MIS 233

**WEB BASED APPLICATION PROGRAMMING**

**PROJECT 2**

Zeynep Sena Tınaz

2019502168

07/01/2022



**INTRODUCTION**

My mission in this project assignment is to construct a simple *Registration Management System* for an academic institution using PHP technology, Apache Web Server andMySQL database. There had to be three types of users in the system: *Administrator, Professors* and *Students*. Each user of the system had to have a unique username and a password and they could do the different operations.

**TOOL DESCRIPTION**

First of all, I had to use XAMPP tool to set up the Apache server and use Google Chrome browser to go to localhost and display my website. Secondly, I used *Visual Studio Code* editor to make my PHP and CSS files and compile and execute my program. Used phpMyAdmin to control my database. I used Bootstrap CDN to style some parts of my website. I am using a Windows 10 operating system but it isn’t necessary for this project. It could be done on a Mac or a Linux operating system.

**PROBLEM SPECIFICATION**

The Administrator can do the following operations:

• Login and Logout

• Define a New Professor/Student (obtain name, surname etc. information regarding the professor/student and set his/her password, the length of the password should be between MinPwd and MaxPwd characters)

• Define a New Course (obtain name, description, quota, final date, credits, consent needed or not, teaching professor etc. information regarding the course, you may assume that a course is taught by only one professor)

• Deactivate an Existing User (Administrator user cannot be deactivated, prior to deactivating a user of professor type, you should check whether that professor is teaching any course and warn the user if that is the case, similarly a user of student type cannot be deactivated if the course list of that student is not empty)

• Reset the Password of an Existing User (Administrator password can also be changed)

• Obtain the List of All Professors/Students/Courses in the System

• Management of System Parameters (set the value of MinPwd, MaxPwd, MaxCourse, MaxStuCourse and MaxProfCourse)

• Get User Statistics Report (present the number of active professors, active students, deactivated professors, deactivated students)

Students can do the following operations:

• Login and Logout

• Change Password (the length of the password should be between MinPwd and MaxPwd characters)

• Ask for Consent (in case the addition of a course requires the consent of the professor)

• Add a Course to Course List/ Remove a Course from the Course List (a student can add at most MaxStuCourse courses and the quota of the courses should also be considered during this operation).

• View Course List (list all the courses enrolled by the student and the corresponding grades).

Professors can do the following operations:

• Login and Logout

• Change Password (the length of the password should be between MinPwd and MaxPwd characters)

• View/Process Consent Requests (the professor can accept or reject the request)

• View Course and Student List (list all the courses taught by the professor and the list of the students enrolled).

• Submit Grades (grades can have three distinct values, namely, Not Submitted, Passed, Failed, and once submitted cannot be changed)

**ALGORITHM/PROGRAM DESIGN**

* Open up the XAMPP control panel and start the Apache and MySQL servers.
* Create a folder inside xampp => htdocs called “233project2” and create “index.php” file.
* Open up “localhost/233project2/index.php” as well as “localhost/phpMyAdmin”.
* In phpMyAdmin, create a table called “users”. Give every user an id, first name, last name, username, password, and the user type.
* Open up Visual Studio Code and design the basic HTML template for index.php.
* Create a new file called “header.php” and design it using a BootStrap navbar template.
* Create the login.php file and use the CSS login page template that can be found here: <https://codepen.io/danzawadzki/pen/EgqKRr>
* Include the navbar template (“header.php”) in “index.php”.
* Create a file called “config.php” so we can include it to every file to connect to MySQL database using PDO.
* Start the session and connect, create a function called user\_detail to store the data inside the user-detail global variable.
* In the “login.php” check if the username the user puts in matches with the info in the database. If it is correct, direct them to “index.php”.
* Create a new file called “navbar.php” and configure the main pages look depending on user’s type. (Since they are able to do different operations, we design the buttons and their names they can click on differently.)
* Include “navbar.php” file in “index.php”.
* Create a file called “change-password.php” and design the changing password form. Name the input fields same as you named them in the database.
* Store the info you got from database from “user\_detail” function in the variables and the new password info you get from the user when they put it in the form.
* Write an if else statement to check if the user left the input fields empty, if they did alert a warning.
* Write another if else statement to check if the user typed their old passwords right. Echo warning if they did not.
* Write another if else statement to update the password using UPDATE SQL command.
* In the “navbar.php” file, in each link for “change password”, direct them to change-password.php.
* Create a new file called “log-out.php”. Destroy the session that was started in “config.php” and locate the user back to “login.php”.
* Create a new file called “manage-users”. As the administrator, we can obtain the List of all professors/students, get user statistics report, reset the passwords of the users, define a new user, or deactivate an existing user here.
* Design the table for the list. Add an a tag to define a new user, it will direct the admin to the “add-new-user.php” page.
* Create a new file called “add-new-user.php”. Include the “config.php” and check the session as well as giving the CSS style to the page.
* Create a form to put in the required inputs. Create the variables to get the info from the form, add new users using the INSERT INTO command.
* In the “manage-users” file create more tags to resetting their password. Direct them to “change-password-2.php” on click.
* Create a new file called “change-password-2.php”. Apply the same process that was applied in “change-password.php”.
* In the “manage-users” file create a button to deactivate the user. Create status code 0 and 1 to add a tick beside the user to show whether they are deactivated or not.
* Create a new file called “delete-user.php” and use the UPDATE command to update the users status to 0 and deactivate them on button click.
* For the courses, create a new file called “manage-courses” and create the table to display required information about the courses.
* Go to phpMyAdmin and add another table called “courses”. Give it id, description, quota, final date, credits, instructor id, and consent fields.
* Add an a tag for to direct the user to “new-course.php” and add new courses.
* Add a <td> for course consent, we will use it later to know if the consent for the particular course is necessary.
* In the table, display the info that’s gotten from the database by creating a foreach loop.
* In the database, courses table, we do not have the professor’s name and surname information. We only have the instructor id. We bound the “u-id” with the “instd-id” so that we can see the instructors name and surname in the table rather than their instructor id.
* In the “new-course.php” file, create a form and a button to input the required information to add the new course.
* Get the information from the form and store them in variables.
* Use INSERT INTO command to insert the new inputs to the database.
* Check if the user left any input field empty by writing an if else statement using the “empty” method.
* In the select box for the professors, we need to fetch the information from the database where “utype = professor”, and create a foreach loop where we take the id of the professor and turn it into first name and last name.
* For the consent part, create a file named “consent.php” and create a form inside that consists of a message box and a submit button. The student submits the consent here. So we INSERT the message into the database.
* In phpMyAdmin create two new tables called “consents” and “st\_course” and give “consents” consent id, course id, user id and a message field. Give “st\_course” user id, id, course id and grade fields.
* We check with an if else statement that the message box is not empty and alert the user if it is.
* Professor needs to view the consent requests. We create a file called “consent-requests.php” and create a table to display the student’s name and message. We SELECT the consent id and user id and do a rowCount.
* We add two buttons for the professor to either accept or reject the request.
* We give each a status code **=>** pending = 0, approve = 1, reject = 2.
* If professor approves, consent id equals to 1. If he rejects consent id equals to 2.
* After professor makes his decision, we echo “approved” or “rejected” in previous button locations.
* We create a file called “add-course,php”. This file is for the student after the professor has either approved or rejected his/her consent. We do the same status check here.
* If the professor approved, the student will see a button called “add” beside the course he/she has requested consent from.
* We create a new file called “view-courses” for the student to see the courses she/he is taking and create a table to display the required info about the course. We add a button to remove the selected courses.
* In “add-courses.php”, we get the course id and user id by SELECT and write an if statement, do a rowcount and if the user already has this course we warn them.
* Else; we use the INSERT INTO command add the course to the students already existing courses.
* We make a quota control and warn the user if the course is at its limit.
* Finally on the professor’s page, we need to be able to view a list of students and submit grades for them.
* Create a file called “view.php” to see the courses that this particular instructor is providing and make a table to display the necessary information. We bound the “u-id” with the “instd-id” so that we can see the instructors name and surname in the table rather than their instructor id. We add a buton called “view students”. When professor clicks on it, it will direct us to the grading page “view-students.php”.
* Finally, create a new file called “view-students.php”. We make a table just with student’s name, course’s name and description and a select box of us grading the student. “P” if they passed, “F” if they failed. Default value will be “Not Submitted”. On the student screen the grade location will be empty until the professor has graded them.
* We put a ‘-‘ and use the explode method to split the grade and user id to an array from ‘-‘ and transfer them to database and UPDATE the user’s grade. The value at the 0th index is the student’s grade. The value at the 1st index is the students id.
* Create a file called “captcha.php” and create your captcha and bind it to “login.php” with a form.
* Lastly, add data to the webpage and its ready to launch.

**INSTRUCTOR FEEDBACK**

This was definitely a difficult project for my level at least. I can say it was really challenging because it required information that we were not shown during the course. It required me to research a lot. The given time was okay, however considering we had other exams and projects going on, personally it was just on the edge for me. The problem we were given was quite clear. Overall, it was a project that took a lot of my time and effort, however, I have learnt a lot thanks to it.

**REFERENCES**

* <https://codepen.io/danzawadzki/pen/EgqKRr>
* https://getbootstrap.com/docs/4.0/components/navbar/