**Carefully read and follow all instructions. Open book. Use Vanier computers!**

**You are encouraged to use git or some USB device to keep your work backed up at all times.**

# Instructions

Using MS Word, complete this exam document by entering your name and answering Questions 1, 2, and 3 in the locations suggested by lines. Do not modify the document otherwise. Your answer for Question 4 is developed with Docker containers, your favorite text editor or IDE, and phpMyAdmin.

To the “Midterm Exam” LEA/Omnivox assignment inbox, submit the zip file created as follows:  
To **the root folder of the application** built for Question 4 (the folder containing .htaccess and index.php) add

* this document, completed with your name and answers and
* your **database export (exported with phpMyAdmin).**

From the parent folder of your application root folder, select the application root folder and create a zip archive of **the application root folder (and all its content).** This will ensure all hidden files are included in the submitted zip file.

***You are responsible for the exam that you are submitting. Double-check that you have submitted the correct version of your solutions by downloading and examining your submitted material. Only your last submission will be graded.***

# Question 1: MVC framework questions (10 points)

**Refer to the provided code** and answer the following (2 points each).

1. Bootstrapping is the process of initializing an application by executing tasks required for its launch. For example, an application can be bootstrapped by forcing calls to its entry point.  
   For our PHP MVC framework, write which file performs bootstrapping.

-For bootstrapping in the PHP MVC framework, typically the index.php file performs bootstrapping. You would write "index.php"

1. Explain the instruction that rewrites requests as requests to index.php.

-The instruction that rewrites requests as requests to index.php is typically done using mod\_rewrite in the .htaccess file.

1. After requests are received by the PHP MVC Framework, the routing algorithm runs. What does the routing ultimately call to make the application logic run and what input does it use to select the correct call?

-After requests are received by the PHP MVC Framework, the routing algorithm ultimately calls the appropriate controller action to make the application logic run. It uses the URI or URL parameters to select the correct call.

1. Why do we prepend some class names with the “\” character in our application code?

In PHP, we prepend some class names with the "" character to indicate the global namespace.

1. How does the salt in the password hashing process enhance security?

The salt in the password hashing process enhances security by adding additional random data to each hashed password, making it more difficult for attackers.

# Question 2: Multiple choice (10 points)

Answer in the space provided to the right, selection the best option among the given choices (2 points each)

1. Which statement below about controller classes is FALSE?  
   a) They call views.  
   b) They manipulate model data.  
   c) They format the views.  
   d) They can be decorated with access filtering attributes. C
2. True or false? Server-side PHP code runs in the browser like JavaScript. FALSE
3. Access filtering runs  
   a) before routing  
   b) after routing  
   c) before controller actions are invoked by routing  
   d) after controller actions are invoked by routing C
4. In PHP, we can delimit strings by using different characters. Consider the following code:  
   $name='Alice';  
   echo '$name';  
   What is the produced output?  
   a) Alice  
   b) $name  
   c) 'Alice'  
   d) '$name' B
5. Which type of MVC component must normally contain business logic, if any?  
   a) Model  
   b) View  
   c) Controller  
   d) Router A

# Question 3: Database design (5 points)

Consider an application with Users and user Profiles.

A business rule for this application is that Users can have multiple profiles.

Database keys and constraints are used to enforce business rules as stated above.

(3 pts) List all key fields and constraints needed for the database to respect the business rule.

* To respect the business rule that users can have multiple profiles, the database requires a primary key **user\_id** in the Users table, a foreign key **user\_id** in the Profiles table referencing Users, and a constraint enforcing the one-to-many relationship between Users and Profiles.

(2 pts) On the code implementation side, is any data from the User required to create a user Profile? If so, what data?

- Yes, on the code implementation side, data from the User is required to create a user Profile. This data may include the user's unique identifier, such as their username or user ID and any other relevant information needed to associate the profile with the correct user account.

# Question 4: Anonymous secure bulletin board (25 points)

Using the provided PHP MVC Web application framework code, implement an application that will run in the XAMPP environment and where users can post, edit, delete, and view messages on a bulletin board. **Users set one password for each message that they create. Users can only modify or delete a message if they provide the correct password for this message.** This way, messages can’t be modified or deleted by users who don’t know the password. Try the application at <https://bulletinboard.cstutoring.ca/>. Post responsibly.

Build on top of the MVC code from the GitHub repository <https://github.com/paquettm/eComH24S1>. Write descriptive class, function, and variable names (points deducted otherwise). You are allowed to copy code from your previous work and static online sources.

**Database:**

* **(2 points) You must submit your solution complete with the .sql database export file.**
* Implement a database called ***bulletin\_board*** that contains one table called ***post***. The ***post*** table has
  + an automatically incremented primary key called *post\_id*,
  + a field of type *text* called *message*, and
  + a field of type *VARCHAR (of sufficient length)* called *password\_hash*

**Models:**

* **(2 points)** Your application must connect to your database correctly*.*
* **(5 points)** The *Post* model class handles the database table described above, extends \app\core\Model, and contains the methods required to perform the actions called from the Controller described below.

**Controller:** The application has the controller class ***Post*** with the following methods:

* **(2 point)** index: Show all posts in an HTML table with hyperlinks allowing to edit and delete them.
* **(2 point)** create: Display a post-creation form with message and password inputs and handle its submittal.
* **(2 point)** edit: Display the post message in a form with message and password inputs and handle its submittal. Modify the post message if the password provided for this post verifies against the password hash for this post.
* **(2 point)** delete: Display the post message with a form with a password input and handle its submittal. Delete the post message if the password provided for this post verifies against the password hash for this post.

**Views:** The application will have all the views to match controller methods described above: **(1 point)** index, **(1 point)** create, **(1 point)** edit, and **(1 point)** delete.

**Routing:**

* **(2 point)** *Enter all routes in the app/routes.php file*. This file gets loaded by the App class. Include a route for the empty address.
* **(2 points)** All hyperlinks and redirects (header('location:…')) must be included to allow navigation of the application at all times without entering URLs in the browser address bar.

Hints: There is no Access filtering requirement in this exercise. You may use parametrized routes or querystring arguments to pass required key information to appropriate controller methods.

Each completed part of the work described above gives points as indicated next to each item.