Week 5 – Final Project

Final Assignment Project

Said Wahid

CST 499 - Capstone for Computer Software Technology

Professor Amr Elchouemi

May 31, 2021

Online Student Enrollment System

Version 1.0.0

Prepared by: Said Wahid

University of Arizona – Global Campus

May 31, 2021



Table of Contents

[Software Requirements Specification 5](#_Toc73315576)

[1. Introduction 5](#_Toc73315577)

[1.1 Purpose 5](#_Toc73315578)

[1.2 Document Conventions 5](#_Toc73315579)

[1.3 Intended Audience and Reading Suggestions 5](#_Toc73315580)

[1.4 Product Scope 6](#_Toc73315581)

[2. Overall Description 6](#_Toc73315582)

[2.1 Product Perspective 6](#_Toc73315583)

[2.2 Product Functions 6](#_Toc73315584)

[2.3 User Classes and Characteristics 7](#_Toc73315585)

[2.4 Operating Environment 8](#_Toc73315586)

[2.5 Design and Implementation Constraints 8](#_Toc73315587)

[2.6 User Documentation 9](#_Toc73315588)

[2.7 Assumptions and Dependencies 9](#_Toc73315589)

[3. Requirements Specification 10](#_Toc73315590)

[3.1 Functional Requirements 10](#_Toc73315591)

[3.2 Interface Requirements 11](#_Toc73315592)

[3.3 Non-functional Requirements 12](#_Toc73315593)

[UML Models and Testing Levels 13](#_Toc73315594)

[4. UML Diagrams 13](#_Toc73315595)

[4.1. Class Diagram 14](#_Toc73315596)

[4.2 Sequence Diagram (Login and Registration) 15](#_Toc73315597)

[4.3 Activity Diagram 16](#_Toc73315598)

[4.4 Use Case Diagram 17](#_Toc73315599)

[5. Testing Levels 18](#_Toc73315600)

[1.1 Introduction 18](#_Toc73315601)

[5.2 Component Testing 18](#_Toc73315602)

[5.3 Integration Testing 19](#_Toc73315603)

[5.4 System Testing 19](#_Toc73315604)

[5.5 Acceptance Testing 20](#_Toc73315605)

[Creating and Testing Preliminary Pages 20](#_Toc73315606)

[6. Creating the Landing, Login, and Enrollment Pages 20](#_Toc73315607)

[7. Running the PHP Files on XAMPP 24](#_Toc73315608)

[Development, Integration, and Implementation 26](#_Toc73315609)

[8. Creating SQL Tables and Completing Implementation 26](#_Toc73315610)

[9. Creating New Database Tables 26](#_Toc73315611)

[10. Coding New PHP Pages 28](#_Toc73315612)

[11. Screenshots of Tables and Pages 29](#_Toc73315613)

[References 38](#_Toc73315614)

# Software Requirements Specification

## 1. Introduction

### 1.1 Purpose

This software requirement specification (SRS) document defines the requirements the online student enrollment system. The purpose of an online enrollment system is to provide students with a self-service web page that allows them to manage their school courses and semester schedule. Online enrollment systems ease the burden of course management on academic staff and offers the students the flexibility they need to shape and modify their own academic calendar.

This document relates to the first approved version of the enrollment website, which must be approved by the school board and the board of education before going online.

### 1.2 Document Conventions

This document encompasses all functional and non-functional requirements of the online student enrollment system. The requirements specification section is divided into four different sections that outline the software’s functional requirements, Interface requirements, platform requirements, and non-functional requirements respectively. Each topic in this document is further divided into subtopics in order to provide better detail of the requirements specification.

### 1.3 Intended Audience and Reading Suggestions

This SRS document is written to deliver information on the purpose and requirements of the online student enrollment system, therefore it is intended for all the software architects, designers, developers, testers, and project managers that will work on the on developing this system. Furthermore, this document is suggested as a supplemental reading for all stakeholders, i.e. school staff, school board member, and reviewing committee members for the online enrollment system.

### 1.4 Product Scope

The main job of the online student enrollment system is to offer students a fast and convenient way of maintaining their semesterly schedule. The website will provide students with the options of creating an account, logging into the system, searching for classes, registering for available classes, dropping a registered class, or using the waitlist feature.

## 2. Overall Description

### 2.1 Product Perspective

This online student enrollment system is commissioned by the school board of University of Arizona – Global Campus as a new and self-contained product that will ease the burden of class registration and schedule management on the educational counselors and administration staff. The system will be hosted as a web page on the current school website and a link to the enrollment system will be provided in the home page.

### 2.2 Product Functions

Based on the scope of online account creation and course/schedule management capabilities, we know that the functional flowchart for any this online student enrollment system should follow the workflow in Figure 1.

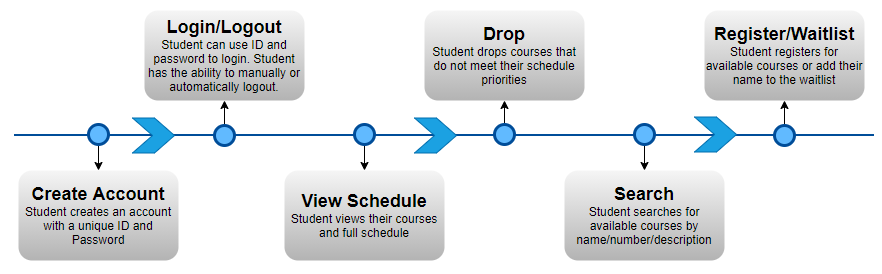


Figure 1: Online Student Enrollment System Functional Flowchart

### 2.3 User Classes and Characteristics

The two main users of the student enrollment system are student and staff, so the user class will have both student and staff type users. The student type users are the main users who can view, register, or drop their classes. The staff types users will include staff/administrators and instructors who are able to view/modify user and course information. Based on the two type of actors, the attributes for the user class are login information, bio, user type, and contact information. The functions for the student type users are createAccount(), login(), and viewProfile(). And, the functions for staff type users are viewUser() viewCourse().

The other main class of the enrollement system is the course class. The attributes for the course class are number, name, description, subject, credits, capacity, openSeats, startDate, endDate, and instructor. And, the functions for the course class are viewCourse(), openCourse(), and closeCourse().

Since registration functions are shared between multiple classes, a separate class callsed actions is used for the providing these functions, which include searchCourse(), registerCourse(), dropCourse(), and waitlistCourse(). Figure 2 shows a preliminary view of the classes.

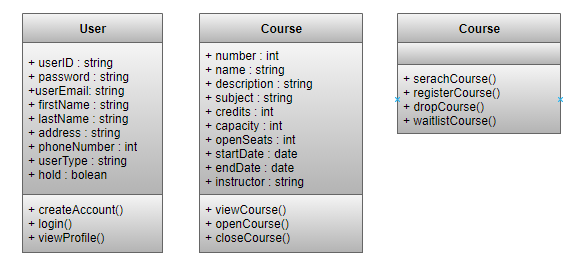


Figure 2: List of preliminary classes for the enrollment system

### 2.4 Operating Environment

The student online enrollment website is considered a course registration kiosk that is available to all students at any time, so it must be compatible and accessible on all platforms including mobile devices. According to Curacubby (2021), the most important aspects of an online enrollment are website portals that are compatible with all operating systems including mobile ones. The website, therefore, must be hosted on the public Internet and should function with all major web browsers, including Google Chrome, Microsoft Edge, Microsoft Internet Explorer, Apple Safari, and Chromium running on Windows, macOS, iOS, Chrome OS, and Android. The website can be a subsystem of the current school website with its links prominently featured on the home page.

### 2.5 Design and Implementation Constraints

The themes, fonts, color, wording, and policies of the website must follow the guidelines provided by the school board and the education board of the state during the requirements specification phase. The mentioned aspects of the website must allow modifiability in case of future policy changes.

Furthermore, the website must exceed or match all the security and privacy guidelines of the similar the main school website websites and comply with all laws and regulations set for school governing body and other local, state, and federal governing authorities.

The timeline for this project is x months and will be considered met if a beta instant of the website is available to the business.

The available budget for this project is $xx,xxx.xx.

### 2.6 User Documentation

User documentation must be created by the stakeholders with heavy involvement from the software development company and users. User documentation and “help” links must be available on the website itself and should exceed or match the detail provided in similar websites. User documentation must discuss functions relating to all user types discussed in Section 2.3.

### 2.7 Assumptions and Dependencies

It is assumed that the school is responsible for providing a working database the courses along with staff user info to be imported at the time of implementation. The website will be dependent on these databases for keeping course and user information up to date. The website must integrate seamlessly with the mentioned any other database systems or student records systems in use at the school. It is further recommended that the website is compatible for integration with the current of future billing systems that are used for financial purposes.

## 3. Requirements Specification

### 3.1 Functional Requirements

Using the flowchart in Figure 1, the list of functional requirements for the student online enrollment system is provided below:

1. User must be able to interact with the system through a User Interface (Interface Requirements are described in section 3.2).
2. New users must be able to create an account using a unique ID and password. Limits of this function are:
   1. Before registration or login, users will have access to the Create Account and Login links only
   2. Users cannot perform any other functions before creating an account and logging in
   3. Users cannot pick an ID that already exists in the system
3. Users must logout once done, or the system will automatically log them out within 5 minutes of inactivity
4. Users must be able to view their profile and course schedule after successful login
5. Within the class schedule view, users must be able to drop a previously registered class.
6. Users must be able to search for available courses for the upcoming semester. Limits for the function are:
   1. Available course open seats must be greater than zero
   2. Number, name, or description of resulting courses must match the search query string
7. Users must be able to register or waitlist for a course that is listed under the result of their search query. Limits for this function are:
   1. If the hold attribute for a user is set to ‘true’, the register and waitlist functions must not be available to the user
   2. The register function must not be available before searching for classes
   3. The waitlist option is only available if the open seats for a class is between -10 and 0.
8. User must receive a notification when their waitlisted course’s open seats are greater than 0.

### 3.2 Interface Requirements

The most important requirement of any public facing website is having a user interface. The user Interface has requirements of its own that are listed below.

1. All pages of the UI must follow a consistent look with non-moving header, footer, links, search, and menu sections
2. Page and link transitions must be consistent and smooth
3. The interface should follow all the theme, color scheme, photo, and marketing specifications of school’s current website
4. The interface should include legal notices and verbiage as requested by the legal team in accordance with the school and department of education rules and regulations
5. Staff should have a separate interface that changes based on their role, such as administrator, counselor, and instructor.

### 3.3 Non-functional Requirements

a list of general and specific non-functional requirement for the student online enrollment system are below:

1. Performance and availability of the website should exceed or match other student enrollment websites for similar size schools. The following maintenance periods are approved by the board during which the website will be offline.
2. 3:00AM – 5:00AM Every Tuesday, Thursday and Sunday.

Following the survey results conducted by DowntimeMonkey.com in April of 2018, an average downtime of 1% is acceptable for an average website such as the student online enrollment system

1. Safety and Security for student personal information and financial information must be of utmost importance and should exceed or match similar school and department of education regulations
2. The development must maintain the budgetary and timeline constraints mentioned in section 2.5 of this document
3. As mentioned in section 2.4, the website must be accessible from and compatible with all operating system and web browsers
4. The website must have a mobile version that has better view and compatibility on mobile platforms
5. The website must have integration with emailing systems to provide notifications for waitlist features

Non-functional requirements for the enrollment system website shape software design to adhere to exact a specific school’s curriculum, governing rules, and business aspects. Therefore, in a project as this, the involvement of legal, marketing, and accounting departments bring about other non-functional requirements that must be specified and verified before getting to the implementation phase.

# UML Models and Testing Levels

## 4. UML Diagrams

Software systems are modeled using UML Diagrams. UML stands for unified modeling language, which is a rich language that uses graphical representations of a system. A visual representation of the system helps in understanding the system's functionality better and makes it easier to explain the system design to the customer.

In this document, the following four models are presented for the online student enrollment system.

1. Class Diagram: According to GeeksforGeeks (2020), "class diagrams are the building blocks of object-oriented programming. These diagrams show the online student enrollment system's objects, classes, and their relationships in detail.
2. Sequence Diagram: Sequence diagrams portray the sequence of interactions between users and the system and interactions between system components. This diagram shows how students can start from creating their accounts to logging in and managing their course schedules as a sequence of actions.
3. Activity Diagram: The activity diagram portrays the flow of control between the actors and different system components. The activity diagram for the online student enrollment system shows the enrollment actions performed by the students and system.
4. Use Case Diagram: Use cases clarify what stakeholders and customers want the system to do or show the essential actions to actors in the system (Filipova & Nikiforova 2019). The use case diagram for the online student enrollment system shows what functions are expected from each actor.

### 4.1. Class Diagram

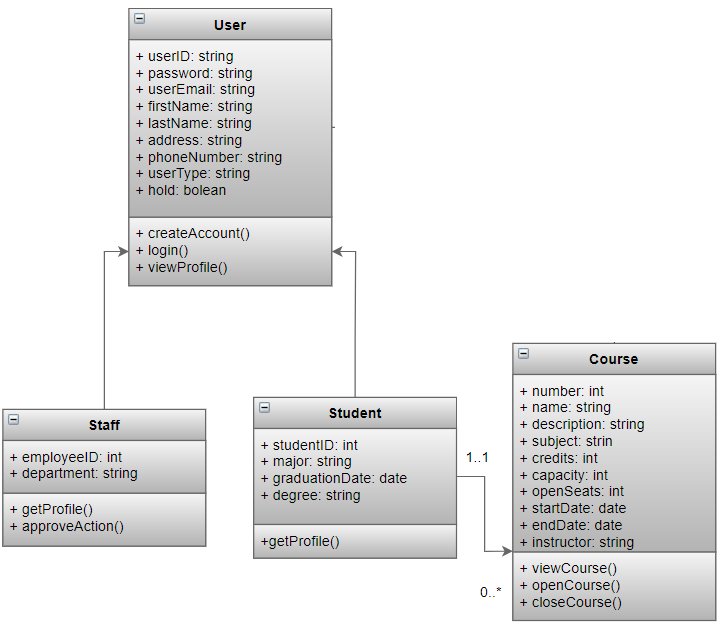


Figure 3: Class Diagram

### 4.2 Sequence Diagram (Login and Registration)

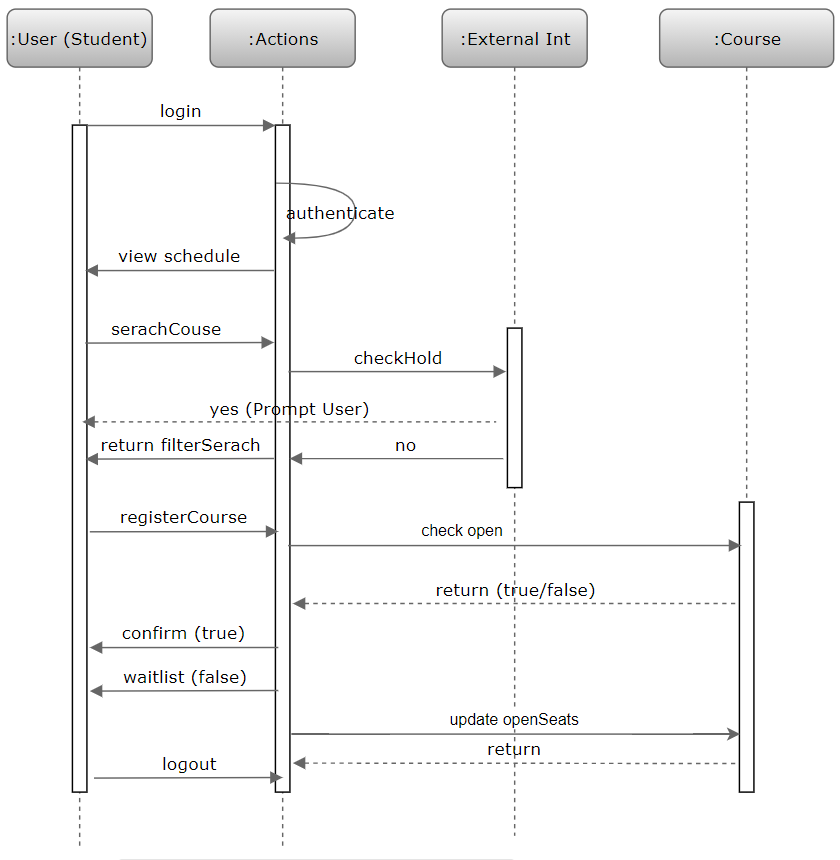


Figure 4: Sequence Diagram

### 4.3 Activity Diagram

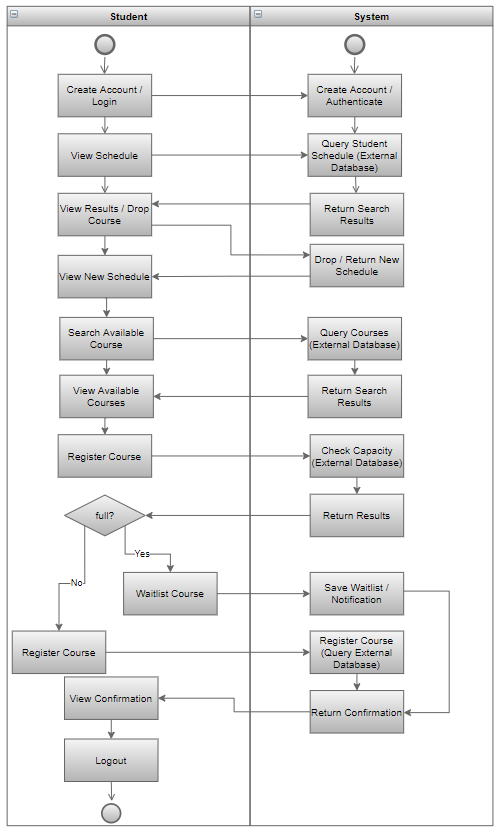


Figure 5: Activity Diagram

### 4.4 Use Case Diagram

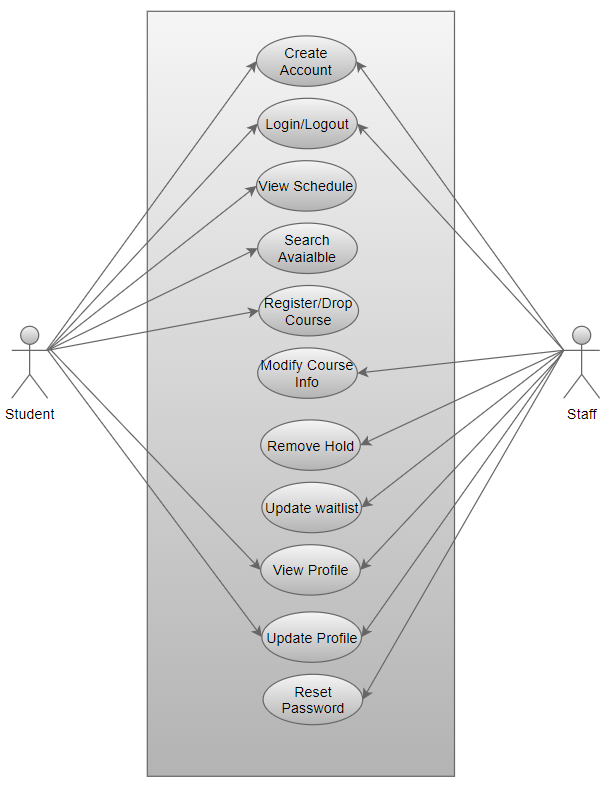


Figure 6: Use Case Diagram

## 5. Testing Levels

### 1.1 Introduction

As we know, software development has many stages, and mistakes, errors, and bugs can occur at any step. Therefore, it is imperative to have a planned testing process in place that is comprehensive and ongoing. The purpose of ongoing testing at every stage of development is to confirm that the system is built to specifications and can solve the problem it was created for (Tsui et al., 2018). Enhanced quality, lower cost, better security, and customer satisfaction are benefits of thorough testing in software development (TestDevLab, 2018).

According to William Miller (2019), a lack of proper software testing causes a bad consumer experience and increases the chances of disapproval by stakeholders and developers. It is also critical that a balanced approach is taken in testing software, where programmers, designers, and certified testers are involved in an ongoing testing process.

Based on the ongoing nature of software testing processes, there are four levels of testing – Component testing, Integration Testing, System Testing, and finally, Acceptance Testing (Spillner et. al. 2014).

### 5.2 Component Testing

Component testing is performed in the specification and design phase and all system components one-by-one. In the case of the online student enrollment system, the essential elements can be retrieved from the use cases shown in (Figure 6). For a student, the use cases are creating an account, login/logout to their account, viewing their course schedule, searching for courses, registering/dropping classes, and registering for a course waitlist.

Based on these components, we can perform two types of tests – static and dynamic. Static testing reviews the code for each class and web page without executing or checking its results. Dynamic testing runs each page separately to ensure it is functioning according to the activity diagram (Figure 5).

### 5.3 Integration Testing

Once component testing is completed, the integration or collaboration of all components is tested. Integration testing checks that information and control are passed between different parts according to the flow in sequence diagram.

Integration testing could also look at the external integration of different components. However, thorough external integration testing is better performed during the system testing phase. In the online student enrollment system case, external integrations to the school databases and school billing system are essential and must be maintained.

### 5.4 System Testing

The online student enrollment system is dynamically tested for proper functionality and integration at the system testing phase. Additionally, the system is tested for non-functional requirements approved in the requirement engineering phase of SDLC.

When testing the online student enrollment system, using a beta environment is an option. However, testing in both beta and production environments is recommended because the online student enrollment system's environmental behavior and external integrations cannot be tested without connection to the central school system. These components must be tested for accepted accuracy, performance, availability, and reliability.

For instance, the developers can test the search function with a test database, but the test might not show live changes to course changes by other users. Therefore, functionality that requires connection to a dynamic external system must be tested with it in a production environment.

### 5.5 Acceptance Testing

Also known as User Acceptance Testing or UAT, this phase of testing verifies that the website meets all the requirements specification in the SRS document version 1.0.0, which was approved in the first phase of SDLC. In other words, it ensures that the contractual agreements have been met and the project is finished within the timeline and budget.

UAT will use students to check that the system gives them the full functionality they need to look, register for, and drop classes quickly. Staff will likewise have to give criticism regarding whether they can efficiently and effectively produce reports and update course parameters. This phase also includes an acknowledgment testing will happen across various working frameworks and gadgets to guarantee that all clients will have a positive client experience. Playing out the four degrees of testing, component, integration, system, and acceptance will help deliver a quality product.

# Creating and Testing Preliminary Pages

## 6. Creating the Landing, Login, and Enrollment Pages

Two things are required to create and deliver web pages for the Student Enrollment Website – a Web Server to deliver web pages and a database management system to hold database tables. To achieve that, XAMPP was installed on a computer with connection to the local network. According to the Apache Friends website (2020), XAMPP is a free distribution that contains the Apache server along with the MySQL database management system. The Apache server allows XAMPP to quickly convert any computer into a web server that can store and serve up web pages. PHP programming language is used to write the code for the needed pages. The PHP code can be written in any text editor, but the Brackets text editor was used in this case because it has better aesthetic support for PHP.

The code for the Landing, Login, and Registration pages was written first. The landing page, coded as home.php, serves as the home page for the website. It is a simple page that welcomes users and non-users alike, and provides them with the links to the login, enrollment, and contact pages. Figure 7 shows a screenshot of the code for home.php file.



Figure 7: Home.php Code

The Enrollment page, called *registration.php*, is designed to take user input for the attributes of the user class and save it into a database table. It is a common registration page that consists of a front-end web page for taking input and a backend database table that stores the input in an organized fashion. The PHP form function is used to take input from user and store it in the MySQL database. Before running and using the enrollment page, the backend database instance *studentenrollement* was created with *tblUser*. The tables are based on the user class and its attributes detailed in the version 1.0.0 of the design document. Figure 8 and 9 show screenshots of the data and structural views for *tbluser*.

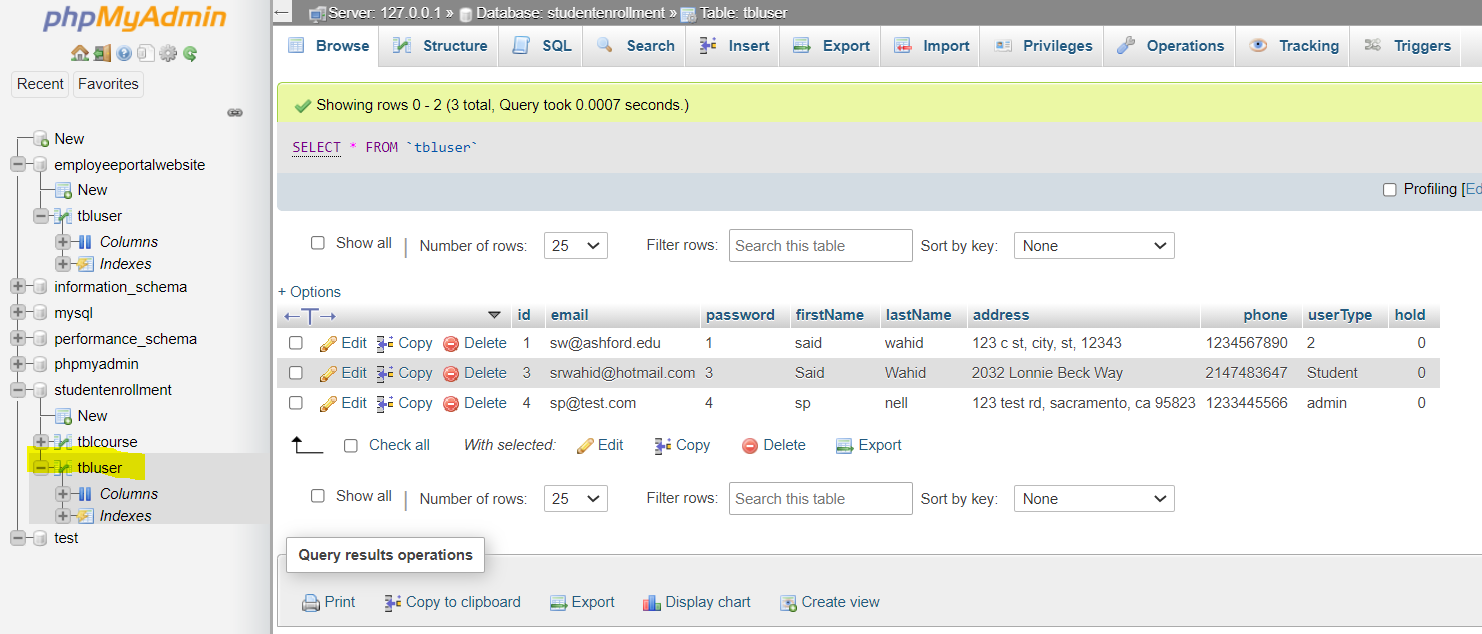
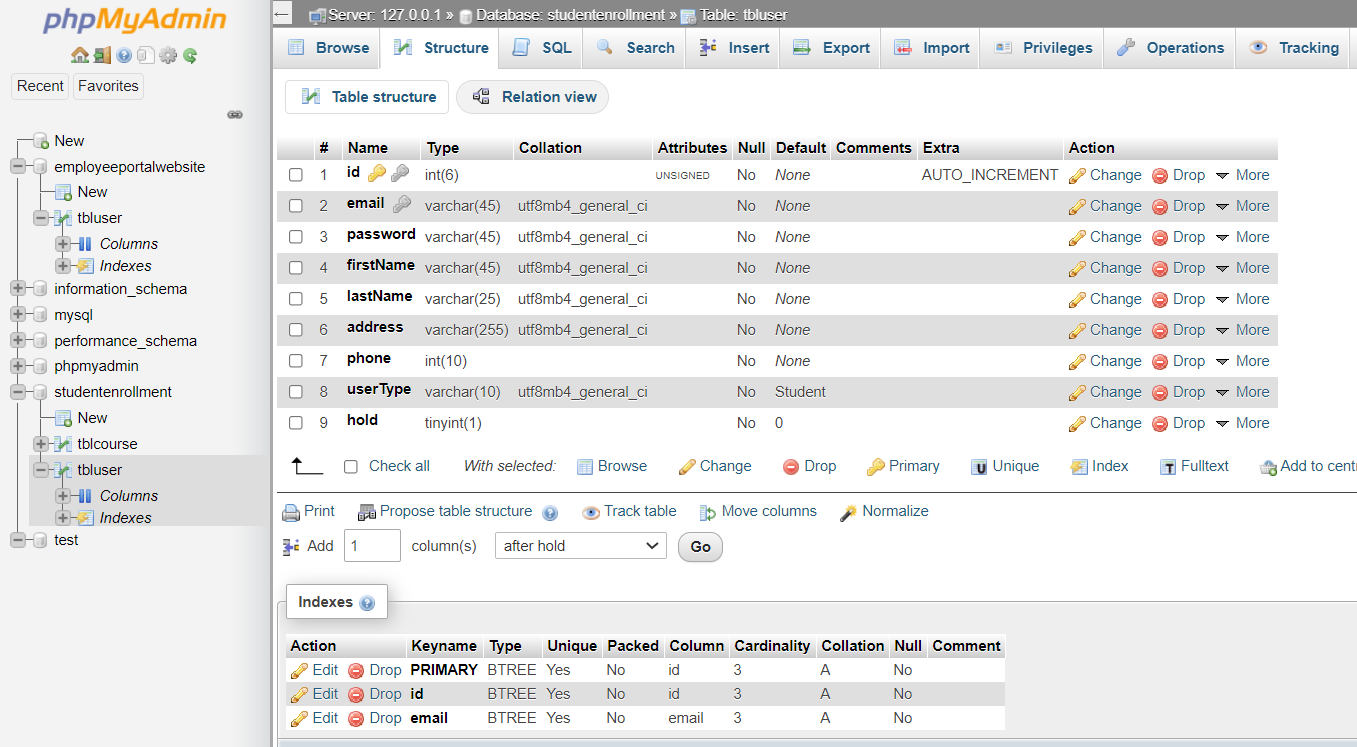


Figure 8: Data View for tbluser



*Figure 9: Structural View for tbluser*

Additionally, a separate database connection is used as a custom class, called dbConnect.php to connect to the MySQL database instance and used the INSERT query for saving the inputs into a table called tbluser. Figure 10 shows a screenshot of the dbConnect.php class.



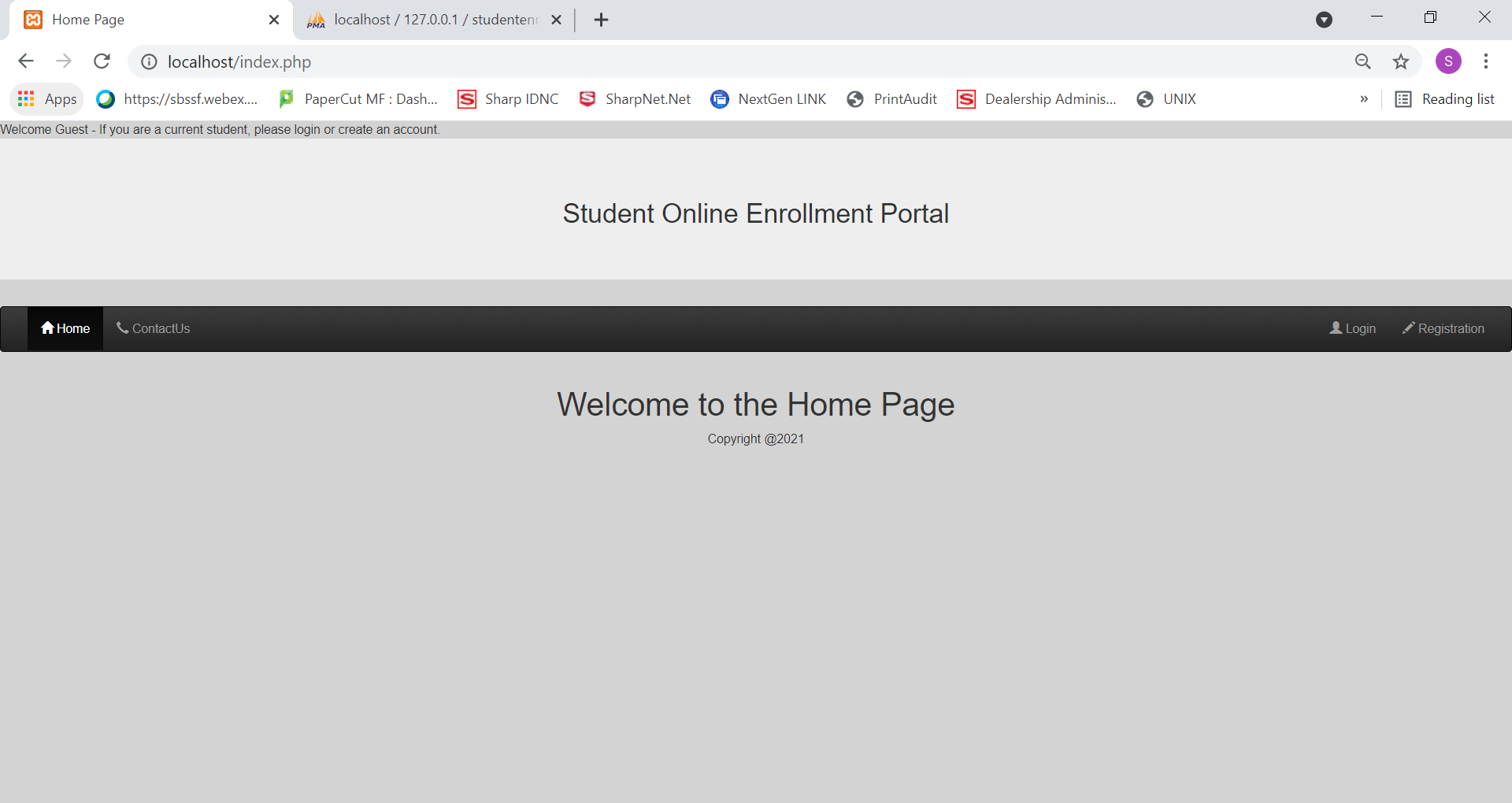
*Figure 10: dbConnect Class*

And finally, the Login page for the Student Enrollment Website was created. This page authenticates a registered user by validating their email and password in the MySQL database. If the user’s provided email and password match with a record on the database, they can view the home page with the added “Profile” button. The method POST is used to take three inputs – email with type email, password with type password, and submit with type submit for the button. Once the inputs are acquired, the page uses the $\_POST method to assign them to local variables. And in the next section validate them tblUser. The database connection class, dbConnect.php, is used again to provide connections to the database instance.

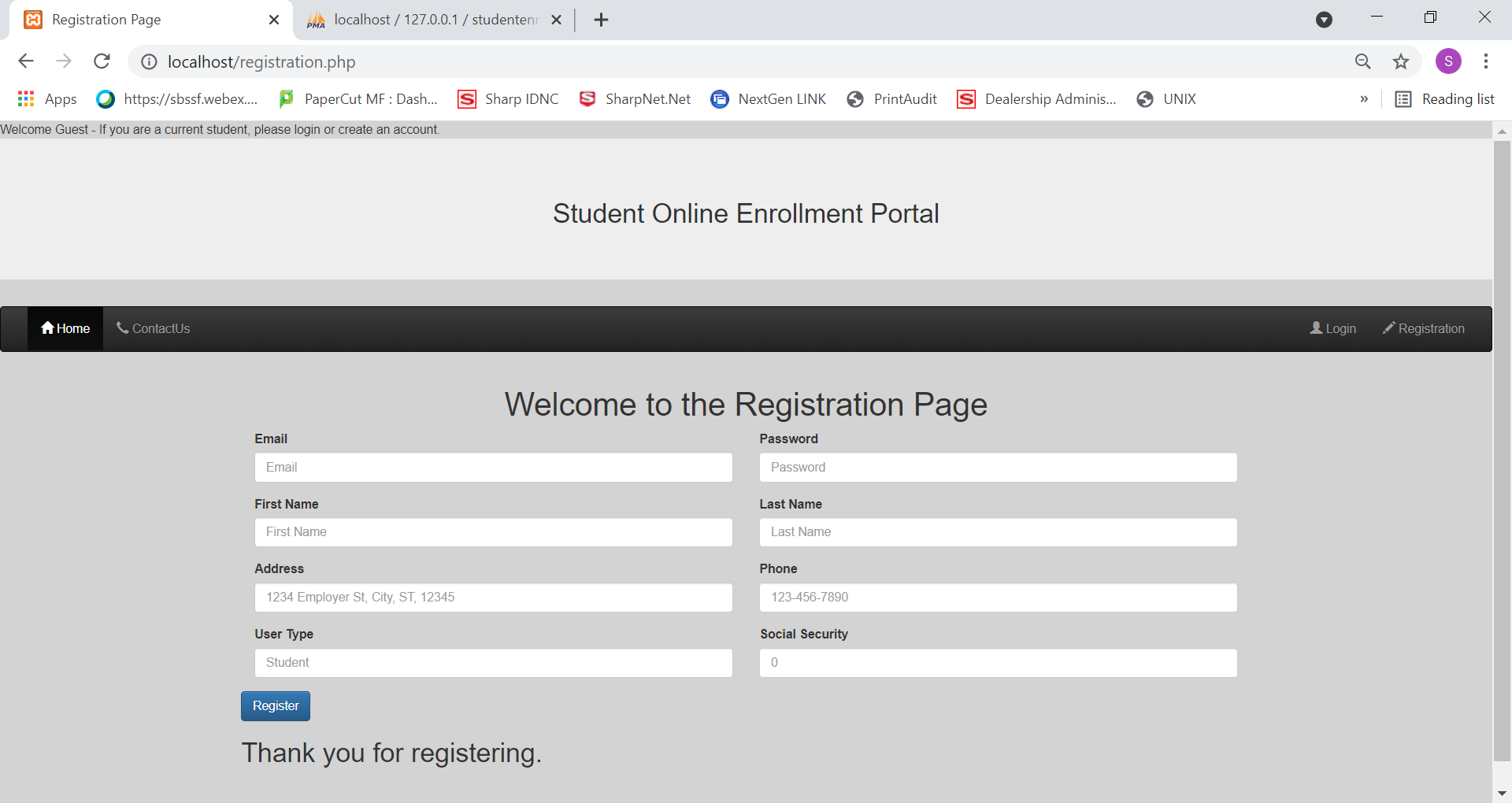
In effort to shorten code timing, the master.php, header.php, and footer.php was used from a previous project. These pages are called from the newly created pages to provide the layout, color scheme, and static texts at the top and bottom of the website.

## 7. Running the PHP Files on XAMPP

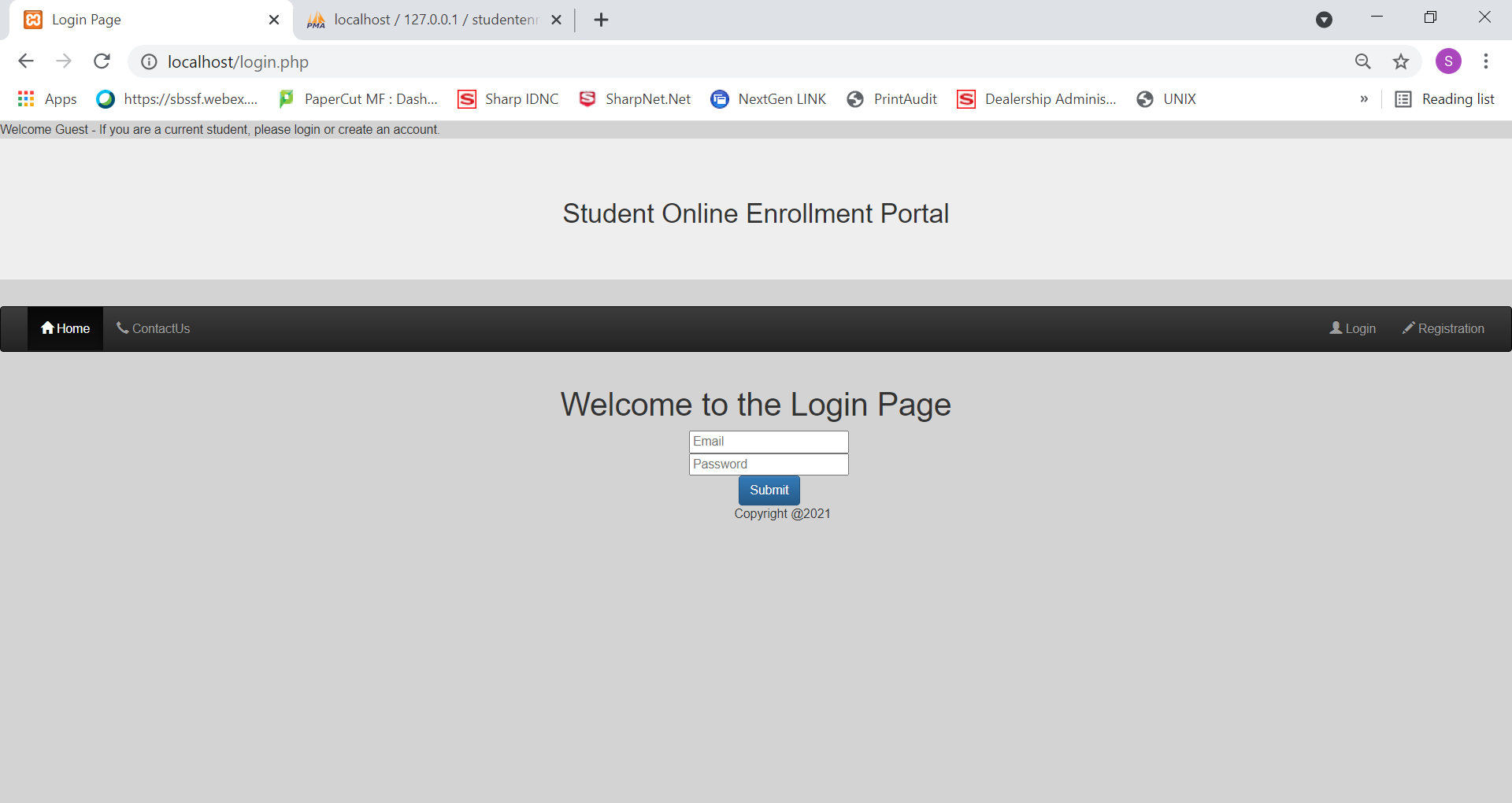
After the pages were created, they must be viewed using a browser to make sure the appearance and function are accurate and to specifications. Web pages can be delivered to a browser by a web server, which is performed by XAMPP’s Apache web server. Through this process, code written in PHP residing on a web server transfers as HTTP onto web browsers. For the web pages to be viewable on a browser, they must be saved in the folder c:/xampp/htdocs by default. To request a page, a user opens any web browser on the local computer with XAMPP, i.e., Google Chrome, and types localhost/pagename.ext in the address bar. If a page by that name exists in the c:/xampp/htdoc folder, it will be displayed on the web browser. The extension for web pages coded in PHP is .php. Figures 11, 12, and 13 show screenshots of all three pages.



*Figure 11: Landing Page Display*



*Figure 12: Enrollment Page Display*



*Figure 13: Login Page Display*

# Development, Integration, and Implementation

## 8. Creating SQL Tables and Completing Implementation

This section needed to provide a front-end UI on the website, the full backend integration with the MySQL database, utilizing and testing of complex database queries, and working with many variables, tables, fields, and placeholders. In this section a list of database tables are created and then populated. In addition, the preliminary pages for Schedule, Profile, Search, and add Course is created and integrated with the backend MySQL.

## 9. Creating New Database Tables

Before starting this section, the only table available was the tbluser. So, the first page the Login page was connected with the database. The login page uses the ISSET() function to carry over the email address and password from user input. It then uses the SELECT \* FROM \* WHERE query to check the inputs against tbluser and, if the result is 1, moves the control to the home.php page with all the user’s information.

Next, the following database tables were:

tblCourse: holds course number, course name, and course capacity.

coursesOffered: shares course number with tblCourse and provide semester and year for the given course.

To test the integration of the pages with the database, first, the registration page created in the previous sections is used to enter multiple users into the users’ table. The login page helps in adding sample data. Then the sample course data is into the course table, including English, Psychology, Math, and Music classes. After that, courses offered a table is populated by assigning different classes to semesters and years. Before populating the tables, the designer view of phpMyAdmin with MySQL database is created to generate table relationships. Generating relationships with the designer’s UI helps visualize table relationships and allows the viewer to capture them for later use.

And finally, the rest of the tables were created as following:

Enrolled: holds student ID and offered course ID to relate courses to a specific student. This table, along with the tbluser, courses offered, and tblcourses, essentially creates the schedule view.

Notify: holds notifications that were generated to students based on their class selection and waitlist possibilities.

Waitlist: hold the waitlist id, student id, and offered course ID. The waitlist table is connected to the Schedule table, so when a registration for a course is accepted or denied, a notification email is sent to the user.

In addition, the Schedule page also provides the functionality of dropping classes for a student. The dropCourse function was added to the PHP file where a DELETE \* FROM \* WHERE query checks the course number and student ID, confirms the existence of such course in the student’s schedule, and deletes from the enrolled table.

## 10. Coding New PHP Pages

Once the tables were populated, and the relationships between them were created, coding of the PHP files was started. The PHP files serve as different pages of the enrollment website. The first page coded and tested was the Search page. The Search page lets the user select the year and semester for the available course. The main feature of this page is a drop-down menu for semester and year, which only allows the selection of valid fields. The fields, semester, and year are coming from the courses offered table. If the courses offered table does not have a year or semester listed, they would not appear in the drop-down menu of the search page. Once the user selects the semester and year and clicks submit, a database query checks for the availability of classes by checking a match for semester/year and a capacity that is greater than zero. A SELECT query then lists the options, with the add button next to them.

The Logout pages allows the user to login by unsetting the variables that carries the email field for the user over to other PHP files.

## 11. Screenshots of Tables and Pages

Graphical user interface, application

Description automatically generated

Figure 14: Structural View of tblUser

Graphical user interface, text, application, Word, email

Description automatically generated

Figure15: Browse View of tblUser

Graphical user interface, application, Word

Description automatically generated

Figure 16: Structural View of tblCourse

Graphical user interface, text, application, email

Description automatically generated

Figure 17: Browse Views of tblCourse

Graphical user interface, application, Word

Description automatically generated

Figure 18: Structural View of offered [Courses]

Graphical user interface, application

Description automatically generated

Figure 19: Browse View of offered [Courses]

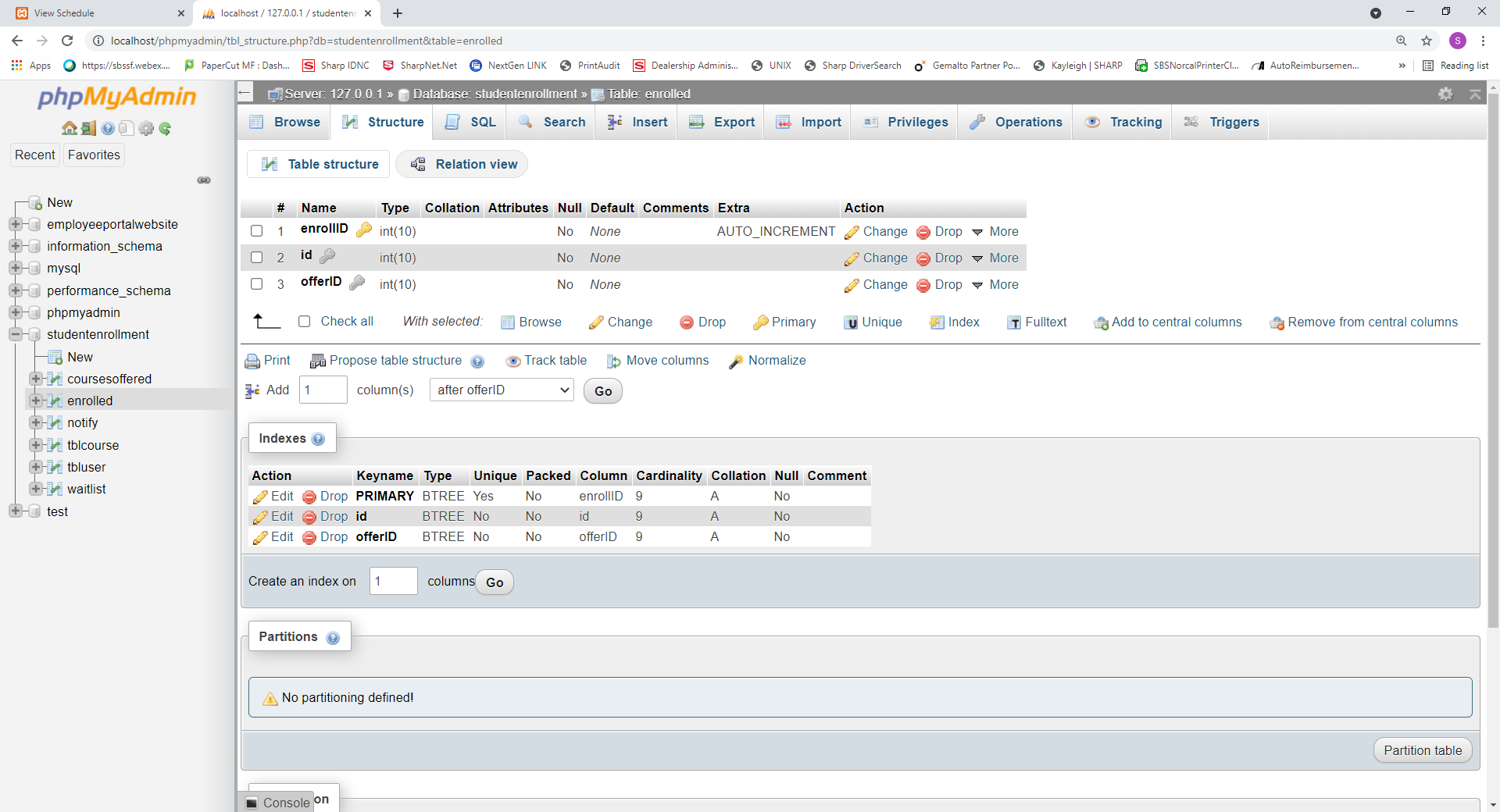


Figure 20: Structural View of enrolled

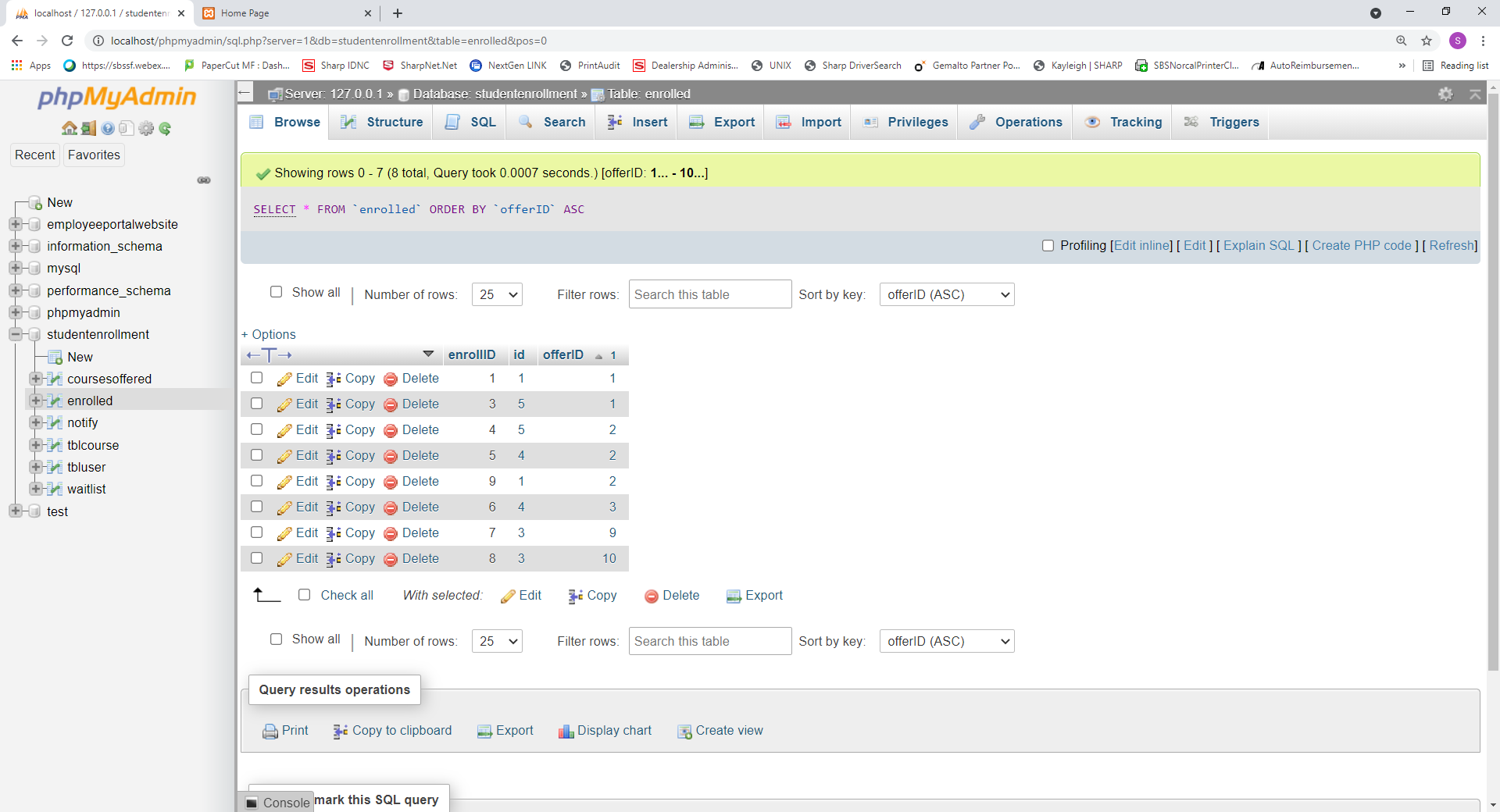


Figure 21: Browse View of enrolled

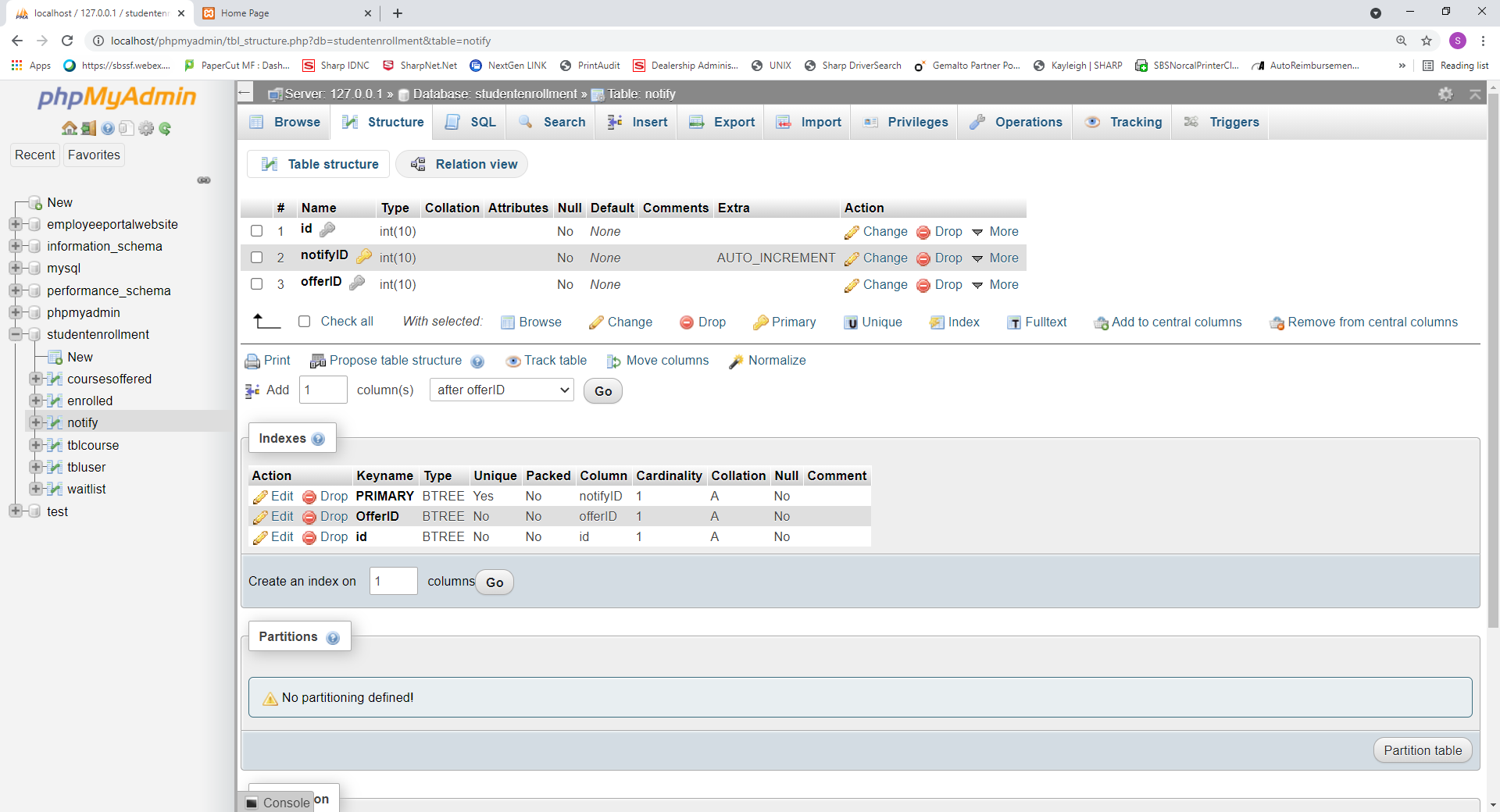


Figure 22: Structural View of notify

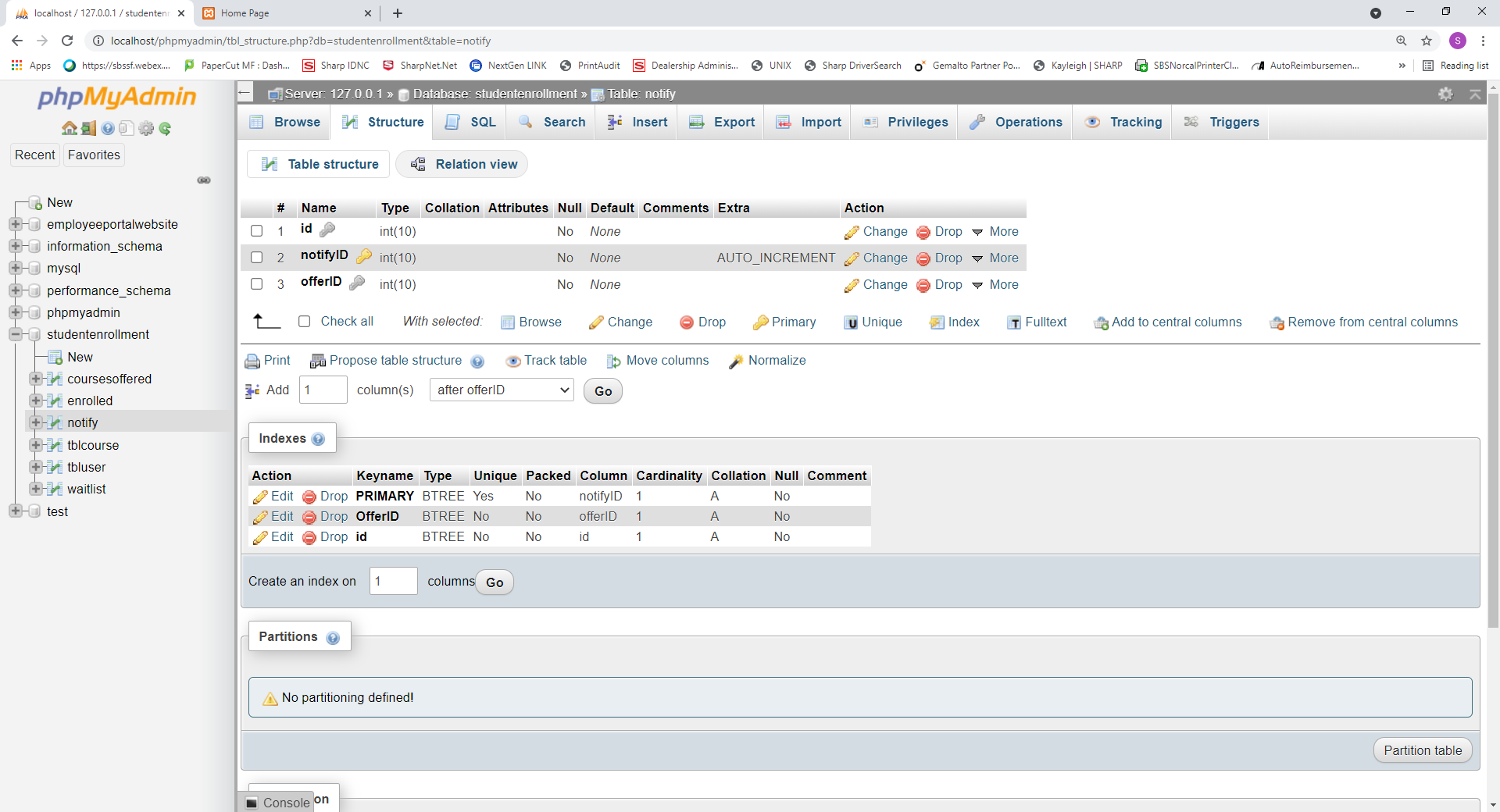


Figure 23: Browse View of notify

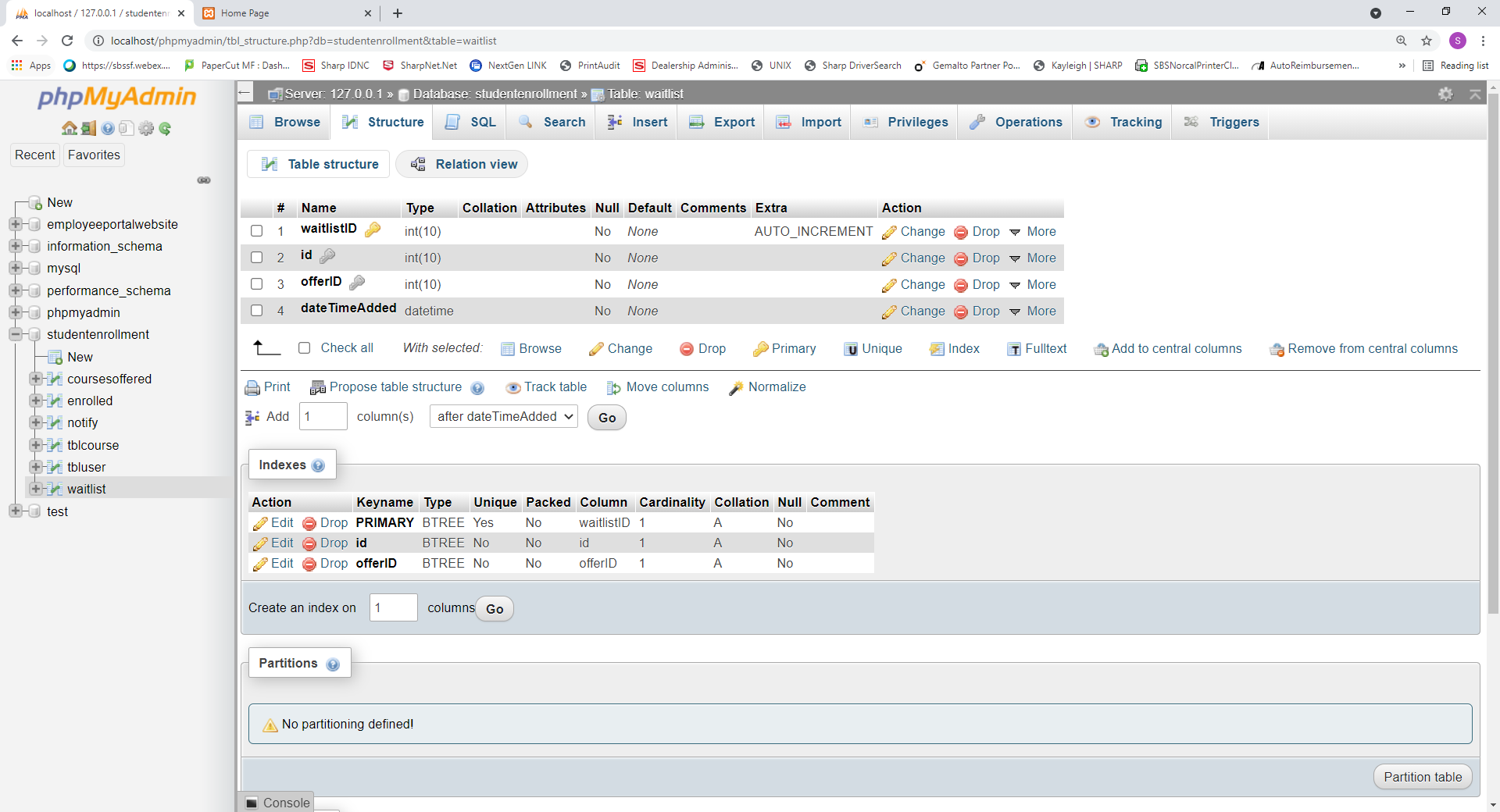


Figure 24: Structural View of waitlist

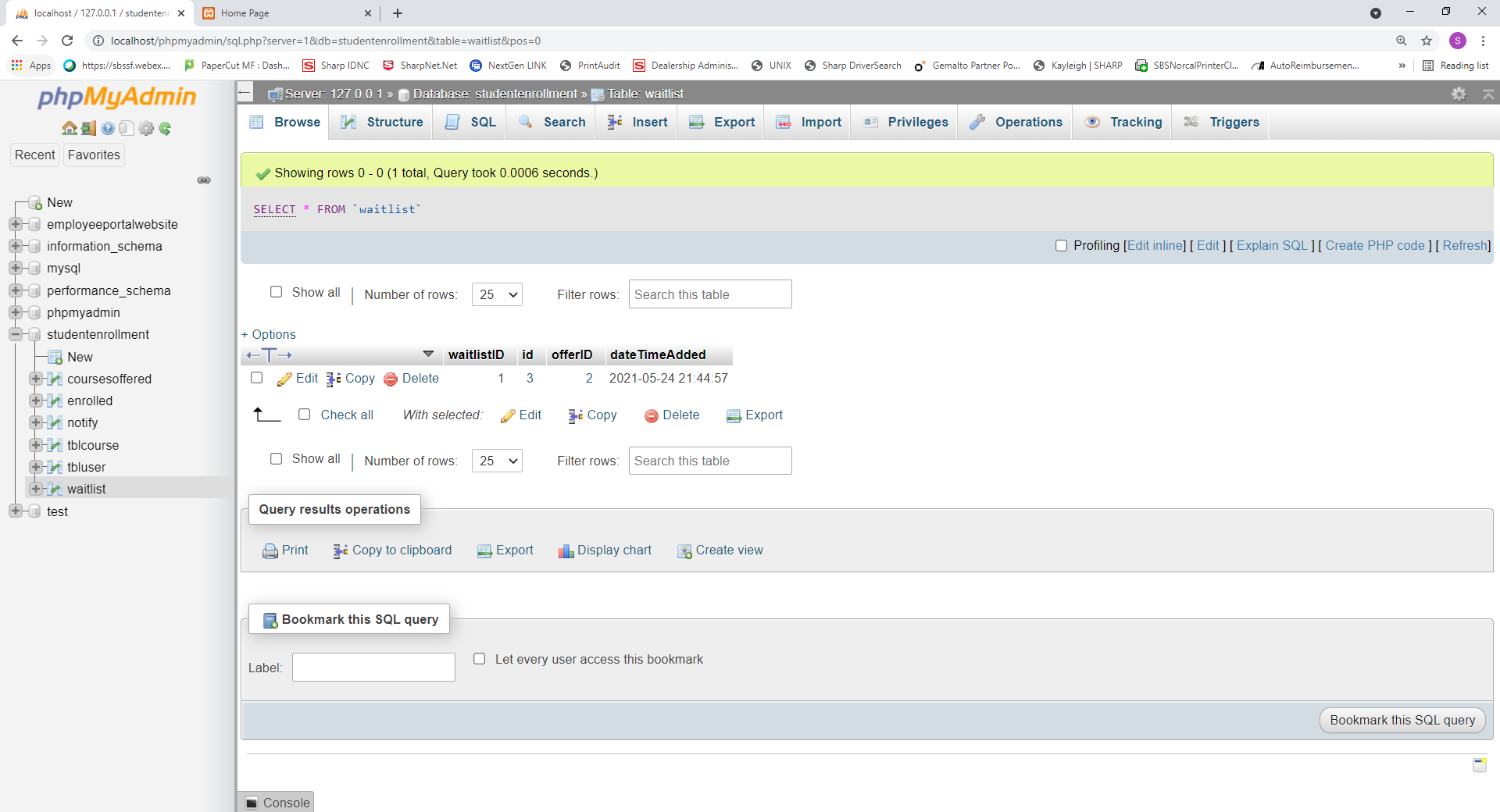


Figure 25: Browse View of waitlist

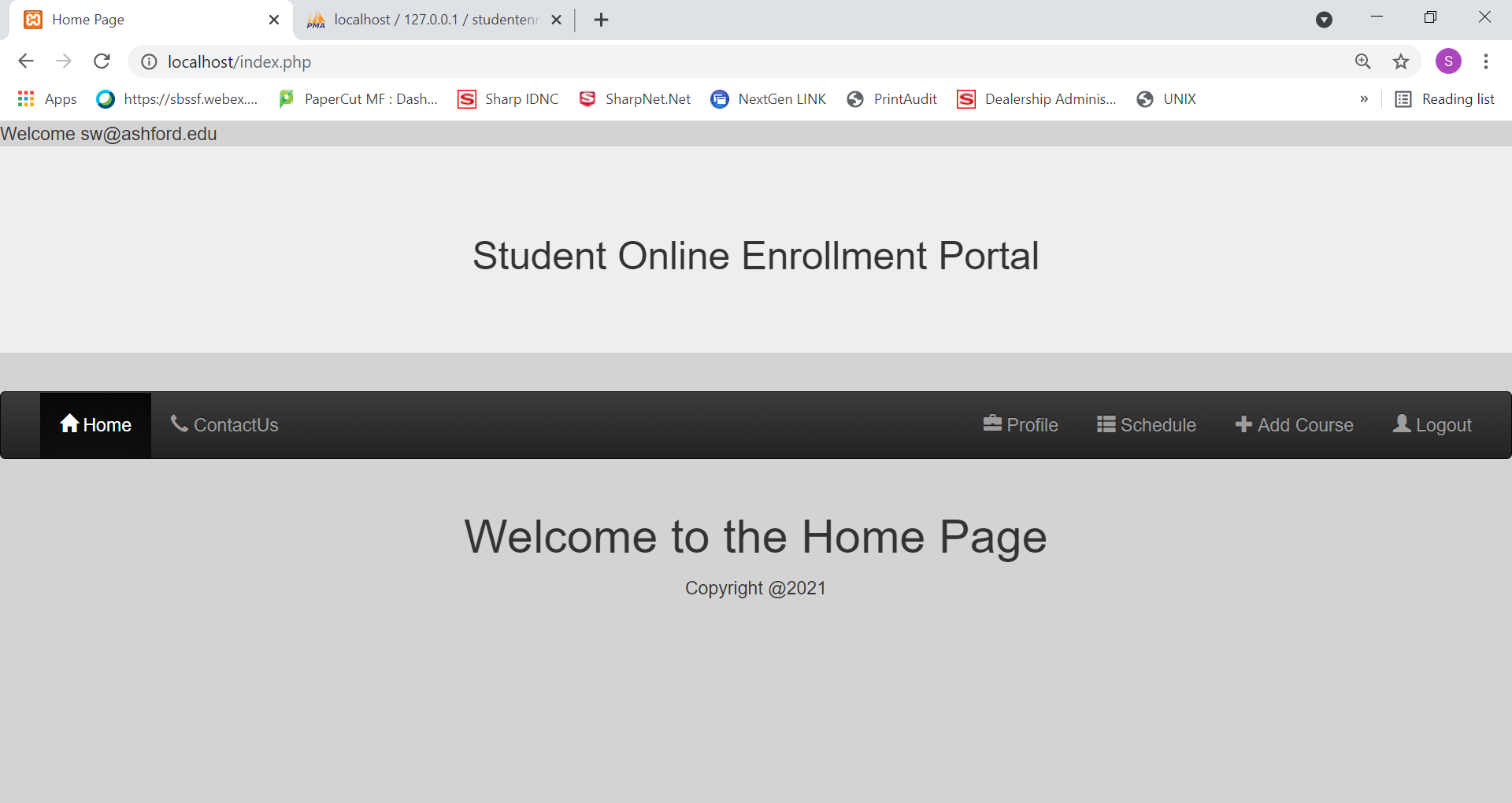


Figure 26: Home Page

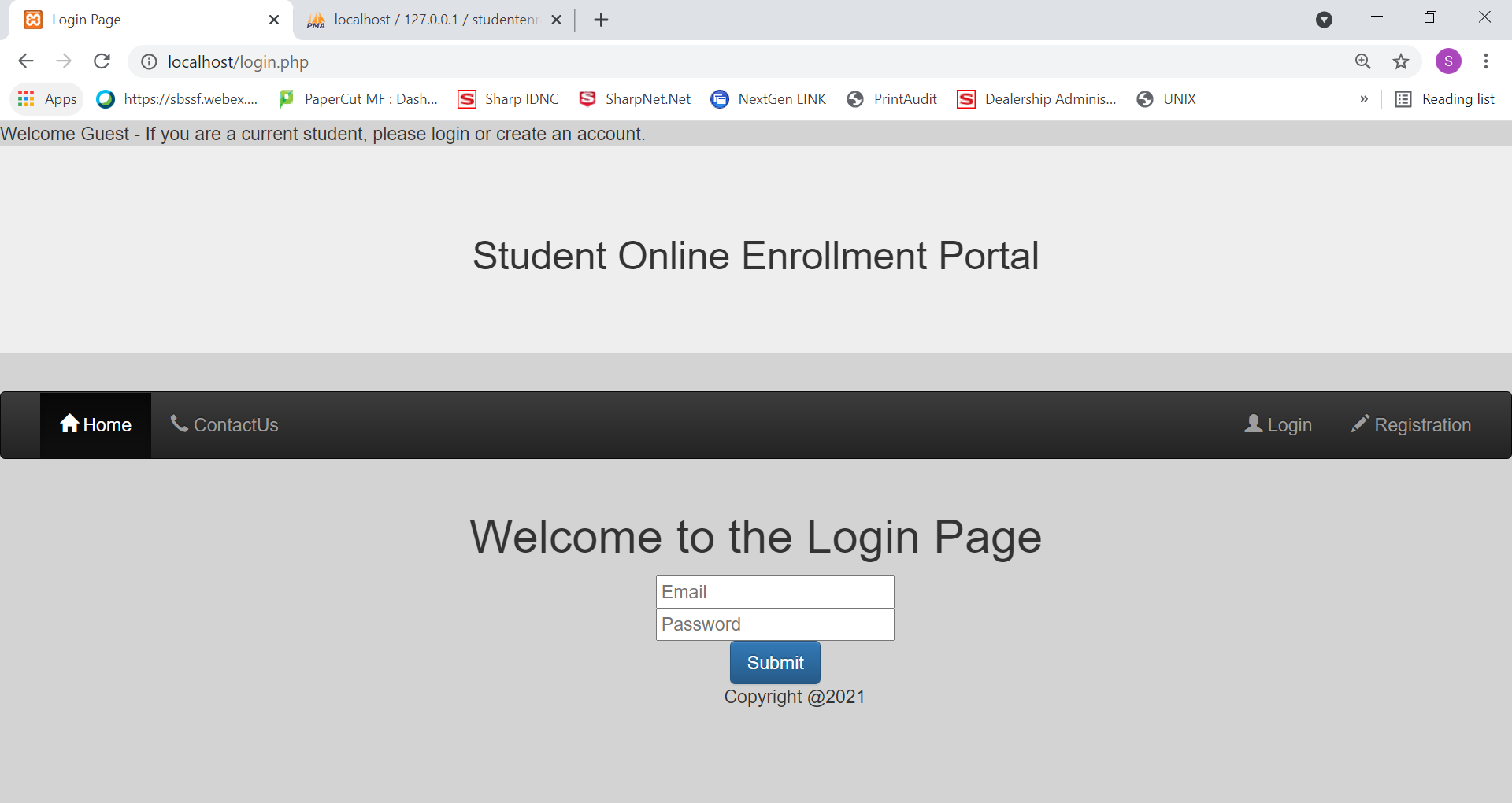


Figure 27: Login Page

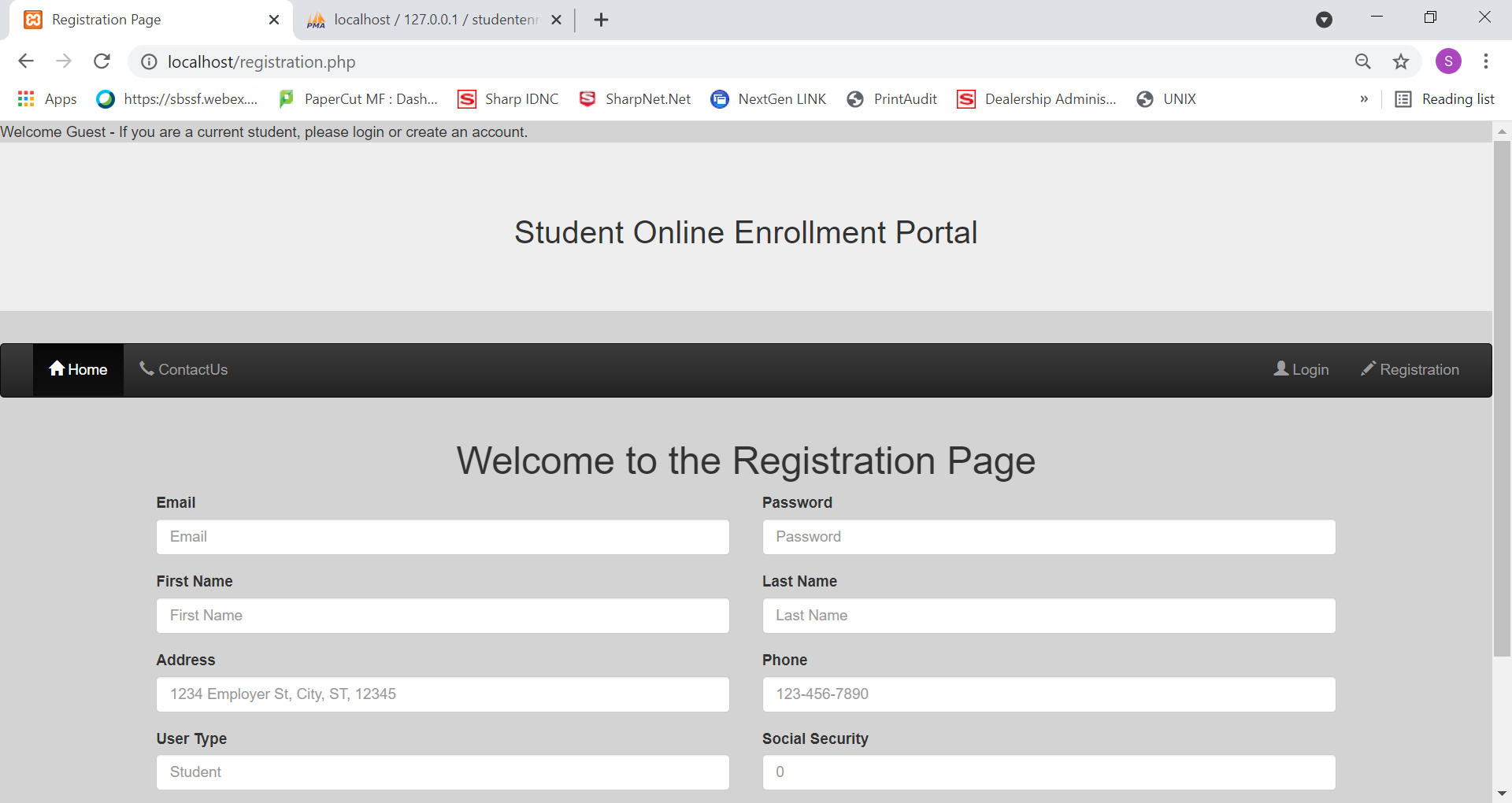


Figure 28: Registration Page

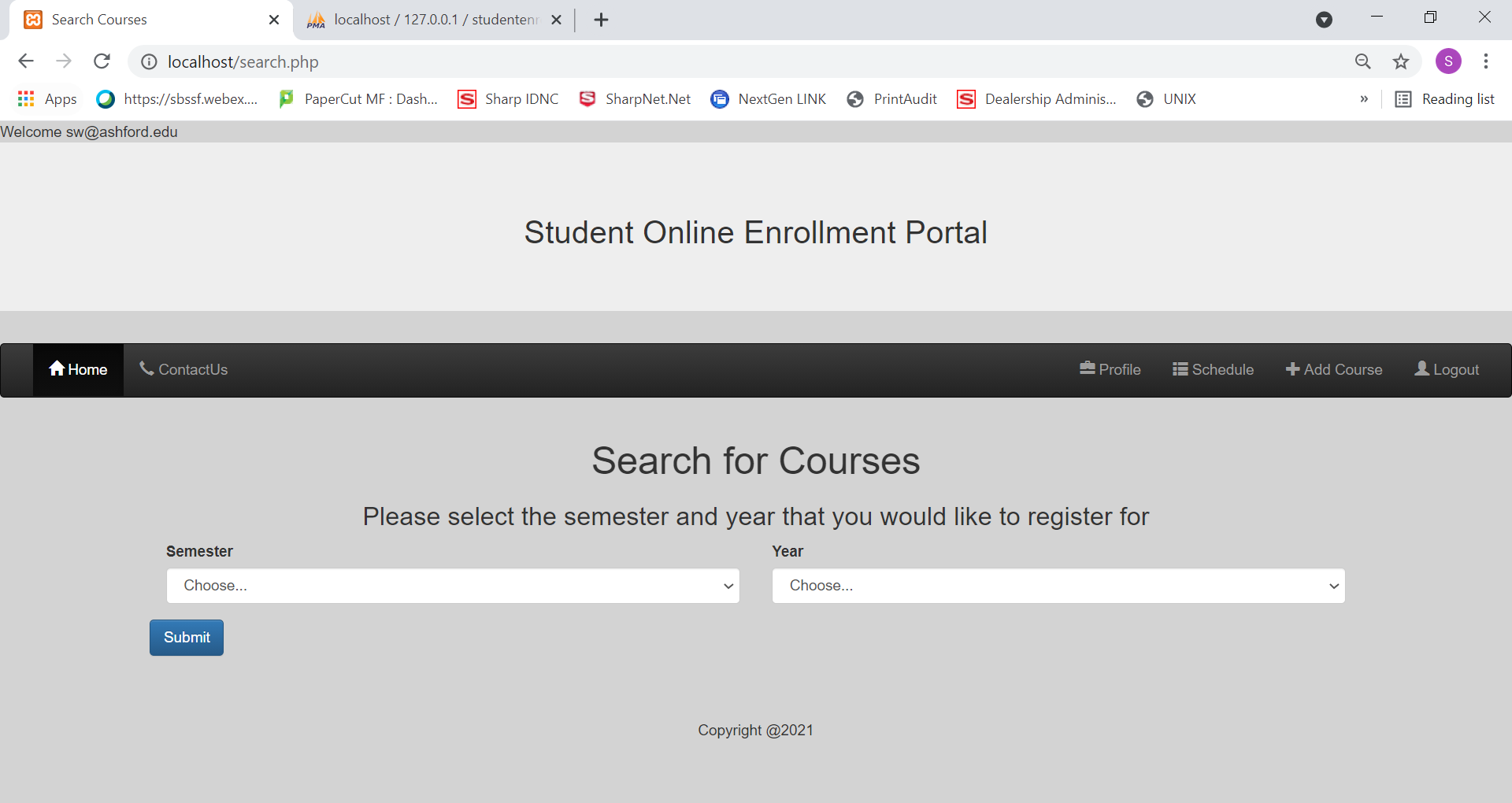


Figure 29: Search Course Page with Drop-down Selection Feature

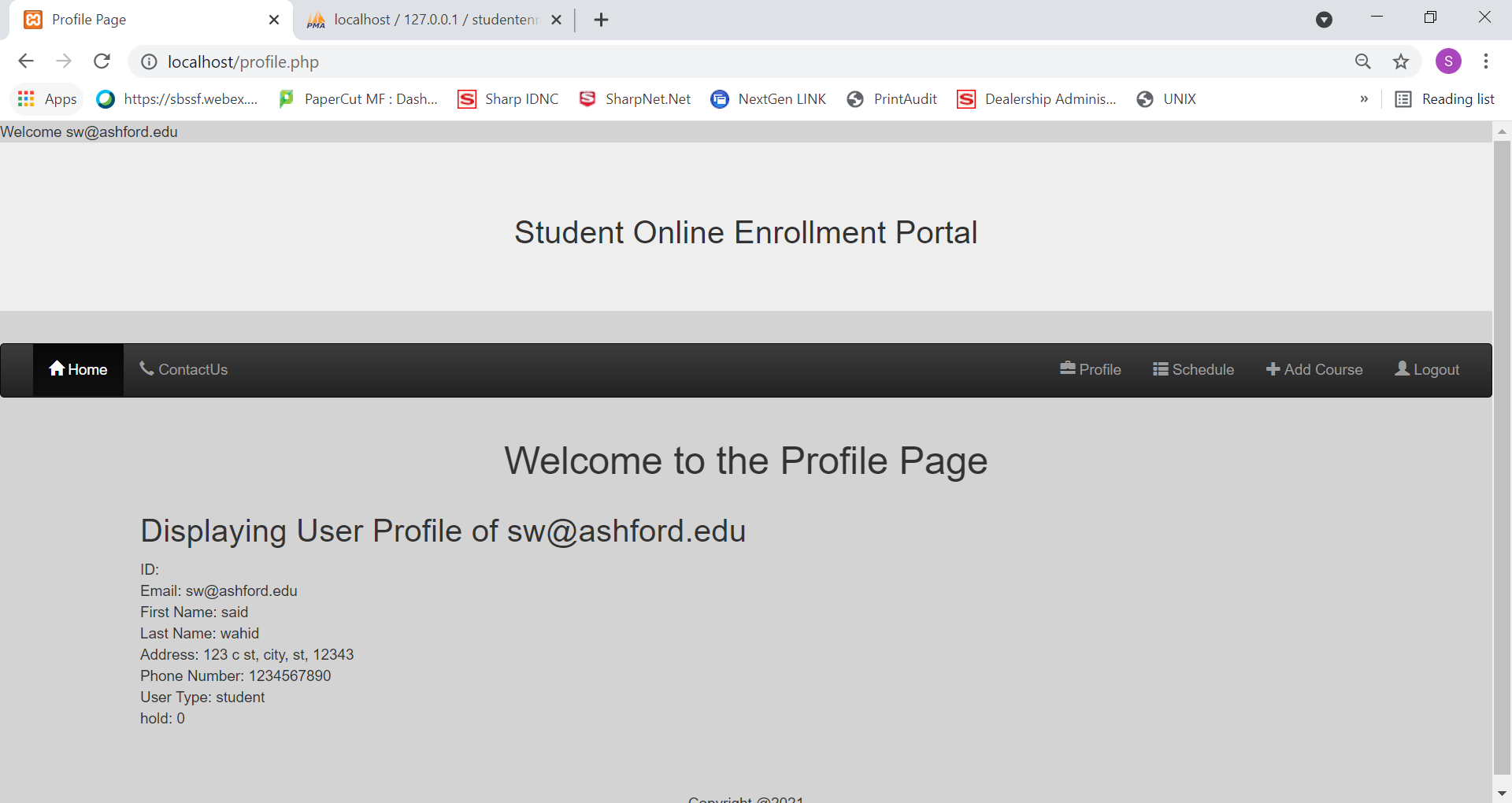


Figure 30: Profile Page

# References

Apache Friends. (2020). What is XAMPP? Retrieved from <https://www.apachefriends.org/index.html>

Curacubby Team. (2020, Nov 19). Why Your School Needs Online Enrollment + How to Set It Up. Retrieved from <https://www.curacubby.com/blog/your-school-needs-online-enrollment>

DowntimeMonkey.com. (2018, April 25). How Much Downtime is Acceptable? [updated 2019, Oct 31]. Retrieved from <https://downtimemonkey.com/blog/how-much-downtime-is-acceptable.php>

Filipova O., & Nikiforova Oksana. (2019). Definition of the Criteria for Layout of the UML Use Case Diagrams. Applied Computer Systems, 24(1), 75–81. <https://doi-org.proxy-library.ashford.edu/10.2478/acss-2019-0010>

Miller, W. (2019, September 11). Risks of Not Testing Software Properly. iBeta Labs [Blog]. Retrieved from <https://www.ibeta.com/risks-of-not-testing-software-properly/>

Rani, B. (2020, Sep 20). Unified Modeling Language (UML) | Class Diagrams. GeeksforGeeks. Retrieved from <https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/>

Spillner, A., Linz, T., & Schaefer, H. (2014). Software testing foundations: A study guide for the certified tester exam (4th ed.). Rocky Nook.

TestDevLab. (2018, July 3). 4 Reasons Why Software Testing is Important. [Blog]. Retrieved from <https://www.testdevlab.com/blog/2018/07/importance-of-software-testing/>

Tsui, F., Karam, O., & Bernal, B. (2018). Essentials of software engineering (4th ed.). Jones & a Bartlett Learning