**Web Programming Project II**

**WEBD3000 Web Application Programming II**

**Evaluation :** 25% of Final Grade

**Assignment Description**

In Web Programming Project II you are going to develop a job board website. Jobs will be pulled from an API and saved to a MySQL database. Jobs will be listed on the homepage, and users will have the ability to apply for a job by filling out an HTML form.

This project must make use of objects and it must also use the Model View Controller (MVC) design pattern. At a minimum, you must have a User and Job Model class, an App class that bootstraps your application, and have at least one Controller class.

Your application must also make use of pretty/clean URLs, meaning none of your URLs will contain “.php” and should supply IDs using route parameters, instead of query strings. Query strings may still be used to pass additional data.

You are welcome to utilize the structure and code we learned in class, but you may add new classes to your liking. You are not allowed to use a prebuilt framework like Laravel, or Symfony to complete this project. You may include additional libraries with composer. PSR-4 namespacing must be used for all your classes.

This project also needs to utilize twig for rendering views.

PDO must be for all interactions with the database.

The API can be found here: <http://p2api.ryanmclaren.ca/api/job-postings>

The requirements are:

* A homepage that lists jobs from a MySQL database
  + If there are no jobs in the database, a message should be displayed to the user letting them no there are no jobs to display.
  + The following fields should be displayed for each job:
    - Title
    - Location
    - Start Date: in the format January 01, 2024
    - Description (but only show the first 150 characters)
  + Users can click on a link to get more details about the job and apply for it. The URL structure for this link should be /jobs/{id}.
  + Job listings must be paginated. Your app should allow for the number of jobs per page to be easily configurable. It must be flexible enough to handle the addition or removal of jobs from the database. For code review, have it show 5 results per page.
  + Pagination should be handled using a query string e.g., ?page=2
  + The pagination buttons can be displayed as just previous/next buttons or can be a list of page numbers.
  + If you are on the first page, the “previous” button should be disabled or hidden.
  + If you are on the last page, the “next” button should be disabled or hidden.
* An admin login page that displays a form with an email and password field (located at /login)
  + Email and password are both required.
  + After successfully logging in the user is redirected to /admin
* An admin page that displays a button that the admin can click to get jobs from the API and save to the database.
  + Each job will save to a new row in the database meaning you are not saving the serialized json results directly to the database.
  + A “jobs successfully fetched from API” message should be displayed to the user.
* All admin related pages (i.e., /admin) are protected from unauthorized access, and if they are accessed directly, unauthorized users are redirected to /login.
* When the user is logged in, an “admin bar” should be displayed at the top that has two links/buttons:
  + Dashboard - links to the /admin page
  + Logout – Logs the user out.
* On the job detail page (/jobs/{id})
  + All job fields are displayed in a user-friendly way.
  + An “apply for this job” form is displayed. This is a form with a Your Name, Your Email, and Resume upload field.
    - All fields are required.
    - The resume upload must only allow pdf, doc, and docx.
    - You do not need to account for the overwriting of files meaning it is okay if a user uploads a file with the same name of one that exists.
    - The files do not need to be secured.
    - File upload should be limited to 4MB.
  + After successfully filling out the form an email is sent to the “contact email” for the job and the user is shown a message of “Application sent successfully”
* The email that is sent must include the Job Title, applicants name, applicants’ email, and a link to the applicants resume. Example below:
* A screenshot of a computer

  Description automatically generated
* The interface should be user friendly.
* Links should be included on each page to make it easy to navigate back and forth.
* Error checking must be included for all forms:
  + All fields are required.
  + Server-Side validation required.
  + Appropriate validation messages should be displayed.
  + Client-side validation is not required.
* Your web app should not crash if there are no jobs in the database table.
* Be sure to include internal commenting throughout your code to explain the logic.
* Git should be used throughout development of your project. Remember to commit often and to include comments that describe what the commit contains.
* To aid in testing, submit your project with no jobs in the database table.

**Requirements (Marks breakdown)**

|  |  |
| --- | --- |
| Application Test Cases |  |
| Application test cases  *A series of test cases that test all requirements listed above* | 13.5 |
| Interface and Code Design |  |
| Database Table and Interface Design  *All table fields included / clean interface that is easy to use*  Overall code and structure  *Use of OOP, MVC, minimal redundancy, no unnecessary code included etc.*  Comments Included Git | 1.5  2  1 1 |
| TOTAL MARK | 19 |

**Other Notes**

* Feel free to use the boilerplate via:

git clone git@github.com:getthefiles-ryan/php-mvc-boilerplate.git

* Be sure to have the entire project folder with you at lesson time on the due date for code review.