INFO/CS 2300: Project 3

Create an photo gallery with user access controls.

1. Overview

You've learned a lot of server-side programming techniques in this class so far. But you've only been able to use them in very controlled situations. This makes it hard to understand how all these different pieces come together to make a fully functional web page. In Project 3 you'll put all of these pieces together to create a photo gallery.

2. Learning Objectives

- Experience building an entire website applying the techniques we've learned thus far.
- Develop the skills to translate client/customer requirements into a working implementation.
- Practice using your *web programmer's toolkit* to solve complex problems.
- One last practice empathizing with a persona before you'll need to use this skill to conduct a cognitive walk-through for your final project.
- Practice structuring a database with multiple tables and foreign keys.
- Practice building and querying relationships between tables using common fields (joins).
- Working knowledge of how to implement support for user accounts for a web site with sessions and cookies.

3. Deadlines & Receiving Credit

Milestone	Points	Grade	Deadline
Project 3, Milestone 1 (p3m1)	20	Feedback (*completion*)	Tues. 3/20, 5:00pm
Project 3, Milestone 2 (p3m2)	20	Feedback (*completion*)	Tues. 3/27, 5:00pm
Project 3, Final (p2final)	**200**	Rubric	Tues. 4/10, 5:00pm

You may use your slip days (see syllabus) for all deadlines. Maximum of 2 slip days per deadline.

Submit all materials, including plans, designs, and reports to your GitHub repository for this assignment. See each milestone's respective **submit.md** file for directions.

3.1. Spring Break

This assignment is due after Spring Break. You have a total of 3 weeks to work on this project **not** including your break. Each milestone is designed to take 1 week. This is sufficient notice and time to complete this project. This

follows the University's policy for work during breaks (http://theuniversityfaculty.cornell.edu/the-new-faculty-handbook/6-policies-and-assistance/6-1-instruction/work-over-break-2/). Given this advance notice and sufficient time to complete the assignment, I will not make any exceptions to accept late work due to the break.

4. Assignment Repository

`git@github.coecis.cornell.edu:info2300-sp2018/YOUR_GITHUB_USERNAME-project-3.git`
Replace **YOUR_GITHUB_USERNAME** in the URL with **your GitHub username**.

5. Description

For Project 3 you will design and implement an online photo gallery in PHP. Your photo gallery will be backed by a database which you will use to store information about the images uploaded to your gallery. To help organize the photos in your gallery, you will implement the ability for users of your site to tag the photos. You will also implement login/logout functionality and user access controls to control who can upload photos on your site.

6. Requirements

Below are the requirements for Project 3.

6.1. General Web Site Requirements

Your website should meet the following requirements.

- Your website should be designed for a specific persona.
- Your design should be aesthetically pleasing and follow visual design principles.
- Your page navigation should be well designed. Take care to ensure your page navigation is appropriate for your persona.
    - There should not be any dead-end pages on your site that require the user to click the *back* button in order to continue navigation.
    - Organize your PHP files so that it is easy to add new pages. Make sure your navigation is created in just one place so that as you change it, you don't have to change it on every page.
- You will implement your web site in PHP with the data backed by a database.
- When generating the HTML for your web page, you should use SQL to retrieve specific data from the database and *directly* use that data for your HTML generation.
    - Do not use PHP to *retrieve or find* specific data. For example, do not retrieve all the records from the database using SQL and then use PHP to loop through and find the data you want. SQL is much more efficient at searching and we expect you use it where appropriate.

- You are required to create your database using SQL. Create your database in your DB initialization script (**init/init.sql**).
    - You are not permitted to use DB Browser for SQLite for creating or modifying your database for this assignment.
    - You may use DB Browser for SQLite to view your development database as well as test your SQL queries.
    - Do not commit your development databases to your Git repository. We have configured your Git repository to ignore all SQLite databases.
- All file uploads should be placed in a sub-folder under the **uploads** folder.
    - All sub-folders should have the same name as their corresponding database table.
    - All files stored in each sub-folder are required to be named as the corresponding database record.
    - Do not commit any uploaded images (except seed images) to your Git repository.
    - **Do not store file uploads in the *images* folder.**
- There is no minimum number of pages you need to implement.

6.2. Photo Gallery

You'll be implementing a photo gallery. Your photo gallery must meet the following requirements.

- Your persona should be able to view *all* images in your photo gallery at once.
- Your persona should be able to view *all* images for a *tag* at once.
- Your persona should be able to view a *single image* and all the tags for that image at once.
- Your persona should be able to upload a new image.
- Your persona should be able to remove (delete) an image.
    - Make sure you clean up any relationships to the image in other tables. (Where the image is a foreign key.)
    - Make sure you delete the corresponding file upload from disk.
- Tags should be unique. You cannot have duplicates of the same tag.
- Your persona should be able to view *all* tags at once.
- Your persona should be able to add an existing tag to an image, add a new tag to an image, and remove a tag from an image.
- You may store and display any other type of data you like for the images. Although none of this is required.
    - example: Which user uploaded an image.
    - example: A user defined description for the image or the date it was taken.
    - example: The file format or size of the image.
- Implementing your gallery
    - Use query string parameters (`gallery.php?tag=balloons` or `gallery.php?tag_id=3` ; `image.php?image_id=8`) to render specific content.
    - You should render any content in your database dynamically. Do not hard-code tag or images pages for your seed data (i.e. `balloons.php`).

Note: Your content can be something other than an image gallery but it must follow a similar structure. If you choose entities other than images and tags, you should explain in your design-plan.md how your entities correspond to the requirements of the assignment as written in the language of images and tags.

6.3. User Access Controls

- You will need to build user account support for your web page.
  - Users with accounts should be able to login with an *existing* username and password.
  - Logged in users should be able to logout.
  - You do not need to implement account creation (sign-up).
  - You do not need to implement changing usernames or passwords.
  - You do not need to implement confirming accounts by sending an email.
- You need to manually implement session support.
  - Upon login, create a session and store it in the database. Send the session ID as a session cookie to the user.
  - Upon logout, destroy the session in the database and reset your session cookie.
  - **You may not use PHP's built-in sessions (** `session_start() or $_SESSION[]` **).** For this assignment you need to manually program it yourself using cookies and your database.
- Your user access controls should have a secure implementation.
  - Passwords must be stored in the database in hashed form.
  - You should generate a unique session ID for each login.
  - You should clean up old sessions and cookies when a user logs out.
  - Due to the limitations of your development setup, you cannot use SSL (HTTPS) to encrypt sensitive information. This is acceptable for this assignment.
- You should implement user access controls for your image gallery.
  - Only a logged in user may upload an image to the gallery.
  - Only the image creator (uploader) may delete that image.
  - Both, anonymous users (not logged in) and logged in users may tag images.
  - Only the user who uploaded an image may delete tags from that image.
  - Hint: You should probably not show options to users that they cannot do. For example, Abby is likely to be confused if she's using your website, she's not logged in, and she see's an option to delete an image.

6.4. Well Organized Database

To implement the photo gallery and user access controls you will need to carefully design your database.

- You should plan your database before you implement anything.
  - Seriously. Planning your database is the key to success for this project. A bad database design can make the basic tasks of this project extremely difficult.
- Your plan should include a schema for each table in your database. Make sure you specify any field constraints.

- Your database is required to implement a **many to many** relationship.
  - Your two main entities, images and tags, should be in a **many to many** relationship. One tag can have many images and one image can have multiple tags.
- There is no requirement for a minimum number of tables.
- Follow the database guidelines for the web as discussed in lecture 12.
  - Example: Do not duplicate data in your database (foreign keys excluded).

6.5. Seed Data (Populated Database + Images)

Your website should ship with some initial data.

- You should provide initial *seed* data for your web site.
  - You should have at least 2 user accounts. Specify the username and passwords in your design-plan so we can test your web page.
  - You should have at least 10 images.
  - You should have at least 5 tags.
  - At least 3 tags must applied to at least 1 image.
  - At least 8 images need to have a tag.
  - At least 3 images need to have *multiple* tags.
- All seed data should be SQL queries in your database initialization script (**init/init.sql**).
  - You are not permitted to add seed data to your development database using DB Browser for SQLite.
- All seed image files should be stored in a sub-folder under the **uploads** folder.
  - The sub-folder should be named the same as the corresponding database table.
  - All seed images should named to correspond to their respective database record.
  - You are required to modify **uploads/.gitignore** to *exclude* your seed images files.
    - If you forget this step, your seed images will not be committed to your Git repository.
    - Forgetting to commit your seed images will result in a point deduction.

6.6. Security

Your code should filter input and escape output as we've practice in lecture, labs, and your projects.

- Filter Input
  - You should thoroughly filter all input.
  - Your design should report all invalid or rejected input back to the user.
  - You are not required to validate your forms (what you did in INFO 1300). Form validation is checking the form input and sending a populated form with error messages back to the user if their submission was incorrect.

- Escape Output
  - Make sure you escape any dirty inputs. Be careful to escape for the appropriate context. You should escape for HTML and SQL.
  - Rigorous SQL injection prevention is required for this assignment.
- Passwords & Sessions
  - Passwords must be stored in the database in hashed form.
  - You should generate a unique session ID for each login.

6.7. Testing

- You should rigorously and thoroughly test your website and forms to make sure there are no errors and everything works as intended.

6.8. Coding Standards & Design Guidelines

- 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 for responsive design.
- A source credit must display near each image unless you created the image yourself in which case the credit can be in an HTML comment.
- Your assignment should follow the coding standards, conventions, and expectations of this class.
- All code should be easy to read and understand by anyone.
  - Use functions and includes to help organize your code.
  - Make use of comments to explain things that aren't obvious.
  - Name your functions and variables appropriately.
  - Your code should be indented properly so that it's easy to see which lines belong to each element (HTML/CSS) or function (PHP/JavaScript).
- Your HTML must validate.
- You must connect and interact with the database using PHP's PDO extension. No other methods/libraries are permitted.
- Your website should be under 10MB (including seed data).
- All code must be your own for this project. You may not use frameworks such as bootstrap or libraries such as jQuery.

7. Milestone 1

20 points.

In Milestone 1, you'll design and plan your website and database. You'll also inform your design based on a persona of your choice. You'll generate *seed* data and build a "skeleton" of your website. Like Project 1, you'll author your plans in Markdown in the *design-plan* folder of your Git repository.

7.1. Pick a Persona

You have 4 personas to choose from located in the **design-plan/personas.pdf** PDF document. You may pick any persona you like. Make a note of your selected persona in your Design & Planning document: **design-plan/design-plan.md**.

7.2. Sketch & Wireframe

Design your Project 3 website. When designing take care to meet the "Photo Gallery" and "User Access Controls" requirements above.

Sketch out ideas for your website. Make sure your sketches meet the requirements for the website. Keep in mind the different types of pages you need. You should also consider how you will handle login/logout on your website. As always, keep your persona in mind when authoring the sketch.

After sketching our your ideas, author wireframes for your page. When finalizing the design details in your wireframe, make sure that it's a design that will work for your persona. Take the time to layout the page elements carefully and to make sure your tables and forms have proper labels.

Feel free to annotate your sketches and wireframes with notes about how the behavior or design changes based on certain conditions. For example, "delete image only visible when logged in..."

There is no minimum number of sketches or wireframes. Include as many as necessary to fully plan out your web site's design.

7.3. Plan your Database

Plan out your database schema. Take care to meet the "User Access Controls" and "Well Organized Database" requirements above. Think about the following questions.

- How many tables will you have?
- How will you structure your tables to eliminate duplication of data (not including foreign keys).
- Are your tables about "one thing"?
- Are you using foreign keys to organize your tables' relationships?
- What fields will you have in your tables?
- What type will each field be?

- What constraints will your fields have? (primary key, not null, auto increment, etc.)

In the **design-plan.md** file, describe your database plan. You may do this anyway you like. We just want to see that you've thoroughly thought through the structure of your database.

Take the time to carefully plan your database. The extra care you put into planning your database will save you time in later milestones with less complex SQL queries. Trust me, poorly planned databases can lead to some pretty impossible SQL queries.

You may feel confused for some of this. You may feel like you don't understand how to apply the *organizing a database guidelines* we discussed in class. You may feel like you want more examples. This is all normal. The only way to learn this is to try and do it for yourself. I want you to try and think through the guidelines on your own before we provide more examples, etc. You will understand this much better if you work through your frustrations and misconceptions first.

7.4. Plan your Database Queries

Think about the types of SQL queries you need to implement your design. Take care to meet the "Photo Gallery" and "User Access Controls" requirements above. Describe your plan for these queries. You may use natural language, pseudocode, or the SQL queries themselves.

You may feel totally lost here. You may feel like there are so many requirements and you don't know where to begin. This is normal. It's difficult learning how to take the tools from your *web programmer's toolbox* and use those to implement something new. Take it one step at a time. Think about how you might write a query for each of the requirements above. After you go through this process, you'll starting feeling a lot more confident about this!

7.5. Structure and Pseudocode

Plan out the structure of your web site. What top-level PHP pages will you have? (i.e. gallery.php, login.php, etc.) What PHP includes will you have?

Tip: Do not have a PHP page for each table in your database. You'll often discover that the way you present information to your user (persona) is different than the way you store that data in the database.

Plan out your pseudocode for each page. Each page's pseudocode should indicate where your planned database queries will execute. They should also take care to *filter input, and escape output*. Make sure your pseudocode fulfills the requirements for "Photo Gallery", "User Access Controls", and "Security" above.

It's okay if your plan doesn't exactly match your final code, although it should be close. The point of this exercise is to have you thoroughly think through how you will implement this before you start coding. It's a lot easier to starting coding once you've thought through how you will code it. I really want to help you develop this habit of planning first, and coding second. It really does save you time in the long run and you'll often be expected to do this for most programming jobs.

Tip: Feel free to utilize user-defined functions to make this process easier for you.

7.6. Database Creation & Seed Data

Create your database based on your plan and populate it with initial seed data. Take care to meet the "Seed Data" requirements above.

You are not permitted to create or populate your database with *DB Browser for SQLite*. For this project you will need to write the SQL queries to create the tables and insert the seed data in the **init/init.sql** file. You will need to look up the reference documentation for creating tables in SQL. We are not covering this in lecture or labs because we want you to have practice with SQL reference documentation.

We are using this script approach (init.sql) so that once Project 3 is over we can help you *deploy* your website to an actual live production server. Recall that production servers do not usually support SQLite. Instead they use MySQL or PostgreSQL. In order to *deploy* your website you'll need to have a SQL script ready in order to create the database on actual web server.

You will also need to upload seed image files for your seed data. Follow the requirements in "Seed Data" above.

IMPORTANT! Make sure you list your usernames and passwords in **design-plan.md**. We'll need these to test your website in later milestones.

7.7. Skeleton Website

Create the file/folder structure for your web site. Create *skeleton* PHP files for each web page. Include any additional files (images, fonts, CSS, JavaScript, etc.) that you may need to get started. We do not expect any styling to be done (CSS can also have empty files), but ask you to link all files in advance.

Why Skeleton? In later milestones you'll actually begin coding. We just want you be ready to start coding as soon as M1 is done. To help with this we'd like you to get all your files ready to start coding. The *skeleton* files can be completely empty or they can have some initial *boilerplate* code. For example, see the **index.php**.

7.8. Grading - Feedback

This is a feedback milestone. Your grade will be based on whether you submitted a complete assignment. If it looks like you tried, even though there are some mistakes, you'll get full credit. If your submission is obviously incomplete, you'll get a 0. There is no partial credit, either you submitted the assignment or you didn't.

We'll provide feedback on your milestone 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). **This does not resolve you of the responsibility of meeting the project's requirements, even if we miss something.** Your feedback will show up as comments for the assignment in CMS. Please read it and use it to improve your learning and your project.

Note: Your Milestone 1 artifacts will be graded for points using the rubric below (not completion) in the final submission. Feel free to update and change these documents based on feedback. In fact, it's a good idea to do so. It will likely improve your grade.

7.9. Submission

See **submit-*m1*.md**.

Failure to follow the submission instructions **exactly** will result in a late submission (2 slip days or a 0). Late submissions will not receive feedback until a least one week after the deadline.

IMPORTANT!: If you submit late, do not *push* work for M2 until after the late deadline passes (48 hours after the deadline). If you do push work after you submit late we may mistake your M2 work for M1 work and take another slip day. You may commit all you want, just don't push.

8. Milestone 2

20 points

In Milestone 2, you'll begin to implement your design and plan from M1. Your entire web page should be authored in HTML and CSS using PHP with the PDO extension to query your database. You may optionally use JavaScript, however it is not required.

8.1. Implementation Requirements

- You do not need to implement anything related to tags yet.
- The user should be able to view all images in a gallery.
- The user should be able to view a single image and it's details (no tags required for M2).
- The user should be able to upload an image.
- Login and logout should be fully working.
- You should implement the user access controls that you can (ignore the tags stuff for M2). (i.e. You can't upload an image if you aren't logged in)

8.2. Connecting to the Database

Since we're no longer using *DB Browser for SQLite*, you'll need handle creating your database differently to support deploying our website after you've finished this Project.

You will need to connect to your database using the `$db = open_or_init_sqlite_db('gallery.sqlite', 'init/init.sql');` function provided for you in **includes/init.php**. This function checks to see if your SQLite database exists, if it does it connects to it. If it does not exists (i.e. there is no gallery.sqlite file) then this function creates the database initializing it with your **init/init.sql** script. If you want to delete your database, just delete the **gallery.sqlite** file and the next time you refresh a page in your web browser, your database will be recreated.

8.3. Grading - Feedback

This is a feedback milestone. Your grade will be based on whether you submitted a complete assignment. If it looks like you tried, even though there are some mistakes, you'll get full credit. If your submission is obviously incomplete, you'll get a 0. There is no partial credit, either you submitted the assignment or you didn't.

We'll provide feedback on your milestone 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). **This does not resolve you of the responsibility of meeting the project's requirements, even if we miss something.** Your feedback will show up as comments for the assignment in CMS. Please read it and use it to improve your learning and your project.

Note: Your Milestone 1 artifacts will be graded for points using the rubric below (not completion) in the final submission. Feel free to update and change these documents based on feedback. In fact, it's a good idea to do so. It

will likely improve your grade.

8.4. Submission

See **submit-*m2*.md**.

Failure to follow the submission instructions **exactly** will result in a late submission (2 slip days or a 0). Late submissions will not receive feedback until a least one week after the deadline.

IMPORTANT!: If you submit late, do not *push* work for the final milestone until after the late deadline passes (48 hours after the deadline). If you do push work after you submit late we may mistake your final work for M2 work and take another slip day. You may commit all you want, just don't push.

9. Final Milestone

200 points.

You will finish coding the website that you planned in Milestone 1 and started implementing in Milestone 2. You will need to implement all support for tags. You should now implement everything necessary to meet the full requirements of the assignment.

Tip: Before you submit, re-read this document and check that you've meet this assignment's requirements.

9.1. Grading & Submission

The final milestone is graded via a rubric.

See **submit-*final*.md** for submission instructions. Failure to follow the submission instructions **exactly** will result in a late submission (2 slip days or a 0).

Tip: Before submission, re-read this document and rubric and check that you've met the requirements.

10. Rubric

200 points.

I reserve the right to change this, but this should be close to the final version.

10.1. Planning (20%)

- Complete and thoughtful Design & Planning document.

10.1.1. Persona

- Selected persona is specified.
- Rationale for persona is thoughtful and justified.

10.1.2. Sketches & Wireframes

- Sufficient sketches included.
- Sketches show idea variation.
- Sufficient and complete wireframes included.
- Wireframes are polished and give a good sense of the final web site design.
- Sketches and wireframes are informed by a persona and follow visual design principles.

10.1.3. Database Schema

- Schema is thoughtful and complete.
- Schema generally follows guidelines presented in lecture where reasonable/applicable.
  - Data organized for how it's used.
  - Schema planned in advance.
  - Every table is about *one* thing.
  - Schema does not duplicate data.
  - Uses web framework convention of surrogate *id* field for primary key.
  - Uses tables for storing data or relations between data.
  - Foreign keys used to organize relationships between tables.
- Schema allows for a many-to-many relationship.
- Schema utilizes proper types and constraints.
- Schema does not include any hard-coded data (not including seed data).
- Schema does not limit the amount of data.
  - There is no limit to the number of tags.
  - There is no limit to the number of images.
  - There is no limit to the number of tags for an image.
  - There is no limit to the number of users.

10.1.4. Implementation Planning & Pseudocode

- File structure is planned and reasonable.
- Includes are planned and reasonable.
- Pseudocode is thoughtful and reasonably complete.
- Plans are sufficient for implementing the full web site.

10.1.5. Content

- Seed data/content is functional (SQL queries are valid and seed images exist in the proper location).
- Seed data includes at a minimum:
  - 2 user accounts.
  - 10 images.
  - 5 tags.
  - 8 images have at least 1 tag.
  - 1 image must have at least 3 tags.
  - At least 3 images need multiple tags.
- Usernames and passwords included in **design-plan.md**

10.2. Design (20%)

- Design and navigation is appropriate for persona.

10.2.1. Design

- The design is pleasant and follows visual design principles. Design uses appropriate color, typography, layout, and positioning.
- Content is legible, easy to read and understand.
- Pleasant to look at; Creative and interesting.
- User access control design is appropriate for persona.

10.2.2. Navigation

- Site has functional navigation.
- Is the site navigation consistent and easy to use?
- Navigation is easy to follow and indicates current page: menu should be consistent on every page and indicate in some way what page is currently being viewed. No dangling pages (user shouldn't need to press the back button).

10.3. Implementation (50%)

- Implementation follows design and plan.
- Website is implemented in PHP.
- Website is under 10MB (excludes *design-plan* or *.git* folders).
- Website renders correctly in Firefox and Chrome (minor differences acceptable).

10.3.1. Database

- Database is created using valid **init/init.sql** file.
- **.sqlite** or **.db** files are not checked into version control. (i.e. Used proper .gitignore, or proper staging).
- The web site connects to a SQLite database using PHP's PDO extension.
  - -10 pts. if PDO is not used.
- Database follows your plan.
- All data is stored in database. (Hard-coded arrays are not acceptable).
- All data is queried using SQL. No "querying" using PHP loops and conditionals.

10.3.2. Gallery

- All images can be viewed at once.
- All tags can be viewed at once.
- All images for a tag can be viewed at once.
- A single image can be viewed with its tags at once.
- New images can be uploaded.
- Images can be deleted.

- Tags should be unique. You cannot have duplicates of the same tag.
- Images should be able to be tagged.
- Tags should be able to be removed from images.
- No hard-coded pages/content. You should render any content in your database dynamically. (i.e. uses query string parameters)
- All content that is modified by a user is properly inserted into the database or placed on disk.

10.3.3. Uploads

- Upload is functional and properly implemented.
- All file uploads are placed in a sub-folder under the **uploads** folder.
  - All sub-folders should have the same name as their corresponding database table.
  - All files stored in each sub-folder are required to be named as the corresponding database record.
    - -5 pts. if files stored with the *uploaded file name* instead of DB primary key.
  - Uploaded images (except seed images) are not committed to Git repository. (Proper .gitignore included)

10.3.4. User Access Controls

- Fully complete and secure login/logout implemented.
  - Sessions implemented using unique values written to DB and cookie.
  - Upon login, create a session and store it in the database. Send the session ID as a session cookie to the user.
  - Upon logout, destroy the session in the database and reset your session cookie.
  - -20pts. if PHP's sessions are used (i.e. `session_start()`)
- Complete and bug-free access controls
  - Only a logged in user may upload an image to the gallery.
  - Only the image creator (uploader) may delete that image.
  - Both, anonymous users (not logged in) and logged in users may tag images.
  - Only the user who uploaded an image may delete tags from that image.

10.3.5. Security

- Web page is secure from injection based attacks.
  - All input is properly and appropriately filtered.
  - All output is escaped for the appropriate context.
  - No SQL injection vulnerabilities.
- Proper Login/Logout security is implemented.
  - Passwords are hashed.
  - Sessions are (semi-secure) unique IDs/hashes.
  - SSL (HTTPS) is NOT required.

10.4. Coding Standards (10%)

- All code should be easy to read and be understood by anyone.
  - Use functions and includes to help organize your code.
  - Make use of comments to explain things that aren't obvious.
  - Name your functions and variables appropriately.
  - Your code should be indented properly so that it's easy to see which lines belong to each element (HTML/CSS) or function (PHP/JavaScript).
- The code follows the standards, conventions, and expectations of this class.
  - All HTML & CSS output validates (http://validator.w3.org).
  - Are the HTML, CSS, PHP, and JavaScript (if any) well-formatted, commented, and readable?
  - Lines are indented correctly.
  - Variable and function names are human-readable.
  - External styling via CSS. **No inline or internal styling**
  - All files are well organized (styles folder, images folder, no redundant code).
  - Website is free of PHP and SQL errors. **Warnings are acceptable if they are minimal.** (a few per page)
  - External resources (images) have appropriate credits.
- No external code or libraries used. All code is your own work.
  - Lecture/Lab code is not copied. (exception: `open_or_init_sqlite_db()`)
  - Login/Logout may be similar to lecture/lab but must be obvious that student did not copy it.