

**Web Programming and Database Development**

**Final Project**

Jesse Valley Portfolio Website

CWEB2121, and CWEB2111 Fall 2017

Instructors: Christopher Fulton, and Amalan Pulendran

***Project Member***

**Jesse Valley, jessevalley@gmail.com, Developer and Creator**

**Approved By:**

**Paula Merns**

**Christopher Fulton**

**Amalan Pulendran**

December 21, 2017

Contents

[**Web Programming and Database Development** 1](#_Toc501549556)

[**Final Project** 1](#_Toc501549557)

[1. Project Overview, Introduction and Requirements Specification 4](#_Toc501549558)

[1.1. Introduction 4](#_Toc501549559)

[1.2. Project Overview 4](#_Toc501549560)

[1.3. Project Scope 5](#_Toc501549561)

[1.4. Project Requirements 6](#_Toc501549562)

[1.5. Acronyms and Abbreviations used in documentation 7](#_Toc501549563)

[2. Analysis 7](#_Toc501549564)

[2.1. Description of Problem 7](#_Toc501549565)

[2.2. Technologies Used 8](#_Toc501549566)

[2.3. System Architecture 8](#_Toc501549567)

[2.4. Business Rules and Descriptions 9](#_Toc501549568)

[2.4.1. Business Rules 9](#_Toc501549569)

[2.4.2 Entity Descriptions 9](#_Toc501549570)

[2.4.3 Assumption 10](#_Toc501549571)

[2.5. Sample Data from Data Planning 10](#_Toc501549572)

[3. Design 12](#_Toc501549573)

[3.1. Architecture design 12](#_Toc501549574)

[3.2. Interface design 12](#_Toc501549575)

[3.3. Source Code 15](#_Toc501549576)

[3.3.1. Password-Protected Gallery 16](#_Toc501549577)

[3.3.2. Logged-in Enabled Gallery 16](#_Toc501549578)

[3.3.3 Password-Protected and Logged-in Enabled Gallery 16](#_Toc501549579)

[3.3.4 Inactive or Banned User Account 16](#_Toc501549580)

[3.3.5 PHP Modules 16](#_Toc501549581)

[3.3.6 JavaScript/JQuery 27](#_Toc501549582)

[3.3.7 HTML5/CSS3 28](#_Toc501549583)

[3.4. Data Model and Storage 31](#_Toc501549584)

[3.4.1 SQL Statements 31](#_Toc501549585)

[3.4.2 ER Diagram 34](#_Toc501549586)

[3.5. Security Considerations 34](#_Toc501549587)

[4. Implementation 35](#_Toc501549588)

[4.1. Mode of Implementation 35](#_Toc501549589)

[4.2. Description of major components for implementation 35](#_Toc501549590)

[4.2.1. Authentication (authenticate.php) 35](#_Toc501549591)

[4.2.2. Validate Gallery Password (valgpass.php) 35](#_Toc501549592)

[4.2.3. Sign Up, Create Account, and Update Account 36](#_Toc501549593)

[5. Testing and On-going Maintenance 36](#_Toc501549594)

[5.1. Overview of Site Test Plan 36](#_Toc501549595)

[5.2. Methods used for testing applications 36](#_Toc501549596)

[5.2.1. Interactive Tests 36](#_Toc501549597)

[5.2.2. User Tests 36](#_Toc501549598)

[5.2.3. External Validations 37](#_Toc501549599)

[5.3. Methods used for maintaining application 37](#_Toc501549600)

[5.3.1. Overview of Maintenance Plan 37](#_Toc501549601)

[5.3.2. Future Improvements 37](#_Toc501549602)

[Appendices 38](#_Toc501549603)

[References 38](#_Toc501549604)

[Versions 38](#_Toc501549605)

# Project Overview, Introduction and Requirements Specification

## Introduction

Hello! Welcome to the Final Documentation for my Final Project in Dunwoody’s curriculum for Web Programming and Database Development. Two main classes were the impetus for this project, which is a portfolio website for my artwork, poetry, and programming projects. Those classes were Database Systems (taught by Christopher Fulton) and Web Publishing (taught by Amalan Pulendran).

All of the work and coding was completed by me, Jesse Charles Valley, with guidance and help from both Mr. Fulton and Mr. Pulendran. The website is located on the worldwide web at <http://jessecvalley.com/>.

## Project Overview

Ultimately, the website will serve as a combination art gallery and portfolio site; it will promote me and help me get a job (both as a computer programmer and a photographer).

One of the primary things that I learned during this project is that modern web programming and database development are truly iterative processes, and there is always room for updates and improvements. The main scope and objectives for my portfolio website are essentially unchanged, but some of the specific modules and capabilities have changed during the semester. There are improvements that I still intend to implement, which I will discuss in section five.

The website will essentially be a portfolio and resume site all in one. It will be a location to present examples of the art I do (photography, painting, pottery, poetry, and programming). Since I do photography as a business, I would like to highlight some sample work and let prospective clients know that I am available for hire. I would also like to host galleries of my clients’ pictures that they can access and review (which could be protected by passwords). Also, it will allow them to contact me if they are interested in buying any of my art.

As far as the website is a part of a resume, I will have either a subdomain or subfolder wherein to point prospective employers or recruiters. This is something that I will probably set up to be protected by a password—the idea being that only people interested in pursuing me as a programmer would get access to the more privileged material. It may link up with samples of my programming on the portfolio site, but with extra access to materials such as source code and explanations.

Finally, I’m currently planning on linking the website to my blog for a couple reasons. It will provide access to a lot of original content I’ve previously created (which helps establish my credibility in the mind of prospective clients), and it would also be a place I could add additional content easily. Right now I am not intending to create my own content management system (CMS) or template, but I want to at least present my previous blog posts.

This website should be an accurate representation of my personal brand. It should also accomplish the following:

* Get me an interview and/or programming job
* Get me hired as a photographer
* Market me as an artist
* Facilitate portraiture sales
* Be responsively designed
* Look clean and modern
* Demonstrate unique interaction with the users (demonstrate some generally untapped interactivity features)
* Password-protection and restricted access for customers without an account
* Easy navigation
* Large, high-quality images

Goals as related to the end user are:

* Users should be able to navigate to the various sections and galleries easily and not get lost
* Users should be able to create and update their account
* Users should be able to view the public galleries without logging in and private galleries if they are logged in and authenticated
* Users should be able to find my contact info easily and contact me
* Users should be able to use the website just as easily on a cell phone as a computer
* Users should enjoy great visual samples of the artwork
* Administrator should be able to view all galleries by default
* Users should be able to discover and enjoy my social media accounts

## Project Scope

I will create some sort of gallery functionality for smaller galleries, and the rest of the website, but not a high capacity gallery with shopping cart integration. Also, I will eventually like to connect to some sort of payment portal (most likely Square or PayPal), but I am not going to create something like that from scratch.

Most of the data requirements are going to be focused on the galleries for my portfolio. I want people to be able to view the different galleries.

I also intend to create a login and log out functionality for my clients. Once logged in, they will be able to see private galleries. I want to get info from them to personalize their experience on the website, such as doing something as simple as a custom greeting. I would also like to wish them a happy birthday or anniversary if that pertains to them. Here are actions that users should be able to take:

* The end-user will be able to create an account.
* The end-user will be able to login to the account they’ve created.
* The end-user will be able to view galleries that are private and linked to their account. They will also be able to view public galleries whether they are logged in or not.
* The end-user will be able to browse public galleries without creating an account.
* Some galleries will be password protected. The end-user will be required to know the correct password to see the gallery. This will enable people to see specific galleries without having to create an account.
* The end-user will be able to update their account info.
* The admin will be able to login to the admin account.
* The admin will be able to see all the galleries by default.
* The end-user will be able to contact me via the web or the contact page.

## Project Requirements

There are really three main audiences I’m targeting right now (which could theoretically grow if I get more well known in any of the individual disciplines). Here are their user stories and the scenarios that will be handled by the website.

1. Acquaintances (whether personal or via social media). These users will have found out about my artwork either through interacting with me or with someone that already knows me and my art. Their ages could be vast, but I would imagine younger children would not be included. These guests would not necessarily need to create a user account, but rather they would need to be able to easily navigate the website and enjoy the artwork. Ultimately, this could lead to them buying artwork or commissioning some sort of service.

2. Current and prospective photography clients and their families. This would mostly be adult women that hire me or are looking to hire a photographer for portraiture, but it certainly includes men and perhaps teenage children (the percentage is just much lower). When this audience includes current clients, it is important that their pictures are easy to find and share. The families of the client will be looking to mostly view the images. This audience will need to be able to log in and easily navigate to the images that they are interested in viewing. For prospective clients, they should be able to easily peruse sample galleries and easily contact me for hiring purposes.

3. Employers/Recruiters. This audience will most likely come to the website through some sort of marketing. It could be that I’m marketing my artwork to a gallery or collector; it may be a hiring manager that is researching me as a potential employee and reviewing my work, and it could be someone from which I’m seeking a commission. Again, information should be easy to read and find, and it should be professional in tone.

There are some main use cases that will be included in the project, which are as follows:

A. Case One: The actor is a recruiter or HR manager who is looking for a web programmer to hire. The ideal situation is that the recruiter has a special link to the subdomain or subfolder that is geared specifically to highlight my web programming portfolio. In all honesty, since the website will be an extension of my personal brand, I am not too concerned if they also peruse the rest of the website and social media links. Ultimately, these other things should give a clear and accurate portrayal of me; plus, since I developed the whole website, it will also demonstrate my programming abilities.

I will want to enable the recruiters to have access to other programs and source code, so I want to protect that access with a password. If a recruiter or potential employer unsuccessfully tries to access that material, they will be prompted to reach out to me via email so that I can give them the password. This will help ensure that the people getting access to those areas really need it, and are not just someone poking around. Finally, I want to have an easy method for them to contact me to set up an interview.

B. Case Two: The actor is an acquaintance that is interested in learning more about my artwork. The carousel slideshow will likely be the first page they encounter. The point of the site and the navigation should be evident and obvious. If they navigate to any of the galleries, I want them to be inundated with large, high-quality images. They should be able to contact me easily and buy something easily if desired. As long as they are able to find their way around the site easily and leisurely look through the galleries, it will be successful for them.

C. Case Three: The actor is a client that wants to look at their photos. The best scenario for this would be similar to the recruiter scenario, with the client having a link directly to their gallery. It will be essential for them to either create an account or log in to their account to actually see the gallery. As long as they can successfully navigate through their gallery and are able to glean the information needed to order their portraits, the site will be successful. After that, if they want to hang around and peruse the galleries, that is just fine.

D. Case Four: The actor is a prospective client. This user will likely land on the homepage (probably because they found me via Google) or specifically on the business page. In either scenario, the navigation should be clear and it should be easy for them to find the information they desire. Ultimately success will be that they contact me to schedule a session, or they bookmark the page and return when they need photos.

## Acronyms and Abbreviations used in documentation

1. CMS: Content Management System
2. LAMP: Linux, Apache, MySQL, PHP
3. PHP: Hypertext Preprocessor
4. HTML: Hypertext Markup Language
5. CSS: Cascading Style Sheets
6. SQL: Structured Query Language
7. UI/UX: User Interface/User Experience
8. PDO: PHP Data Objects
9. MVC: Model, View, Controller
10. IDE: integrated development environment
11. XAMPP: Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P)
12. JV: Jesse Valley

# Analysis

## Description of Problem

Ultimately, this website was made to overcome multiple problems. First, I wanted the website itself to demonstrate the skills and abilities I have learned over the course of the program to find employment. Second, I wanted the site to be an accurate representation of myself as a personal brand, which is strongly influenced by the UI/UX features, visual design and overall quality of the content. Third, I wanted the website to be a place where I can show the various art that I create, especially as I see programming to be technology mixed with art. And finally, I wanted to create a place where current and prospective photography clients can be updated on my work and view their images.

A tremendous amount of effort for this project really went into merging these four problems into one solution, and I feel very satisfied with the final project.

## Technologies Used

One of the exciting aspects of this project was in using standard, open source technologies to create a handsome, modern website. It is exciting to know that I have developed the skill set to produce such a dynamic website and that I do not have to pay someone else to make it.

Primarily, I used PHP mixed together with HTML and CSS. PHP was used to query and display results via HTML. When PHP interacted on the backend with MySQL, I was able to access specific data quickly and efficiently.

In order to type out the code, I primarily used Adobe Dreamweaver. Even though it is not a free IDE (integrated development environment), there are other options that could be used for PHP coding that are open source. In order to access the Apache Webserver on the Linux hosting machine, I needed to use the C-Panel webpage from the service provider, which is TMD hosting in this case. I was able to create and manipulate the database through both MySQL Workbench (remotely) and PHPMyAdmin (via the web portal).

Most of the time, I was relying on Apache’s XAMPP (Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P)) to serve, test and deploy the website locally. Not only did this let me test changes immediately, but it also let me work on the website without being connected to the internet. For the most part this worked great, however I did have a few issues when exporting my test database to merge it with the production database, because it changed the casing of my table names from uppercase to lowercase, which affects SQL queries. I also had to remove some extra PHP “echos” I created while testing some PHP modules in order for them to work on the actual website. I’m not totally sure why this happened, but I am fairly certain it relates to some sort of configuration on either my host’s server on in XAMPP itself.

The only specific API that I used on this website was from Google Maps so I could show my location on the index page. This was fairly straightforward but has a nice effect.

In order to really create a responsive, dynamic website, I had to use JavaScript, JQuery, Bootstrap, the new features of HTML5, and customized CSS. Although these don’t necessarily seem like “technologies,” they were an integral part of the final result.

Additionally, I relied on Google Chrome’s Development Tools to really understand different breakpoints for media queries and to visualize the webpage on different devices.

## System Architecture

The overall architecture of the website follows the normal LAMP stack, using Linux, Apache2, MySQL and PHP (each of the initial letters spells out LAMP).

A Linux server hosts the website and is the physical storage for the database. The webpage is served via an Apache2 module on the server, running the PHP scripts to render the HTML. Data flows both to a client’s computer (where it is displayed and can be manipulated on their end with JavaScript and JQuery), and back to the server in response to their actions. The information and responses that they input are sent back to the server over the internet and processed on the server-side with PHP. MySQL is used to for all of the C.R.U.D. procedures on the database.

## Business Rules and Descriptions

### Business Rules

1. A USER can generate one GALLERY\_LIST, and a GALLERY\_LIST is generated by only one USER.
2. A GALLERY exists in one GALLERY\_LIST, and a GALLERY\_LIST has the existence of one or more GALLERIES.
3. A FILE is contained in one GALLERY, and a GALLERY can contain many FILES.
4. A FILE exists in one FAV\_FILE, and a FAV\_FILE has the existence of one or more FILES.

### 2.4.2 Entity Descriptions

USER: A USER represents an admin or registered user who can access web galleries on the Jesse Valley (JV) Portfolio site. Each registered user will have a username, password, first name, last name, middle name (optional), address, birthday, phone (optional), email, anniversary (optional), gallery list id, user id, account creation time, and other fields that track data such as if they are a current or past client, if they have given permission to receive text messages or an email newsletter, and if they are an active account or not. If the user type is an admin, he or she will be able to add any gallery to their gallery list, as well as be able to create new galleries.

GALLERY: A GALLERY represents one gallery on the JV site. A gallery will have a gallery id, gallery name, gallery description, password (optional), number of views, file list id, and a field that tracks whether or not someone must be a registered user to view the gallery (in other words, be logged in to the site). There can be multiple files in a gallery, but there will not be any duplicate files.

GALLERY\_LIST: A GALLERY\_LIST represents the list of galleries that are accessible to a specific user. A gallery list will have an id and a gallery id that connects it to a specific user. Numerous galleries can be attributed to one gallery list.

FILE: A FILE represents a file that will be displayed within a specific gallery on the JV site. A file will have a file id, a creator, a title, a description, a file type, a category, a price, a quantity available, and other fields to track if it is for sale, if it is downloadable, and if it is a user’s favorite or not. A specific file will only be displayed in one gallery.

FAV\_FILE: A FAVE\_FILE represents the list of files that are favorited by a user. A fav file list will have a user id and a file id that connects it to a specific user. Numerous files can be attributed to one file list.

### 2.4.3 Assumption

Can a FILE be in more than one GALLERY? No, it cannot.

## Sample Data from Data Planning

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User** | **Data Type** | **Description/Assumptions** | **Sample 1** | **Sample 2** | **Sample 3** |
| uName | NVARCHAR | Username: must be unique | jValley | mary\_jones | m\_jones |
| pWord | NVARCHAR | Password: at least 6 characters long, encrypted | \*\*\*\*\*\* | \*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\* |
| admin | BOOL | True for admin users; False otherwise | TRUE | FALSE | FALSE |
| fName | NVARCHAR | First name | Jesse | Mary | Mark |
| lName | NVARCHAR | Last name | Valley | Jones | Jones |
| mName | NVARCHAR | Middle name: Can accept NULL | Charles | Ann | NULL |
| birthday | DATE | User can enter or pick from calendar | 25-Feb-80 | 10-Oct-00 | 21-Jun-66 |
| street1 | NVARCHAR | Would like to verify via USPS | 2821 Georgia Ave S | 123 Main St. | 123 Main St. |
| street2 | NVARCHAR | Can accept NULL | NULL | #101 | #101 |
| city | NVARCHAR | Auto generate via zip code | St. Louis Park | Minneapolis | Minneapolis |
| state | CHAR(2) | 2 letter state, will be auto-generated via zip code | MN | MN | MN |
| zip | VARCHAR | Would like to verify via USPS; validate format | 55426 | 55402 | 55402 |
| phoneHome | VARCHAR | Can accept NULL; must be valid format | NULL | 612-111-1234 | NULL |
| phoneCell | VARCHAR | Can accept NULL; must be valid format | 612-382-2867 | 612-222-1234 | 612-333-1234 |
| getTexts | BOOL | True if they agree to texts; False otherwise (default) | TRUE | TRUE | FALSE |
| email | NVARCHAR | Required, must be validated | [valjesc@dunwoody.edu](mailto:valjesc@dunwoody.edu) | [mj@hotmail.com](mailto:mj@hotmail.com) | [mr\_jones@gmail.com](mailto:mr_jones@gmail.com) |
| newsletter | BOOL | True if they agree to be on the email list; False otherwise | TRUE | TRUE | TRUE |
| id | INT | Auto increment and unique; primary key | 1 | 2 | 3 |
| createTime | TIMESTAMP | Timestamp value when account was created | 1504797024 | 1504807024 | 1504797158 |
| active | BOOL | True if account is activated; False if suspended/blocked | TRUE | TRUE | TRUE |
| anniversary | DATE | Can accept NULL; required if past wedding client | 23-Apr-05 | NULL | 13-Jun-66 |
| client | BOOL | True if current or past client; False otherwise | FALSE | TRUE | TRUE |
|  |  |  |  |  |  |
| **Gallery** | **Data Type** | **Description/Assumptions** | **Sample 1** | **Sample 2** | **Sample 3** |
| g\_id | INT | Auto increment and unique; primary key | 1 | 2 | 3 |
| gName | NVARCHAR | Gallery Name | Photography | Paintings | Jones Family |
| gDesc | NVARCHAR | Discription of the Gallery | Fine Art Photos | Fine Art Paintings | Family Portraits |
| pw | NVARCHAR | Can accept NULL; main portfolio pages won't have a pw | NULL | NULL | jfam |
| loggedIn | BOOL | True if must be logged in to view; False otherwise | FALSE | FALSE | FALSE |
| numViews | INT | Increases with each unique session | 5366 | 85 | 3 |
|  |  |  |  |  |  |
| **File** | **Data Type** | **Description/Assumptions** | **Sample 1** | **Sample 2** | **Sample 3** |
| f\_id | INT | Auto increment and unique; primary key | 1 | 2 | 3 |
| creator | NVARCHAR | Generally will be myself | Jesse Valley | Jesse Valley | Jesse Valley |
| title | NVARCHAR | Title of the artwork. NULL changes to "Untitled" | Red Sunset | Untitled | Untitled |
| desc | NVARCHAR | Description of artwork | Photo of red sunset. | Abstract Painting | Family Photo |
| category | NVARCHAR | Category of artwork | Photograph | Painting | Photograph |
| forSale | BOOL | True if for sale | TRUE | FALSE | FALSE |
| price | INT | Price of object for sale (insurance value if NFS) | 10 | 2500 | 10 |
| dLoad | BOOL | Downloadable file (to be set by admin user) | FALSE | FALSE | FALSE |
| qty | INT | Quantity for sale | 1 | 0 | 0 |
|  |  |  |  |  |  |
| **Fav\_Files** | **Data Type** | **Description/Assumptions** | **Sample 1** | **Sample 2** | **Sample 3** |
| id | INT | Primary Key to user id | 1 | 1 | 2 |
| f\_id | INT | Primary Key to file id | 1 | 2 | 78 |
|  |  |  |  |  |  |
| **Gallery\_List** | **Data Type** | **Description/Assumptions** | **Sample 1** | **Sample 2** | **Sample 3** |
| id | INT | Primary Key to user id | 1 | 1 | 2 |
| g\_id | INT | Primary Key to gallery id | 1 | 2 | 5 |

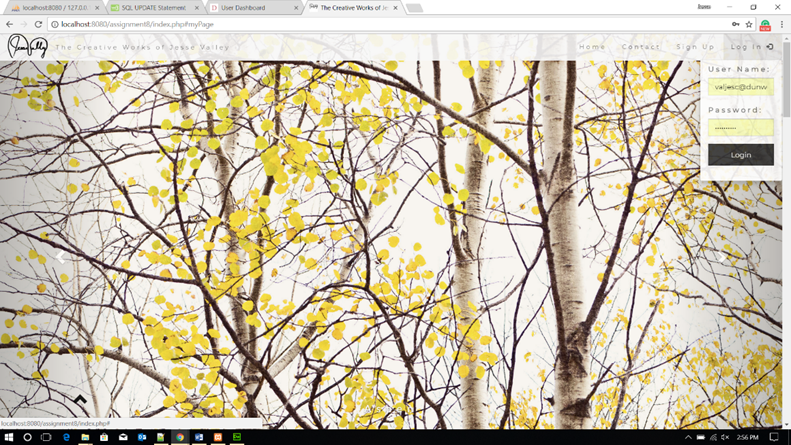
# Design

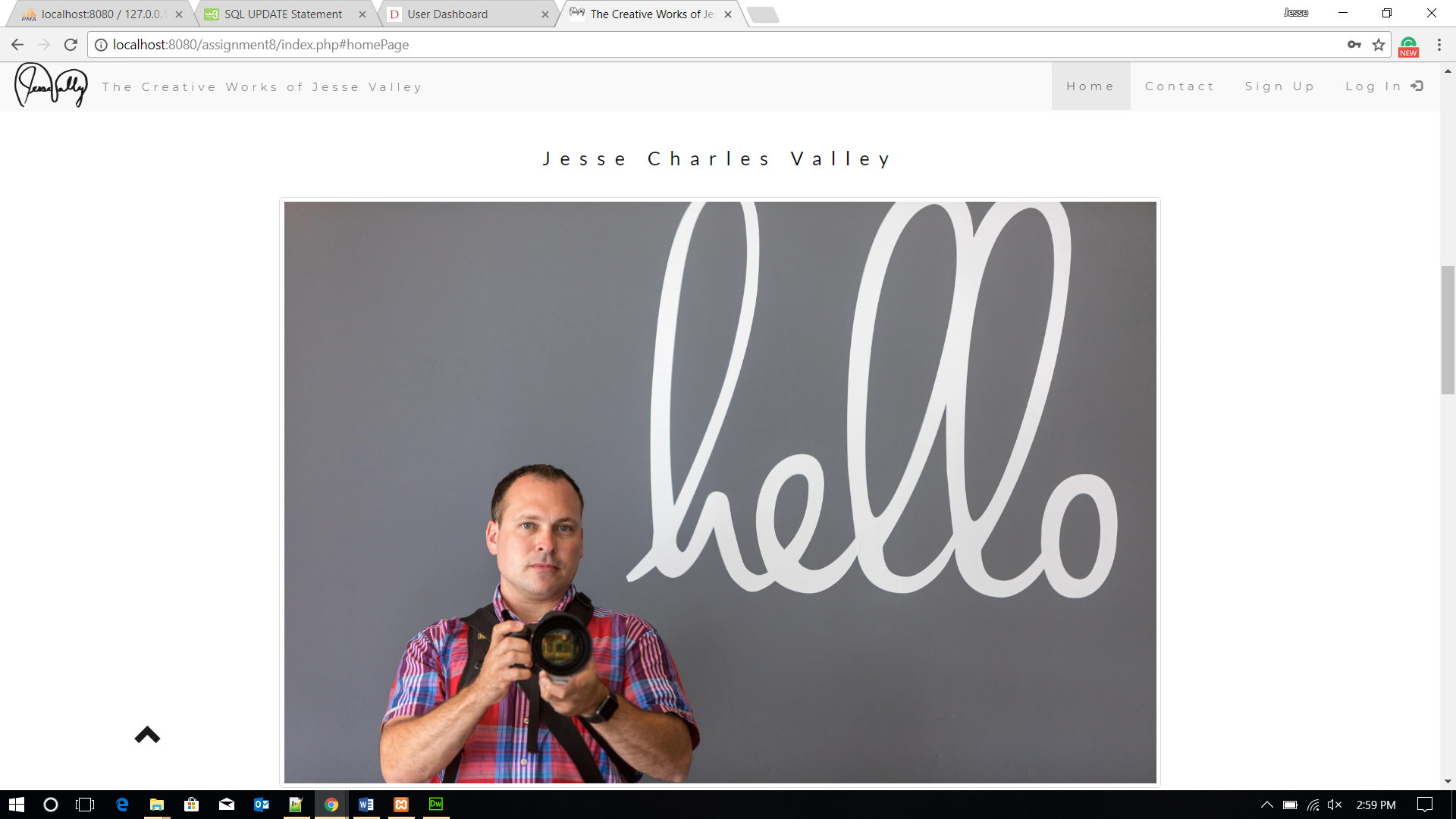
## Architecture design

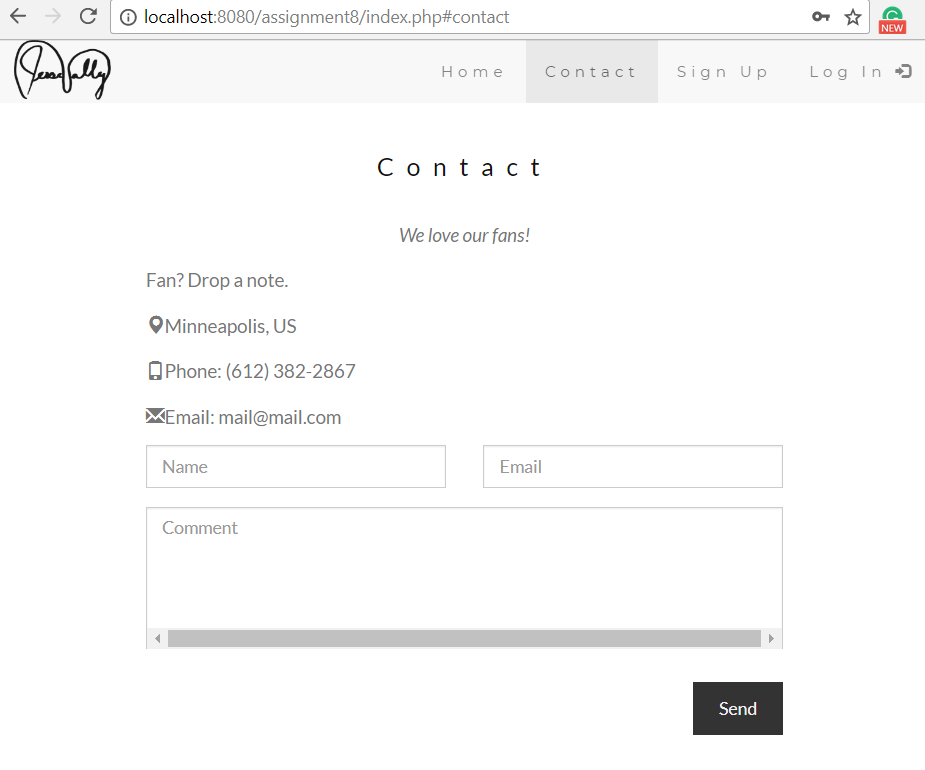
The directory structure consists of the main page “index.php” and all the gallery and php pages within the same main folder. There is an “img” folder and “css” folder that holds the various images and css files respectively. All of the files link to each other, and I opted to have each gallery become its own webpage. The main reason I did this is so I can easily link to certain galleries based on the target audience. For example, if I know a recruiter wants to see samples of my developing work, then I want to send them right to that gallery—they can still peruse the other galleries at their leisure, but I don’t want to hinder them, either. As the galleries increase in number, I may have to have more options for sub galleries and searching, but as the site currently exists, I prefer the cleanliness of the layout and design.

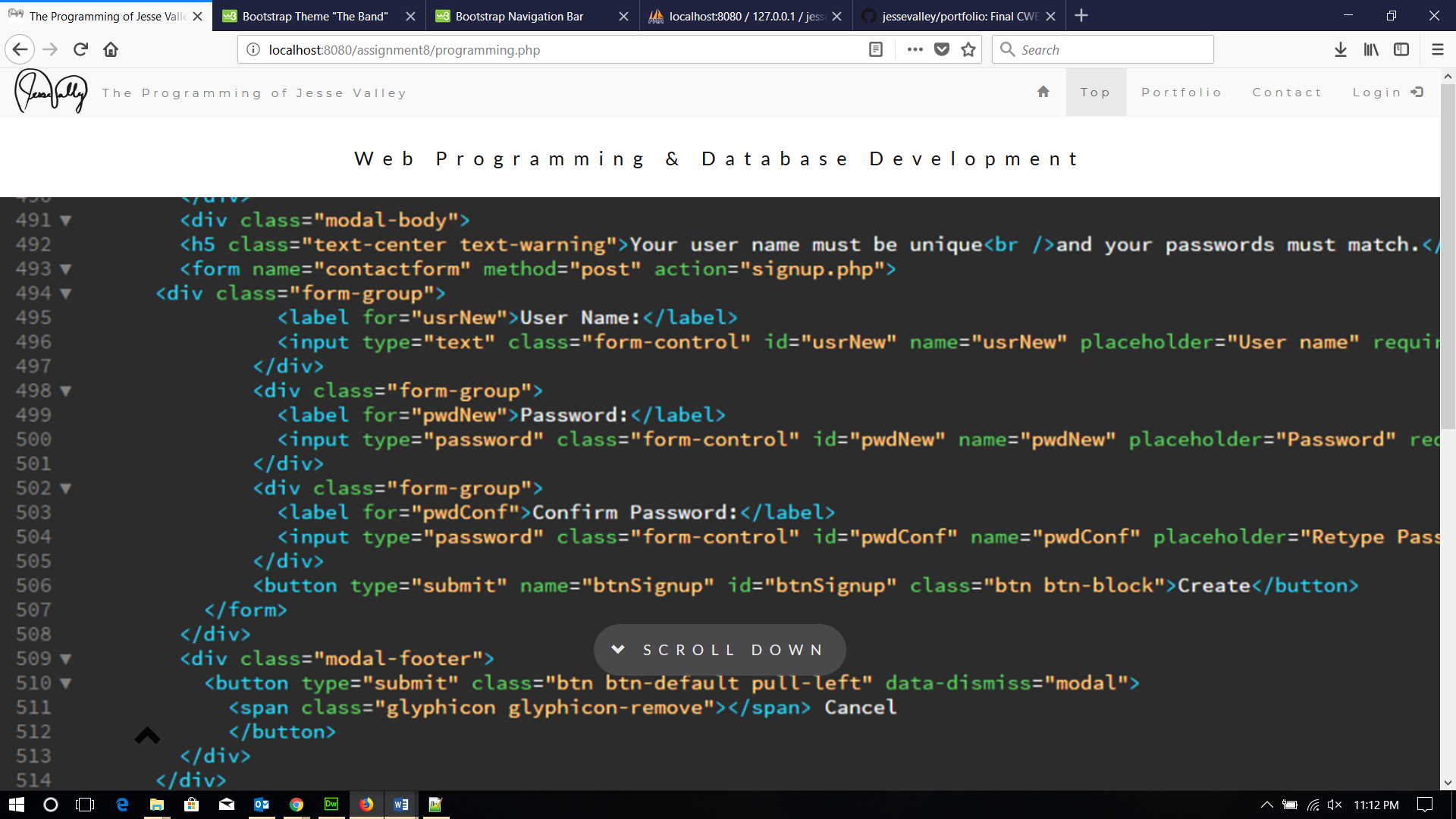
## Interface design

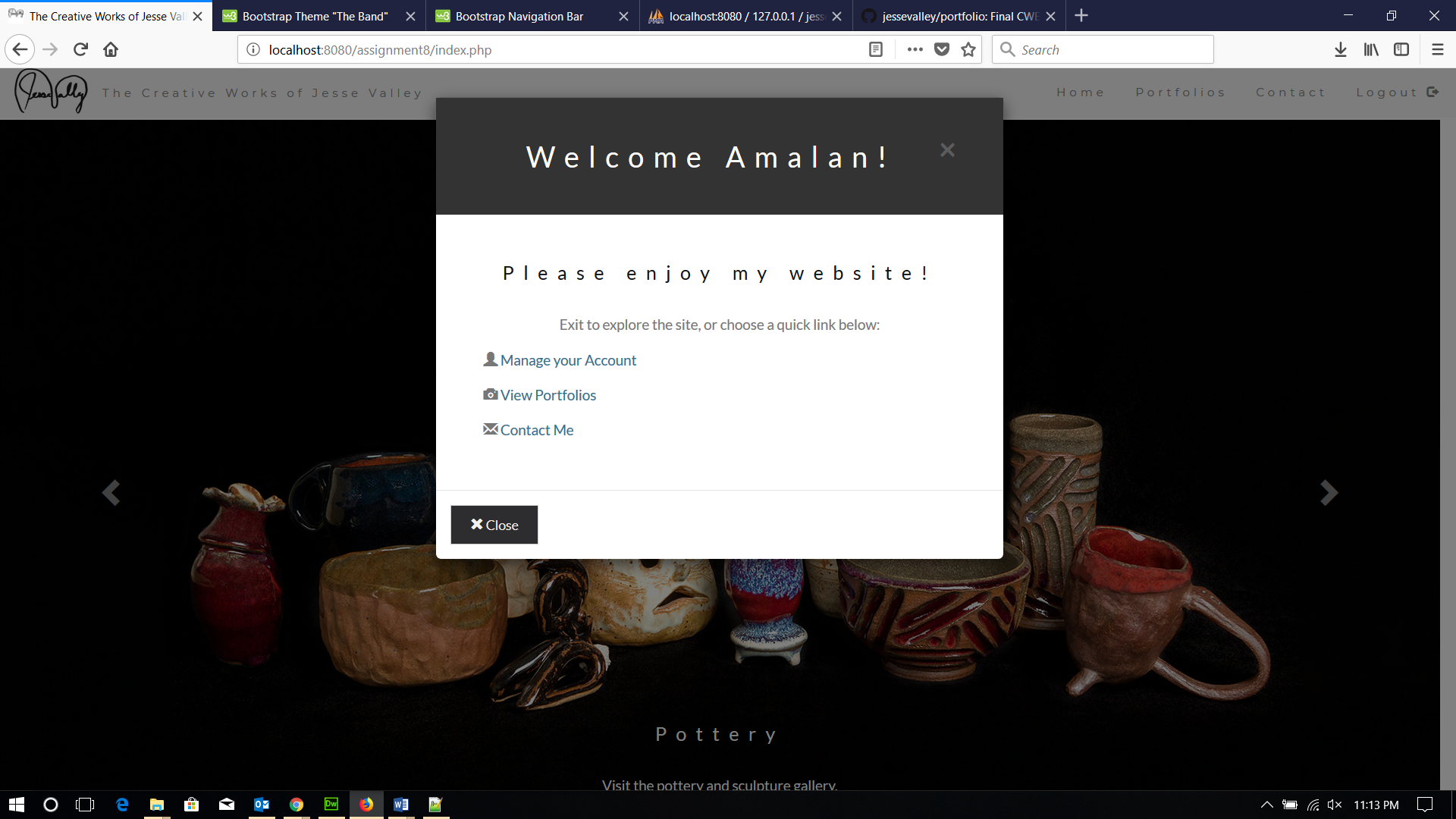
Currently, the overall design of the interface was established through multiple pages with the navigation and footers (and some of the content) copied and pasted inside. Ideally, I would want to create things like a “nav.php” and “footer.php” that I could include on the various pages. This way, if I made a change to the footer or header I would only need to do it one time. I did some initial mockups to get me started, which are included below.

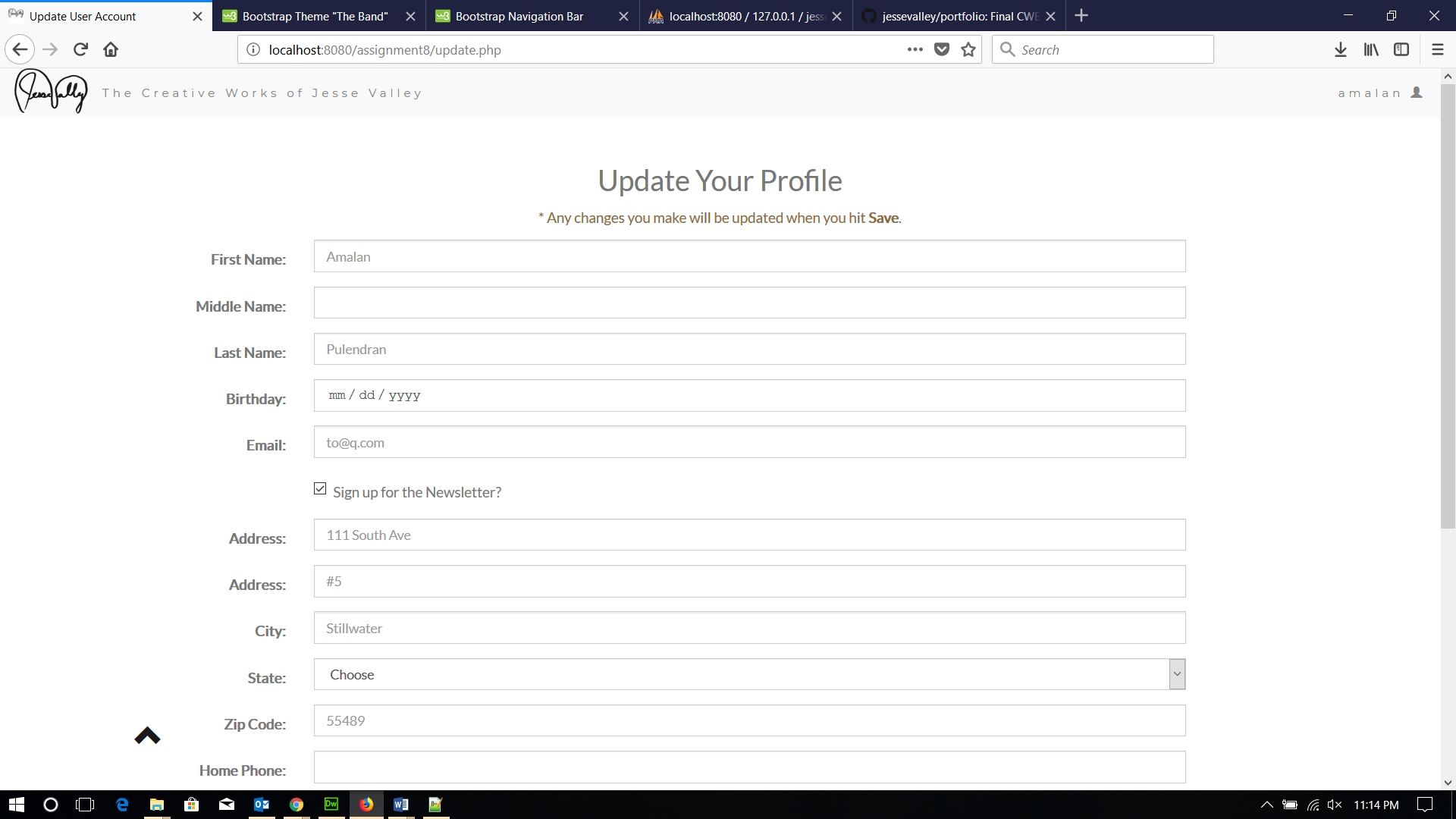












## Source Code

The entire project is on my GitHub account (<https://github.com/jessevalley/portfolio>) so that it can be checked over by my instructors and prospective employers. For the sake of demonstrating role-based authentication features, I have given special Boolean data values for specific galleries and users. Following is that info if you would like to test out those features. By default, I have set it up so that the Administrator accounts can look at any gallery as long as they are logged in. The credentials for the Administrator account is “admin” for the user name and “Password01” for the password.

### Password-Protected Gallery

The “Programming Portfolio” is password-protected. In order to see this gallery, type “pw” in the password field. Once you hit enter, you will be able to view that gallery until a new user session is initiated (I do not think it is good for the user experience if they have to type in the password every time they visit that gallery during the same session).

### Logged-in Enabled Gallery

The “Poetry Portfolio” is only available when the user named “lav” is logged in (the password for that account is “pw”). I created this protection so that I could have galleries that are only available to certain users (so they’re still secure), but that does not make them type a password for the applicable galleries.

### 3.3.3 Password-Protected and Logged-in Enabled Gallery

The “Portraiture Portfolio” is only available if the user named “jesse” is logged in (the password for the login is “pw”) and if that user types in the correct password (“pw” in this case). This simply offers more flexibility and protection, which I could see some clients appreciating in some circumstances. Let’s say that a wife got pictures taken for a gift, so she doesn’t want her husband (who may use the same account to look at their other pictures) to see them, so she can have them also password-protected.

### 3.3.4 Inactive or Banned User Account

I don’t foresee using this a lot, but I still wanted the functionality to keep an account in the database but to revoke the user’s privileges. To see how this works, login with the username of “vi” and the password “ow” (they get redirected to the index screen with a custom warning).

### 3.3.5 PHP Modules

A lot of the PHP deals with accessing the database, providing validation and creating a more dynamic website. I also created a contact form that works with PHP to send me an email from website user. Following are some of the main modules.

#### Database Query via PDO

<?php

session\_start();

$role = $\_SESSION['admin'];

if($role!=1 && $role!=2){

header('Location: index.php?err=2');

}

include("database\_config.php");

$uname = $\_SESSION['uname'];

$query22 = $dbh->prepare("SELECT \* FROM USER WHERE UNAME='$uname';");

$query22->execute();

$r = $query22->fetch(PDO::FETCH\_ASSOC);

?>

#### Update User Data

<?php

require 'database\_config.php';

session\_start();

function test\_input( $data ) {

$data = trim( $data );

$data = htmlspecialchars( $data );

return $data;

}

if($\_POST){

$username = $\_SESSION['uname'];

$md5\_pass = $\_SESSION['pword'];

$fname = test\_input( $\_POST[ "fname" ] );

if (empty($\_POST['mname'])){

$mname = "";

}else{

$mname = test\_input($\_POST['mname']);

}

$lname = test\_input( $\_POST[ "lname" ] );

$bday = test\_input($\_POST['bday']);

$street1 = test\_input( $\_POST[ "street1" ] );

if ( empty( $\_POST[ "street2" ] ) ) {

$street2 = "";

}else{

$street2 = test\_input( $\_POST[ "street2" ] );

}

$city = test\_input( $\_POST[ "city" ] );

$state = $\_POST[ "state" ];

$zip = test\_input( $\_POST[ "zip" ] );

if(empty($\_POST['phone\_home'])){

$phone\_home = "";

}else{

$phone\_home = test\_input( $\_POST[ "phone\_home" ] );

}

if(empty($\_POST['phone\_cell'])){

$phone\_cell = "";

}else{

$phone\_cell = test\_input( $\_POST[ "phone\_cell" ] );

}

$email = test\_input( $\_POST[ "email" ] );

if(empty($\_POST['anniv'])){

$anniv = NULL;

}else{

$anniv = test\_input($\_POST['anniv']);

}

$texts = $\_POST['texts'];

$news = $\_POST['news'];

if(empty($\_POST['client'])){

$client = 0;

} else {

$client = $\_POST['client'];

}

try{

$query = $dbh->prepare("INSERT INTO user (UNAME, PWORD, FNAME, LNAME, MNAME, BDAY, STREET1, STREET2, CITY, STATE, ZIP, PHONE\_HOME, PHONE\_CELL, TEXTS, EMAIL, NEWS, ANNIV, CLIENT)

VALUES ('$username', '$md5\_pass', '$fname', '$lname', '$mname', '$bday', '$street1', '$street2', '$city', '$state', '$zip', '$phone\_home', '$phone\_cell', $texts, '$email', $news, '$anniv', $client);");

$query->execute();

}catch(PDOException $e){

echo $e->getMessage();

}

include 'authenticate.php';

}

?>

#### Database Configuration

<?php

$host = 'localhost';

$user = 'root';

$pass = '';

$database = 'jessecva\_portfolio';

$dbh = new PDO('mysql:host='.$host.';

dbname='.$database, $user, $pass);

$dbh->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

if(!$dbh){

echo "unable to connect to database";

} ?>

#### PHP Email Form

<?php

if(isset($\_POST['email'])) {

// EDIT THE 2 LINES BELOW AS REQUIRED

$email\_to = "jessevalley@me.com";

$email\_subject = "From the interwebs";

function died($error) {

echo '<meta name="viewport" content="width=device-width, initial-scale=1">';

echo '<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">';

echo '<link href="https://fonts.googleapis.com/css?family=Lato" rel="stylesheet" type="text/css">';

echo '<link href="https://fonts.googleapis.com/css?family=Montserrat" rel="stylesheet" type="text/css">';

echo '<style>';

echo '.btn {

padding: 10px 20px;

background-color: #2d2d30;

color: #f1f1f1;

border-radius: 0;

transition: .2s;

}

.btn:hover, .btn:focus {

border: 1px solid #2d2d30;

background-color: #fff;

color: #2d2d30;

}';

echo '</style>';

echo '<div class="signin-form">';

echo '<div class="container">';

echo "<div class='alert alert-danger'>";

echo "<button class='close' data-dismiss='alert'>&times;</button>";

echo '<strong>Sorry!</strong> There were errors with the form you submitted.<br />';

echo $error."<br />";

echo 'Please go back and fix these errors.<br />';

echo '</div>';

echo '<a href="index.php"><button class="btn btn-primary" id="back">';

echo '<span class="glyphicon glyphicon-backward"></span> &nbsp; Back to Home';

echo '</button></a></div></div>';

die();

}

// validation expected data exists

if(!isset($\_POST['first\_name']) ||

!isset($\_POST['last\_name']) ||

!isset($\_POST['email']) ||

!isset($\_POST['comments'])) {

died('I am sorry, but there appears to be a problem with the form you submitted.');

}

$first\_name = $\_POST['first\_name']; // required

$last\_name = $\_POST['last\_name']; // required

$email\_from = $\_POST['email']; // required

$comments = $\_POST['comments']; // required

$error\_message = "";

$email\_exp = '/^[A-Za-z0-9.\_%-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$/';

if(!preg\_match($email\_exp,$email\_from)) {

$error\_message .= 'The Email Address you entered does not appear to be valid.<br />';

}

$string\_exp = "/^[A-Za-z .'-]+$/";

if(!preg\_match($string\_exp,$first\_name)) {

$error\_message .= 'The First Name you entered does not appear to be valid.<br />';

}

if(!preg\_match($string\_exp,$last\_name)) {

$error\_message .= 'The Last Name you entered does not appear to be valid.<br />';

}

if(strlen($comments) < 2) {

$error\_message .= 'The Comments you entered do not appear to be valid.<br />';

}

if(strlen($error\_message) > 0) {

died($error\_message);

}

$email\_message = "Form details below.\n\n";

function clean\_string($string) {

$bad = array("content-type","bcc:","to:","cc:","href");

return str\_replace($bad,"",$string);

}

$email\_message .= "First Name: ".clean\_string($first\_name)."\n";

$email\_message .= "Last Name: ".clean\_string($last\_name)."\n";

$email\_message .= "Email: ".clean\_string($email\_from)."\n";

$email\_message .= "Comments: ".clean\_string($comments)."\n";

// create email headers

$headers = 'From: '.$email\_from."\r\n".

'Reply-To: '.$email\_from."\r\n" .

'X-Mailer: PHP/' . phpversion();

@mail($email\_to, $email\_subject, $email\_message, $headers);

?>

<!doctype html>

<html lang="en">

<head>

<title>Email Success</title>

<meta charset="utf-8">

<meta name="description" content="Creative Works of Jesse Valley">

<meta name="keywords" content="Photography, Paintings, Poetry, Pottery, Art, Programming, Portfolio, Jesse Valley">

<meta name="author" content="Jesse Valley">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<link href="https://fonts.googleapis.com/css?family=Lato" rel="stylesheet" type="text/css">

<link href="https://fonts.googleapis.com/css?family=Montserrat" rel="stylesheet" type="text/css">

<link rel="stylesheet" href="http://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.4.0/css/font-awesome.min.css">

<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">

<link rel="icon" href="favicon.ico" type="image/x-icon">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

<script type="text/javascript">

$('document').ready(function()

{

window.setTimeout(function(){

window.location.href = "index.php";

}, 6000);

$("#back").click(function(){

window.location.href = "index.php";

});

});

</script>

</head>

<body id="myPage" data-spy="scroll" data-target=".navbar" data-offset="50">

<nav class="navbar navbar-default navbar-fixed-top">

<div class="container-fluid">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-left" href="#myPage"><img src="img/sig.png" class="img-responsive" style="height: 48px;" alt="Jesse Valley's Autograph" title="Home"></a>

<p class="navbar-text" id="navtext">The Creative Works of Jesse Valley</p>

</div>

<div class="collapse navbar-collapse" id="myNavbar">

<ul class="nav navbar-nav navbar-right">

<?php if(($\_SESSION['admin']) == 2){ ?>

<li><a href="#homePage">Home</a></li>

<li><a href="#tour">Portfolio</a></li>

<li id="portal"><a href="adminPortal.php">Admin Portal</a></li>

<li id="logout"><a href="logout.php">Logout <span class="glyphicon glyphicon-log-out"></span></a></li>

<?php }elseif(($\_SESSION['admin'] == 1)){ ?>

<li><a href="#homePage">Home</a></li>

<li><a href="#tour">Portfolio</a></li>

<li><a href="#contact">Contact</a></li>

<li id="logout"><a href="logout.php">Logout <span class="glyphicon glyphicon-log-out"></span></a></li>

<?php }else{ ?>

<li><a href="#homePage">Home</a></li>

<li><a href="#portfolio">Portfolio</a></li>

<li><a href="#contact">Contact</a></li>

<li id="login" class="dropdown"><a class="dropdown-toggle" data-toggle="dropdown" href="#">Login <span class="glyphicon glyphicon-log-in"></span></a>

<div class="dropdown-menu">

<form id="formLogin" class="form container-fluid" action="authenticate.php" method="POST">

<div class="form-group">

<label for="usr">User Name:</label>

<input type="text" class="form-control" id="usr" name="usr" placeholder="Username" required>

</div>

<div class="form-group">

<label for="pwd">Password:</label>

<input type="password" class="form-control" id="pwd" name="pwd" placeholder="Password" required>

</div>

<button type="submit" name="submit" id="btnLogin" class="btn btn-block">Login</button><br />

<p class="text-center">New User?

<a class="text-center" href="signup.php"> Sign Up</a></p>

</form>

</div>

</li>

<?php } ?>

</ul>

</div>

</div>

</nav>

<div class="signin-form">

<div class="container">

<div class='alert alert-success'>

<button class='close' data-dismiss='alert'>&times;</button>

<strong>Success!</strong> Thank you for the email. I will be in touch soon.

</div>

<button class="btn btn-primary" id="back">

<span class="glyphicon glyphicon-backward"></span> &nbsp; Back to Home

</button>

</div>

</div>

</body>

</html>

<?php

}

?>

#### PHP around the Welcome Modal

<?php if((($\_SESSION['admin'] == 1) || ($\_SESSION['admin'] == 2)) && ($\_SESSION['done']!=1)){ ?>

<div class="modal fade" id="welcome" role="dialog">

<div class="modal-dialog">

<!-- Modal content-->

<div class="modal-content">

<div class="modal-header">

<?php if(($\_SESSION['admin'] == 1) || ($\_SESSION['admin'] == 2)){

$\_SESSION['done'] = 1;} ?>

<button type="button" class="close" data-dismiss="modal">×</button>

<h4 class="modal-title">Welcome <?php echo $\_SESSION['fname']; ?>!</h4>

</div>

<div class="modal-body text-center">

<?php

$bday = new DateTime($\_SESSION['bday']);

date\_default\_timezone\_set( "America/Chicago" );

$date = date("m-d");

if(($bday->format('m-d')) == $date){ echo '<h3><span class="glyphicon glyphicon-star-empty"></span> Happy Birthday! <span class="glyphicon glyphicon-star-empty"></span></h3>'; }?>

<?php if($\_SESSION['admin'] == 1){ ?>

<h3>Do cool admin stuff:</h3>

<p>Visit your <a href="adminPortal.php" class="text-info">Admin Portal</a></p>

<!--<p>Create a <a href="newGal.php" class="text-info">New Gallery</a></p>-->

<?php }else{ ?>

<h3>Please enjoy my website!</h3>

<p>Exit to explore the site, or choose a quick link below:</p>

<div class="containter">

<p class="text-left"><span class="glyphicon glyphicon-user"></span> <a href="update.php" class="text-info">Manage your Account</a></p>

<p class="text-left"><span class="glyphicon glyphicon-camera"></span> <a href="#portfolio" class="text-info" onClick="$('#welcome').modal('hide');">View Portfolios</a></p>

<p class="text-left"><span class="glyphicon glyphicon-envelope"></span> <a href="#contact" class="text-info" onClick="$('#welcome').modal('hide');">Contact Me</a></p>

</div>

<?php } ?>

</div>

<div class="modal-footer">

<button type="submit" class="btn btn-default pull-left" data-dismiss="modal">

<span class="glyphicon glyphicon-remove"></span> Close

</button>

</div>

</div>

</div>

</div><?php } ?>

### 3.3.6 JavaScript/JQuery

Using JavaScript and JQuery, I was able to do some neat UI/UX related things. For example, JQuery is used to smooth the scrolling when users click a link that is hashed on the same page (for example, clicking the “TO TOP” arrow in the bottom left corner). JavaScript is the main method for loading modals when certain triggers happen. For example, when a user logs in and gets redirected to index.php, it will load the welcome modal. Following are some examples.

#### JQuery Function to redirect to index.php after sending the email

$('document').ready(function()

{

window.setTimeout(function(){

window.location.href = "index.php";

}, 6000);

$("#back").click(function(){

window.location.href = "index.php";

});

});

#### JQuery to initialize the tooltip and enable smooth scrolling

$(document).ready(function(){

// Initialize Tooltip

$('[data-toggle="tooltip"]').tooltip();

// Add smooth scrolling to all links in navbar + footer link

$(".navbar a, div#up a[href='#myPage']").on('click', function(event) {

// Make sure this.hash has a value before overriding default behavior

if (this.hash !== "") {

// Prevent default anchor click behavior

event.preventDefault();

// Store hash

var hash = this.hash;

// Using jQuery's animate() method to add smooth page scroll

// The optional number (900) specifies the number of milliseconds it takes to scroll to the specified area

$('html, body').animate({

scrollTop: $(hash).offset().top

}, 900, function(){

// Add hash (#) to URL when done scrolling (default click behavior)

window.location.hash = hash;

});

} // End if

});

})

### 3.3.7 HTML5/CSS3

Obviously, HTML5 and CSS3 play huge roles on a website. Even though I used Bootstrap 3 for the framework for this site, there was still a large amount of customization that needed to occur to make the site have the desired look. I used Bootstrap’s grid system to make the site responsive, in addition to media queries where I did not find the responsiveness sufficient. I relied on the form input types in HTML5 (text, password, tel, email, date, et cetera) to help validate data before it even gets sent to PHP or the database. Following are some different examples of HTML and CSS.

#### CSS sample showing media queries and how-to create the parallax effect

.parallax {

/\* The image used \*/

background-image: url("img/aspen.jpg");

/\* Set a specific height \*/

min-height: 650px;

/\* Create the parallax scrolling effect \*/

background-attachment: fixed;

background-position: center;

background-repeat: no-repeat;

background-size: cover;

opacity: .85;

}

.caption {

position: absolute;

z-index: 950;

left: 0;

top: 600px;

width: 100%;

text-align: center;

color: #000;

}

.caption span.border {

background-color: #2d2d30;

color: #fff;

padding: 18px;

font-size: 15px;

letter-spacing: 8px;

opacity: .9;

border-radius: 100px;

}

@media only screen and (max-device-width: 1024px) {

.parallax {

background-attachment: scroll;

}

.caption {

display: none;

}

}

@media only screen and (max-device-width: 415px) {

.parallax {

background-attachment: scroll;

}

.caption {

top: 80%;

display: block;

}

}

#### HTML Section showing part of a form to update a user’s profile

<form class="form-horizontal" action="updateAccount.php" method="post">

<div class="form-group">

<label class="control-label col-sm-2" for="fname">First Name:</label>

<div class="col-sm-10">

<input type="text" class="form-control" id="fname" placeholder="<?php echo $r['FNAME']; ?>" name="fname">

</div>

</div>

<div class="form-group">

<label class="control-label col-sm-2" for="mname">Middle Name:</label>

<div class="col-sm-10">

<input type="text" class="form-control" id="mname" placeholder="<?php echo $r['MNAME']; ?>" name="mname">

</div>

</div>

<div class="form-group">

<label class="control-label col-sm-2" for="lname">Last Name:</label>

<div class="col-sm-10">

<input type="text" class="form-control" id="lname" placeholder="<?php echo $r['LNAME']; ?>" name="lname">

</div>

</div>

<div class="form-group">

<label class="control-label col-sm-2" for="bday">Birthday:</label>

<div class="col-sm-10">

<input type="date" class="form-control" id="bday" name="bday">

</div>

</div>

<div class="form-group">

<label class="control-label col-sm-2" for="email">Email:</label>

<div class="col-sm-10">

<input type="email" class="form-control" id="email" placeholder="<?php echo $r['EMAIL']; ?>" name="email">

</div>

</div>

<div class="form-group">

<div class="col-sm-offset-2 col-sm-10">

<div class="checkbox">

<input type="hidden" value="0" name="news">

<label><input type="checkbox" value="1" checked name="news"> Sign up for the Newsletter?</label>

</div>

</div>

</div>

## Data Model and Storage

As a part of using a standard LAMP (Linux, Apache2, MySQL, and PHP) stack for this website, I used MySQL for the database, which I both connected to remotely via MySQL Workbench and through the web via PHPMyAdmin. I tried to pay close attention to data types and how much space they would take up. For instance, I used “tinyint” values for things that were binary in essence. Also, I limited the field for “State” to only 2 characters since I could control that in the HTML.

The SQL Statements I used are all included below. I created the ER Diagram in MySQL Workbench and forward-engineered the tables there, too.

The portfolio galleries are generated based on the data in the database. Users and Admins will be able to edit their user profiles.

There is also an image of my final ER Diagram that shows the relationships between my tables in the database below.

### 3.4.1 SQL Statements

Query to check if file is favorited: (prerequisite that a user is logged in and is authorized to view the gallery. If a favorite for the user, one result will be returned, otherwise no rows will return; will use if/else logic in PHP to show appropriate icon)

“SELECT \* FROM `FAV\_FILES` WHERE FAV\_UID=2 AND FAV\_FID=3;”

Query to login: (should only return one match (authenticated user) or no matches (wrong username or password). MD5 hashed passwords via PHP. This query is returning Admin user).

“SELECT \* FROM ‘USER’ WHERE UNAME = ’admin’ AND PWORD = ’797cb93f8b1159e6dc68b2b7fddd6c55’;”

Query to create user account: (validation will occur via PHP; some fields will be optional and have default values that are inserted if left blank or will be chosen via dropdown, radio button, etc)

“INSERT INTO `USER` (`UID`, `UNAME`, `PWORD`, `ADMIN`, `FNAME`, `LNAME`, `MNAME`, `BDAY`, `STREET1`, `STREET2`, `CITY`, `STATE`, `ZIP`, `PHONE\_HOME`, `PHONE\_CELL`, `TEXTS`, `EMAIL`, `NEWS`, `ACTIVE`, `ANNIV`, `CLIENT`, `CREATE\_TIME`) VALUES (NULL, 'cfulton', '797cb93f8b1159e6dc68b2b7fddd6c55', '0', 'Chris', 'Fulton', NULL, '2017-11-29', '123 Main Street', NULL, 'Minneapolis', 'MN', '55403', NULL, NULL, '0', 'cfulton@gmail.com', '0', '1', NULL, '0', CURRENT\_TIMESTAMP);”

Query to show all files in specific gallery: (This is showing the one test file I entered into the Photography Gallery; other queries to show gallery files will be similar but with a different GALLERY\_GID number)

“SELECT \* FROM `FILE` WHERE GALLERY\_GID=1;”

Query to show files searched for by user in a specific gallery: (This query is showing any photo in the Photography Gallery that has the word “red” in it)

“SELECT \* FROM `FILE` WHERE GALLERY\_GID=1 AND TITLE LIKE '%red%';”

Query to get info about the selected gallery: (will be used to see if the “LOGGED\_IN” flag is true, if there is a password for the gallery or not, and how many gallery views exist).

“SELECT \* FROM ‘GALLERY’ WHERE GID=’1’;”

Query to update number of gallery page views:

“UPDATE GALLERY SET NUM\_VIEWS=(NUM\_VIEWS + 1) WHERE GID = 7;”

Query to see if selected gallery can be viewed by the current user: (If the gallery flag “LOGGED\_IN” is true and someone is logged in, then it will check the database to see if the user (UID) and the gallery id (GID) are returned in the “GALLERY\_LIST” table; if not, an error will be returned).

“SELECT \* FROM `GALLERY\_LIST` WHERE GLIST\_UID=1 AND GLIST\_GID=4;”

Query to Update User Info: (will mostly involve PHP logic to only update the fields the user populates like the following sample)

“UPDATE `USER` SET STREET2='Suite 1' WHERE UID = 5;”

Query to Delete User Account:

"DELETE FROM `USER` WHERE `UID` = 5;"

Query to Create a new Gallery: (validation will occur via PHP; some fields will be optional and have default values that are inserted if left blank or will be chosen via dropdown, radio button, etc.)

“INSERT INTO `GALLERY` (`GID`, `GNAME`, `GDESC`, `GPW`, `LOGGED\_IN`, `NUM\_VIEWS`) VALUES ('8', 'Amalan', 'Photos of Mr. Amalan Pulendran by Jesse Valley.', NULL, '1', '8');”

Query to Update Gallery Data: (will mostly involve PHP logic to only update the fields the admin populates like the following sample)

“UPDATE `GALLERY` SET LOGGED\_IN='0' WHERE GID = 8;”

Query to Delete Gallery:

"DELETE FROM `GALLERY` WHERE `GID` = 8;"

Query to Add a new file to a gallery: (validation will occur via PHP; some fields will be optional and have default values that are inserted if left blank or will be chosen via dropdown, radio button, etc.)

“INSERT INTO `FILE` (`FID`, `TITLE`, `FILE\_NAME`, `CREATOR`, `FDESC`, `CATEGORY`, `FORSALE`, `PRICE`, `DOWNLOAD`, `QTY`, `GALLERY\_GID`) VALUES (NULL, 'Hello, Me', 'me.jpg', 'Jesse Charles Valley', 'Self portrait while photographing a wedding in San Diego in 2016.', 'Photography', '0', '50', '0', '1', '1');”

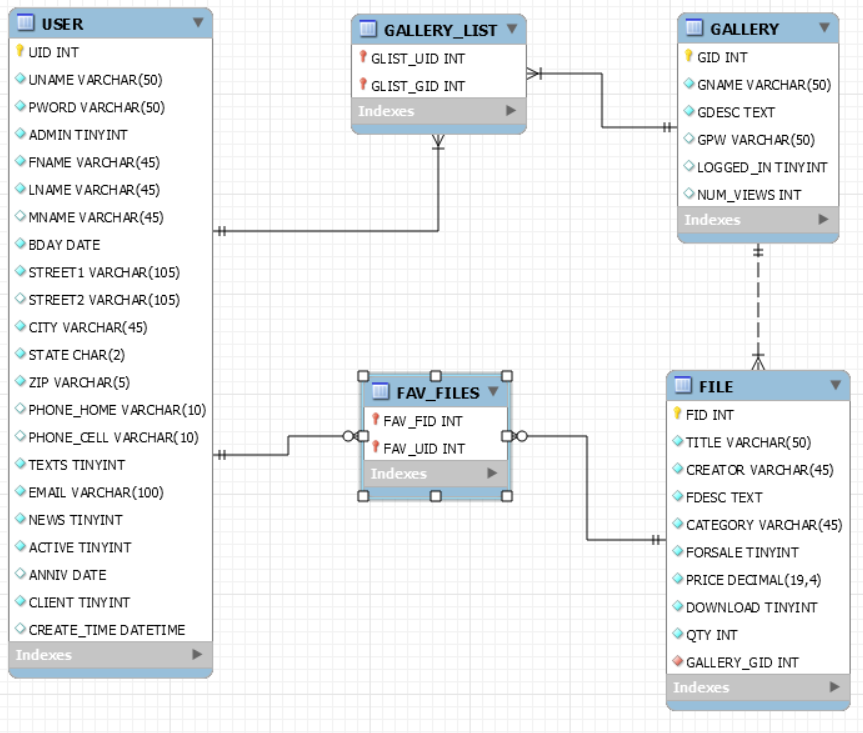
Query to Update File Data: (will mostly involve PHP logic to only update the fields the admin populates like the following sample)

“UPDATE `FILE` SET FORSALE='1' WHERE FID = 4;”

Query to Delete File:

"DELETE FROM `FILE` WHERE `FID` = 4;"

### 3.4.2 ER Diagram



## Security Considerations

In order to increase security, I did a couple worthwhile things. First, I used md5 hashing of the passwords and the password input type attribute with the forms so that the information the user types in is private.

When a user enters data into a form, I first have it validate through the HTML form type attribute, where I can make sure digits are entered for zip codes and phone numbers, et cetera. Additionally, I created a PHP function that trims and uses htmlspecialchars on the data that users enter before it can get put into the database. Both of these features help ensure no funky data or hacks are being entered into the database. Here is an example of that function:

function test\_input( $data ) {

$data = trim( $data );

$data = htmlspecialchars( $data );

return $data;

}

if(isset($\_POST['save'])){

$fname = test\_input($\_POST['fname']);

$mname = test\_input($\_POST['mname']);

$lname = test\_input($\_POST['lname']);

$email = test\_input($\_POST['email']);

$city = test\_input($\_POST['city']);

Finally, I used prepared statements and PHP Data Objects (PDO) to protect against SQL Injections. Essentially, since the entry of the user is being converted to a string value and run separately from the query, there is no way for a true SQL Injection to occur. Here is a sample of that technique:

$q = 'SELECT \* FROM user WHERE uname=:username AND pword=:password';

$query = $dbh->prepare($q);

$query->execute(array(':username' => $username, ':password' => $md5\_pass));

# Implementation

## Mode of Implementation

For this website, I did not use any fancy framework or Model, View, Controller (MVC) system. Instead, I used Adobe Dreamweaver to type out the various pages and manage any revisions. I do feel like there is a similar mode going on where there are pages that are primarily outward facing HTML/CSS (this is similar to a view), and then there are the PHP only pages that work on the backend. I will list some examples in the next section. Ultimately, PHP and MySQL are working together on the backend, and PHP and JavaScript are working on the standard HTML to present the actual webpage that the user experiences. Additionally, media queries ultimately effect what is displayed through the screen, and can be seen as a controller of sorts. Again, it is not a standard MVC system, but it definitely has some similarities.

## Description of major components for implementation

### Authentication (authenticate.php)

Like I mentioned already, the login modal is built around Bootstrap using HTML and CSS. Once data is entered and the user posts the data via the “submit” button, then it gets passed to “authenticate.php” and processed there. If it is successful, the user will be logged in; if not, there will be an error message displayed.

### Validate Gallery Password (valgpass.php)

This is very similar to the authentication in the previous model. First the user will be asked to enter a password to view the portfolio. Once “submit” is clicked, the system moves to the backend and processes the PHP code. Success or failure results in different views being returned via headers.

### Sign Up, Create Account, and Update Account

All three of these pages have some similarities in that on the user’s end, a slick form is presented for them to fill out that takes advantage of the UI features built into HTML5 and CSS3. Once the data is entered and initial client-side verification is done, it gets passed to the appropriate PHP pages that process the data and redirect the user based on success or failure of what the user entered.

# Testing and On-going Maintenance

## Overview of Site Test Plan

Like I mentioned above, one of my primary concerns was that this website would be responsive and work well across many users and devices. To help accomplish this, I tested the site manually on different devices, including a Linux machine, iPhone, iMac and Windows 10 laptop. In order to test the functionality of the website, I relied primarily on manual tests that I created as I wrote the code. In order to do this effectively, I would test the various components out individually to ensure that they were working as planned. I also made sure to incorporate the various user stories into my tests, which I’ve described earlier in section 3.3 regarding the Source Code. Finally, I tested the logic and usability through my peers and my instructors actually using the website.

## Methods used for testing applications

### Interactive Tests

Like I mentioned above, I primarily used manual testing on the website. Since I had such a strong grasp on the data I needed to use and gather, I feel like this was successful for me throughout this project.

Most of the interactive testing I completed was regarding UI/UX features and the responsiveness of the website. A huge tool to help in that regard is the Developer Tools in Google Chrome. It was extremely helpful to use the “responsive” setting to see exactly where funky issues would pop up with the website. While some people may consider this a little bit of overkill, I have experienced when a web browser it not maximized and those weird layout issues ensue. In supposing a potential employer or customer would open the website at a weird width, I want to make sure that the site still makes an excellent first impression. Like we learn in so many other regards, you do not get the chance to earn back a first impression.

### User Tests

In manually testing my website, I focused on the different users and scenarios that they would be under when coming to the website. All of the initial work that we did to really focus on that data and the use cases of potential users helped me to feel confident in the manual testing I did on the website. Since I had such a strong grasp of what feature I wanted to implement and I knew that my SQL code was correct, I just needed to test the actual website and see if it reacted as expected.

### External Validations

I used two main modes of external validation for this website. First, I wrote out every query that I used in MySQL and ran them on test data to be assured they were correct. Mr. Fulton, the instructor who helped organize and normalize our databases, also looked over the ER Diagram and SQL code. Secondly, I used the W3C Markup Validation Service (<https://validator.w3.org/>) to validate the HTML files. Once I remove the internal CSS styling and save them to an external sheet, I will be also validation the CSS through W3C.

## Methods used for maintaining application

### Overview of Maintenance Plan

The primary plan that I have regarding the maintenance of this website is twofold. First, I intend to modularize the webpages so that the generation of new portfolios is easier and more efficient to maintain. Secondly, I intend to clean up the code by creating an external CSS stylesheet and separate PHP files for repeated items such as the header/navigation and the footer.

Eventually, I would also like to learn and implement more modern, best practices for managing and serving the images on the website. For example, I do not want a mobile user to wait to download the high-resolution image that is necessary for the website to look good on a new, 5K resolution iMac, because their screen has a much smaller resolution.

### Future Improvements

Ultimately I want to create the modules to maintain, create and update the portfolios directly through the website. Currently I have been coding each site and manually entering the data. In addition to this, I want to make the images within the galleries searchable, especially as they grow in volume.

Secondly, I want to realize a better solution for displaying and organizing the images within the portfolio. It is currently a little tricky to show images of different orientation, widths and aspect ratios in the clean, symmetrical manner I desire. Ideally, I would want to upload a folder of images that are all different and have them display beautifully.

Third, I want to eventually incorporate a shopping cart and payment portal so that I could sell my services and art online. As a part of this implementation, I want to enable users to favorite or rate images so that they can more easily find and order what they want.

Finally, I want to update my personal blog and copy over my previous posts into the new format. This would mainly be for the sake of continuity, and I believe I will have to tackle it programmatically in order to do it as efficiently as possible.

Overall, I am very happy with the results of this project, and I am excited to share it with the world and see where it goes from here!

# Appendices

## References

1. Chris Fulton & Amalan Pulendran
2. Stack Overflow
3. W3Schools
4. FreeContactForm.com

## Versions

Project Overview\_0.0

Project Overview\_1.0

Project Overview\_2.0

Analysis\_0.0

Analysis\_1.0

Analysis\_2.0

Design\_0.0

Design\_1.0

Implementation\_0.0

Implementation\_1.0

Testing\_0.0

Testing\_1.0