

**GAZİ UNIVERSITY FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING**



**Secure Student Information System Using PHP And PostgreSQL**

**Project Report**

**181180762 Erkin Berk TÜRE**

**181180060 Ceren Umay Özten**

**181180761 Doğukan Okçu**

**Dr. Uraz Yavanoğlu**

**CENG 367 SCRIPTING LANGUAGES**

January 2021

## TABLE OF INDEX

TABLE OF INDEX .....	1
1. INTRODUCTION .....	2
2. GENERAL STRUCTURE OF THE SYSTEM.....	2
3. MODULES AND USER OPERATIONS .....	2,3
3.1 STUDENT OPERATIONS .....	4
3.2 ADMIN(LECTURER) OPERATIONS .....	6
4. WORKING ORDER OF THE SERVICES.....	7,8
5. SECURITY ADDITIONS IN TERMS OF SECURE CODING .....	8
6. DESIGN OF DATABASE .....	9

## 1.INTRODUCTION

The methods used in creating our project, which we prepared within the scope of the scripting languages course, in a secure way, the structures created, the work plan in which the relations of these structures are collected, the graphics with their representation, the development process document and the working plan of the system are presented and explained in this report.

## 2. GENERAL STRUCTURE OF THE SYSTEM

In the process of developing the system, first the modules in the structure of the system were designed and then the parts were combined by applying. The system has been designed with the aim of enabling students and teachers to log in securely and then, similarly, enable to safely do course selections, add and drop course events for students on the other hand create and delete course events by using their admin privileges for lecturers,

## 3. MODULES AND USER OPERATIONS

- **index.php**

Users can reach the login screen of the application by typing the "index.php" module in the url bar or in the search engine as an extension which if they are already a registered user, will enable every user to log in first, consisting of students and faculty members who are specified as admin. If the users are not an already registered user, they can use our services by creating a new user registration provided that they can show the register form next to login form with the register button as shown below in figure 1 and 2. Users register by entering User Number that is created automatically while registering, full name, registered department, user type, first registered year and password information.

- **config.php**

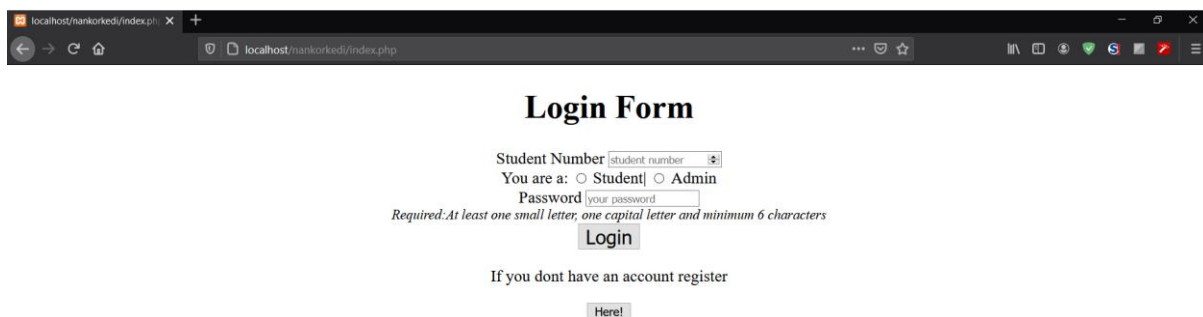
This module makes the necessary server connections in config.php which are certainly coded and must be filled with own dbname and dbpassword for each person who wants to use the system as an obligation, thanks to xampp in localhost.

- **coursecrud.php**

This module carries out crud operations. If the login process is completed successfully, they can add, drop, delete, add and view the registered course information in the system within the authorization and limits specified for the user type. If it is a student it can add and drop courses. If it is an admin then it can create and delete courses which students enroll,

- **profile.php**

If the login process is completed successfully, through profile.php module they can also view the available and enrolled course information in the system within the authorization and limits specified for the user. According to the user type if it is admin then shows admin's profile view which has manage courses and listing courses and If it is student then shows students enrolled courses, all courses and student informations. The information of each student, lecturer and course registered to the system is kept in an SQL-based database to be managed from the PostgreSQL database management system using pgAdmin. For the user, whose process is finished, there exists a LogOut button which has a post request for logging out inside as well.



The screenshot shows a web browser window with the address bar displaying 'localhost/nankorked/index.php'. The page content is a login form titled 'Login Form'. It contains the following elements:

- A 'Student Number' dropdown menu with 'student number' as the placeholder text.
- A 'You are a:' section with two radio buttons: 'Student' (selected) and 'Admin'.
- A 'Password' input field with 'your password' as the placeholder text.
- A note below the password field: 'Required: At least one small letter, one capital letter and minimum 6 characters'.
- A 'Login' button.
- A link below the button: 'If you dont have an account register Here!'.

**Figure 1: Login Form without Registration Form**

The image shows a web browser window with two forms. The top form is titled "Login Form" and contains the following elements: a "Student Number" input field with a placeholder "student number", a "You are a:" label with radio buttons for "Student" and "Admin", a "Password" input field with a placeholder "your password", a "Login" button, and a link "If you dont have an account register" with a "Here!" button. The bottom form is titled "Registration Form" and contains: a "Close registration form" button, a "User Number(Copy it, do not forget)" input field with the value "181462981", a "Full Name" input field with the value "John Doe", a "Choose a department:" dropdown menu with "Computer Engineering" selected, a "You are a:" label with radio buttons for "Student" (selected) and "Admin", a "First registered year:" dropdown menu with "2017" selected, a "Password" input field with masked characters "\*\*\*\*\*", and a "Submit" button. Both forms include a required password policy: "Required: At least one small letter, one capital letter and minimum 6 characters".

**Figure 2: Login Form with Registration Form**

### 3.1. STUDENT OPERATIONS

Students can log in at the login screen by choosing their user type as student with their own randomly generated student numbers starting with the number “181” of each given while the students create their membership on the user registration screen and their passwords consisting of at least one small letter, one capital letter and minimum 6 characters. It includes the ability to view the courses registered with student numbers on the profile screen and add and drop the desired course on the student course screen(coursecrud.php module) as shown below in figures 3 and 4. The student, whose process is finished, can exit by pressing the LogOut button.

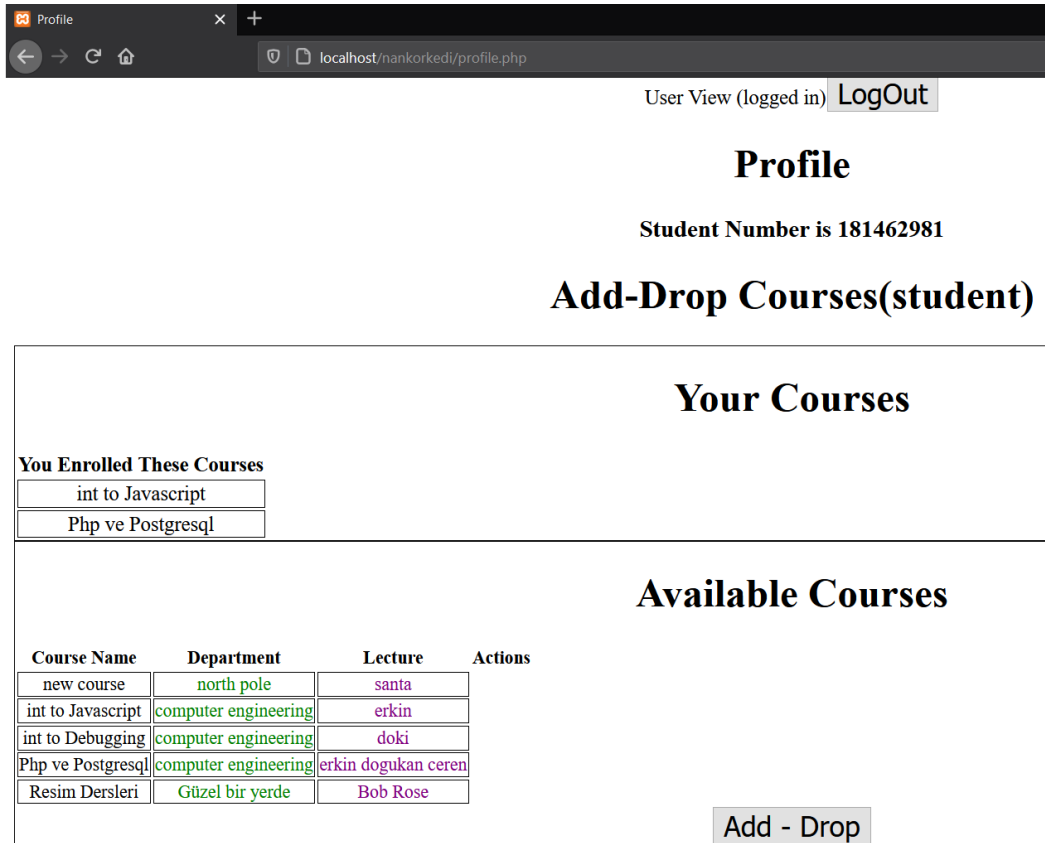


Figure 3: Profile screen for students

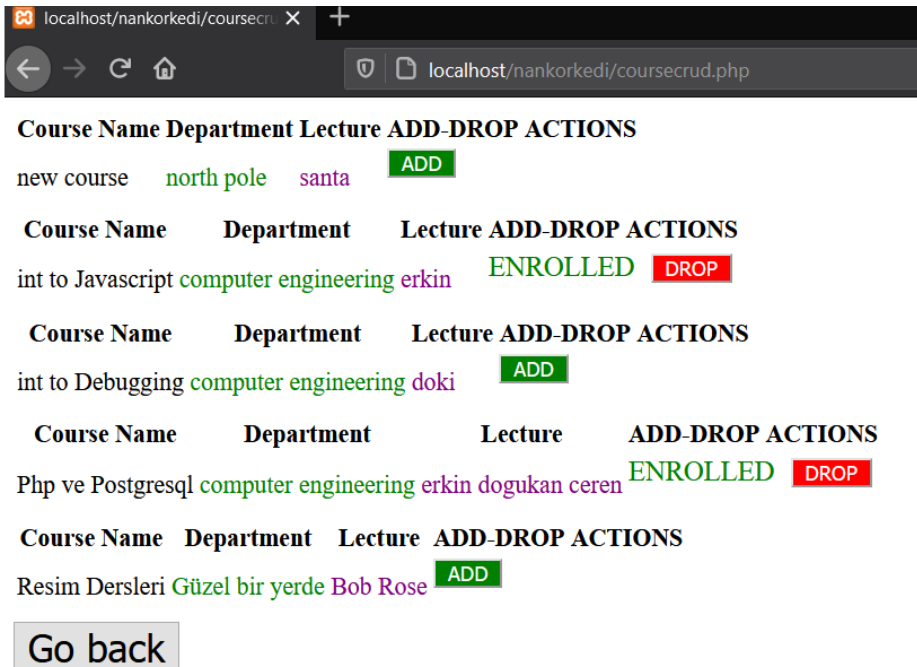
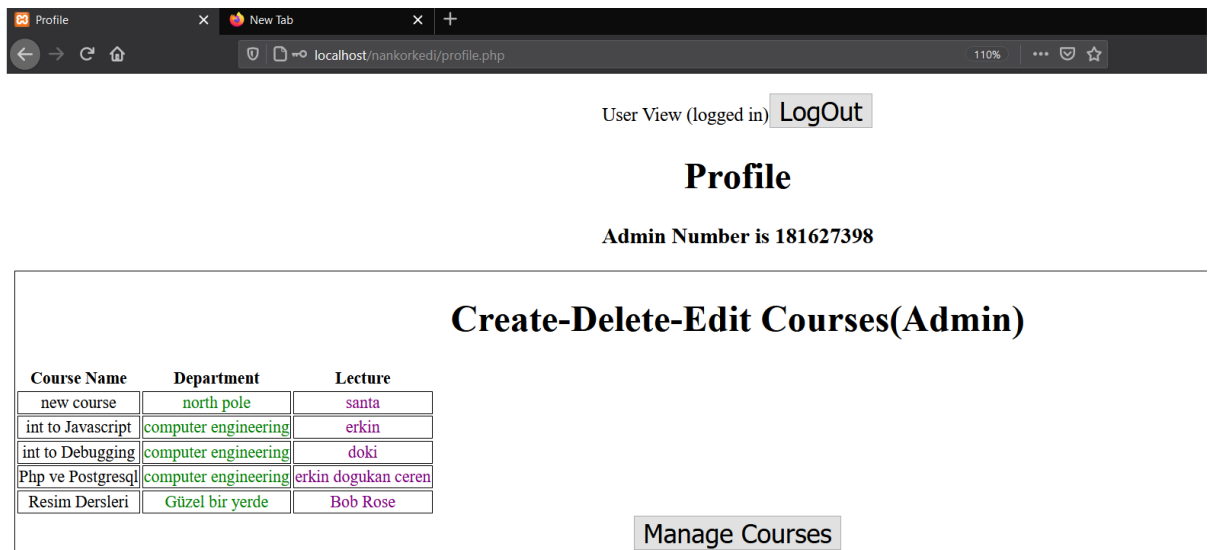


Figure 4: Course add drop screen for students

### 3.2. ADMIN(LECTURER) OPERATIONS

Lecturers which can be called as admins as well can log in at the login screen by choosing their user type as admin with their own admin numbers randomly generated starting with the number 181 of each given while creating their membership on the user registration screen, and their passwords consisting of at least one lowercase and one uppercase letter and minimum 6 characters. After logging in, it includes the ability to view the profile and the courses registered in the system with their admin numbers on the profile screen(profile.php module), provided that they fill in the required relevant course name, lecturer and department fields on the screen that appears by pressing the manage courses button, as well as the ability to delete the registered courses on the course screen(coursecrud module) as shown below in figures 5 and 6. The faculty member who is finished can log out by pressing the LogOut button.



**Figure 5: Profile screen for admins(lecturers)**

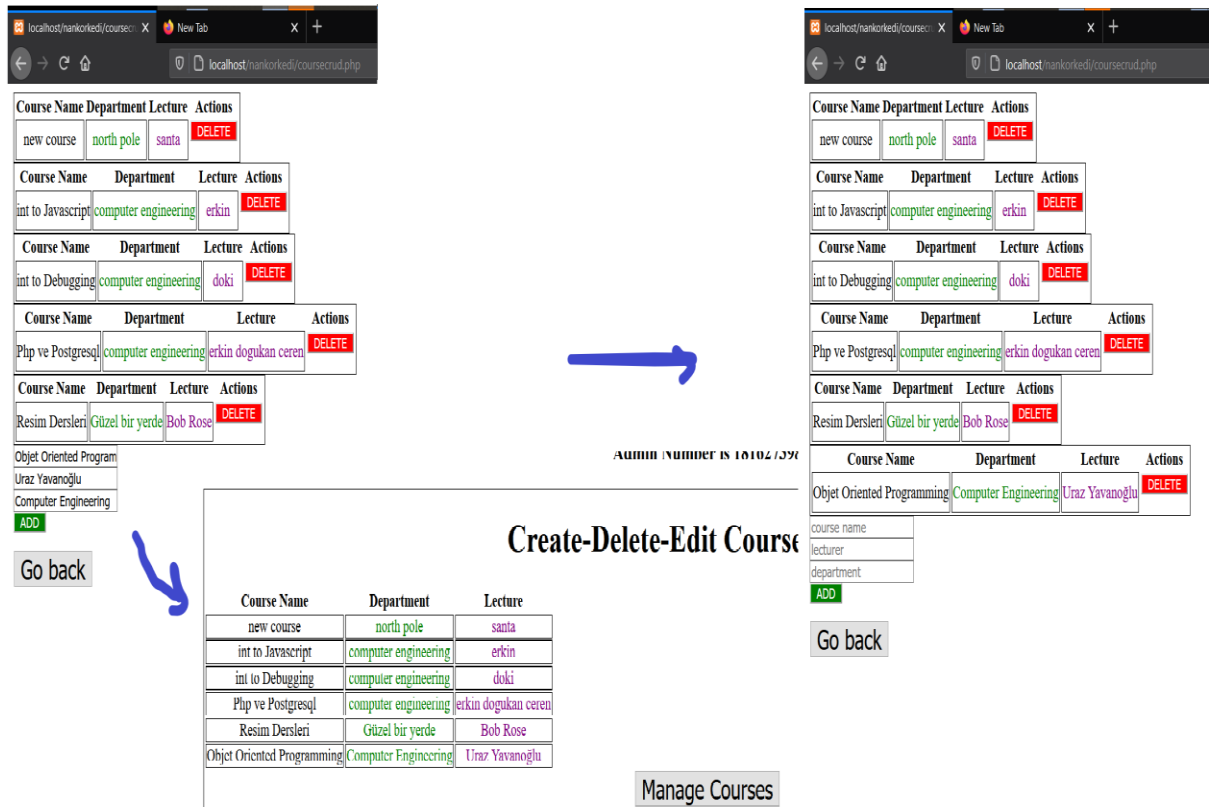


Figure 6: Course add drop screen for admins(lecturers)

#### 4. WORKING ORDER OF THE SERVICES

The services provided by the student information system include registration and login according to the user number given by the system, password and user type, storing the encrypted versions of the existing passwords with the md5 hashing algorithm after trimming the input that user types to store and later compare password values in as shown below in figure 1 and in addition creating, reading, deleting through CRUD operations as shown below in figure 7, 8, 9 and 10.

```
$password = md5(trim_modified($_POST['password']));
```

Figure 7: Trimming and md5 hashing function example code

```
$sql_q1 = "INSERT INTO course (coursename, lecture, department) VALUES ('$coursename','$lecture','$department')";
```

Figure 8: Create (i.e. insert) operation example



```
$sql = "SELECT * FROM public.student WHERE studentno='$studentno' AND password='$password' AND user_type='$user_type' ";
```

**Figure 9: Read (i.e. select) operation example code**

```
$sql_q2 = "DELETE FROM course WHERE coursename='$coursename' AND lecture='$lecture' AND department='$department' ";
```

**Figure 10: Delete operation example code**

In terms of user-friendliness, our services allow users to be fast, secure and as confident as possible about their password security. Users can sign up and use different features depending on the user type. During the course addition, deletion operations of the tables in the database, which are called enrollment while updating the course list, the relations between them is build up with an intermediate table that keeps the common features as a key.

## 5. SECURITY ADDITIONS IN TERMS OF SECURE CODING

We have made a few security additions to our project that we use within the scope of secure coding. Among them, there are prepare and execute functions as shown below in figure 12 instead of query functions that will execute the queries directly against the possibility of SQL Injections, that we use in connections via PDO (PHP Data Objects), which can work with other different databases as well, md5 hashing as we already have shown in figure7 above and PHP Form Security used during database storage and later comparison of the password created during user registration. On behalf of `$_SERVER["PHP_SELF"]` global variable, there are `htmlspecialchars` functions that we use to prevent malicious scripts from running on the client side by using Cross-site scripting (XSS) vulnerability as shown below in figure 11. By the agency of The `htmlspecialchars()` function we have converted special characters to HTML entities. Through this method we are able to prevent attackers exploiting the code by injecting HTML or Javascript code to our login, register or coursecrud form.

```
<form action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']); ?>" method="POST">
```

**Figure 11: XSS(Cross-site scripting) protection example code**

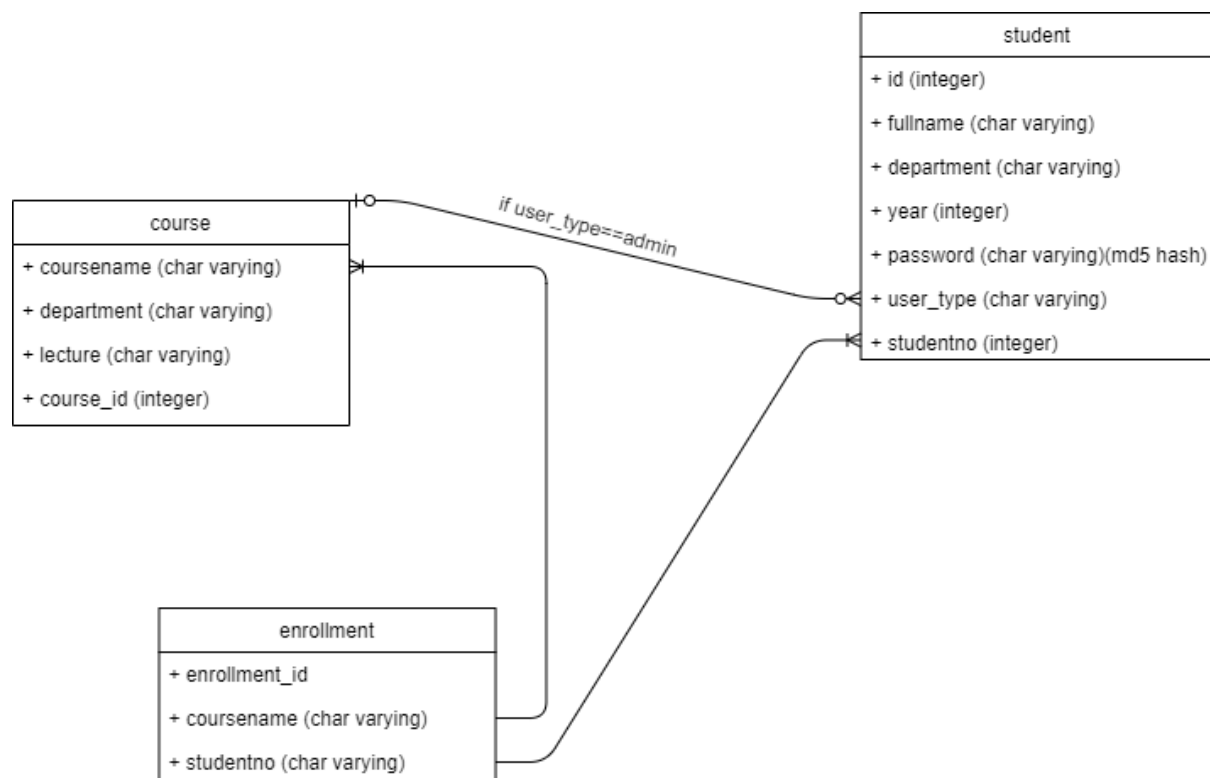
```
// $data = pg_query($conn, $sql);
$user = pg_prepare($conn, "my_query", $sql);
$data = pg_execute($conn, "my_query", array());
```

Replaced with

**Figure 12: SQL injection protection example code**

## 6. DESIGN OF DATABASE

User informations are stored with 3 tables relationship in the database. With the request and authority, the user can access and update information from the database. The student table holds the information for both student and admin, the course table holds the information for the courses enrolled, dropped and to be deleted from the database and lastly the enrollment table holds the information for students and teachers to associate with enrollment table in order to perform the crud operations. The relationship diagram can be explained as below in figure 13.



**Figure 13: Database relations UML diagram presentation**