Project #3

Class: CIS 4004

Semester: Fall 2025

Due Date: 2025-11-30

A LAMP-Stack Web Application

With this assignment, students will create a web application using the LAMP stack, around a theme of their choice. Students will apply their knowledge of front-end and back-end development using MySQL and PHP. The learning outcomes for this assignment are as follows:

- Students will be able to develop a multi-page web application using PHP and MySQL.
- Students will be able to facilitate CRUD operations for MySQL.
- Students will be able to combine front-end and back-end development to create a full-stack solution that allows users to perform CRUD operations through multiple PHP pages.

Students are expected to work alone on the assignment. You may discuss the assignment with others, but please try to complete the assignment by yourself before you seek help. This assignment is not meant to trick you in any way. If something is not clear, remember that you have resources available to you including my office hours. We want you to learn and succeed, so please reach out for any assistance!

Submission Requirements

Students will locally develop a full-stack solution using LAMP technologies, on their local machine (Windows, Linux, MacOS are all OK). Students are not required to deploy the solution to a server. Instead, the project deliverables will be packaged in a .zip folder and submitted to WebCourses. Students may choose any theme around which the solution must be based; this can be a hobby, interest, or fictional business. While the theme is open to student selection, the functionality must meet the following requirements:

1. The web application must be developed using Apache, MySQL, and PHP.
2. There should be at least two types of users: Administrator and Standard User. Users must login to the application. Additional user roles are OK.
3. The first screen shown by the application should be the login page; new users should be able to be created by taking in a username and password. No duplicate usernames should be allowed.
4. Subsequent screens should be shown to the correct users; for example, the Standard User should not see a screen to manage inventory because that is an Administrator task.
5. Administrators should be given a separate list of functionality than Standard Users.
6. Administrators must be able to perform all CRUD operations in some capacity (Create, Read, Update, and Delete).
7. At the very least, Standard Users must be able to Read data. Your application may warrant additional functionality for Standard Users.
8. There should be at least one "One-to-Many" relationship in the data model. For example, a Team has many players; a Player can only play for one team.

Example Project

Consider the following example, using the theme of Pickup Basketball. One (or many) users could be administrators, who manage the data. Here, they are responsible for adding Teams to the database, and then adding Players to the Team. In addition, they may denote a Coach for the team. Lastly, they may schedule a game between 2 teams, and when the game is done, they may insert data for how many points, rebounds, etc. the players earned in the game. On the other side, other End Users may wish to see the Win-Loss record for each team, and they might want to view the average points for a given player. Therefore, we might have the following screens:

- Register New User (Unauthenticated User)
- Login (Unauthenticated User)
- Main Menu (Admin and End User; but the options are different depending on user)
- Add/Update/Delete Team (Admin Only)
- Add/Update/Delete Player (Admin Only)
- Add/Update/Delete Coach (Admin Only)
- Manage Team (Admin Only)
- Schedule Game (Admin Only)
- View Teams (End User Only)
- View a Player (End User Only)

Submitting Your Assignment

1. There are multiple required deliverables for this assignment, and you will put them all in a .zip folder and submit to WebCourses. The following are required:
    a. A MySQL "data dump", which is a SQL script that will generate the tables.
    b. A source folder for all PHP, HTML, and other resources to be used for your application.
    c. Your Apache httpd.conf file, and any other files you added/edited for your Apache server.
    d. Your php.ini file, and any other files you added/edited for your PHP configuration.
    e. Any other files required to run your application.
    f. Documentation that describes how to configure and run your application.
2. In WebCourses, navigate to the assignment and upload a single .zip file containing your entire submission (see above). **Be careful to include everything needed to run your application.**
3. Please include **documentation** for your project, inside a "README.txt" file:
    a. How can the grader navigate to the application?
    b. What are the steps required to test your application? Be thorough with your steps.
4. Please refer to the syllabus for late submission policy.

Grading Your Assignment

Your assignment will be scored based on multiple factors, described below. Your grade will be posted to WebCourses. If you have any questions about your grade, please contact the grader; if you have further questions, please contact me. Any concerns about grades must be resolved within 1 week of the grade being posted.

Grading Criteria:

If the solution does not use the LAMP stack, the submission will be given a 0.

- **(80%) Required Functionality** – the project will be scored, in part, based on the submission's adherence to the required functionality as described above. Functionality will be inspected to ensure all CRUD operations are supported in some capacity.
- **(20%) Visually Appealing Design** – the project will be scored, in part, based on the submission's design. Students have the choice to write custom CSS or to leverage Bootstrap, or a combination of the two.