**Week 3 Assignment**

Justin Stewart

CST499: Capstone for Computer Software Technology

Prof. Elchouemi

3/20/2023

**Explanation of Project**

Throughout this CST499: Capstone for Computer Software Technology course, we, as students, have been tasked with the creation of an Online Course Enrollment System. Over the course of five weeks we will design and implement the system end-to-end. The Online Course Enrollment System is a website that allows users to register, login, view their profile information, and log out of the system. Next week, the system will allow Administrators to add courses to a course catalog, and Students to register for the courses added by administrators. The Online Course Enrollment System utilizes a locally hosted database to store and retrieve user information through a combination of SQL, PHP, and HTML forms that have been styled with CSS/Bootstrap.

**Explain how to run a PHP file in XAMPP.**

In order to run a PHP file in XAMPP, one must first install both XAMPP and some form of text editor. Personally, for this assignment, I utilized Notepad++. Otherwise, PHP must be installed locally to be used within popular Integrated Development Environments (IDEs) such as VSCode. Once the prerequisite software has been installed, one must navigate to where they have installed XAMPP, and locate the htdocs folder. Within this folder, one must create a new project folder containing the PHP files they would like to run. In order to run and see the application locally, one must navigate to [http://localhost/projectfoldername/phpfilename.php](http://localhost/projectname/homepagefile.php) in their browser with the Apache module of XAMPP running.

**Online Course Enrollment System Database**

The Online Course Enrollment System’s database was created in accordance with a class diagram produced in Week 1 provided below:

**Note:** It has become clear to me that the administrator class is somewhat unnecessary, and is being consolidated into User class next week. This week’s database is made to follow the diagram.  
**Diagram

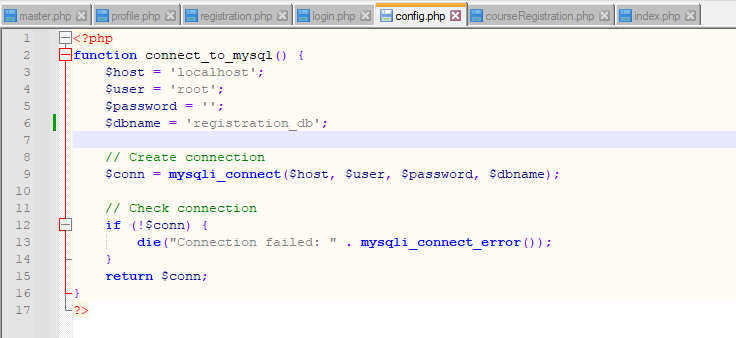
Description automatically generated**

In this diagram, each Class is a table that must be created and associated with its corresponding tables. The following SQL query was used to create the Database itself:

Text

Description automatically generated with low confidence

To connect to the database, the system uses the following Config.php file containing a function to connect to the database that gets “required once” whenever access to the database is required:

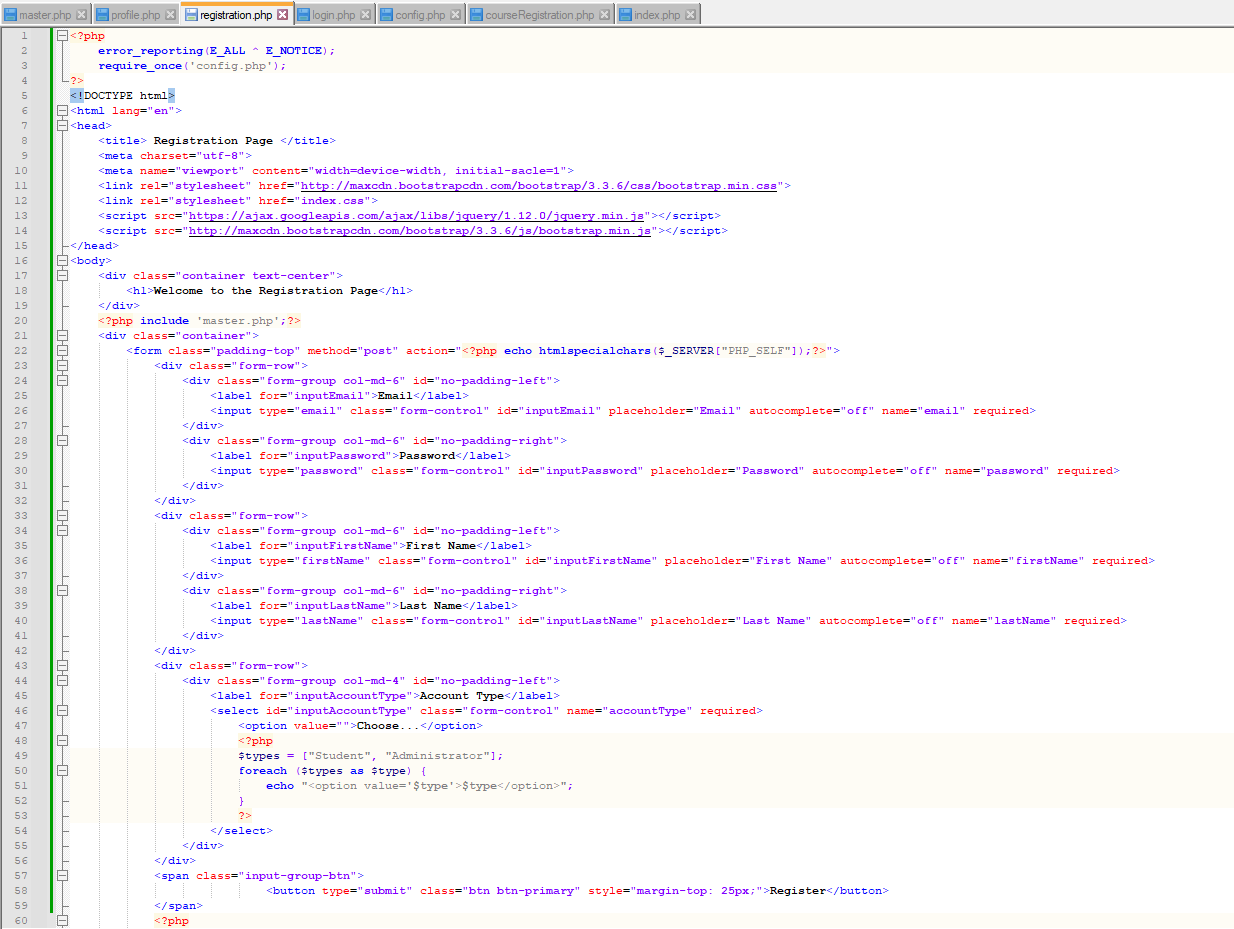


**Create the landing page, login page, and registration page for new users.**

**Note:** I created additional pages such as Profile, Logout, and ContactUs

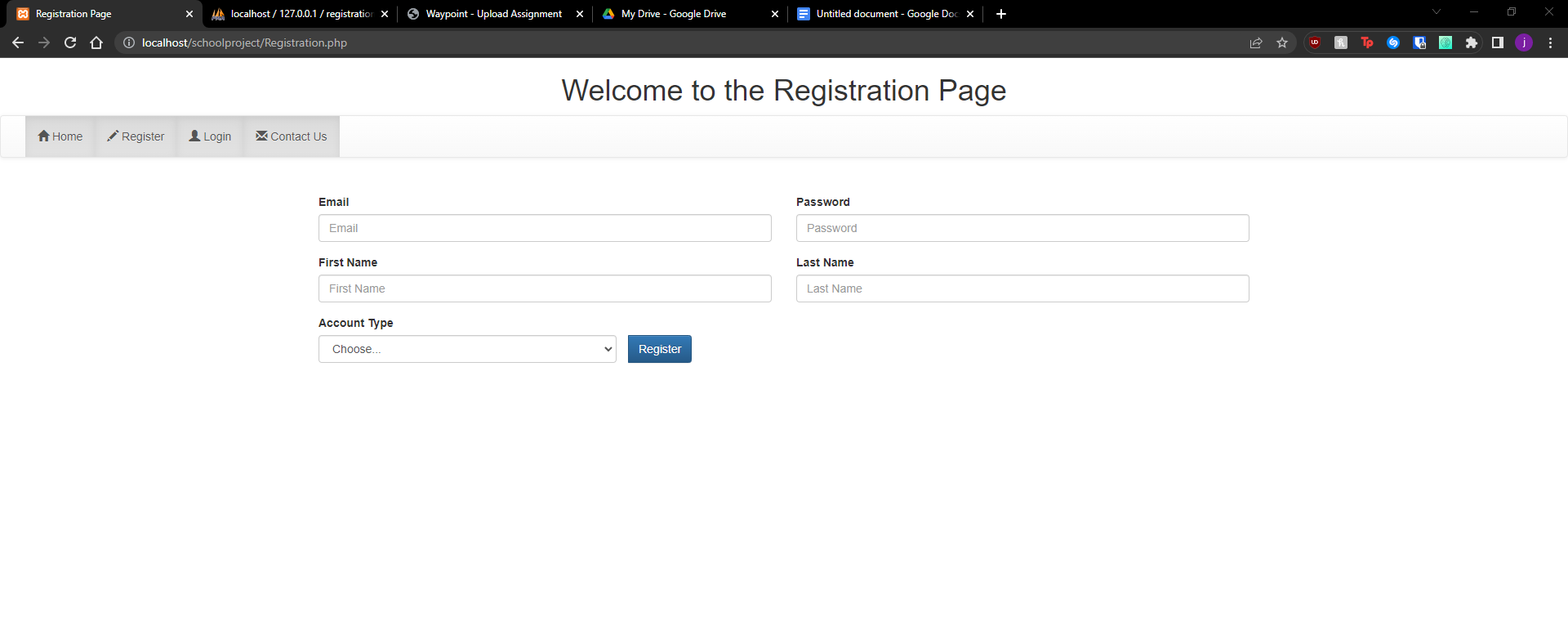
The registration page was created through a multitude of forms to take user input for email, password, first name, last name, and user type (student or administrator) using the POST method. Once the user clicks the register button, preprocessing is performed on the provided user information to prevent SQL injection attempts, and is finally stored within our database to later be retrieved by our login and profile pages. The source code for the registration page is included on the next page.

**Registration Page Source Code:**



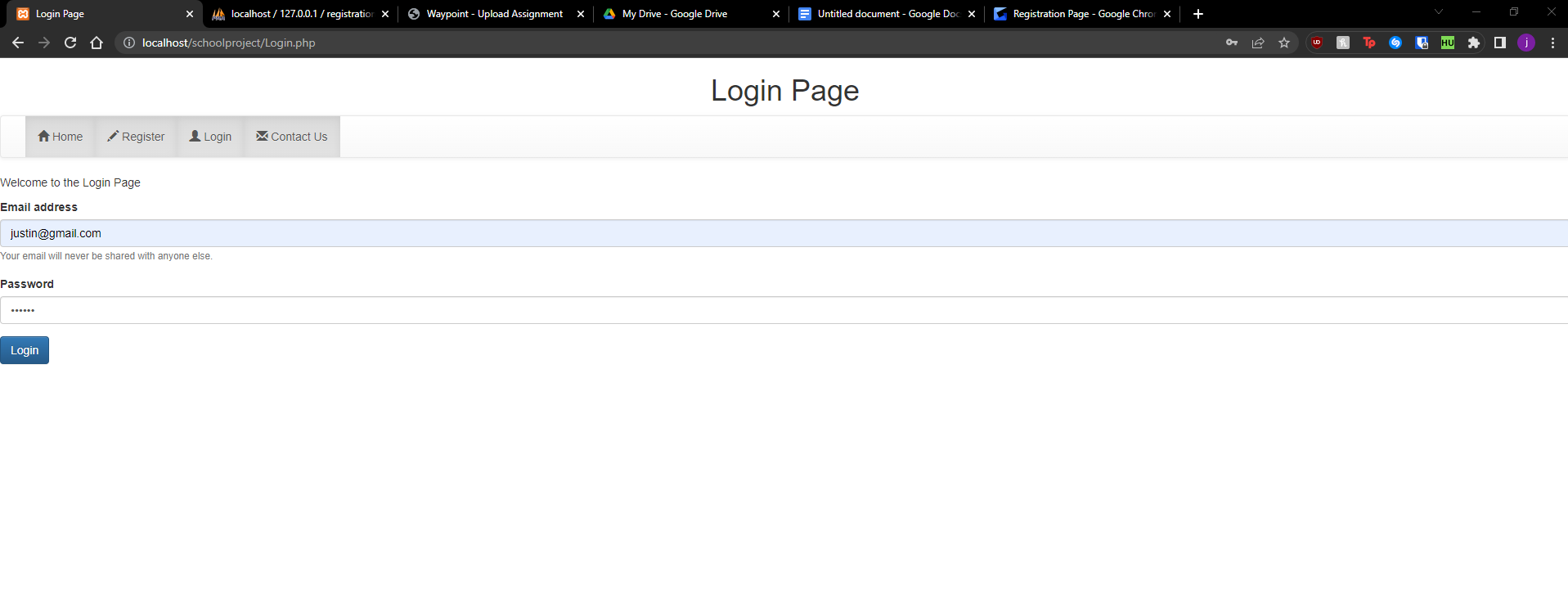


**Resulting Registration Page**

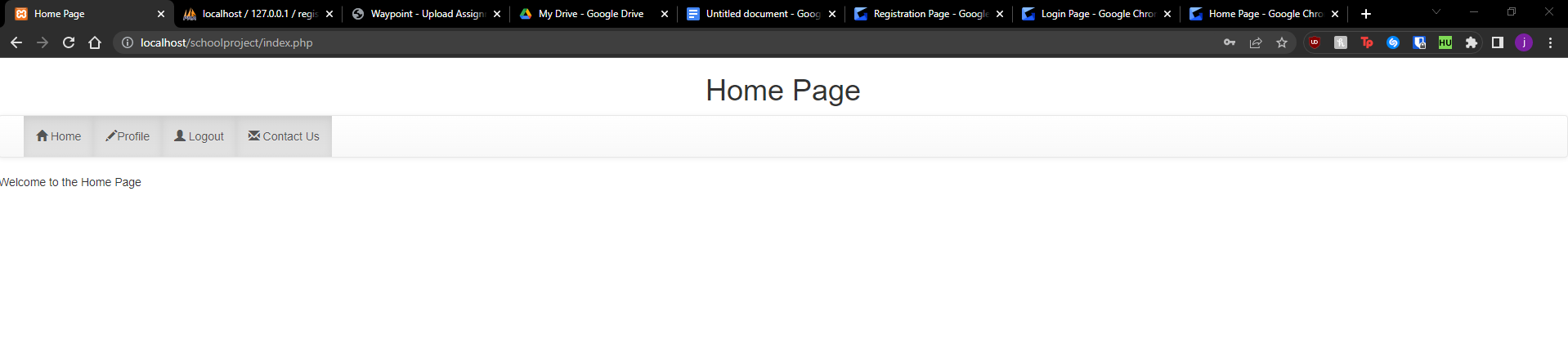
****

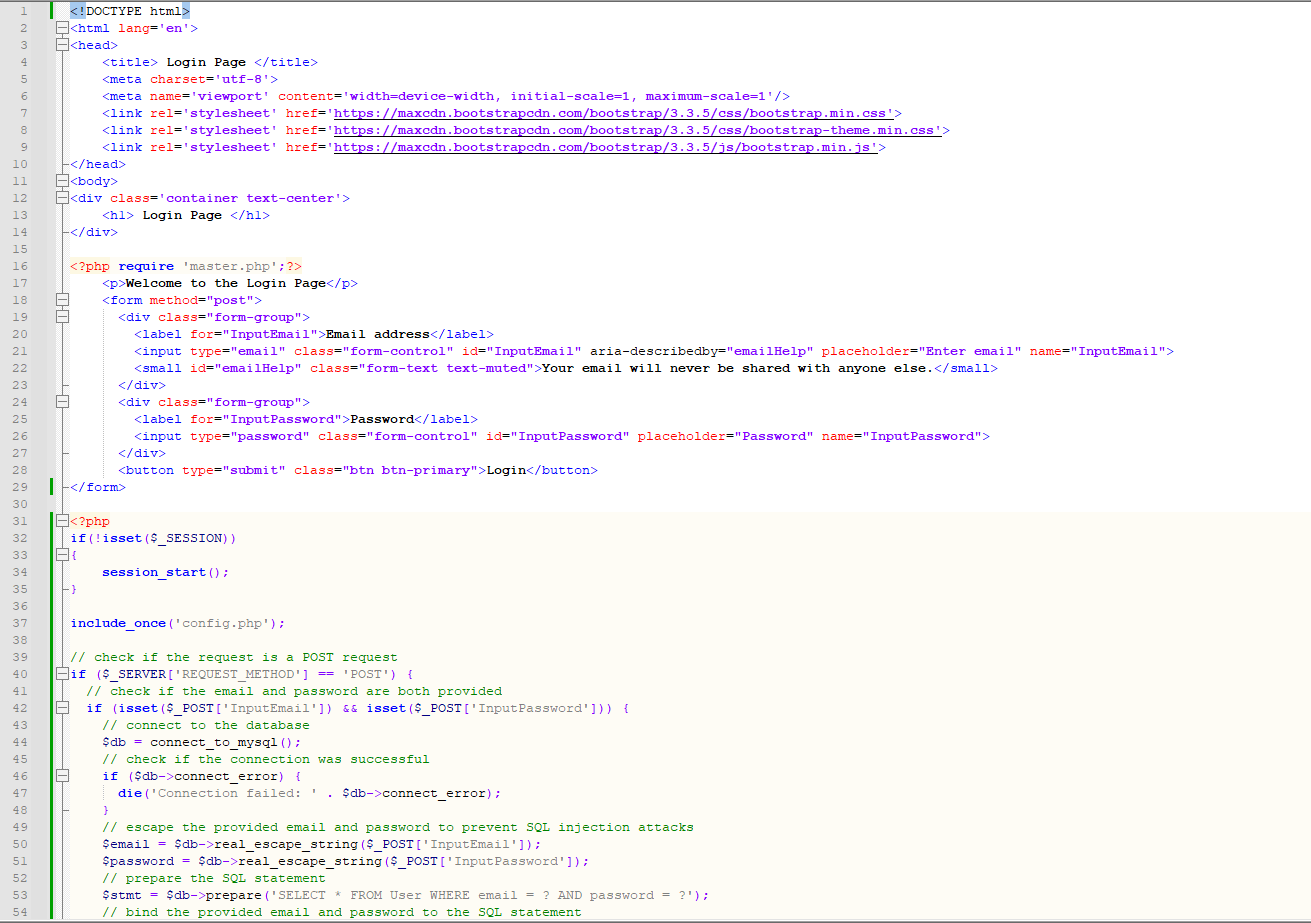
Following the creation of the registration page, users must be able to log into the system. Upon logging in, the user should no longer see the “Login” and “Register” options in the navbar, and instead be able to navigate to their “Profile” page, and log out of the system.

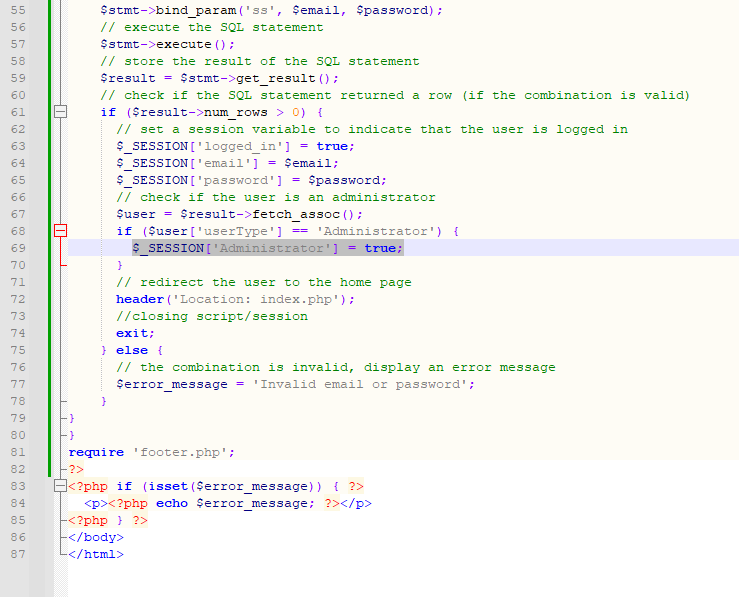
**Login Page (Logged Out)**

****

**Home Page w/ Different Navbar (Logged-in)**

****

**Login Page Source Code**

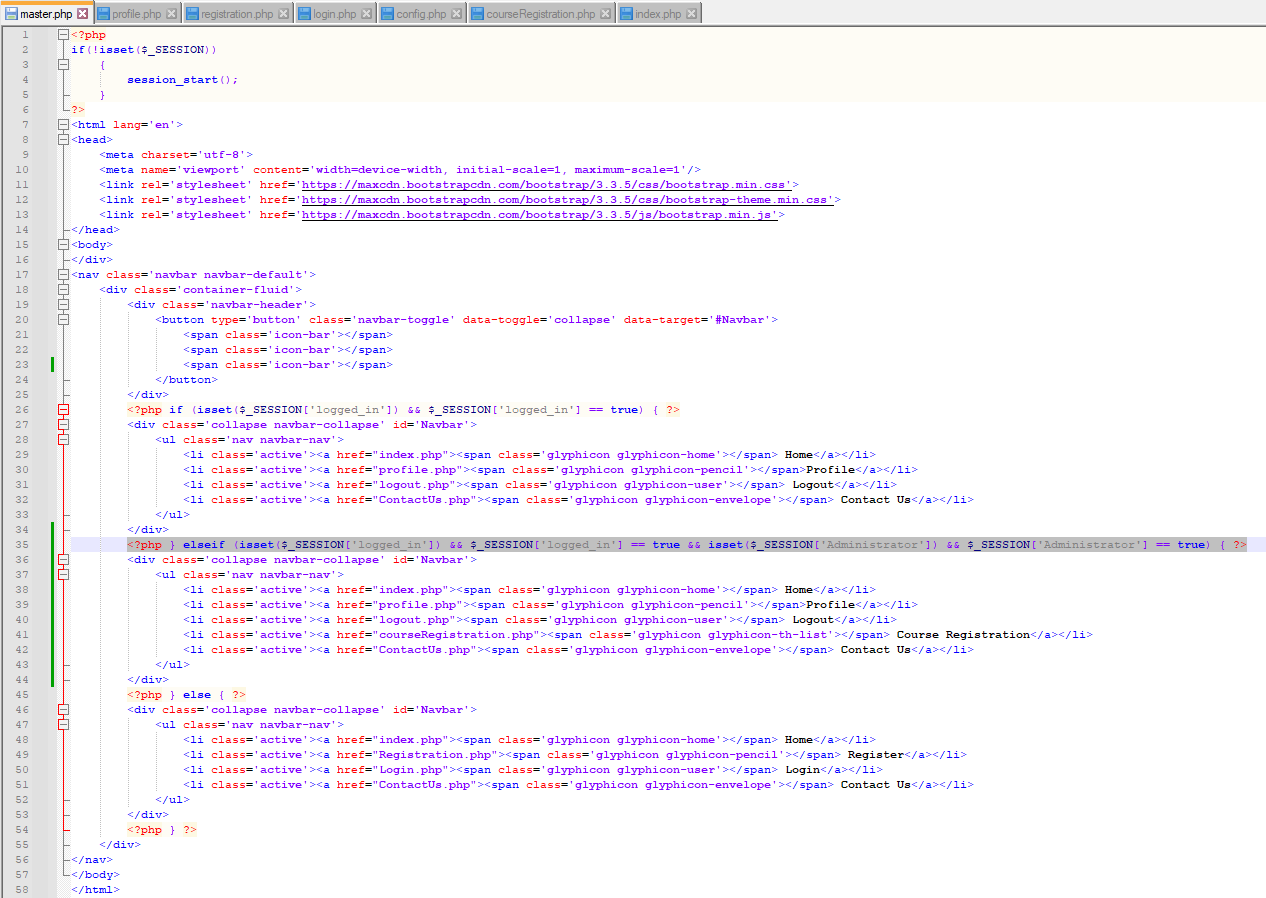
****

To accomplish this task, I check if there is already a session active, and if not, a session is started in order to access the `$\_SESSION` variable and check the value of the `logged\_in` and `Administrator` session variables. With a session started, I required my config file containing the necessary information to establish a connection to the database. After establishing that connection, I use my form elements to escape the user-inputted email and password to prevent SQL injection attacks, prepare a SQL statement to query the database for the pair of user-inputted information, bind the inputted information to the statement, and execute the SQL query. The result of the query is stored in a variable so that the efficacy of the query may be checked. If the query is successful, the `$\_SESSION['logged\_in']` variable is set to true to indicate the user has successfully logged in, if the user that logged in as the ‘Administrator’ userType value associated with their record a `$\_SESSION['Administrator']` variable is set to True, and they are redirected to the home page using the header() function. If the query cannot find the pair of user-inputted information, it tells the user that they inputted an invalid email and password, or no email and password at all.

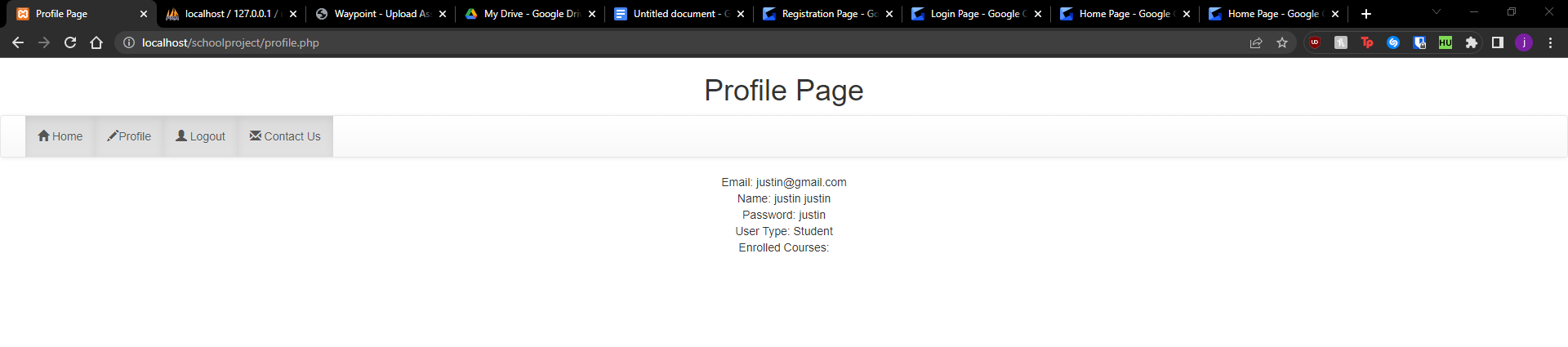
Now that the login functionality is working, the navbar must be altered to adjust what it displays depending on if users are logged in or not, and if the user is an Administrator or Student. To implement this functionality, I created my master.php file, which contains the main navbar of the site that is included on every other page of the site, to have the same session logic I described earlier. With a session established, the master.php file creates the navbar and checks if the user is logged in through the following if/else statements, although the administrator functionality is still a work in progress:

* `if (isset($\_SESSION['logged\_in']) && $\_SESSION['logged\_in'] == true)`
* } elseif (isset($\_SESSION['logged\_in']) && $\_SESSION['logged\_in'] == true && isset($\_SESSION['Administrator']) && $\_SESSION['Administrator'] == true) {

This logic allows us to generate separate navbars depending on if the user is logged in or not, or if the user is an Administrator or not, providing a form of Role-based Access Control (RBAC) to the system. If the user is logged in, the navbar changes the Register and Login buttons/hrefs to Profile and Logout. If the user is not logged in, they will be met with Login and Register. Due to these changes, preliminary log-out, and profile.php files were created to allow users to retrieve their account information and log out of the system.

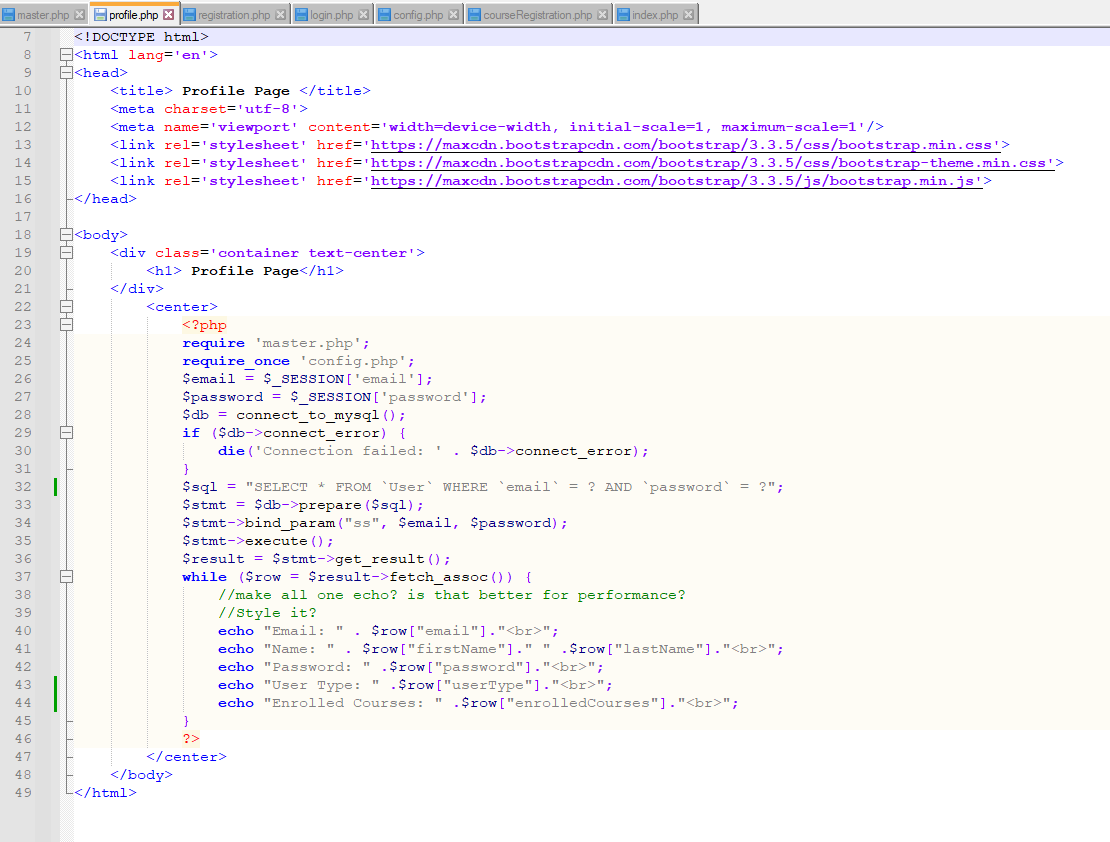
**Master.php Source Code**

**Profile Page**



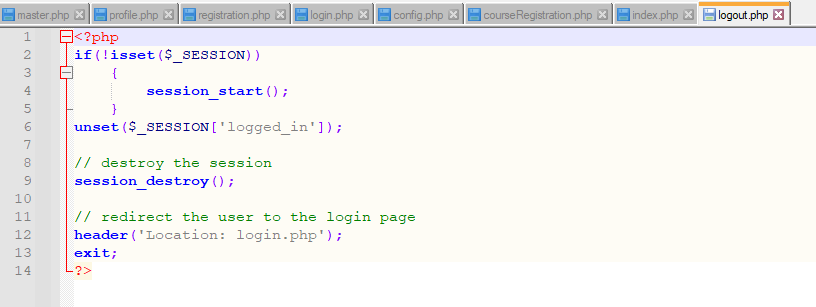
Now that users can only see what they’re allowed to access in the navbar, I created the profile page functionality. In order to implement the profile page functionality, I had to alter my login.php to create the $\_SESSION[‘email’] and $\_SESSION[‘password’] superglobal variables when a user successfully logs into the system. By creating these two superglobal variables, I am able to pass the logged-in user’s email and password into the following SQL query in profile.php: "SELECT \* FROM `tbluser2` WHERE `email` = ? AND `password` = ?";. This query searches the system’s database for all fields associated with the user of that session’s email and password combination. Once all fields associated with the logged-in user have been successfully retrieved, they are now able to be returned to the user in their profile page or profile.php through simple echo statements. I also use this query to fetch the associated records of the query to determine if the User is an Administrator or not, and set the $\_SESSION['Administrator'] session variable to True or False.

**Profile Page Source Code**

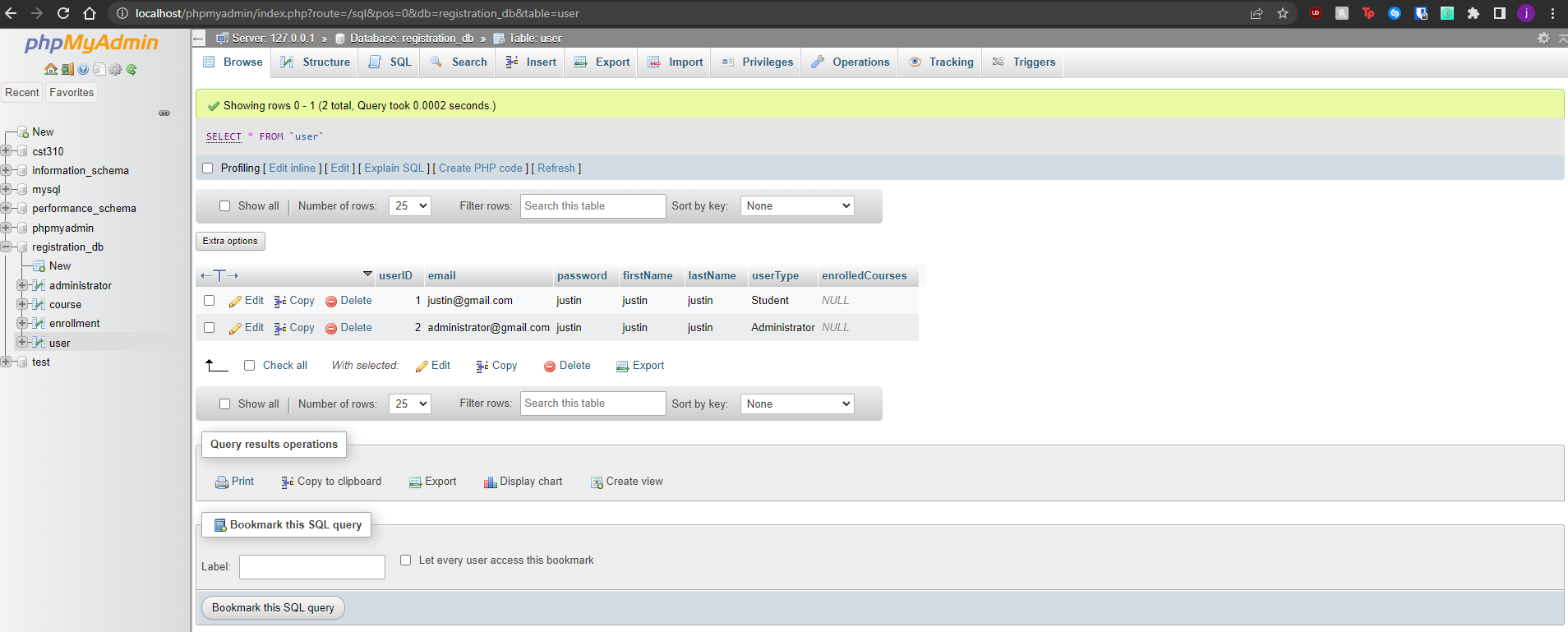
****

With the Profile page complete, the final functionality I implemented was the logout feature. The logout feature was one of the simpler features to implement throughout the whole project, as all the script really does is check if there’s an active session (which there will be if the user is logged in), then destroys the session, unsets the $\_SESSION['logged\_in'] variable, and redirects the user back to the login page through a header() function. With all of these functionalities combined, the online course enrollment system we have created allows users to navigate multiple pages, register an account, log into their account, view their account information, and log out of their account, all while taking the necessary measures to protect our database from various forms of attack, such as SQL injection attacks.

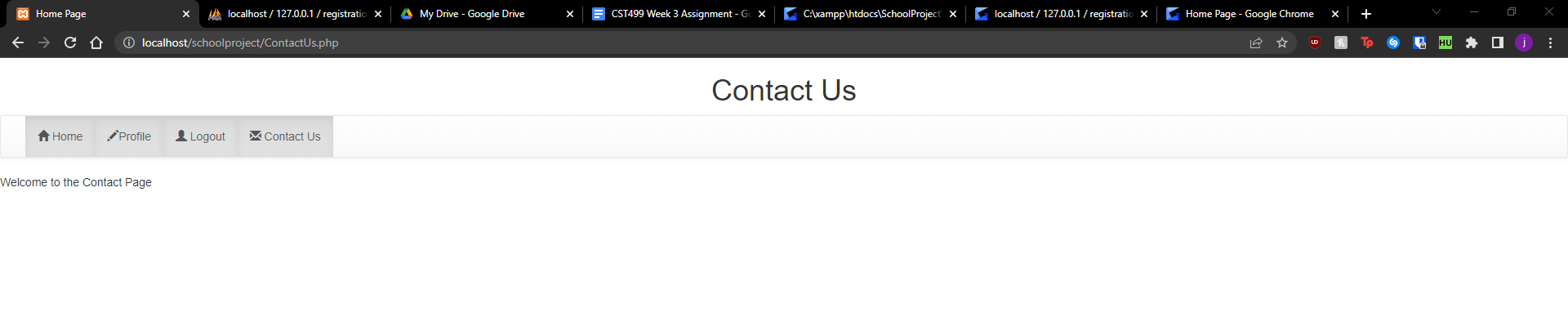
**Logout.php Source Code**

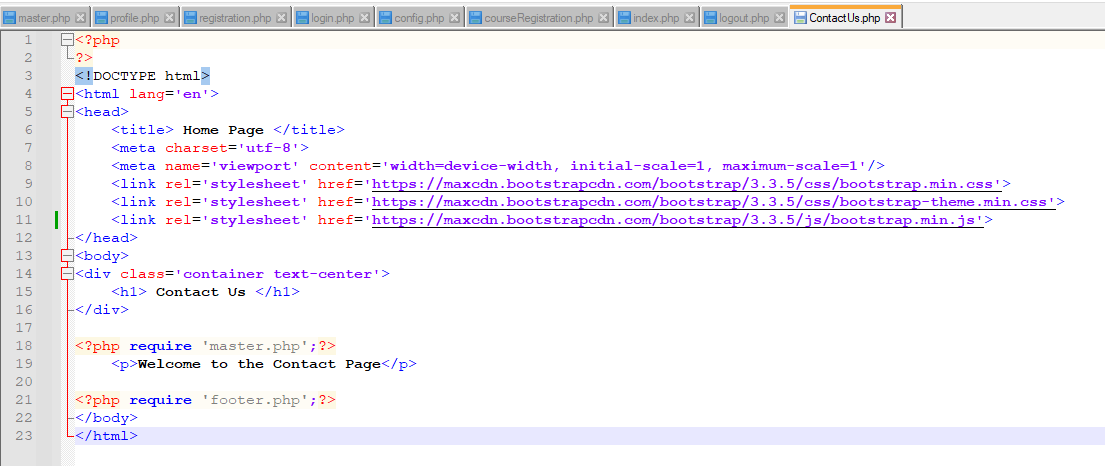
****

**MySQL Database**

****

**Contact Us Page**

****

**Contact Us Page Source Code**