

E-GraB Electronic Grade Book

**Instructor: Amer Hadzikadic**

**Authors: Subic Amna**

**Lokmic Kaid**

**Pekaric Edah**

**Sarajevo School of Science and Technology**

**Sarajevo, October/January 2017/2018**

Table of Contents

1 Introduction 3

2 ERD 4

3 Sessions 6

3.1 Login\_page.php 7

3.2 Session.php 8

4 Profiles 9

4.1 Student profile 9

4.2 Professor profile 12

4.3 Admin profile 14

5 ADD 15

4.1 ADD Professor 15

4.2 ADD Student 16

4.3 ADD Course Assignment 16

6 EDIT 17

4.1 EDIT Professor 17

4.2 EDIT Student 18

4.3 EDIT Course Assignment 19

7 DELETE 20

4.1 DELETE Professor 20

4.2 DELETE Student 20

4.3 DELETE Course Assignment 21

8 VIEW ALL 22

4.1 VIEW ALL Professors 22

4.2 VIEW ALL Students 23

4.3 VIEW ALL Course Assignment 24

9 25

# 1 Introduction

In Bosnia and Herzegovina, schools use paper grade books for taking attendance and for assigning grades to students. Due to this, we decided to create a system which will contain grades and attendance of students. Our goal is to make a system that will import and export grades and attendence of the students. Our database system has 8 tables, connected with each other.

The Electronic Grade Book Project “E-GraB” is focused on providing students and professors with readily information of students enrolled at the university, such as grades, attendance, etc. This project includes: profiles for students, professors and admin with unique content and options. The student profile allows enrolled students to view their personal data, enrolled classes, as well as grades and attendance related to those classes. In addition, they are able to see all their colleagues that are taking the class with them and course information. The professor profile allows professors to also view their personal data and courses that they are assigned to teach. However, unlike the students, professors can assign students to their courses, add attendance, and grades. The admin profile, as the name suggests, allows only one person, which is selected as admin to add students and professor, view all students and professors, as well as assign courses to professors.

Our initial plan was to also make parents involved in their offspring’s education. Parents were supposed to be able to see all their children and their enrolled courses, including attendance and grades. However, while in the process of developing the site and database, we came to the conclusion that there were to many components in the project itself and that we would not be able to complete everything by the set deadline.

To conclude, this report will focus on the database part of the project. All SQL queries used in this project and ER – diagram of the database will be described to the reader. Furthermore, we will provide screenshots of the implementation of those queries in the PHP.

# 2 ERD

# 

1. «Departments» have many «students», «professors» and «course assignments». «Students», «professors» and «course assignments» belong only to one «department».
2. «Professors» can have many «course assignments», but one «course assignment» can belong to only one «professor».
3. «Course assignments» and «students» can have many «enrollments», but one «enrollment» belongs to only one «course assignment» and only «one student».
4. «Enrollments» can have many «grades» and «attendance». One «attendance» belongs to one «enrollment». One «grade» belongs to one «enrollment».

Students and professors have similiar attributes, and to avoid redundancies, could be placed in one table. However, since «professors» have «course assignments», which are not directly connected to «students», and «students» have «enrollments» that are not directly connected to «professors», we decided to keep these two tables separate. In addition, admin is stored in «professors» table as a BOOLEAN variable, which is obtained in php through a sql query, to check which profile and options will be passed. Alongside the admin in «professors», «students» have two columns that are not found in «professors» table, and those are mother and father. For these reasons, we decided to keep these tables seperate, and thus somewhat make the retrieval of data easier.

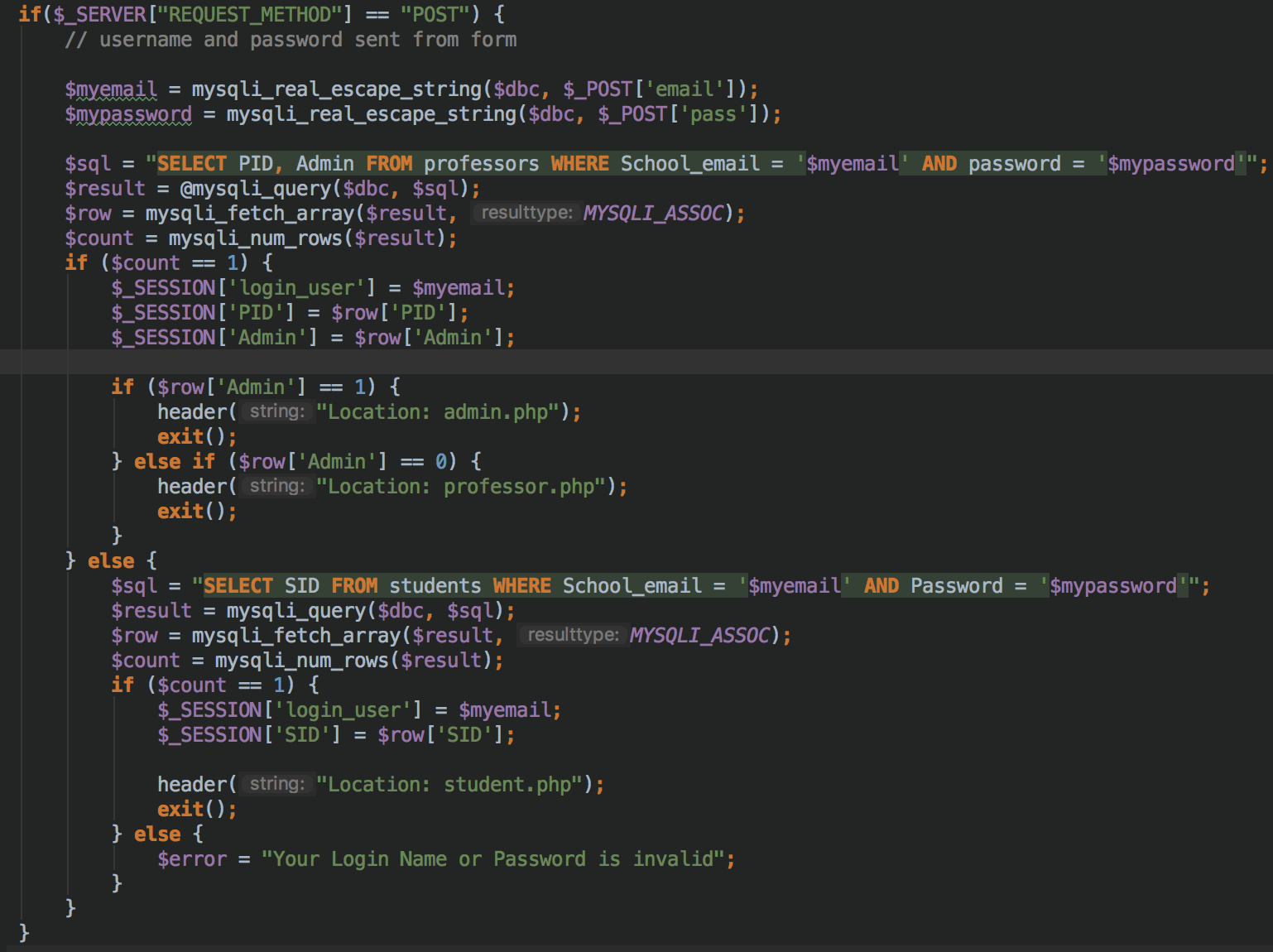
3 Sessions

To make data available for the user on multiple pages of Web site, required of us to make sessions or cookies. Sessions allow data to be stored on the server, not in the Web browser, and session identifier is then used to locate a particular user's record. Sensitive data, such as user's ID, name and etc., always remains on the server, while the identifier is stored in user's Web browser via a cookie. When choosing between cookies and sessions, we opted out for sessions, since they are more secure. Sessions store all recorded information on the server, while cookies send this information back and forth between the server and the client. In addition, sessions provide more storage for data and, while they are designed to work with a cookie, they can work fine without them, if the user chose to not accept cookies or turned them off.

In order to make sessions work, we needed three files: login\_page.php, sessions.php, logout.php, alongside the connection file for the database (which is included in most files). Logout.php, as the name suggest, terminates the session, thus signing the user out. This file does not use any queries.

3.1 Login\_page.php

To create sessions we first needed to create a login page which contains login form and a loop with queries that checks all the inputs.

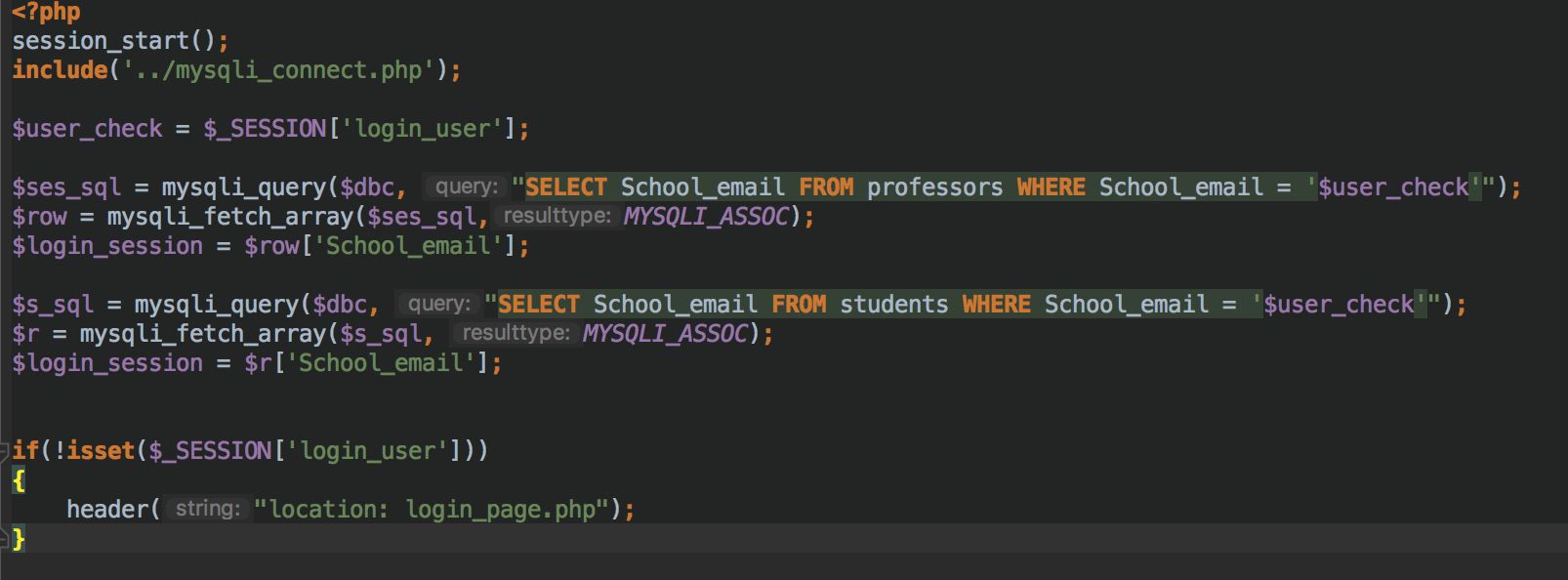


The user enters email and password. Variables $myemail and $mypassword are set with mysqli\_real\_escape\_string command. The sql query then selects «professor» primary key and Admin (boolean variable) form table «professors» if the email matches School\_email and if password matches password from the database. If the result generates one row it will proceede to the next loop where it will check whether professor is admin or not. If admin is true, user will be taken to admin's page, else the user will be redirected to professor's page.

If the first query failed to generate results, another query will be executed. This time it will select «student» primary key from table «students» if the email matches School\_email and if password matches password from the database. If it generates a result, user is redirected to student's page. If there is no result then email and password were invalid.

3.2 Session.php

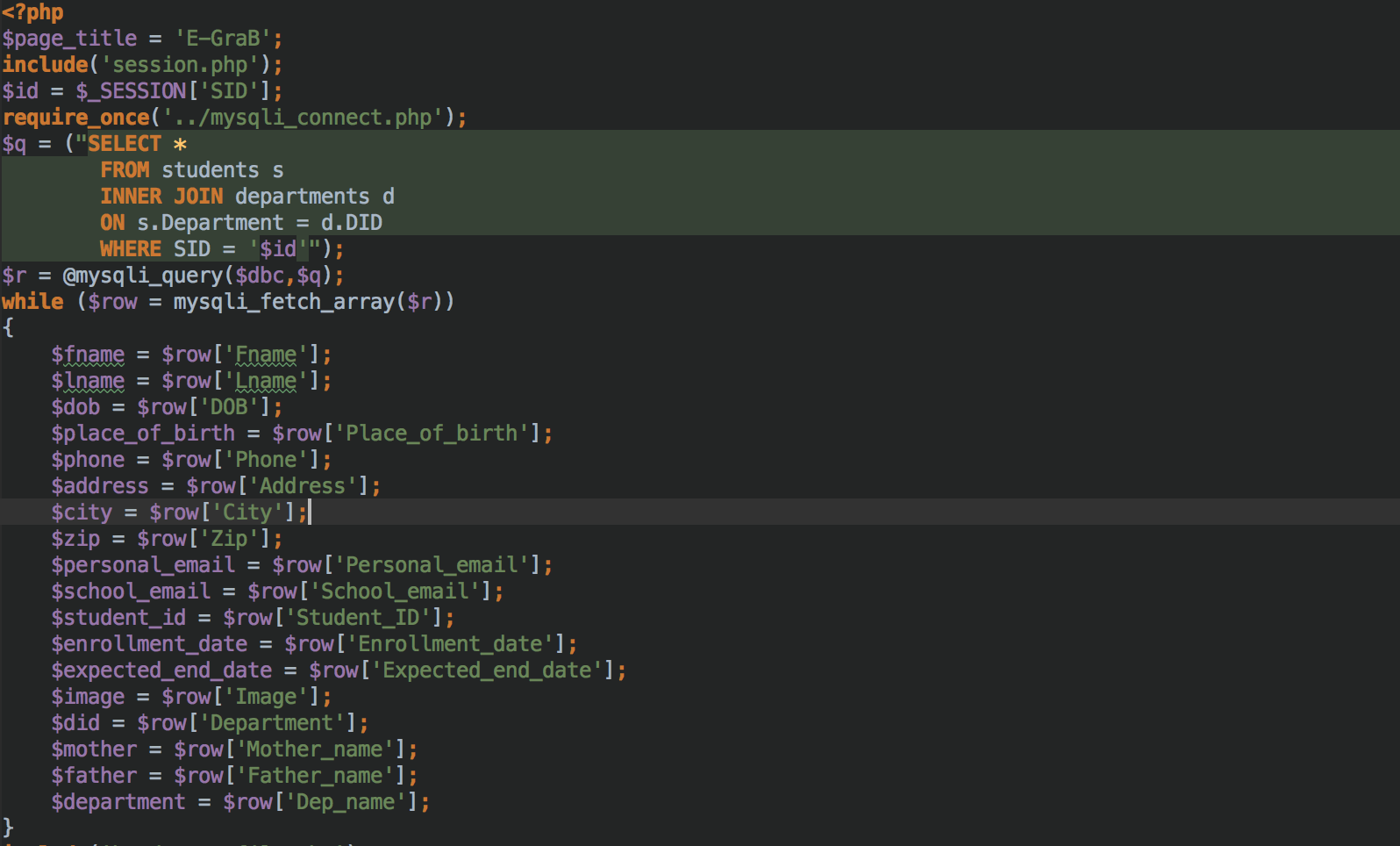
In login\_page.php, after successfully generating a result, session data has been set. Now, in session.php, we check that data once more.



We have set a variable $user\_check and passed on the value of session data - $\_SESSION['login\_user']. Now we start two queries, one for a student and one for a professor, which check whether school emails match variable $user\_check. If not, the user will be redirected to the login page.4 Profiles

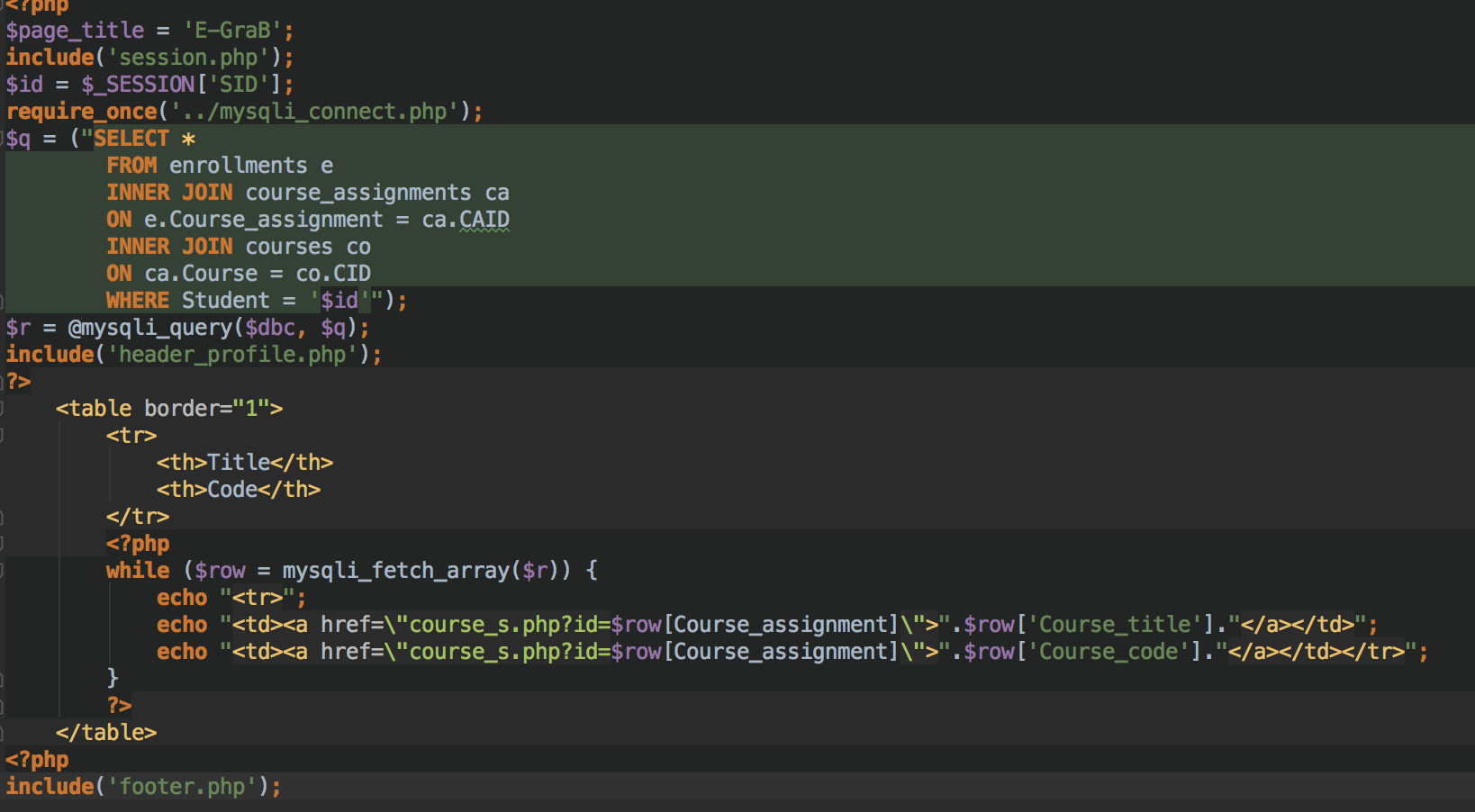
**After successfully loging in, the user is redirected to the appropriate page. As mentioned earlier, each page has its unique data and options.**

****4.1 Student profile****

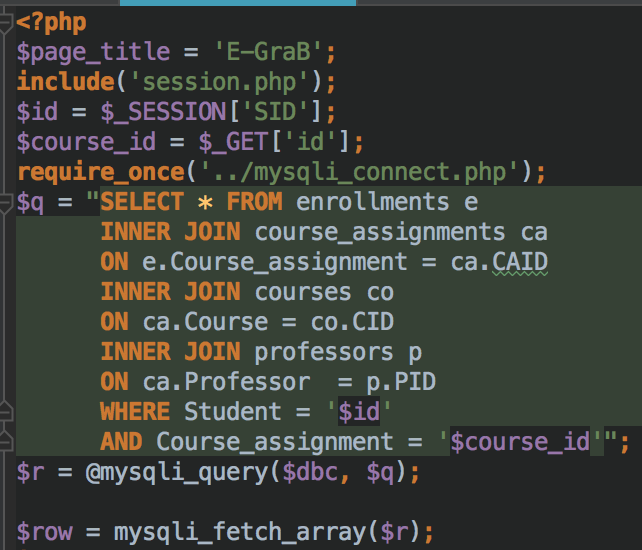
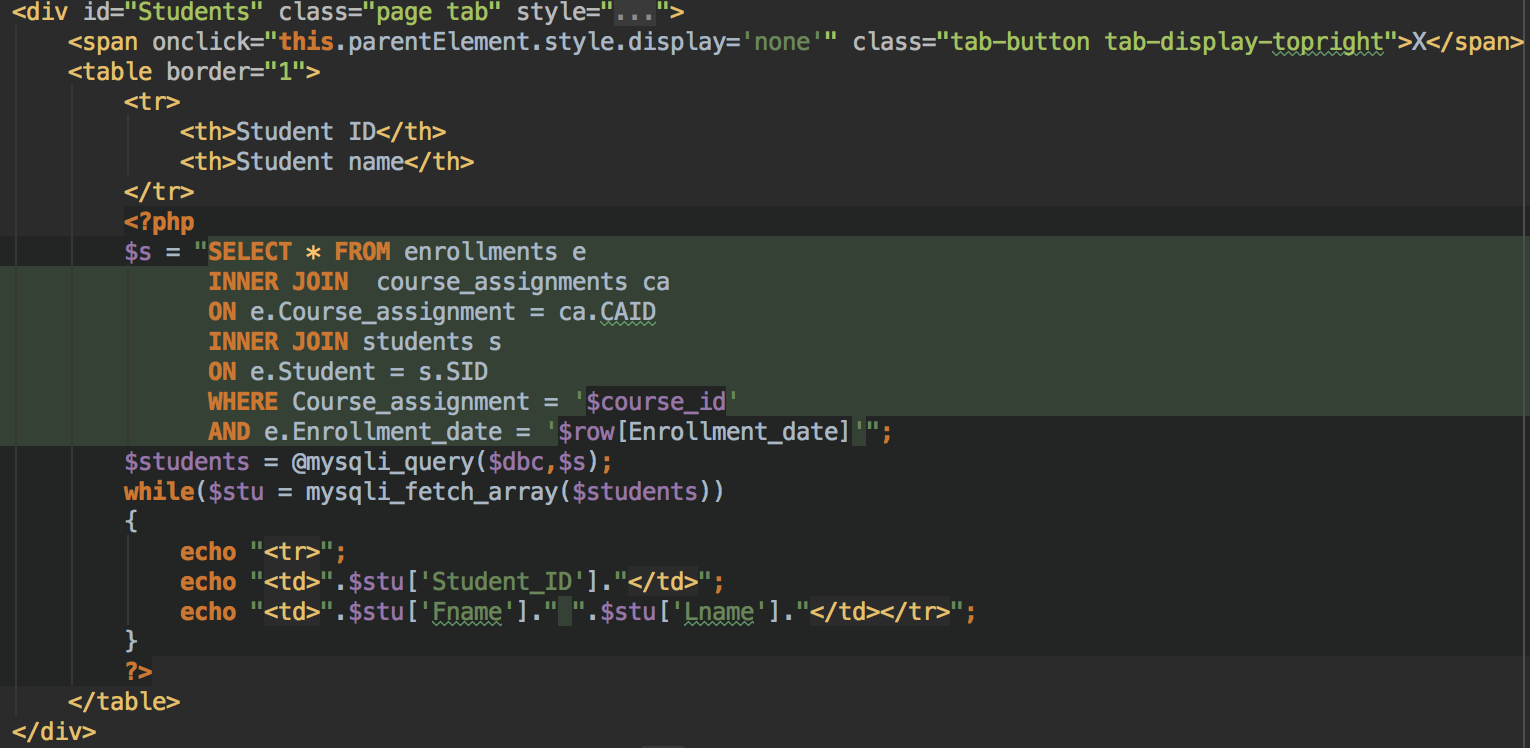


We will start off with the student profile. After starting its session, variable $id retrieves session data, which is student primary key. It then runs a query, which select everything from table «students» and INNER JOINS it with table «departments' ON s.Department = d.DID WHERE «students» primary key (SID) matched the variable we set at the begining of the session. After runing a query, it then sets all variables seen in the while loop and echos them in their assigned places in the student profile.

Every profile has a side navigation, that contains buttons according to the type of profile, in this case: profile (leads back to student profile) and courses(lead to all enrolled courses).

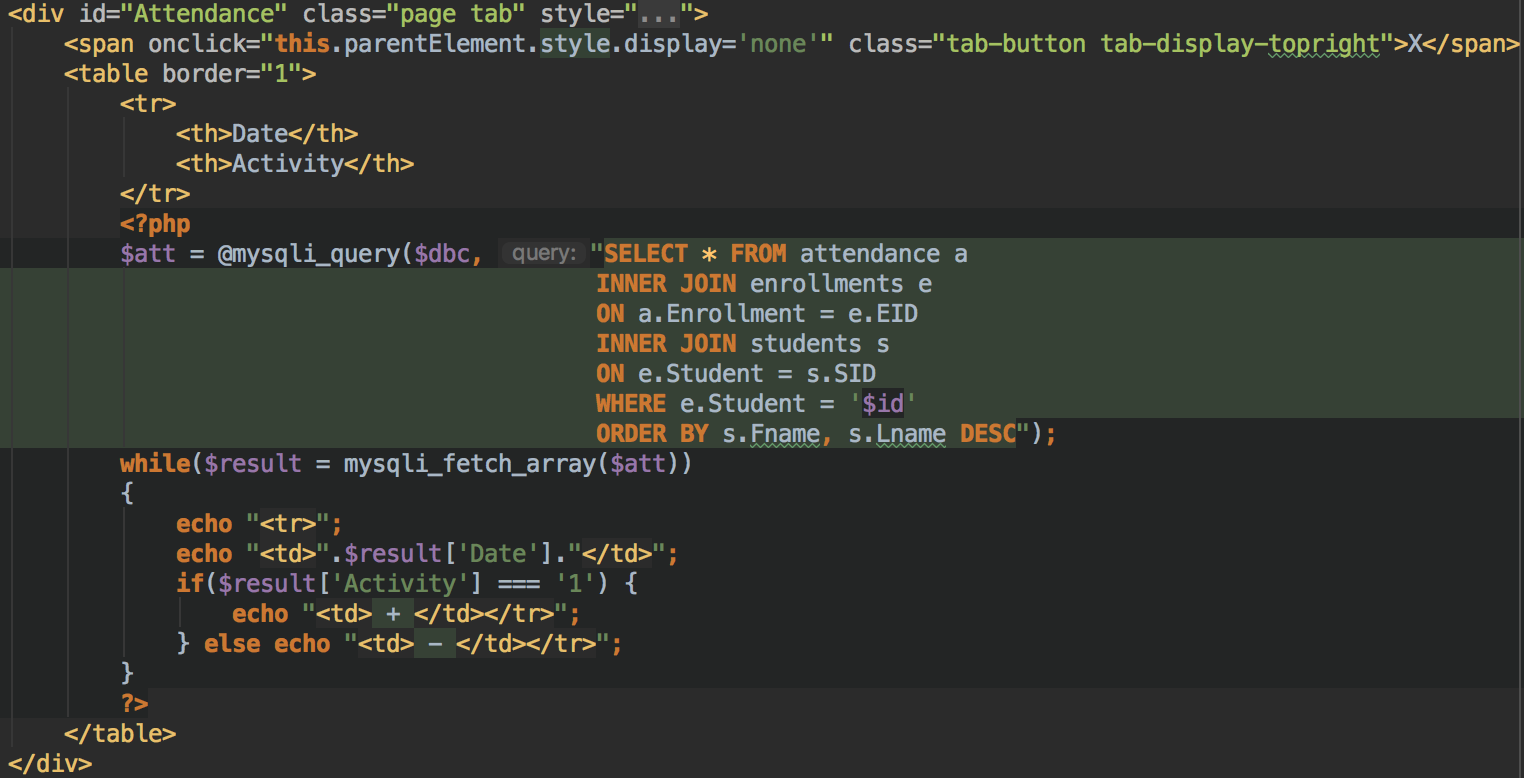


When courses button is clicked, it leads us to a page containing all student's enrolled courses. A query is run which selects everything from «Enrollments» and INNER JOINS it with «Course assignments» ON e.Course\_assignment = ca.CAID and also INNER JOINS with «Courses» ON ca.Course = co.CID WHERE Student matches set session data (in our case student primary key). After that a loop is run which displays Course title and Course code as a link which directs us to appropriate course page.

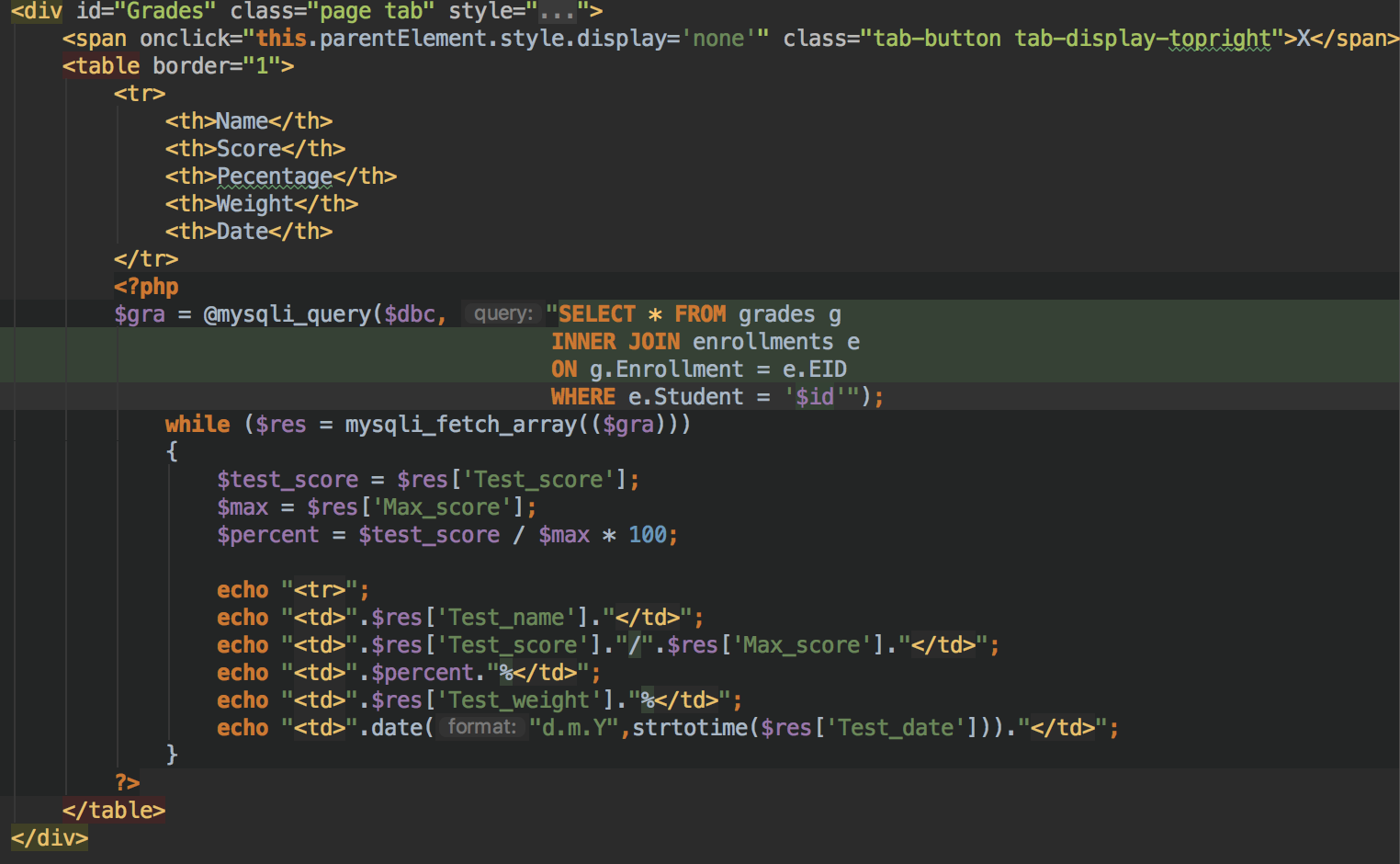
 

After a course is clicked, we are redirected to course page that contains four tabs: course info, students, attendance and grades. In the pictured listed above, we can see similar queries being run with INNER JOINS WHERE Student must be equal to session data variable ($id) and Course assignment must be equal to variable $course\_id. The reason behind this is, that in the link we have set the destination to have id which equals to the appropriate «course assignment» key.

The first query allows us to display the course information. The second query does another INNER JOIN with «course assignments» and «student» on appropriate primary/foreign keys WHERE course\_assignment must equal $course\_id and enrollment\_date must be equal to the result of Enrollment date from the first query. That is, all students must have the sam enrollment dates as the student that is currently in session.



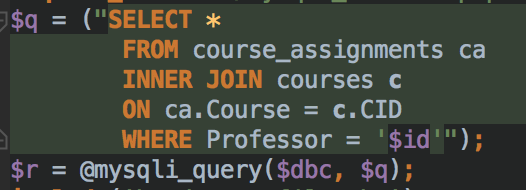
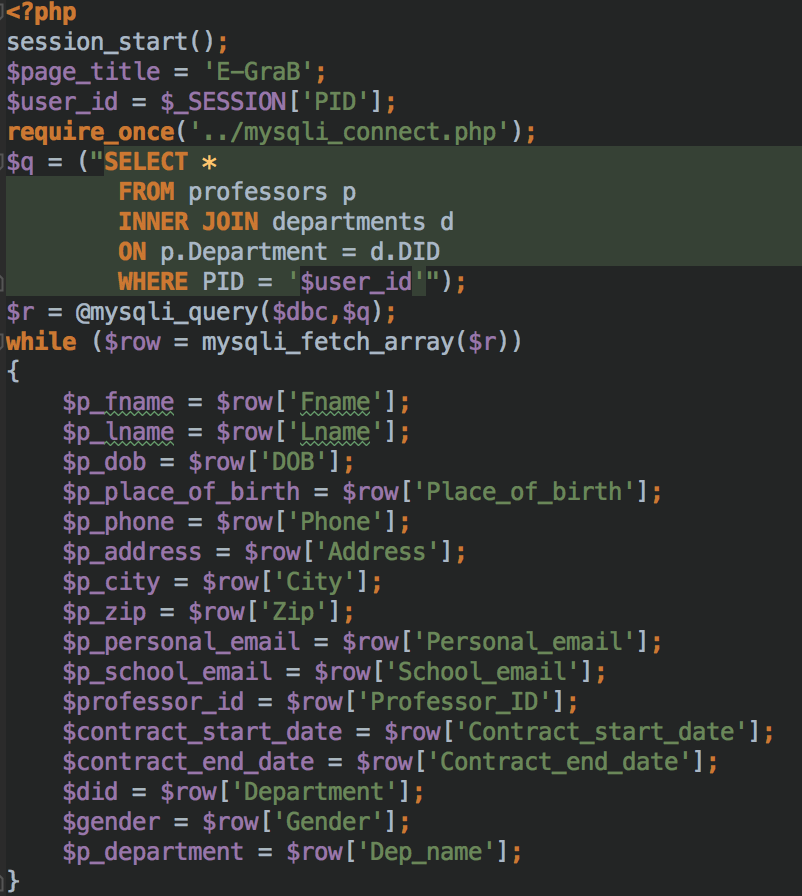
Attendance tab displays all attendance for the currently dispalyed course. It runs a query that select everything from «attendance» and INNER JOINS «enrollments» ON a.Enrollment = e.EID and «students» ON e.Student = s.SID WHERE e.Student = $id (session data) and orders them by student First name and Last name in descending order.



Grades tab displays all grades for the currently selected course. The query again selects everything from grade, does an INNER JOIN on enrollments and checks whether Student = $id (session date).

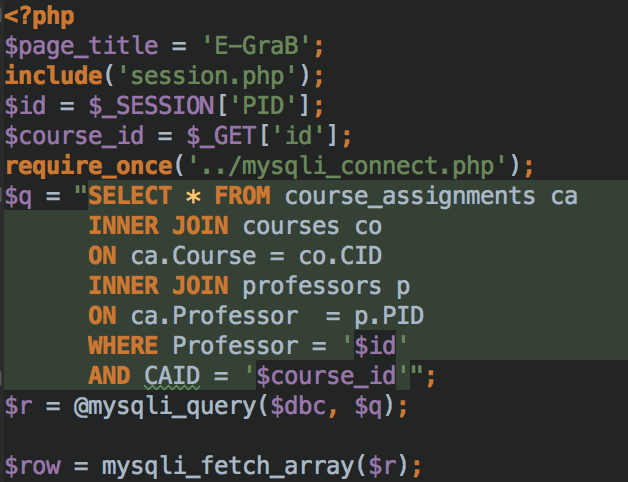
4.2 Professor profile

Professor profile includes mostly similar queries and options. The professor profile was planned to have an option to add students to courses, add attendance and grades fro students, however, we failed to implement these features correctly. As with students, professor page has profile button that returns to professor profile page and courses button that redirects to page with all courses which that professor teaches.

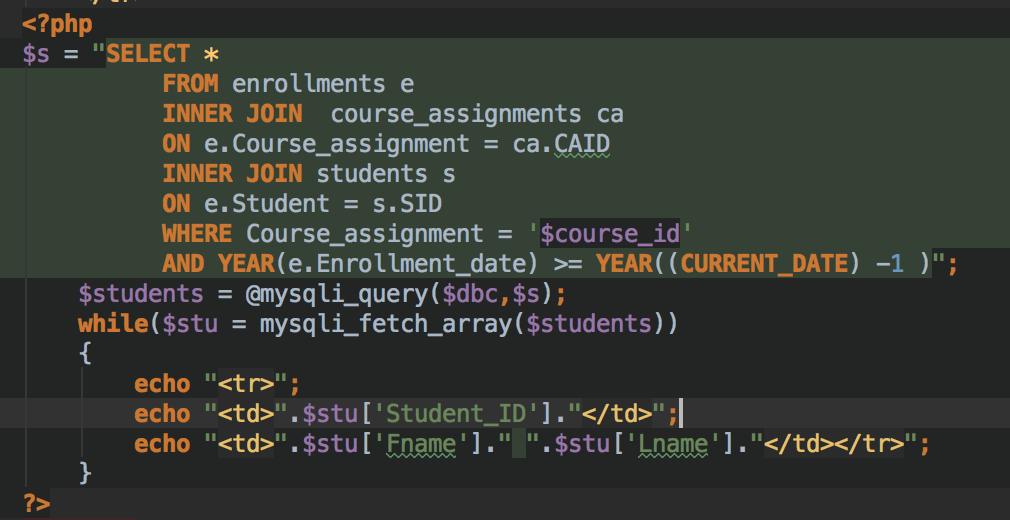


Query for Professor information Query for all courses of the professor

The code after the query for courses is the same as with the student, since it loops the result and displays courses with codes one by one as a link. Again the links use id which equal to course assignment key, so that we can easily view and update data (if possible).



Query to get course info for professor



Get all students enrolled in this course for the current professor in the last year

4.3 Admin profile

Admin profile is mostly for managing the database. It includes the following options:

1. add professor
2. add student
3. add course assignment
4. edit professor
5. edit student
6. edit course assignment
7. delete professor
8. delete student
9. delete course assignment
10. view all professors
11. view all students
12. view all course assignments

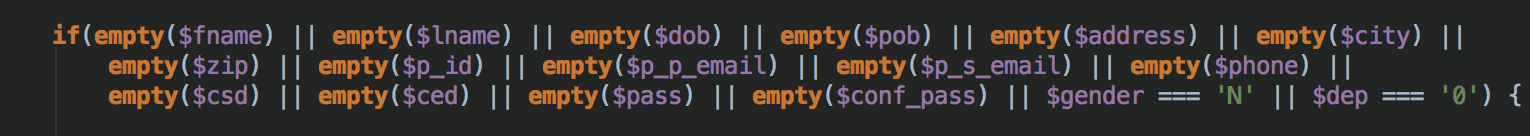
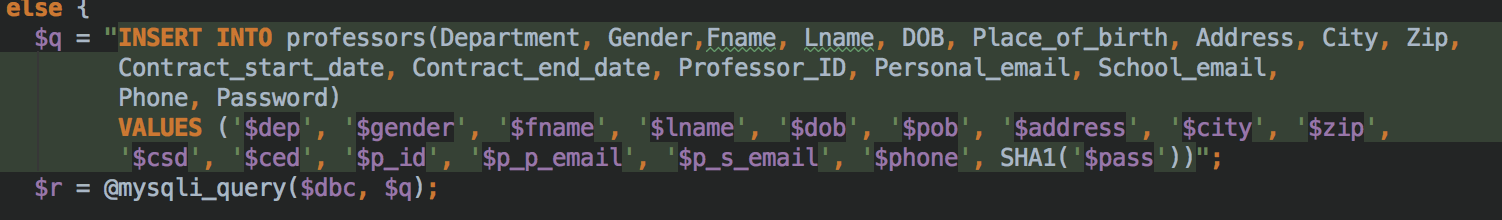
These groups will be explained individually in the next part

# 5 ADD

In this part we will explain how students, professors and course assignments are added into the database. After the forms are filled out, values are passed on to a separate file that checks whether the input is valid and if it is submits it to the database.

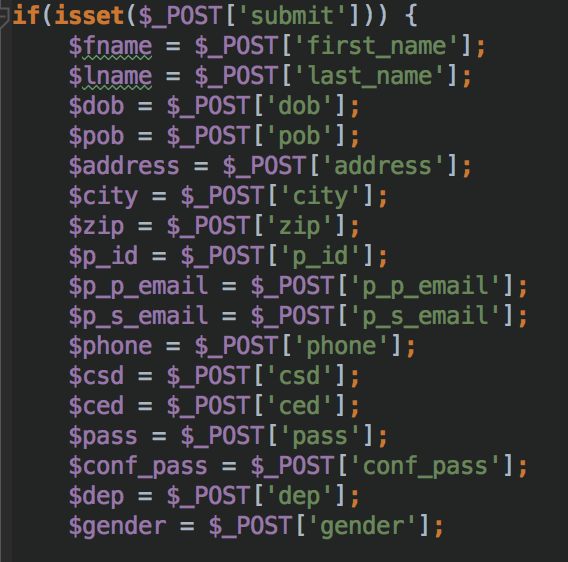
5.1 ADD Professor

3. IF NO ERRORS OCCURED INSERT DATA INTO PROFESSORS



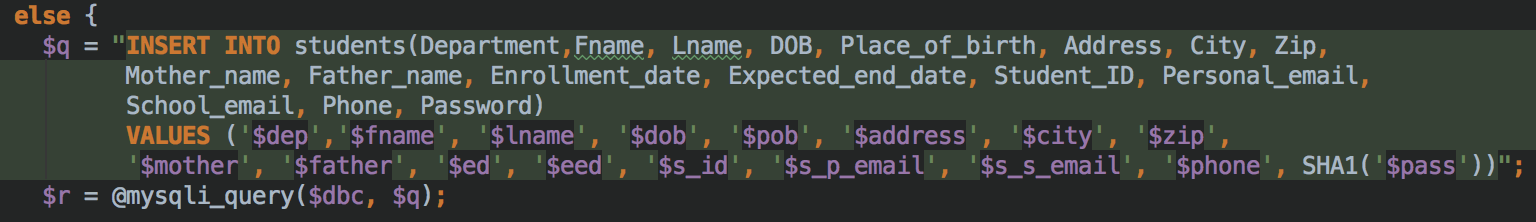
2. CHECK FOR ERRORS

1. GET ALL INPUT VALUES AS VARIABLES



5.2 ADD Student

First two parts of code for «add student» are the same as in «add professor». The only parts that differ are two variables, but the concept is the same. The query changes, now all data is inserted into «students».



5.3 ADD Course Assignment

The concept behid adding course assignment is the same as with student and professor. The only thing that differs is that they have less data to input, making the error check smaller and the data is added to «course assignments table».



# 6 EDIT

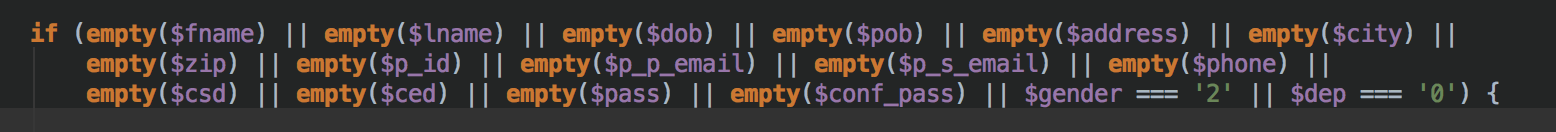
In this part we will explain how students, professors and course assignments are edited. After the update button is pressed, all current values are obtained from the database and placed into the form and the admin is allowed to change those values that he wishes. After inputing new values, confirm password is required to submit all changes.

6.1 EDIT Professor

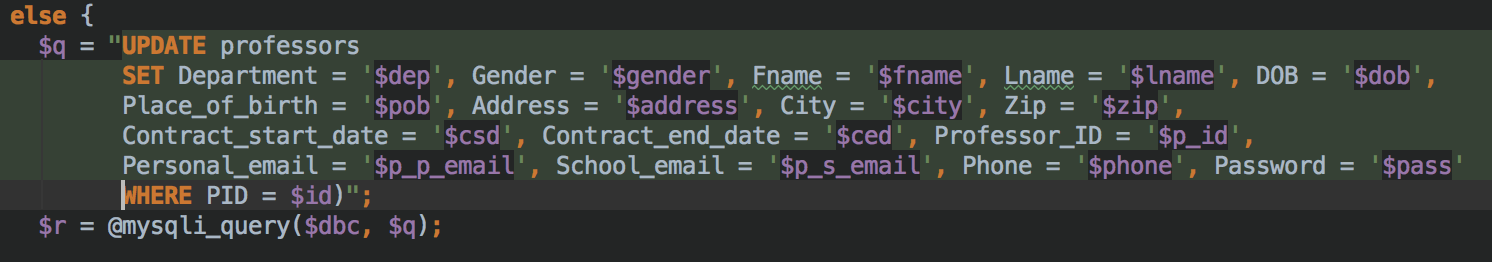
As with ADD option, values are passed into variables.



Then the program checks for errors. If there are no errors occuring the program continues on to the UPDATE statement.



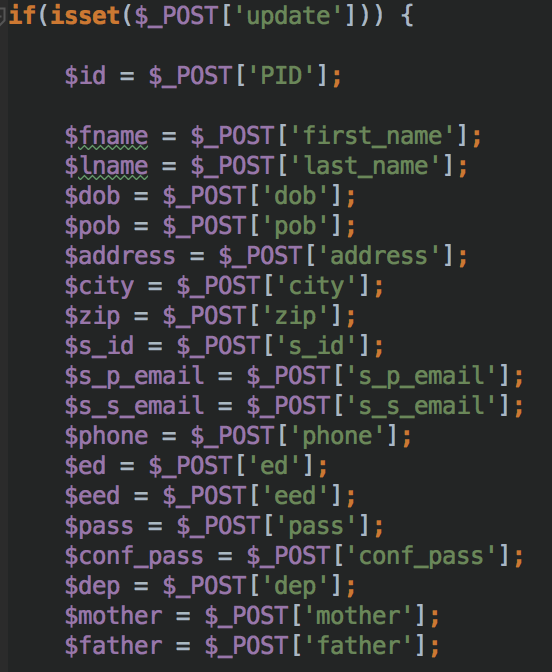
(loop checks for errors)



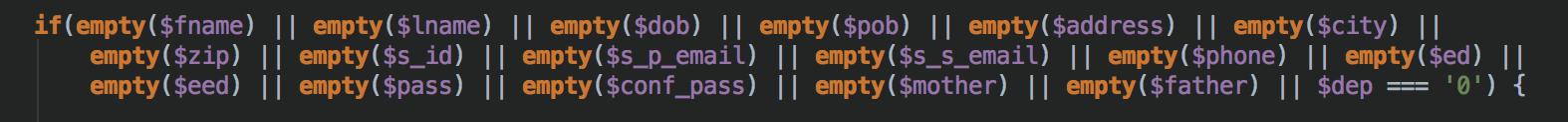
(query for updating professors if there are no errors)

EDIT Student and EDIT Course Assignment are almost the same, except for the variables and the table which is UPDATED.

6.2 EDIT Student



(values passed on into variables)

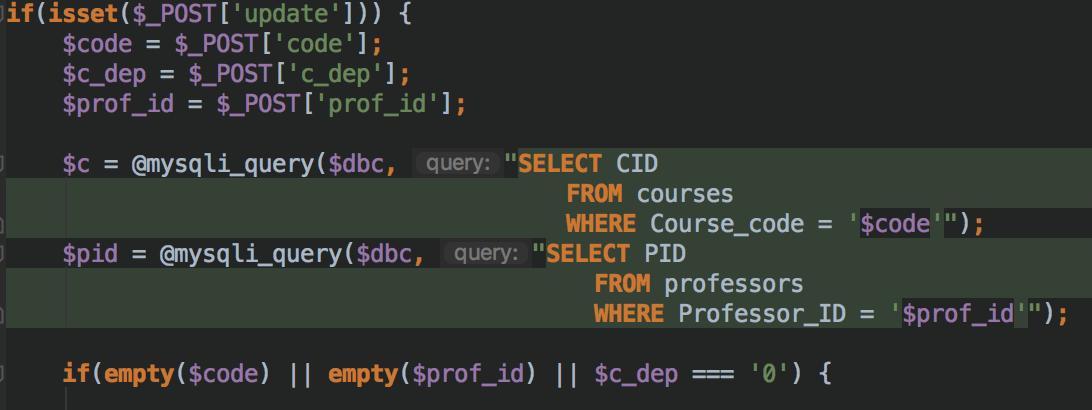


(loop checks for errors)

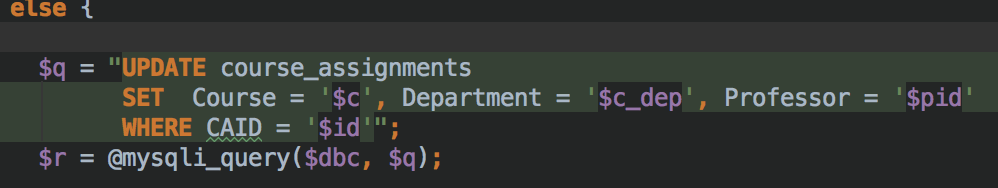


(query for updating students if there are no errors)

6.3 EDIT Course Assignment



(values passed on into variables and loop checks for errors)

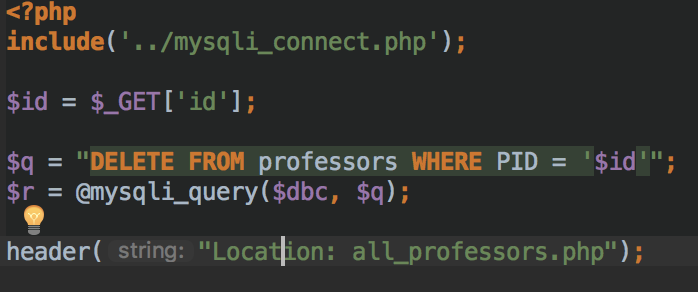


(query updates Course Assignments if there are no errors)

# 7 DELETE

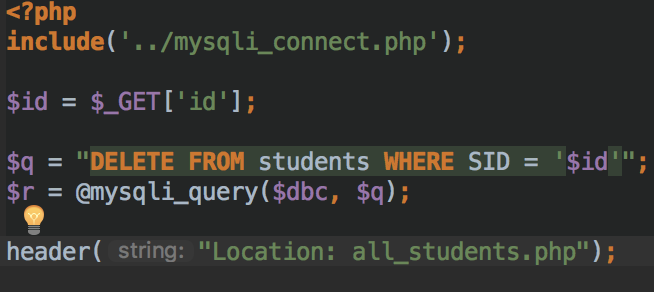
In this part we will explain how students, professors and course assignments are deleted from database. After the delete button is pressed, a pop-up bubble appears prompting us to to press either 'OK' or 'Cancel'. I we choose OK the entire row (student/professor/course assignment) will be deleted.

7.1 DELETE Professor



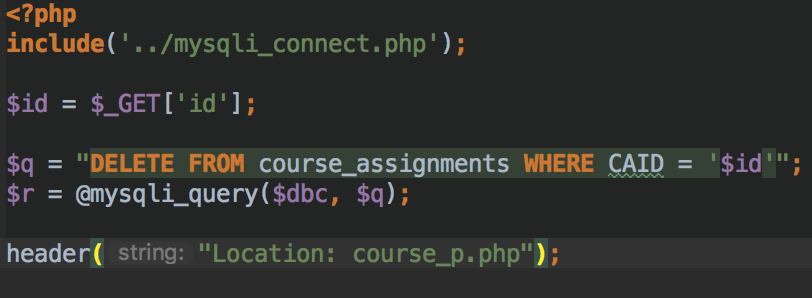
(professor is deleted if his primary key is equal to $id = id from url)  
delete\_p.php?id=$row['PID]

7.2 DELETE Student



(Student is deleted if his primary key is equal to $id = id from url)  
delete\_s.php?id=$row['SID']

7.3 DELETE Course Assignment

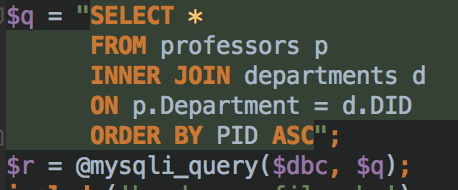


(Course Assignment is deleted if his primary key is equal to $id = id fro url)  
delete\_ca.php?id=$row['CAID']

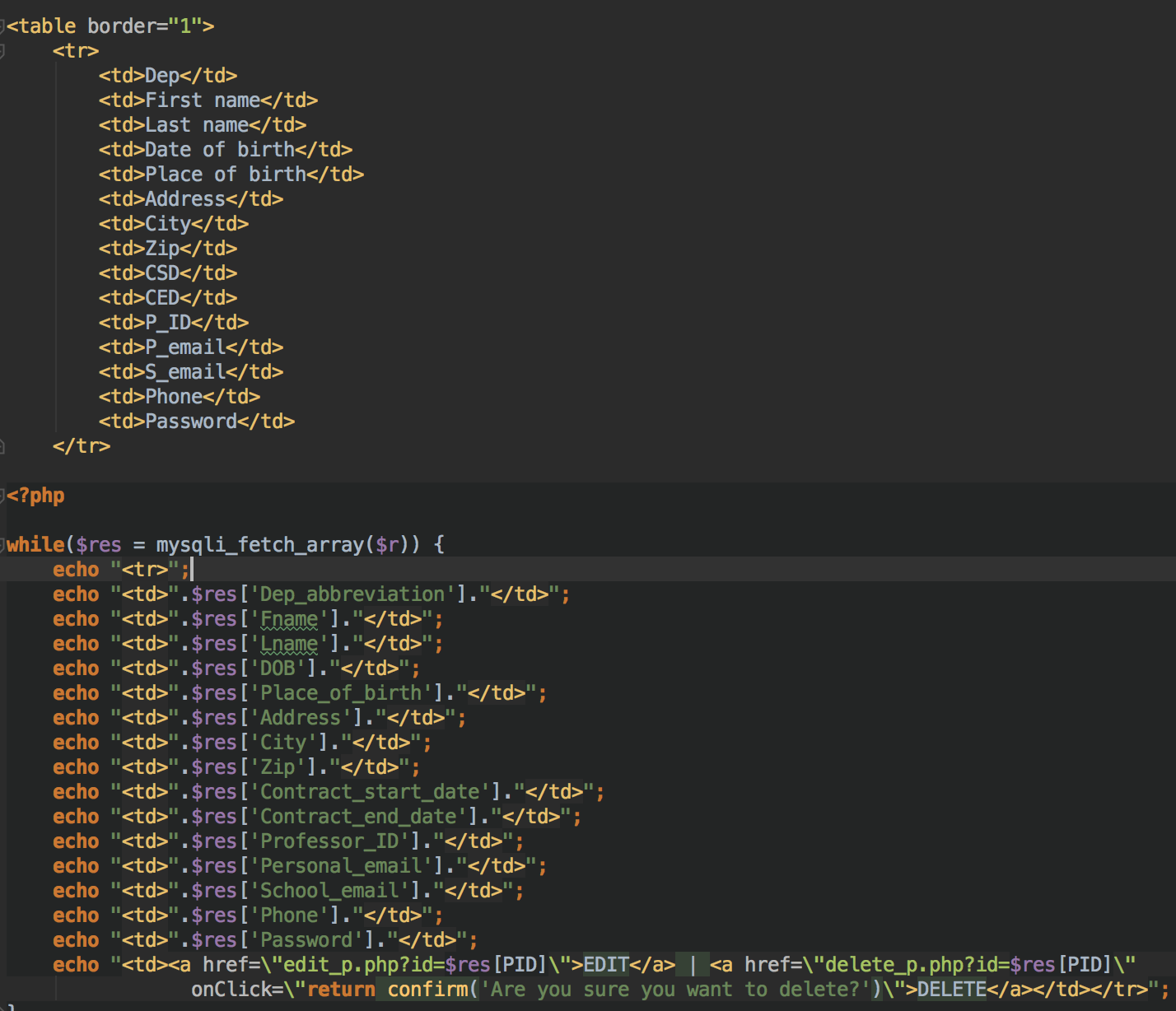
# 8 VIEW ALL

In this part we will explain how students, professors and course assignments are viewed. In the side navigation of admin page, there are three buttons; one for students, one for professors, and one for course assignments, with each leading to its own page and displaying all information depending on the page that is opened.

8.1 VIEW ALL Professors

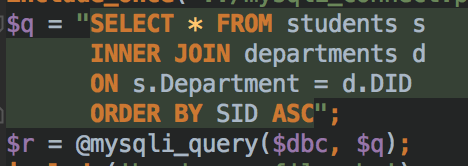


(query for displaying all professors)

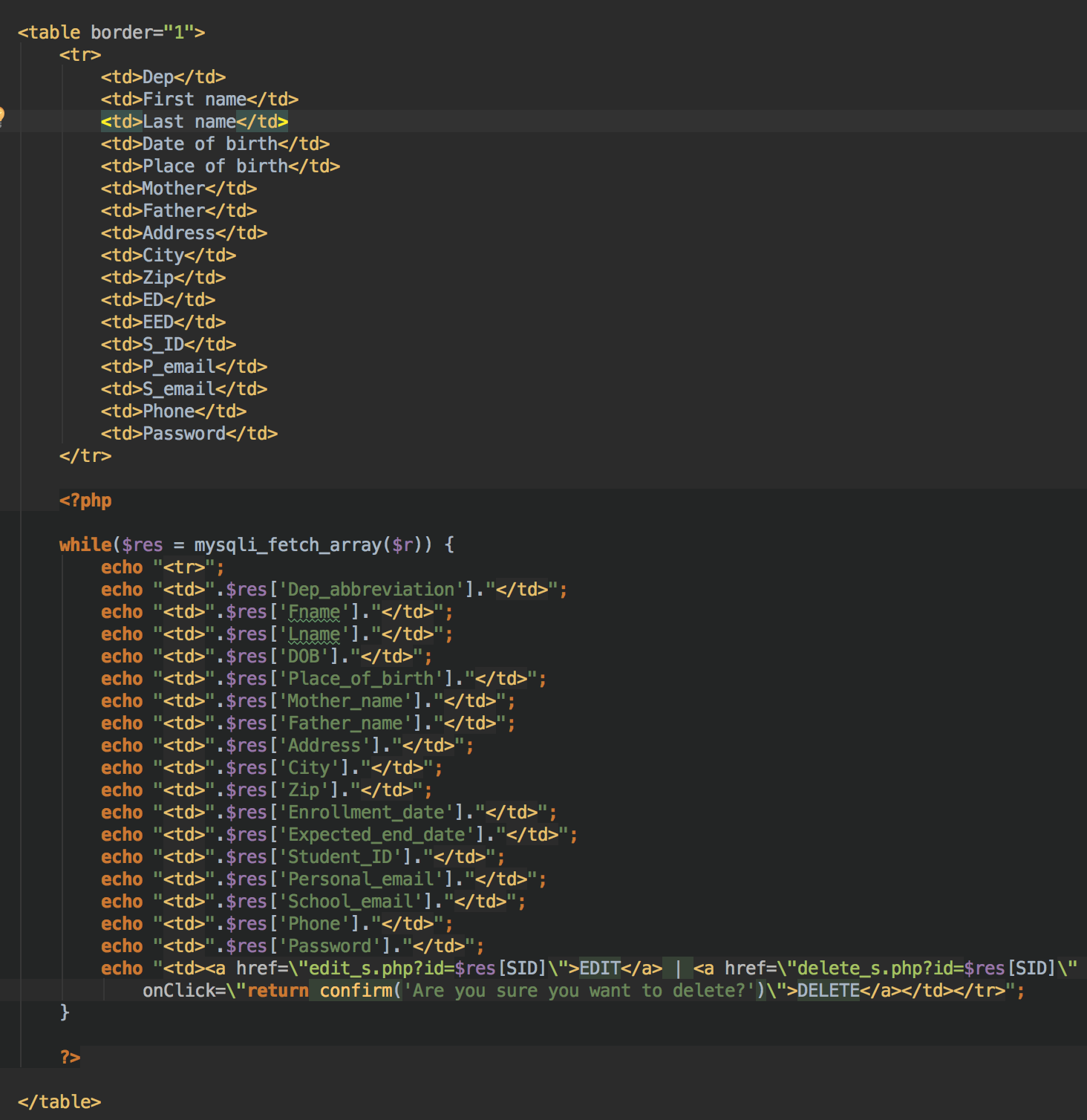


(table code for displaying all the data from the query)

8.2 VIEW ALL Students

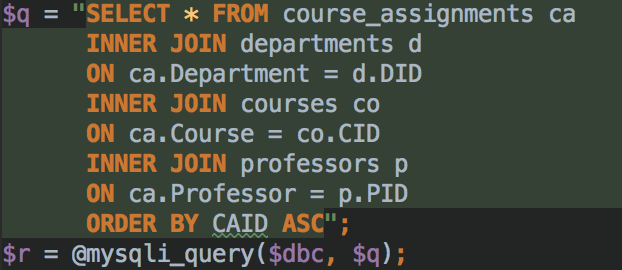


(query for displaying all students)



(table code for displaying all data from the query)

8.3 VIEW ALL Course Assignments



(query for displaying all course assignments)



(table code for displaying all data from the query)