html. Also, instead of validating the raw .php file, you'll need to validate the HTML that is generated (i.e., use view source in the browser and copy-paste the source into the validate by direct input feature on the validator on W3C: https://validator.w3.org/).

To use PHP includes, you want to copy "duplicate" HTML code to a new PHP file. For example, you may want to take all the `<nav>...</nav>` code and move it to i**ncludes/navigation.php**. Then you would remove the `<nav>...</nav>` in your main files and replace it with this:

```
<?php include("includes/navigation.php"); ?>
```

Observe the **includes** directory in the above example. Just like **images** or **styles** directories for images and CSS respectively, it's a good idea to create a directory for your PHP **includes**. This is *not optional;* the includes directory is a convention of this class.

Note: Some of you may have made the active page in your navigation bar look different using a CSS `id` or `class`. This is a little more difficult to do with PHP. Do not worry about doing this for Project 3. It's okay if the active page does not look *active* in the menu bar for Project 3.

2. Client-Side Web Form

Determine what purpose the form for your site will accomplish, where it will go, and what form elements, types, and attributes you plan on using (include at least 6 distinct form elements and at least 3 distinct input types). We are looking for you to have creative and relevant forms that serve real users' needs. There are some examples below regarding the Apple Harvest Festival and your Project 1 which you can use for inspiration and guidance.

- Apple Festival - vendor request (name and contact info of company, type of product, booth size, which days/times? etc.)
- Apple Festival - feedback form (what days/times did you go? how did you like the festival? any improvements you'd like to recommend? were there issues you'd like to bring up? etc.)
- Apple Festival - sponsor request (request to be a sponsor, what capacity? what kind of presence? etc.)
- Apple Festival - performer request (request to be a performer, what kind of performance? do you need space? overhead coverage needed? etc.)
- Apple Festival - trivia quiz
- Store/Service - special/large requests, feedback/inquiries, order page
- Portfolio/Project - sample/project inquiries/feedback (ask questions or give feedback about certain photos from a photographer's portfolio)
- Blog - recommendations/inquiries based on topic (recommend new travel spots for a travel blog)
- Personal - detailed contact (more than just a simple contact form, who do they represent, what is the intent of contact?[recruiting, project question, general inquiry, etc.], their message, contact information, best times to reach, etc.)
- Other - polls/voting/quizzes

After determining what purpose your form will have (and why it's a good purpose for the site's goals), you have to decide where to put it. It is probably easiest to create a separate page, but this is entirely up to you. If you have a form that is about a specific item/section then you may want to consider embedding it into that page.

Your form will not need to do any client or server side validation for this milestone. For now, your form only needs to direct you to a separate response page (using the action attribute of the form tag). We won't be covering how to do this in class or lab. Just like how you learned CSS, part of working with the web is learning how to gather this information from the always changing specifications or examples on the web. Google is your friend here. Try to figure this out on your own (you'll learn so much more if you do). And as always, you can ask for help if you get stuck. We are always happy to give you hints.

Note that the form and response design should match your site design. Your response page should make it obvious that I am still on your website, not just a blank page that says "form submitted".

3. Rationale

Create a rationale.pdf (notice this is a PDF file) which includes the following questions with your answers below each question (bullet points are accept for your answers):

1. Which part(s) of your website did you use PHP include for and justify why you chose them. (1-2 sentences)
2. Explain where you think is a good place to incorporate a form into your website, what form elements, types, and attributes you plan on using (6+ distinct form elements and 3 distinct input types), what purpose you want it to serve, and why it is an appropriate and useful form. Using sketches/storyboards to help communicate the where and what elements aspects is fine; text is also fine. (1-3 sentences)
3. (optional) Include any additional information, justifications, or comments we should be aware of. If you have questions, please limit to office hours where you can get your questions answered. (1-2 sentences)

Tips

- Read this Project write-up carefully.
- Here is a handy reference for HTML forms: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Forms
- Get second opinions on your form from another student or a TA. The earlier you get these basic parts of Project 3 down, the easier everything else will be.
- On lecture, labs, and looking up things on the web: if you're comfy, fine. But many people often skip these things and spend way more time frustrated on their own and in office hours than they "saved". Beware short term gain vs. long-term pain, even in the face of competing priorities.
- I want you to learn how to look up information on web development. This is a vital skill for learning web programming because it changes so frequently. Do not be afraid to do this, especially since it's a required skill to succeed as a web developer.

- Feel free to work ahead. We understand that you are working on both this project and your final project and might want to get a jump start on later parts. We won't penalize you for being smart with your time and planning ahead.

Grading

10 points.

The grade will be based on whether you did the assignment. We will also provide written feedback. If it looks like you tried, even though there are some mistakes, you will be fine. If you have lots of issues or it's pretty obviously incomplete, you won't get full credit.

Our feedback is designed to catch large problems (which we sometimes miss). We are trying to find things you need to watch out for to help you improve your final project (and thus your grade). This does not resolve you of the responsibility of meeting the project's requirements, even if we miss something.

Submission

1. Submit your rationale as a **PDF** to CMS separate from your website.
2. Place your website in a folder called **p3m1-NetID** (example: p3m1-kjh235).
3. Zip up your website's folder and submit the zip file to CMS. **Only include your website's files in the zip.**
4. Max file size for your website is 20MB. If you are working on Project 1, you may need to go back and reduce the size of your website based on what we learned in Project 2.

If you submit late, do **not** send info1300-prof@cornell.edu an email stating that you are using a slip day.

We grade on Tuesdays. If you submit late you should not expect that you will get a grade or feedback before the next Tuesday. **Not receiving a grade or feedback quickly is a consequence of submitting late.** This means you should not email us asking us for feedback if you submitted late.

Milestone 2: Form and Client-Side Validation

Due: Tuesday, November 7th @ 17:00

Forms are a very important part of websites which provide you the ability to collect and interact with information as well as allow users to communicate with you and your website at a deeper level. Common uses of forms include logins, signups, contact information, placing orders, search, surveys, and many more.

Based on your Milestone 1 and the corresponding feedback you received, the goal of this milestone is for you make your form robust on the client side and do something reasonable with the data on the server side.

HTML5 has useful built in validation tools that will help you here: https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

Requirements

1. Take your Milestone 1 and make sure that your form is well structured and provides all information the user needs to fill it out. This includes effective labels, clear indications of required and optional elements, guidance for data formats, and other design elements that make it easy to fill out the form.
2. Your HTML form must prevent the user from submitting, and give useful feedback about the problems, if it is not valid as outlined in your rationale. You can use built-in HTML5 form element checks and/or jQuery to help you validate your form and give good feedback. Use google here to help you figure this out. I want you to get comfortable being able to code something that you were not directly taught in class. If you get stuck, ask for help, that's what we are here for and I really like helping you!
3. The form must direct the user to a response page that echoes back the data that was submitted. If your eventual use case doesn't require echoing back the data, for now you could just create a 2 column table with field names on the left and values on the right, plus whatever else you plan to do for a valid submission. If the use case does include echoing back the information, feel free to use your real planned design for a valid submission. You do not need to store any data on the server side, just `echo` it.
4. Note that you don't need server-side validation yet, but you are welcome (encouraged, even) to start working on it.
5. Create a rationale.pdf (notice this is a PDF file) which includes the following questions with your answers below each question (bullet points are accept for your answers):
    1. List the form elements, types, and attributes that you ultimately used and briefly justify your choices. (1-2 sentences)
    2. Outline the validation requirements your form needs in order to be submitted successfully from the client side and why you chose them. (1-3 sentences)

3. (optional) Include any additional information, justifications, or comments we should be aware of. If you have questions, please visit office hours. (1-2 sentences)

Tips

- Think about the user's experience as well as your form's requirements here. How can you help them complete the form successfully, quickly, easily, enjoyably?
- Revisit which form elements are most appropriate for the tasks you're asking people to do and the validation needs around them.
- Reminder about lectures, labs, and class announcements: if you don't really feel comfy with the stuff, you skip them at your peril.
- Beware that not browsers are created equal. Test on multiple browsers. Our class browsers are Firefox and Chrome.
- You don't have as much time as you think between this and the final version. So working ahead towards server-side validation would be smart.

Grading

20 points.

We will grade on a rubric with minimal written feedback. The rubric covers the basic requirements outlined here. If you meet the requirements you will get full credit. We will take off points for missing pieces or deduct lots of points if it's incomplete.

Our grading or feedback is designed to catch large problems (which we sometimes miss). We are trying to find things you need to watch out for to help you improve your final project (and thus your grade). This does not resolve you of the responsibility of meeting the project's requirements, even if we miss something.

Submission

1. Submit your rationale as a **PDF** to CMS separate from your website.
2. Place your website in a folder called **p3m2-NetID** (example: p3m2-kjh235).
3. Zip up your website's folder and submit the zip file to CMS. **Only include your website's files in the zip.**
4. Max file size for your website is 20MB.

If you submit late, do **not** send info1300-prof@cornell.edu an email stating that you are using a slip day.

We grade on Tuesdays. If you submit late you should not expect that you will get a grade or feedback before the next Tuesday. **Not receiving a grade or feedback quickly is a consequence of submitting late.** This means you should not email us asking us for feedback if you submitted late.

Final Submission: Server-Side Validation and Sticky Form

Based on your Milestone 2 and the feedback you received, the goal of this milestone is for you to extend your form functionality. This includes protecting your server against invalid submissions with server-side validity checks. Client-side validation helps users, but you also need server-side validation because attackers can bypass your form (Javascript can disabled if you use it for validation) and because not all browsers implement all of the validation checks in HTML5. It also includes helping users resume incomplete or failed form submissions. This is on top of all of the fundamentals and basics that you've acquired throughout the course thus far.

Requirements

1. Take your Milestone 2 and convert your form into a "sticky" form. This means that in addition to your client-side validation, you will need to have server side logic to check if the submitted form was valid or not. If the form was not valid then the user should be directed back to the form, pre-filled with the previously entered information along with feedback that helps them detect and correct errors.

   Note that this means you will likely need to have the same kinds of validation checks both on the client side and the server side. Client side validation is useful for improving the user's experience, but server-side validation is important for protecting your site's security. Good sites have both.

2. Create a version of your form for testing server side validation. This should be a fully functioning page that works just like your regular form page, just with all client side validation disabled. You will need to do this for your own testing as well as to make it easier for us to test your work.

   There are multiple ways to create a version of the form that doesn't do validation.

   1. If you only use basic HTML5 form validation then you can make a copy of your completed form and add `novalidate` to the form tag. This will probably be sufficient for the requirements above.
   2. If you have additional validation (i.e., through Javascript or jQuery) then you probably want to make a copy of your completed form and strip all of your validation.

4. 5. Create a rationale.pdf (notice this is a PDF file) which includes the following questions with your answers below each question (bullet points are accept for your answers):

   1. Tell us which pages we should look at for your form and response, including the version that doesn't have client-side validation. (1 sentence)
   2. List the requirements that you've implemented on the client and the server side to validate your form. If there are differences between the client and server side, justify them. (1-3 sentences)

3. How does your design present errors to the user and why is this a good choice that makes it easy and clear for your user. (1-3 sentences)
4. Include any additional information, justifications, or comments we should be aware of. (1-2 sentences)

Tips

- You can make your code a lot simpler if you write small functions for repetitive things like generating bits of HTML that compose error messages.
- PHP provides a lot of useful functions for server-side validation. Here are some functions that you may find useful.
    - For handling spaces and empty fields, `trim` (to remove leading and trailing spaces) and `isset` and `empty` (to check whether variables contain data).
    - For dealing with HTML input, `strip_tags` (to remove HTML tags, if appropriate), and `htmlspecialchars`/`htmlentities` (to handle HTML special characters like &).
    - For data type validation, `filter_var` (email, numeric values, URLs) and `dateparse` and `checkdate` (for dates).
    - More info and examples at http://php.net/manual/en/funcref.php (or your favorite search engine).
- Here are some useful things to think about when doing server side validation:
    - Is it there?
    - Is it the right type?
    - Does this have only the things it's supposed to have?
    - Is it within the right range/scope that I want?
- Using placeholder text instead of form labels in forms is tempting, but works badly when there are errors or when users have the form filled out: is this a phone number? Is it home or work? Is it optional or required? Placeholders for formats and labels for the data type is one common, winning strategy for being helpful.
- Beware saucy error messages that are funny to you but may be confusing or insulting to people using your design. http://cs3240team16.wordpress.com/2012/09/01/error-message-design-good-and-bad-examples/
- Don't arbitrarily use functions that strip special characters or encode strings unless you have specific reasons for doing so in your validation. Use the right tools for the job.

Grading

100 points + Milestone 1 (10 points) + Milestone 2 (20 points) = 130 points

We will grade on a rubric with no written feedback.

Submission

1. Submit your rationale as a **PDF** to CMS separate from your website.
2. Place your website in a folder called **p3final-NetID** (example: p3final-kjh235).
3. Zip up your website's folder and submit the zip file to CMS. **Only include your website's files in the zip.**
4. Max file size for your website is 20MB.

If you submit late, do **not** send info1300-prof@cornell.edu an email stating that you are using a slip day.

Rubric

This is the rubric for the final milestone. You should keep this rubric in mind while working on your milestones. I reserve the right to tweak this, but this should be close to the final version.

Rationale (10 points)

- No Credit: A poor justification/explanation results in no credit for the talking points for that section.
- Deduction: Not there, not PDF, or poorly addressed all 4 talking points.

Web From, Validation, Response (60 points)

Form/Response requirements [10 points]

- Deduction: Form - Does not contain at least 6 distinct form elements and 3 distinct input types, unless justified. Check the PHP documentation. This is all or nothing: if they don't meet the requirement.
- Deduction: Form or response - Poor formatting/design (overlap, ordering, improper alignment, spacing, etc), or design isn't consistent with the website. This is all or nothing: if the design is not good.
- Deduction: Form - No validation-free version for testing.

Validation requirements [15 points]

- Deduction for each inadequate requirement. This means that there is a reasonable requirement that should be included but is not. For example: on a request form where the plan is to reply by email, it's logical and reasonable to require this input.

Client side validation [10 points]

- Deduction for each individual error. An individual error is any instance of broken validation related to the validation requirements. For example, if the validation requirements state that an email input is required but there isn't a check for a valid email or that the field is empty.

Server side validation [10 points]

- Deduction for each individual error. An individual error is any instance of broken validation as related to the validation requirements, similar to the client-side checks.

Success Response Page [5 points]

- No Credit: No success response page.
- Deduction: Inappropriate/unclear response page.
- Deduction: Visual design is poor and/or does not match the rest of the website.

Sticky form [10 points]

- Deduction: Does not show pre-filled form containing any of the previously entered information.
- Deduction: Shows pre-filled form containing some but not all of the previously entered information, unless justified.
- Deduction: Does not clearly indicate errors and what's needed to fix them.

PHP Includes (10 points)

- No Credit: Not used or incorrectly used, without good justification.
- Deduction: Used, but poorly, improperly, or incompletely.

Fundamentals (20 points)

- Code following standards, conventions, and expectations of class.
- Valid HTML/CSS/JS.
- Code Formatting.
- Folder/File Structure.