**TalkShop Code Documentation**

The database for TalkShop contains two tables: speakers and teachers. Each table contains the appropriate columns to hold the information for accounts that do or do not display active profiles.

The website is set up using all PHP files. This allows us to break up the code, especially code that would typically be repeated across several files. By doing this, we are able to use PHP include statements instead of using repeated code. This reduces the number of lines of code overall, as well as makes it easier to make changes. A needed change can be made in one spot and can affect many files rather than having to make the change across all of the files individually.



*htmlHeader.php*

A good example of this is shown in the above file. Each file in the website that uses HTML will use this file in a PHP include statement as the very first. It contains the opening HTML and opening head tag. Notice it does not include the closing head tag. This allows us to use this file across the website, but use a different title tag on various pages.



*headerBar.php*

This file contains the code for the bar at the top of each page that contains the logo. The script at the top controls the drop down box for the login form. The first part of the if statement checks to see if someone is logged in, if they are, then it will display that code which prints the logged in user’s first name and well as a drop down setting menu.

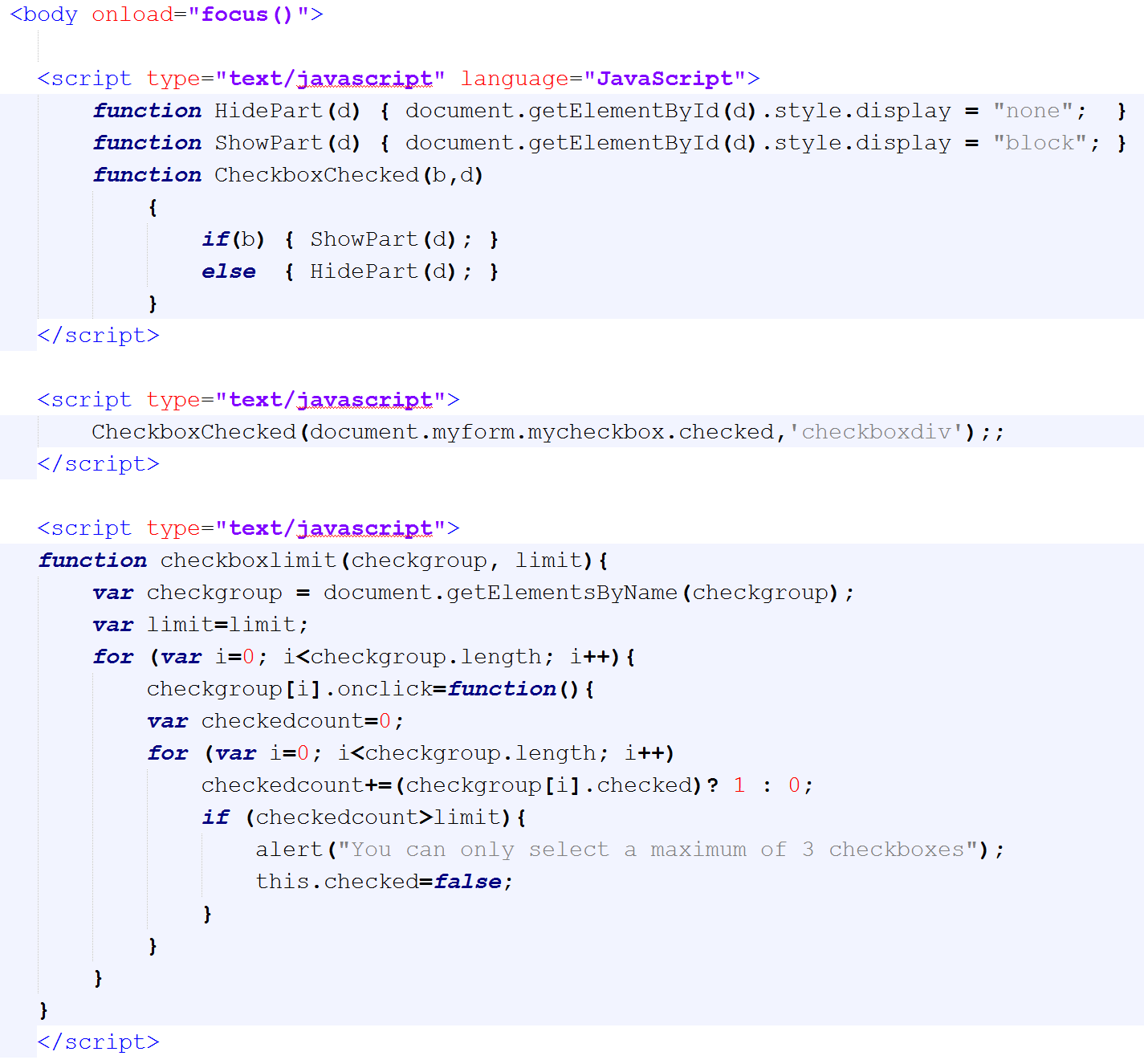
For users that are not logged in:



*headerBar.php*

The else statement is for users that are not logged in. It displays a login and register option.

The files teacherRegistration.php and speakerRegistration.php contain the HTML for the registration forms (too long to picture here). Everything in these files is easy to understand, as it contains mostly basic HTML for form functionality.

*teacherRegistration.php*

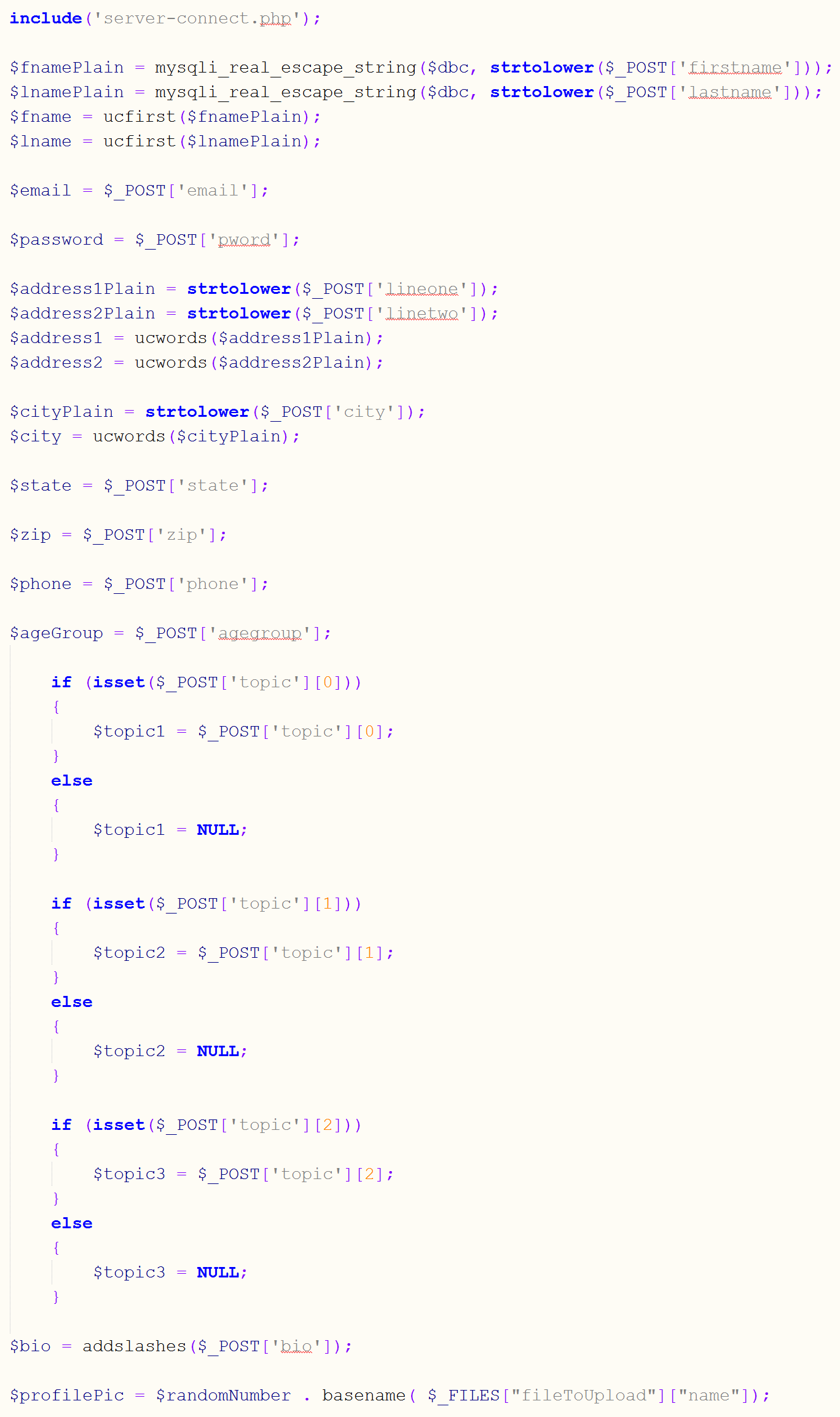
JavaScript functions at the top of these files control the expansion of the form when the user checks the “Make Active Profile” checkbox. The other function ensures that the users are only selecting a maximum of three options in the topic areas section.

Once the user passes validation for either registration form, we use PHP to do a little bit more validation and upload the image. The file opens with an if statement which checks to see if the user clicked to make an active profile. If so, the code to upload the image is executed and the proper variables are set and proper SQL statement is executed.



*teacherConfirm.php*

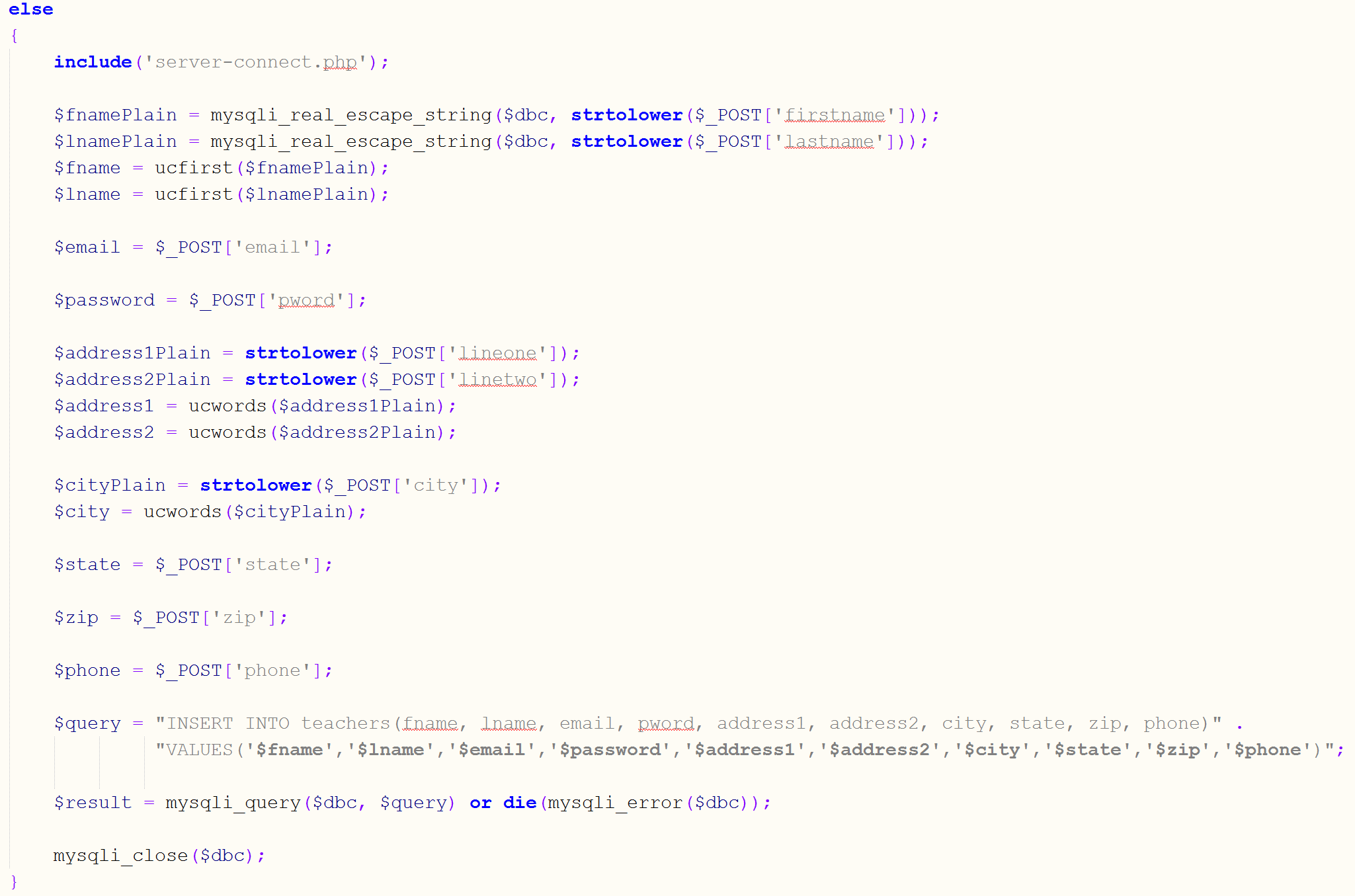
This PHP takes the uploaded image, adds a random number in front of the file name (to eliminate duplicate files names and causing problems). It also chooses the destination to store the file on the server. It will either be a folder named teacherUploads or speakerUploads, depending on which type the user is registering for. Other validity checks include the file size and file format.



*teacherConfirm.php*

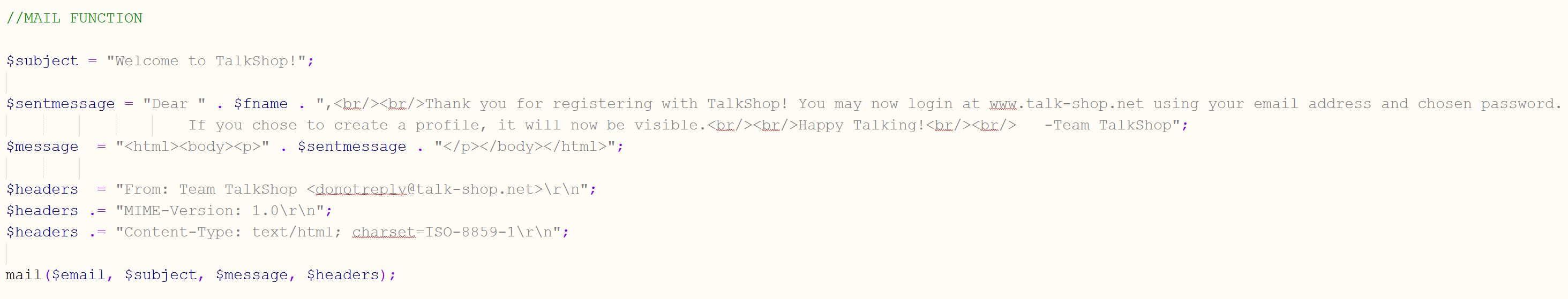
When setting the variables to insert into the database, we needed to make sure they were formatted correctly in the case that users entered all, none, or a mixture of capital letters. For the names, we escape the special characters, make everything lowercase, and then uppercase the first letter. This ensures the names will all look the same. We do the same thing with the address and city to ensure consistency. The if statements for the three topics check for values in the topic array and set the variables accordingly. The PHP addslashes() function for the bio ensures that special characters are escaped and users can input special characters in their bio without causing other issues with the database. The name of the profile picture is given the same name as the uploaded picture with the random number placed in front of the original file name. This ensures the appropriate picture is referenced in the database.

If the user doesn’t choose to create an active profile, then the else statement will be executed that does not try to upload an image and insert variables that will be null:



*teacherConfirm.php*

Once the picture is uploaded (if necessary) and the information is inserted into the database, the user will receive an email to let them know that have been successfully registered for TalkShop.

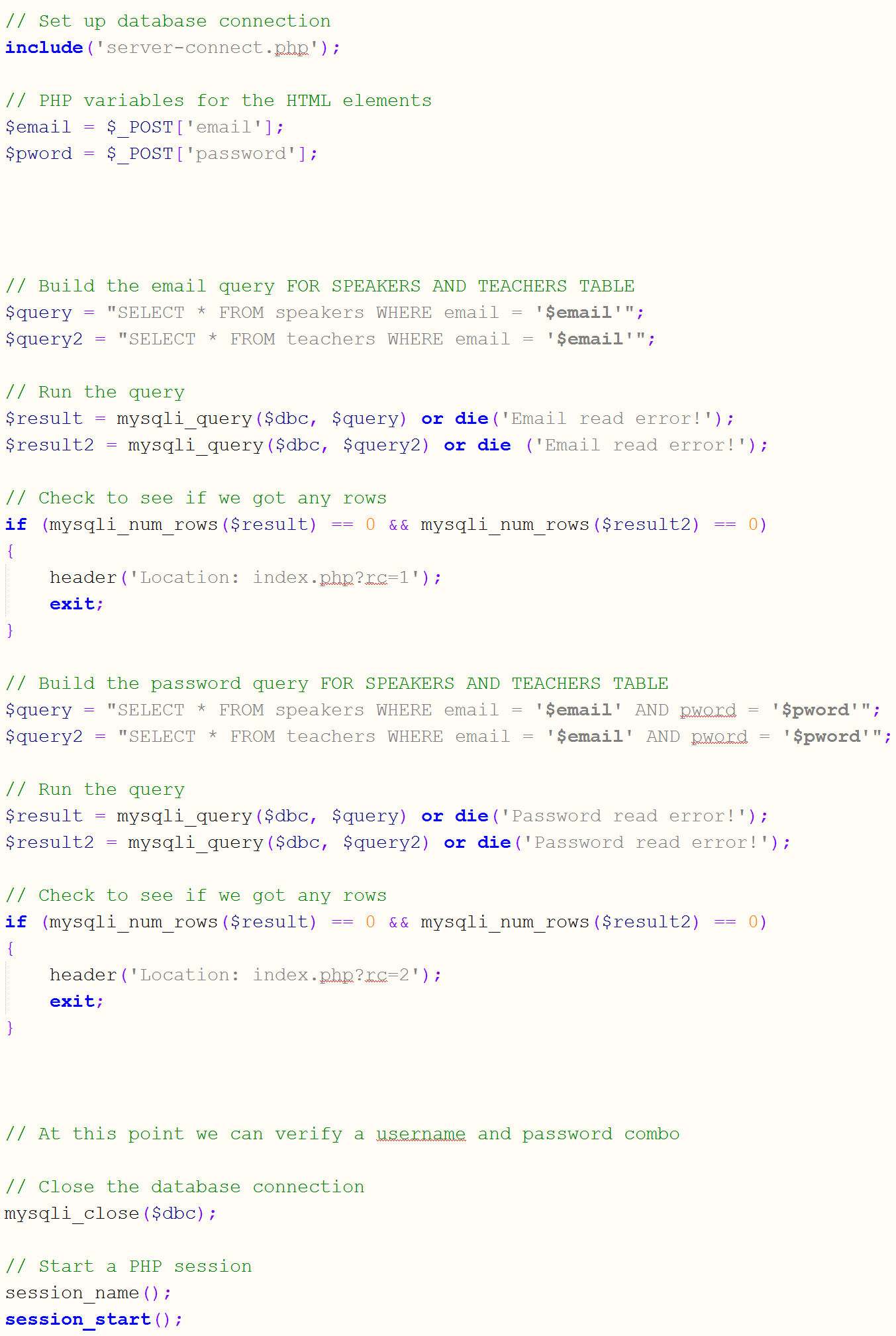


*teacherConfirm.php*

We customized an email message to use the user’s name in order to make the message more personal.

At this point, the user has successfully registered and can now login.

The code for logging in was a little tricky, since we needed to check two tables (speakers and teachers) for an email and password. As shown below, we use two of each variable by simply adding a ‘2’ for the second variable:

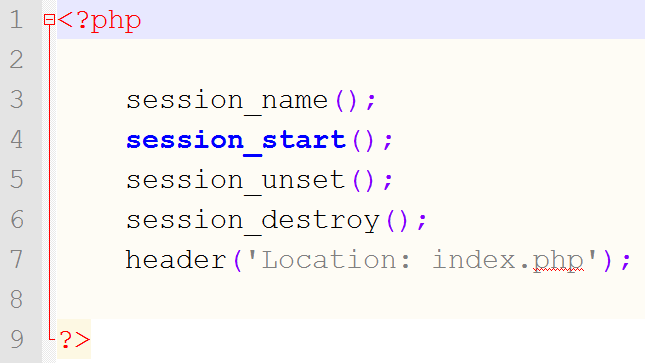
*loginProcess.php*

Two queries: one for checking the teachers table and one for checking the speakers table. The if statement below checks to see which table found a matching email address and password.



*loginProcess.php*

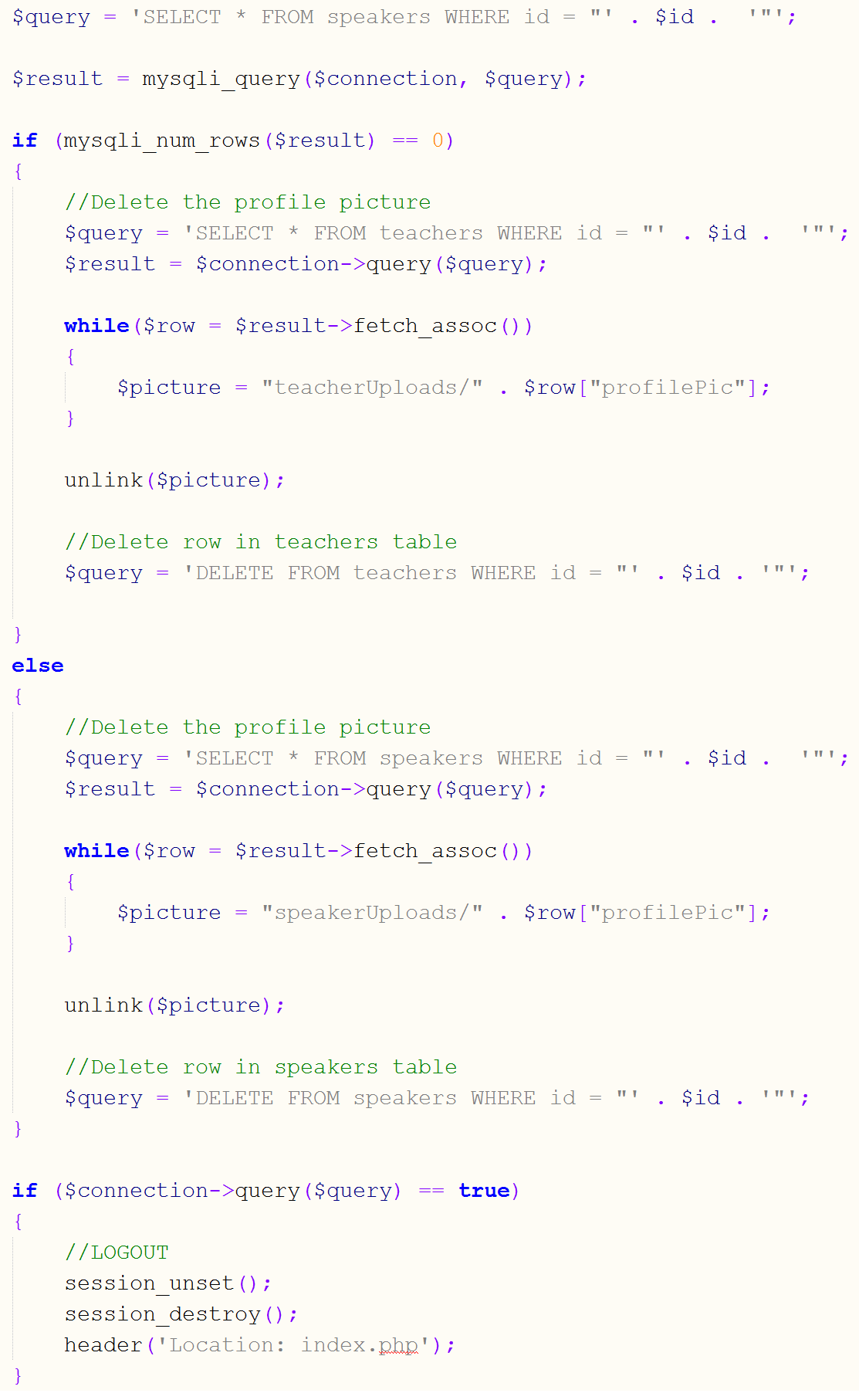
Once a user is logged in, the will have the option to logout or delete their account. If they choose to logout, the file logout.php will run:



*logout.php*

The code if very simple; it simply destroys the active PHP session and loads the homepage.

If the user wants to delete the account, the code is a little more complex. We must check to see which type of account they are registered for in order to delete the information from the correct table (speakers or teachers). To do this, we run a query to see if the account exists in the speakers table. If so, then delete the matching row in the speakers table. If it doesn’t exist in the speakers table, then it must exist in the teachers table.



*deleteLogout.php*

After the appropriate table is decided, the code will delete the picture from the correct folder on the server and delete the row in the correct table. After this, the user is logged out by destroying the active session.

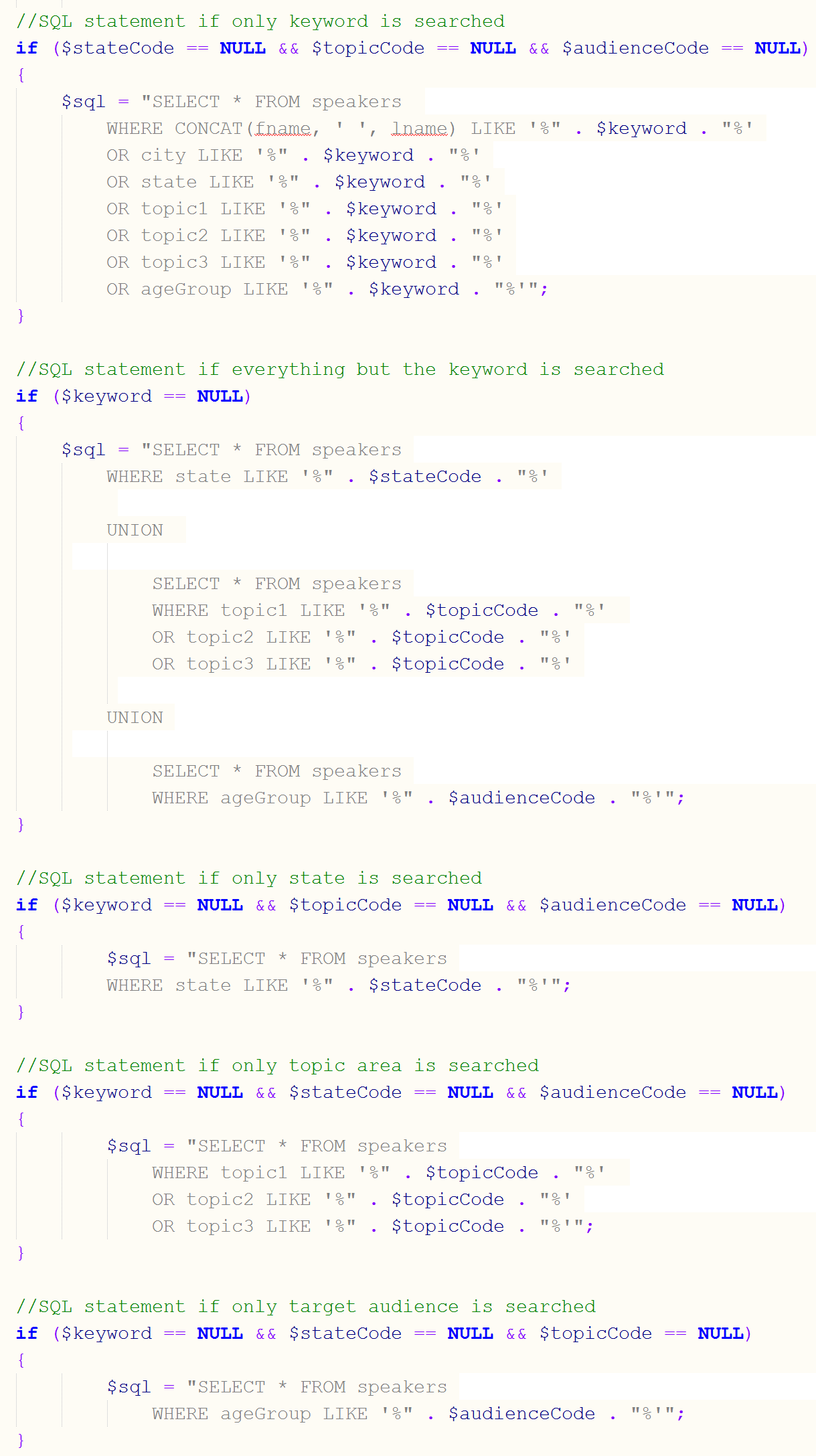
When someone wishes to view and look for listings, they are must choose to view speakers or teachers. The code of each one is about the same, but the main difference is which table the information is getting pulled from. The file that shows all listings is condensedSpeaker(Teacher)Profiles.php. It contains the search form HTML as well as the PHP to display the profiles in a list. The SQL query selects the rows where the profile picture is not null. Since users that do not choose to create an active profile will have a null value in the profile picture column, their information will not be selected to display as a profile.



*condensedSpeakerProfiles.php*

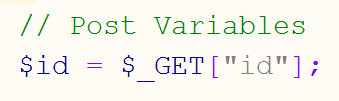
Placing an HTML div outside of the while loop enables the profiles to appear as a list, as all of the information is being pulled from the table.

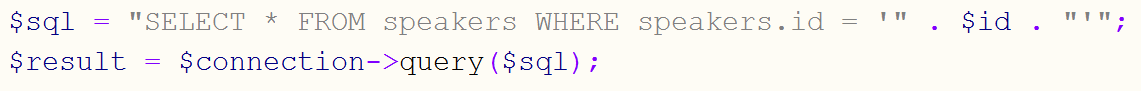
The search form on the left side of the condensed profile listings page allows users to search for a specific person. Upon clicking the Search button, the page refreshes to speaker(teacher)ProfilesSearch.php. This page is the same as condensedSpeaker(Teacher)Profiles.php, but has a different search query to only display the appropriate profiles. We use if statements to determine what fields the user is using to search to run the appropriate query.



*speakerProfilesSearch.php*

When a user clicks on one of the condensed profiles, it will link them to fullSpeaker(Teacher)Profiles.php and pass the corresponding user ID to display the correct information for the full profile. We use GET to get the id from the previous page and run the query to select the row with the corresponding ID.





*fullSpeakerProfiles.php*

The variables are set and then used to fill in the HTML like so:



*fullSpeakerProfiles.php*

If a user wishes to contact a speaker or teacher, the must do so by clicking the contact button at the bottom of the full profiles. The user must be logged in order to send a message:



*teacherEmailForm.php*

The user ID is passed to the email page as well, as shown previously. It displays a simple form that allows a user to type a message and then send the message. Upon clicking send, the code in confirm.php executes. Confirm.php works with both speakers and teachers since the email setup is the exact same for both situations. The email function is a little more complex than the other email function because of the setting and use of many variables to make the email personal.



*confirm.php*

Upon the sending of the email, the user sees a confirmation page for the sent email.

**Important GoDaddy Information**

GoDaddy Account:

USERNAME: talkshopgroup

PASSWORD: Asu275833

CUSTOMER #: 98555524

PIN: 3751

EMAIL: [jkhumpherys@gmail.com](mailto:jkhumpherys@gmail.com)

Linux cPanel:

USERNAME: talkshopadmin

PASSWORD: Asu275833

Email Accounts:

ACCOUNT: info@talk-shop.net

PASSWORD: Asu275833

INCOMING SERVER: p3plcpnl0180.prod.phx3.secureserver.net

OUTGOING SERVER: p3plcpnl0180.prod.phx3.secureserver.net

ACCOUNT: DoNotReply@talk-shop.net

PASSWORD: Asu275833

Connect to Database (GoDaddy):

$servername = 'localhost';

$username = 'talkshopconnect';

$password = 'Asu275833';

$db = 'talkshop';