Seneca | FACULTY OF CONTINUING EDUCATION AND TRAINING

SENECA WEB PROGRAMMING PROGRAM
SUMMER 2021 SESSION

Module Number: WEB 306
Assignment #2

Instructor: Tim Lai
Title: PHP MVC CRUD Application
Assigned: Sunday July 4
Due date: 11:59pm Wednesday July 28
Value: 40%

Submission details: This assignment MUST be submitted in at least one of 2 ways, as specified below. If the assignment does not follow the specified instructions, it will be considered incomplete. *Failure to upload this assignment will result in a grade of zero*.

1. This assignment should be uploaded to a **GitHub repository** which I will invite you to. When I download this GitHub repository I must be able to access all your source files so that I can see your PHP code and connect your site to a database.
2. If you have difficulty uploading to GitHub, then you may alternatively upload your assignment as a ZIP file to **MS Teams** in your **personal channel's Files tab**, or an online cloud storage site such as **Google Drive** or **WeTransfer** and send me a link on **MS Teams** in your **personal channel's Posts tab**. *DO NOT EMAIL ME A ZIP FILE AS AN ATTACHMENT. I WILL NOT RECEIVE IT.*

Instructions: You will be creating a fully-functional **MVC CRUD application** which allows users to create, read, update and delete **objects** which can belong to a **class of your choice** (i.e. Spaceship, Dog, Clown etc.) *as long as it is NOT a Car or Elephant – be creative*. This application will store the objects in an **SQL database**. Your SQL database should have at least two columns: an **INT** called **id**, which will act as a primary key, and a **BLOB** for storing serialized objects. You should connect to your SQL database using a **Connector singleton class** and the **PHP Data Object (PDO)**. The host should be **localhost**, the username should be **root** and the password should remain blank (unless you need one). You should also export an **SQL file** from **PHPMyAdmin** which includes your database, your table and at least **1 item added** to the file, **1 item updated** and **1 item deleted**. Include this file in the root directory of your repository when you submit.

Your application should have a distinct **Model**, **View** and **Controller**, and should follow proper **object-oriented** and **MVC** principles to the best of your ability. It should also use logical folder structure and naming conventions. Your Model must be defined as a class with a **constructor method** which defines at least **3 properties** – a **name**, an **image** and something **unique to your class** – and should only make use of **private** or **protected** properties (no public properties) which are accessed using getter and setter methods. Use of the **__get()** magic method is encouraged but each setter method should perform necessary sanitization based on the property's data type.

Sanitization should be done using a **static Sanitizer utility class**. Any strings must be **filtered and validated** to prevent XSS attacks and hacking. Names should be formatted with **capital first letters** with the **rest lowercase**. Image files should be checked against an **array of approved files**.

The application should start on a **View Objects** page which displays the **name** of each of the objects which have been added to the database. The user should also be able to view an **image** which is associated with each object. There should be a clear **navigation item** or **button** that links to an **Add Object** page which displays a **form** where users can enter **properties** for a new object, including a **name** and an **image**. If the user does not fill out all the required fields, or there is an error sending data to the database, then they should be presented with an **error message** and should not be able to proceed. If the user correctly fills out all the required fields and the data gets sent to the database then they should be taken back to the View Objects page where they are presented with a **success message** and can see all the objects in the database, including the one they just added. Each object displayed on the View Objects page should have an **Edit button** and a **Delete button** near it. When the Edit button is clicked, the user should be taken to an **Edit Object** page where there is a **form** which is **already filled out** with that **object's properties**. If the user edits any of the properties, then they should be taken back to the View Objects page where the changes take effect. If the user leaves one of the required fields blank, they should be presented with an error message and should not be able to proceed. When the Delete button is clicked the selected object should be **deleted** from the database. If the object is successfully deleted, they should be presented with a success message. If there is an error, they should be presented with an error message. *I MUST be able to add objects using the form on the Add Object page, edit objects using the form on the Edit Object page, and delete objects using the Deletion buttons.* Any strings must have HTML **escaped** before being displayed to prevent XSS attacks and hacking.

The **Controller** should be used to transfer data from the Model to the View and from the View to the Model using a **Controller class** with distinct **create**, **read**, **update** and **delete** scripts. You should also use a separate **routes file** to control which controller methods get called based on the GET parameters which are passed to the URL.

The application should have an intuitive and functional **user interface** which uses valid HTML and CSS. Nothing complicated or fancy is required but the user interface should be easy to navigate and use. It should make use of **buttons**, **alerts** and **colour-coding**. Accessibility considerations should be made for non-standard users. Responsiveness and use of libraries such as Bootstrap, Foundation and Font Awesome is NOT required but is strongly encouraged.

Deliverables:

- A **Connector singleton class** which connects to an **SQL database** which stores all of this data; the host should be **localhost**, the username should be **root** and the password should remain blank (unless you need one)
- A **Model** defined as a **class** with at least **3 private or protected properties**, a **constructor method** and corresponding **getter and setter methods**
- A **static Sanitizer utility class** which filters, formats, sanitizes and escapes data
- At least 3 View page types: a **View Objects (index.php)** page, an **Add Object** page and an **Edit Object** page (you should rename the pages to match your class i.e. Add Dog)
- Common **header** and **footer** elements included using **PHP templates**
- An external CSS file
- A functional and intuitive **UI/UX design** which makes use of **buttons**, **alerts** and **colour-coding** (use of Bootstrap/Foundation/Font Awesome is encouraged)

- Accessibility considerations for non-standard users (blind, assistive devices, etc.)
- An **Add form** on the **Add Object** page which takes in and sanitizes **object property** data including a **name**, an **image** and something **unique to your class**
- A **new object** which is created, **serialized**, **encoded**, saved to an **SQL database**, retrieved, **decoded**, **unserialized** and displayed on the **View Objects** page using **PDO** every time the **Add Object** form is submitted
- A set of **buttons or links** for each object on the View Objects page which allow the user to select a method for the object to perform
- An **Edit form** on the **Edit Object** page which updates existing **object property** data using **PDO**
- A **Delete button** for each object on the **View Objects** page which deletes an object using **PDO** when clicked
- A **Controller** with distinct **create**, **read**, **update** and **delete** scripts
- **An SQL file** with a **database** and **table** created, **1 item added**, **1 item updated** and **1 item deleted** which has been exported from PHPMyAdmin

Grading breakdown:

- Inclusion of static Sanitizer and Connector singleton utility classes **(5%)**
- Creation of a Model with properties, getter and setter methods and sanitization **(25%)**
- Creation of a View which retrieves and displays objects from the SQL database **(25%)**
- Creation of a Controller with distinct create, read update and delete scripts **(25%)**
- Creation of a usable UI/UX design using HTML, CSS and PHP templating **(10%)**
- Good coding/file organization practices (property names, bracket indentation etc.) **(10%)**

Your files should contain only valid (as determined by the W3C validator) HTML code. You can use invalid code in the HTML but if you do, you must include a comment beside the invalid code explaining your decision. To properly validate your HTML in your PHP files, you MUST do a "view source" of the page in a web browser. The W3C validator does not understand PHP code. *You will lose 2% per type of uncommented validation error*. You should not have any PHP errors. *You will lose 2% per PHP error which could have been fixed*.

Late Policy

All late assignments will be given a grade of zero.

Plagiarism

There are serious penalties for cheating and plagiarism offences and you are expected to be aware of our Academic Honesty Policy. Please refer to the Academic Policy at http://www.senecacollege.ca/ academic-policy/acpol-09.html for more information.