**Student Portal Website**

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

Dennis Walters

University of Arizona Global Campus

CST 499: Capstone for Computer Software Technology

Dr. Amjad Alkilani

November 15, 2021

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

The purpose of the Student Portal website is to provide a means for each student to register, login, maintain their information, and enroll in classes. At some point in the future additional functionality may be added to the Portal to provide additional information, or access to other applications like financing, or student resources.

**How to Run a PHP File in XAMPP**

There are couple of different ways to run a PHP file. The first is by accessing it via a web browser which requests the PHP file through Apache. The second way is by running the PHP command line interpreter, php.exe. The best way to test if PHP is running properly is to create a PHP file called phpinfo.php in the htdocs directory under the XAMPP root directory (i.e. on Windows, C:\xampp\htdocs\phpinfo.php). Inside the phpinfo.php file add the following line and save:

<?php phpinfo(); ?>

Once you have created the above phpinfo.php file, open your web browser and type <http://localhost/phpinfo.php> into the URL bar and press Enter. You should see a long list of information about PHP including the version, settings, and extensions loaded. If you do, PHP is running properly.

**Creation of MySQL Tables and Custom Connection Function**

To create the MySQL database, I took the following steps. Via phpMyAdmin, I created a new database called studentportal. I then ran the following SQL query to create the tbluser table (Figure 13).

CREATE TABLE `tbluser` (

`id` int(11) UNSIGNED NOT NULL,

`firstName` varchar(30) NOT NULL,

`lastName` varchar(30) NOT NULL,

`streetAddr` varchar(50) DEFAULT NULL,

`city` varchar(30) DEFAULT NULL,

`state` varchar(2) DEFAULT NULL,

`zipcode` varchar(10) DEFAULT NULL,

`email` varchar(200) NOT NULL,

`homephone` varchar(20) DEFAULT NULL,

`cellphone` varchar(20) DEFAULT NULL,

`ssn` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

PRIMARY KEY (id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

I then created a database user, "spuser", and granted that user all privileges on the studentportal database. The "spuser" and password are I what I use to access the database from the PHP code.

I chose to use the PHP Data Objects (PDO) extension for creating the database connection class in the Model.php (Figure 21) file. The reason for this choice is that PDO offers a consistent interface regardless of which database specific PDO driver you are using. In this case I am using the MySQL driver for PDO. Because of the way PDO is designed, it is pretty easy to create a single function that supports connecting to multiple database types relying on configuration settings stored in a separate class to determine which database driver is to be used. When the class is instantiated, it checks to make sure that the PDO MySQL driver is loaded before trying to connect to the database. It then establishes a connection to the database using the database type, name, user, and password that are stored in a separate Config class as constants. The separation of those elements provides additional security and a single place to store all configuration values for the entire system. Once the connection is established, it is returned back to the calling function for use.

**Steps Taken to Create Registration Page**

To create the Registration page, I chose to use the Twig templating engine within the Model, View, Controller framework I’ve built the Student Portal website around. In addition, I have made use of the Bootstrap CSS framework to design and build the look and feel of the registration page (Figure 2). The form also makes use of built-in HTML attributes to require each input field to be populated before allowing the form to be submitted.

**Steps Taken to Save User Information to the Database**

Once the user enters their information into the registration page then clicks the Register button, the data is passed to the create method in the User model (Figure 22) where a validation is performed on several of the fields (i.e. phone numbers, SSN, email address, and password). If any of these fields fails the validation check, all of the data, with the exception of the password, is passed back to the registration page to pre-populate those fields in the registration form so the user does not have to reenter them. In addition, an error will be displayed above the form letting the user know why registration failed. The final validation check is to verify the entered password against the entered password confirmation. If they match, then the data is prepared including encrypting the SSN and password. The prepared data and SQL statement are then passed to the executeQuery function in Model.php (Figure 21).

The executeQuery function executes the query using the PHP Data Objects (PDO) database connection established by the create method in the User model (Figure 22). If the data is inserted into the database table successfully, a TRUE Boolean value will be returned to the createAction method in the Register controller (Figure 20). If the returned value is TRUE, the user is redirected to the Login page (Figure 4) where a message stating that registration was successful is displayed to the user. If the value returned is FALSE, then the user is returned back to the registration with all of their data, except the password and password confirmation, pre-populated and an error message displayed above the form.

**Screenshots**

**Figure 1**

The initial landing page.

**Graphical user interface, text

Description automatically generated**

**Figure 2**

The registration page.

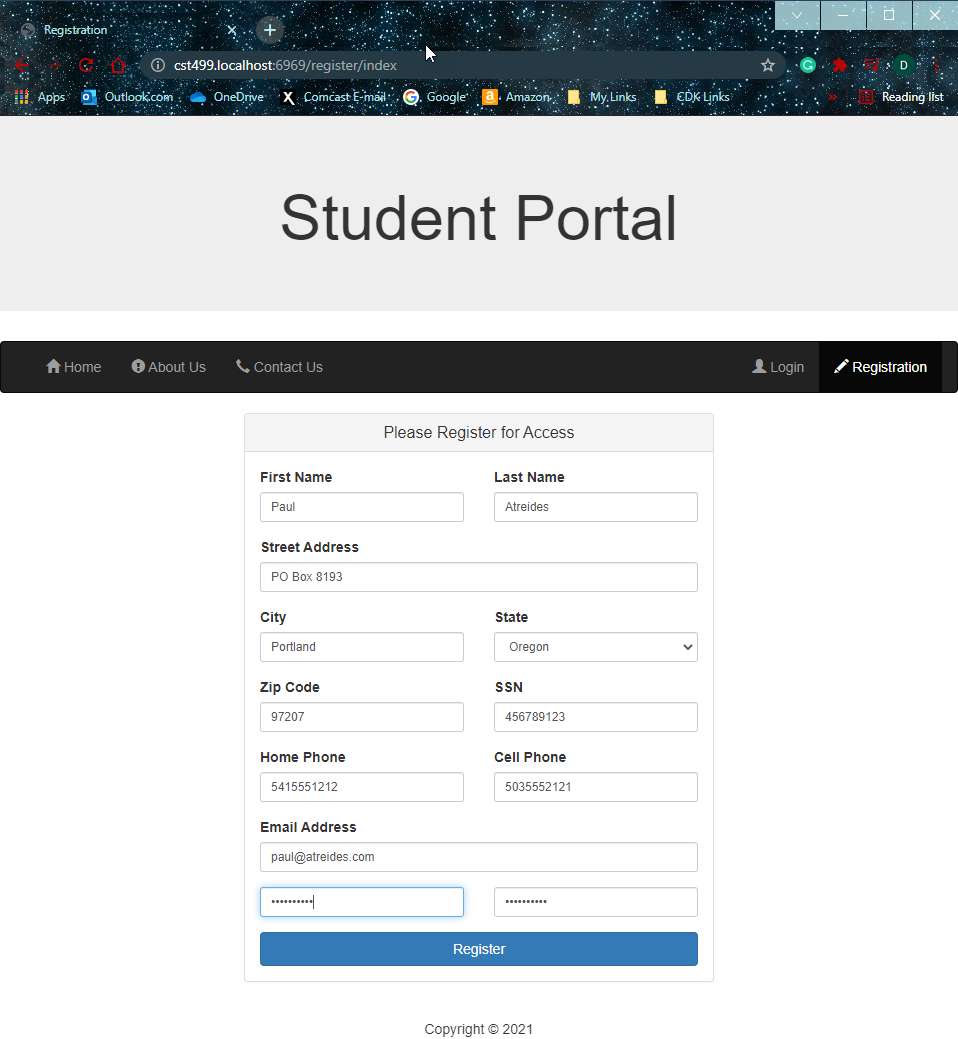
**A screenshot of a computer

Description automatically generated**

*Note.* The Student Portal Website registration page collects the users first and last names, mailing address, phone numbers, social security number, email address and password. The user must enter their password twice to confirm it otherwise registration will fail.

**Figure 3**

The registration page filled in.

****

**Figure 4**

The login page after the user has successfully registered.

**Graphical user interface, website

Description automatically generated**

*Note.* The Student Portal website login page after the user has successfully registered and where the user can gain access to the site. The user enters their email address and password to access the functionality of the portal.

**Figure 5**

The login page accessed by clicking the Login button.

**Graphical user interface

Description automatically generated**

*Note.* The Student Portal website login page accessed by clicking the Login button on the menu bar and where the user can gain access to the site. The user enters their email address and password to access the functionality of the portal.

**Figure 6**

The user home page.

**Graphical user interface, text, application

Description automatically generated**

*Note.* The Student Portal website home page after the user has logged in. A welcome message is printed personalized with the user’s first name.

**Figure 7**

The student profile page.

**Graphical user interface, table

Description automatically generated**

*Note.* The Student Portal website Student Profile page accessed by a logged in user by clicking on the Profile button in the menu bar. From here the user can view their information, edit their information, and change their password. The SSN and password fields are masked by default and can be unmasked by clicking on the eye icon next to each field.

**Figure 8**

SSN and password fields unmasked.

**Graphical user interface, website

Description automatically generated**

*Note.* The Student Portal website Student Profile page with the SSN and password fields unmasked.

**Figure 8**

The edit student profile page.

**A screenshot of a computer

Description automatically generated**

*Note.* The Student Portal website Edit Student Profile page which allows the user to edit all of their information except for the password. Note that the SSN field is no longer masked.

**Figure 9**

The student profile page after update.

**Graphical user interface, website

Description automatically generated**

*Note.* The Student Portal website Student Profile page showing that the user successfully updated their profile.

**Figure 10**

The change password page.

**Graphical user interface, website

Description automatically generated**

*Note.* The Student Portal website Change Password page allows the user to change their password. The user must enter the password in both fields before clicking the Change Password button.

**Figure 11**

The student profile page after password change.

**A screenshot of a computer

Description automatically generated**

*Note.* The Student Portal website Student Profile page after the user has successfully changed their password. This screenshot shows the SSN and password unmasked; however, the default is for them to be masked after the password has been changed.

**Figure 12**

The log out page.

**Graphical user interface, website

Description automatically generated**

*Note.* The Student Portal Website login page after the user has successfully logged out.

**Figure 13**

MySQL tbluser table structure.

Graphical user interface, application

Description automatically generated

**Figure 14**

MySQL tbluser table data view.

Graphical user interface, text, application, email

Description automatically generated

**Figure 15**

View: base.html

Text

Description automatically generated

**Figure 16**

View: Home/index.html

Text

Description automatically generated

**Figure 17**

View: Users/register.html

Text

Description automatically generated

**Figure 18**

Controller: Home.php

Text

Description automatically generated

**Figure 19**

Controller: Login.php

Text

Description automatically generated

**Figure 20**

Controller: Register.php

Text

Description automatically generated

**Figure 21**

Core: Model.php

Text

Description automatically generated

**Figure 22**

Model: User.php

Text

Description automatically generated

**References**

*Bootstrap · The world’s most popular mobile-first and responsive front-end framework. 3.3.6*

*Documentation—BootstrapDocs*. (n.d.). Retrieved from <https://bootstrapdocs.com/v3.3.6/docs/>.

Connolly, R., & Hoar, R. (2018). *Fundamentals of web development* (2nd ed.). Pearson.

Retrieved from <https://ashford.instructure.com/courses/87470/modules/items/4425331>*.*

*Documentation—Twig—The flexible, fast, and secure PHP template engine*. (n.d.). Retrieved

from <https://twig.symfony.com/doc/3.x/>.

*Html reference*. (n.d.). W3Schools. Retrieved from

<https://www.w3schools.com/TAGS/default.ASP>.

Oh, S. (n.d.). *Bootstrap: Tutorial 7* [Presentation slides]. Retrieved from

<http://www.cs.toronto.edu/~mashiyat/csc309/Tutorial/6/Bootstrap.pdf>.

Tsui, F., Karam, O., & Bernal, B. (2018). *Essentials of software engineering* (4th ed.). Jones & Bartlett Learning.