**Landing, Login, and Enrollment Pages Development**

Bill Maggs

The University of Arizona Global Campus

CST499: Capstone for Computer Software Technology

Dr. Charmelia Butler

October 21st, 2024

Web page development is a fundamental skill for creating dynamic and interactive websites, allowing users to perform various actions such as registering for accounts and logging in. This paper will walk through the process of setting up a local development environment using XAMPP to run PHP files, creating essential web pages such as landing, login, and registration pages, and establishing a MySQL database to store user information. This paper will also explain how to build a custom class to connect the web application to the MySQL database and outline the development of the registration page, including its layout and PHP source code. This project not only demonstrates the technical steps involved in creating a fully functional web application but also highlights the importance of database connectivity and user management in modern web development.

**Running a PHP file in XAMPP.**

All of the XAMPP software should be installed from previous coursework, if not it can be downloaded and installed from Apache Friends (Apache Friends, 2024), with an example tutorial for installation and testing provided by Udemy (Mikoluk, 2013).

To execute a .php script file, a webserver such as Apache and a database server such as MySQL should be started first. After starting both servers, you must write a .php script in an editor such as Visual Studio Code (VSC) or Notepad. After writing the .php script, save the script to the default /htdocs location. Apache will look for a default index.php when navigating to localhost, if one does not exist an index of the htdocs folder will be shown. As an example, the simple “helloworld.php” shown in figure 1 is created in VSC and saved in "C:\myapps\htdocs\helloworld.php". The XAMPP Control panel is initialized in figure 2, with an Apache server started. A browser is opened to the default URL [localhost/helloworld.php](http://localhost/helloworld.php), the browser output shown in figure 3.

**Figure 1:**

*Sample helloworld.php script*

A screen shot of a computer

Description automatically generated

Figure 2:

*XAMPP control panel with Apache webserver started.*

A screenshot of a computer

Description automatically generated

**Figure 3:**

*Browser output*

A screenshot of a web page

Description automatically generated

**Creating the landing page, login page, and registration page for new users.**

The following html/php scripts were created using VSC to show a landing page, login page, and registration page for new users. A system note - for ongoing development of multiple projects, the default index.php page shows multiple project landing pages as in figure 3. In future references, the link to the University Enrollment Project will be project landing page as shown in figure 4, with the login and registration pages shown in figures 5 and 6 respectively.

**Figure 3:**

Projects landing page

A screenshot of a computer

Description automatically generated

**Figure 4:**

University Enrollment Project landing page

A screenshot of a computer

Description automatically generated

**Figure 5:**

Login page

A screenshot of a computer

Description automatically generated

**Figure 6:**

Registration Page

A screenshot of a computer

Description automatically generated

**Creating the MySQL database and tables.**

From the XAMPP control panel, start MySQL and then select admin as shown in figure 7. The MySQL phpMyAdmin database tool will open in a browser window.

**Figure 7:**

*XAMPP with MySQL admin*

A screenshot of a computer

Description automatically generated

**Figure 8:**

*phpMyAdmin database tool*

A screenshot of a computer

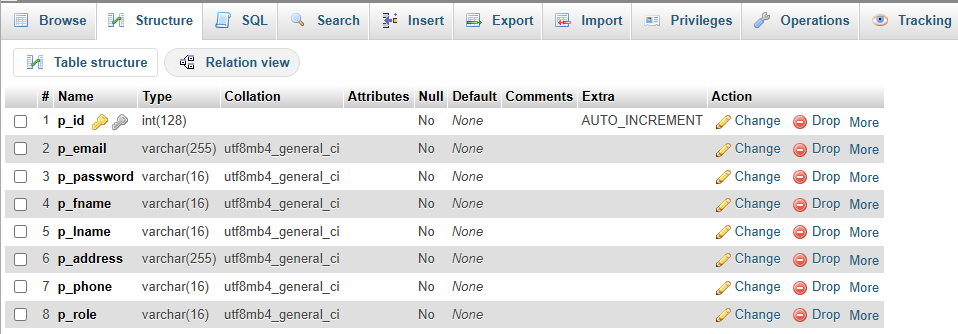
Description automatically generated

Creating a basic user database.

1. Click new in the left column and enter a name for the new database (in this example “university”) as shown in the red ovals in figure 8.
2. Create a new table for User data, adding columns and data types for each component.
3. The finished table will look as in figure 9. Note p\_id is autoincremented as the User ID.

**Figure 9:**

*tbluser*

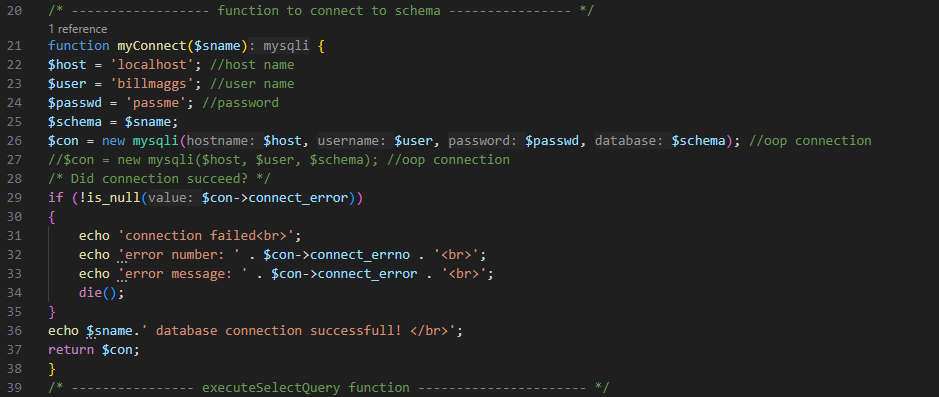


**Discuss the MySQL database functions that you used and the steps you took to create the database connection custom class.**

The .php coding for connecting to a database/schema is shown in figure 10.

Figure 10:

Connecting to database/schema



A database name is passed to the function using the variable $sname. The $con object is created using the new mysqli connection (php, 2024), passing the localhost, user, password, and database name. If there is an error, a message is sent to the console, else a success message is output. The full output will be shown in figure 18.

**Develop the registration page layout.**

The registration page is shown in figure 6. The html for the page is shown in figure 11. CCS scripts are not used, as the focus at this time is on functionality. Note the post method is used to transfer all of the input data to a php script with open, add, and list functions.

**Figure 11:**

Registration page html

A screen shot of a computer

Description automatically generated

**Develop the registration page PHP source code.**

Two php scripts were involved in this page.

1. registration.php is the html page which uses a form to gather the user inputs and POST the data to the php script (addReg.php) which performs the actual table updates and displays results.
2. addReg.php is the more complex script which uses both html and php to generate the mysql command from the POST data, execute the query, and then message the user that the data was entered into the table successfully. There is also a “verbose” option that can be enabled by the developer to show the entire table once the user has entered data. This would be disabled by the developer for normal use to avoid sharing data with unauthorized users.

**registration.php**

As describe in figure 11, registration.php is a pure html entry form. There is a <head> section giving the tab title, some <style? Information, and the <meta> content with author information. The body contains a <nav> block with links to the other pages. The new content is in the <form> block which uses the POST attribute to submit the information. The information is all entered, and once the user completes entry and clicks on the submit button, the data is POSTed to addReg.php for processing. The page originally used the integrated example from w3schools.com (w3schools, 2024, PHP -$\_POST) with the html and php integrated in the same script, but there were some issues with the php running before the submit (example, opening and closing the connection before submit), so it was decided to keep the files separate for clarity. There is also the option to reset the form.

So why use POST instead of GET? There is personal information such as password and social security numbers involved. Since POST keeps the information invisible as they are contained within the HTTP request, this seemed to be the better option (W3schools, 2024, PHP Form Handling).

**addReg.php**

This was the complex portion of the assignment. Since the source code is too lengthy for a single screenshot, this will be explained how it functions in multiple blocks. The submit button from the previous page will open this new webpage which would give the user confirmation that the data was accepted into the database. It can also be used for debugging purposes (more on that later). In figure 12, the script starts with a html header to keep the appearance the same as the previous pages and provide a link back to the homepage once the operation is complete.

**Figure 12:**

*Lines 1-14*

A computer screen with text

Description automatically generated

Figure 13 shows the functions for connecting to the database, executing a query returning a result (such as SELECT), and executing a query to modify the database (such as INSERT).

**Figure 13:**

*Database functions*

A screen shot of a computer program

Description automatically generated

Finally, html is mixed with php to setup a table border and headings, and then uses embedded php commands to fetch the rows from the passed variable $result and echo them into the columns. Figure 14 shows the final function. Since the html and php are mixed, ending and beginning the php with the ‘<?php’ and ‘?>’ tags are critical to understand. Even the closing parenthesis in line 108 required its own tags since it is associated with the php while function to scroll through the rows. Other critical elements are the <th> and <tr> elements, with examples from w3schools (w3schools, 2024, tr tag).

**Figure 14:**

*displayTableAlt*

A screen shot of a computer program

Description automatically generated

Now for the main body of the script. Following through figure 12, the script starts with a debug flag $myVerbose. As covered in part 1, the script can output either just status to the user, or output the entire table using the displayTableAlt function. The script then uses the $\_POST superglobal to assign the user data to the php variables. The connection to the schema is then attempted using the myconnect function. Once the connection is successful, the $sql variable is used to put together the INSERT INTO statement. The $con and $sql are then sent to the executeQuery function to add the table row.

**Figure 15:**

*Main script*

A computer screen shot of text

Description automatically generated

If $myVerbose is set, then the updated table is displayed as was shown in figure 4, using the updated displayTableAlt function. The end of the script is “housekeeping” – memory is freed from the $result array, and the connection to the database is closed.

**Develop the table that saves the user information in the database.**

At this time, there is only one table in use for the user registration information and is shown in figure 9. As more functionality is developed such as adding classes, more associated tables for the database will be created.

**Explain the steps taken to create the registration page and save the user information in the database.**

The source code development is covered in “Develop the registration page PHP source code” and associated figures 12-15. However, figures 16- 18 show the steps a user would follow. From the landing page shown in figure 16, select Registration.

**Figure 16:**

*Landing page*

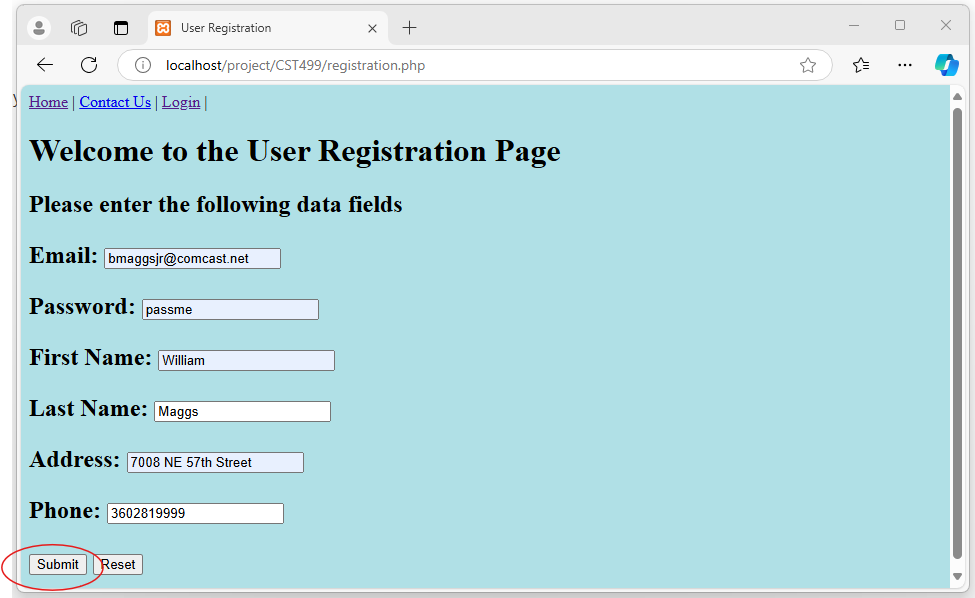
A screenshot of a computer

Description automatically generated

Once all of the user information is entered, click on submit as shown in Figure 17.

**Figure 17:**

*Submit user data*



After submission, a confirmation screen will appear, with a link back to the landing page as shown in figure 18.

**Figure 18:**

*Submission confirmation*

A screenshot of a computer

Description automatically generated

The user may now login from the landing page in figure 16.

**Provide screenshots of all developed pages, database, tables, layout, and source code.**

All screenshots and source code have been provided in figures 1-16. Full source code files are available on github at [bmaggsjr/CST499: GITHUB for CST499 Assignments](https://github.com/bmaggsjr/CST499)

**Conclusion**

Developing a web application that includes user registration and login functionality requires a solid understanding of both frontend and backend development. By setting up a local environment using XAMPP, designing web pages, and creating a MySQL database, I was able to build a system that allows users to register and store their information securely. The use of PHP and MySQL provides a flexible and reliable platform for developing dynamic web applications. Through this project, I learned the importance of integrating server-side scripting with database management, which forms the backbone of many web-based systems today. The steps and methods discussed in this paper serve as a foundation for more complex web development projects.

**References:**

Apache Friends (2024) Download XAMPP for Windows. Apache Friends. Retrieved from [Download XAMPP](https://www.apachefriends.org/download.html)

Mikoluk, K. (2013). *XAMPP Tutorial: How to Use XAMPP to Run Your Own Web Server. Udemy*. Retrieved from [XAMPP Tutorial: How to Use XAMPP to Run Your Own Web Server - Udemy Blog](https://blog.udemy.com/xampp-tutorial/)

Php.net (2024). Connections. Php.net manual. Retrieved from [PHP: Connections - Manual](https://www.php.net/manual/en/mysqli.quickstart.connections.php)

W3schools (2024). $\_POST in HTML Forms. W3schools. Retrieved from [PHP $\_POST (w3schools.com)](https://www.w3schools.com/Php/php_superglobals_post.asp)

W3schools (2024). PHP Form Handling. W3schools. Retrieved from [PHP Form Handling (w3schools.com)](https://www.w3schools.com/php/php_forms.asp)

W3schools (2024). HTML tr tag. W3schools, retrieved from [HTML tr tag (w3schools.com)](https://www.w3schools.com/tags/tag_tr.asp)