**Database Development and Class Registration**

Bill Maggs

The University of Arizona Global Campus

CST499: Capstone for Computer Software Technology

Dr. Charmelia Butler

October 27th, 2024

This paper builds on the development of a university enrollment website designed to streamline the process of student registration and course management. In the initial phase, essential elements such as user registration, login, and basic database functions were successfully implemented. However, to expand functionality and create a fully interactive system, additional features were necessary. The latest project iteration introduces backend database tables and integrates dynamic interactions that allow students to register for classes, view their enrolled courses, add additional courses, and modify their schedules by dropping classes. This integration with MySQL and PHP extends the website’s capabilities, creating a user-friendly and efficient platform for university enrollment. This paper will show how the enhanced functionality supports real-time management of student schedules and improves user accessibility, showcasing the website’s scalability and potential as a comprehensive enrollment solution.

**Revise work in Week 3 based on the feedback you received from your instructor.**

* Feedback #1 - Elaborate on error handling in your connection class.

The previous code would simply stop the script if an error was detected, sending unintelligible information to the user. This update is a much improved user experience from the week 3 feedback using exception handling (W3schools, 2024, PHP Exception Handling). Try/Catch was implemented on the object creation to catch any error (Stack overflow, 2019, Try/catch in mysqli). On any exception, a User message is sent out to the page to contact IT.

**Figure 1:**

*Updated source code from addReg.php*

A screen shot of a computer program

Description automatically generated

**Figure 2:**

*Updated user output on any error*

A screenshot of a computer

Description automatically generated

* Feedback #2 - Include the SQL commands used for creating the database and tables.

An export was completed from myphpadmin on the entire University database showing the table creation. For brevity, the file has been edited to remove comments and blank line spaces. The full university.sql file can be downloaded from the git repository for the project (Maggs, 2024).

*--*

*-- Table structure for table `tblcatalog`*

*--*

*CREATE TABLE `tblcatalog` (*

*`p\_cid` int(128) NOT NULL,*

*`p\_cname` varchar(255) NOT NULL,*

*`p\_csemester` varchar(128) NOT NULL,*

*`p\_cseats` int(16) NOT NULL,*

*`p\_cfilled` int(16) NOT NULL*

*) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;*

*--*

*-- Table structure for table `tblenrollment`*

*--*

*CREATE TABLE `tblenrollment` (*

*`p\_id` int(128) NOT NULL,*

*`p\_cid` int(128) NOT NULL*

*) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;*

*--*

*-- Table structure for table `tbluser`*

*--*

*CREATE TABLE `tbluser` (*

*`p\_id` int(128) NOT NULL,*

*`p\_email` varchar(255) NOT NULL,*

*`p\_password` varchar(16) NOT NULL,*

*`p\_fname` varchar(16) NOT NULL,*

*`p\_lname` varchar(16) NOT NULL,*

*`p\_address` varchar(255) NOT NULL,*

*`p\_phone` varchar(16) NOT NULL,*

*`p\_role` varchar(16) NOT NULL*

*) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;*

*--*

*-- Table structure for table `twaitlist`*

*--*

*CREATE TABLE `twaitlist` (*

*`p\_id` int(128) NOT NULL,*

*`p\_cid` int(128) NOT NULL*

*) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;*

*--*

*-- Indexes for table `tblcatalog`*

*--*

*ALTER TABLE `tblcatalog`*

*ADD PRIMARY KEY (`p\_cid`);*

*--*

*-- Indexes for table `tblenrollment`*

*--*

*ALTER TABLE `tblenrollment`*

*ADD KEY `p\_id` (`p\_id`),*

*ADD KEY `p\_cid` (`p\_cid`);*

*--*

*-- Indexes for table `tbluser`*

*--*

*ALTER TABLE `tbluser`*

*ADD PRIMARY KEY (`p\_id`),*

*ADD KEY `p\_id` (`p\_id`);*

*--*

*-- Indexes for table `twaitlist`*

*--*

*ALTER TABLE `twaitlist`*

*ADD KEY `p\_id` (`p\_id`),*

*ADD KEY `p\_cid` (`p\_cid`);*

*--*

*-- AUTO\_INCREMENT for table `tblcatalog`*

*--*

*ALTER TABLE `tblcatalog`*

*MODIFY `p\_cid` int(128) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;*

*--*

*-- AUTO\_INCREMENT for table `tbluser`*

*--*

*ALTER TABLE `tbluser`*

*MODIFY `p\_id` int(128) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;*

*--*

*-- Constraints for table `tblenrollment`*

*--*

*ALTER TABLE `tblenrollment`*

*ADD CONSTRAINT `tblenrollment\_ibfk\_1` FOREIGN KEY (`p\_id`) REFERENCES `tbluser` (`p\_id`),*

*ADD CONSTRAINT `tblenrollment\_ibfk\_2` FOREIGN KEY (`p\_cid`) REFERENCES `tblcatalog` (`p\_cid`);*

*--*

*-- Constraints for table `twaitlist`*

*--*

*ALTER TABLE `twaitlist`*

*ADD CONSTRAINT `twaitlist\_ibfk\_1` FOREIGN KEY (`p\_id`) REFERENCES `tbluser` (`p\_id`),*

*ADD CONSTRAINT `twaitlist\_ibfk\_2` FOREIGN KEY (`p\_cid`) REFERENCES `tblcatalog` (`p\_cid`);*

*COMMIT;*

**Create your tables within the MySQL database related to rest of your design.**

Figure 3 shows the University Schema, the tables used in the database, their data types and relationships. These tables were created using phpMyAdmin, and the Designer tool used to show the relationships.

The first table created within the database is tbluser, with a primary key of p\_id assigned during the registration process. The remainder of the fields and datatypes can be seen in the figure, however of note should be the p\_role field which allows for access control. By default this field is assigned as ‘student’, however the database administration can change the field to other role types such as ‘admin’.

The second table of primary importance is the tblcatalog, which has all of the course ID’s as the primary key (again, auto-generated so to remain unique), the course name, semester offered, and fields for how many seats are available and filled. As a student registers for a course, the filled field will be updated.

There are two additional tables both related to the primary tables. The tblenrollment is the full enrollment listing with each row being a foreign key of the user id and course id. The other is tblwaitlist which also contains foreign keys of the user id and course id being waited on.

**Figure 3:**

*Database tables, data types, and relationships*

A screenshot of a computer

Description automatically generated

**Create the different pages related to the rest of the requirements and the design per your work in Week 1 and Week 2**.

The pages were completed, and the page design will be documented in the remainder of this paper. The general flow of the complete website and pages is shown in figure 4.

**Figure 4:**

*Website flow showing pages and associated php scripts*

A diagram of a website

Description automatically generated

**Generate screenshots of the database and the tables you created.**

The screenshot of the overall database and tables are in figure 3. The following in figures 5 and 6 are the current default populated tables; contents will change as key operations are performed which will be shown in the next section.

**Figure 5:**

*Tbluser holding user information*

A screenshot of a computer

Description automatically generated

**Figure 6:**

*Tblcatalog holding course information*

A screenshot of a computer

Description automatically generated

The enrollment and waitlist tables from figure 3 are empty at this stage, screenshots will be provided as they receive content during a user’s enrollment process in the next section.

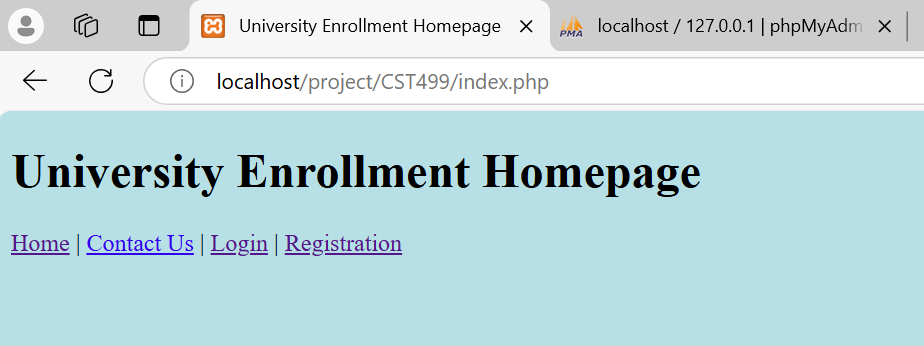
**Make sure to include the database before and after each key operation you did to show the success of your implementation. Generate screenshots of the different pages that you created.**

These two objectives are in a blended format, since created pages are shown with the table updates. This will all together be considered a walk-through of the updated functionality of the website. An example will be shown of each individual operation, as it would be redundant to show (for example) the updated user table every time a new user registers, the function will need only shown once with the table change before and after.

Using the website homepage in figure 7 as a given starting point, an example of a user logging in, adding several classes, dropping a class, and placing a class on a waitlist will be shown, along with the associated database changes. The action of adding a user through registration was shown in the previous paper, along with the associated database change.

**Figure 7:**

*Homescreen with no current login*



Once a user is logged in, the screen will change to allow forward movement into the course registration system. Figure 8 is the login screen and following confirmation screen, figure 9 shows the home page after a user is successfully logged in.

(remainder of page intentionally blank for formatting)

**Figure 8:**

*Logging in and confirmation screens*

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Figure 9:**

*Updated home screen*

A screen shot of a computer

Description automatically generated

Note the addition of Course Selection and View My Courses links, this is a conditional screen which will be covered in the php code overview.

A student may now go through the course selection process and enroll in open courses. As shown in figure 10, the enrollment table is currently empty, as no students have registered for any courses.

**Figure 10:**

*Empty enrollment table*

A screenshot of a computer

Description automatically generated

To begin enrollment, a student would choose the Course Selection link to view the enrollment actions as shown in figure 11.

**Figure 11:**

*Enrollment actions*

A screen shot of a computer

Description automatically generated

First, a student should View Courses to see the school catalog. At this time, there is no option yet for an instructor to be able to add or remove courses from the catalog, this is a feature that will be implemented if time allows. It is currently an admin feature supported through phpMyAdmin, therefore the tblcatalog in figure 6 will not be re-shown as it will not change. The View Courses tab displays the tblcatalog as shown in figure 12. Note each course displays the current capacity, and how many seats have been taken. Since this is the first enrollment, all current enrolled seats are at zero.

(remainder of page intentionally blank for formatting)

**Figure 12:**

*Course catalog*

A screenshot of a computer

Description automatically generated

The student may now select the Add Course link, with the page following in figure 13.

**Figure 13:**

*Adding a course*

A screenshot of a computer

Description automatically generated

Once the request is submitted, the course is checked for capacity. If there is an open seat, a confirmation is given as shown in figure 14.

**Figure 14:**

*Confirmation*

A screen shot of a computer

Description automatically generated

Two tables are affected by the registration. Note in figure 15 tblcatalog p\_cfilled has been incremented by one (dbvis, 2024), and the student and course ID’s have been added to tblenrollment.

**Figure 14:**

*tblenrollment and tblcatalog affected by enrollment.*

A close up of a number

Description automatically generatedA screenshot of a computer

Description automatically generated

The student may now return to the home screen, and view their current courses as shown in figure 15.

**Figure 15:**

*Current courses*

A screenshot of a web page

Description automatically generated

The student has now decided they wish to drop the course. Using the Drop Course link, the student is led to the page shown in figure 16.

Figure 16:

Drop a course

A screenshot of a computer

Description automatically generated

Once the request is submitted, a confirmation is given along with the updated enrollment for that course as shown in figure 17.

**Figure 17:**

*Confirmation and updated enrollment*

A screenshot of a computer

Description automatically generated

This again affects two tables. The tblcatalog is now decremented, and the student and course removed from the tblenrollment as shown in figure 18.

**Figure 18:**

*tblenrollment and tblcatalog after dropping a course*

A white rectangular sign with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

Next to be shown is waitlist. In this scenario, the tblcatalog has been artificially filled to capacity for the Fall offering of Algebra 101, and the student will attempt to enroll. Figure 19 shows tblcatalog before the student attempts to enroll.

**Figure 19:**

*Fall Algebra is filled to capacity*

A screenshot of a computer

Description automatically generated

The student attempts to add the full course as shown in figure 18. The confirmation screen will now show unable to register, and recommends the link to waitlist as shown in figure 19..

(remainder of page intentionally blank for formatting)

**Figure 18:**

*Attempting to register for a full course.*

A screenshot of a computer

Description automatically generated

**Figure 19:**

*Notification of full class, recommend waitlist.*

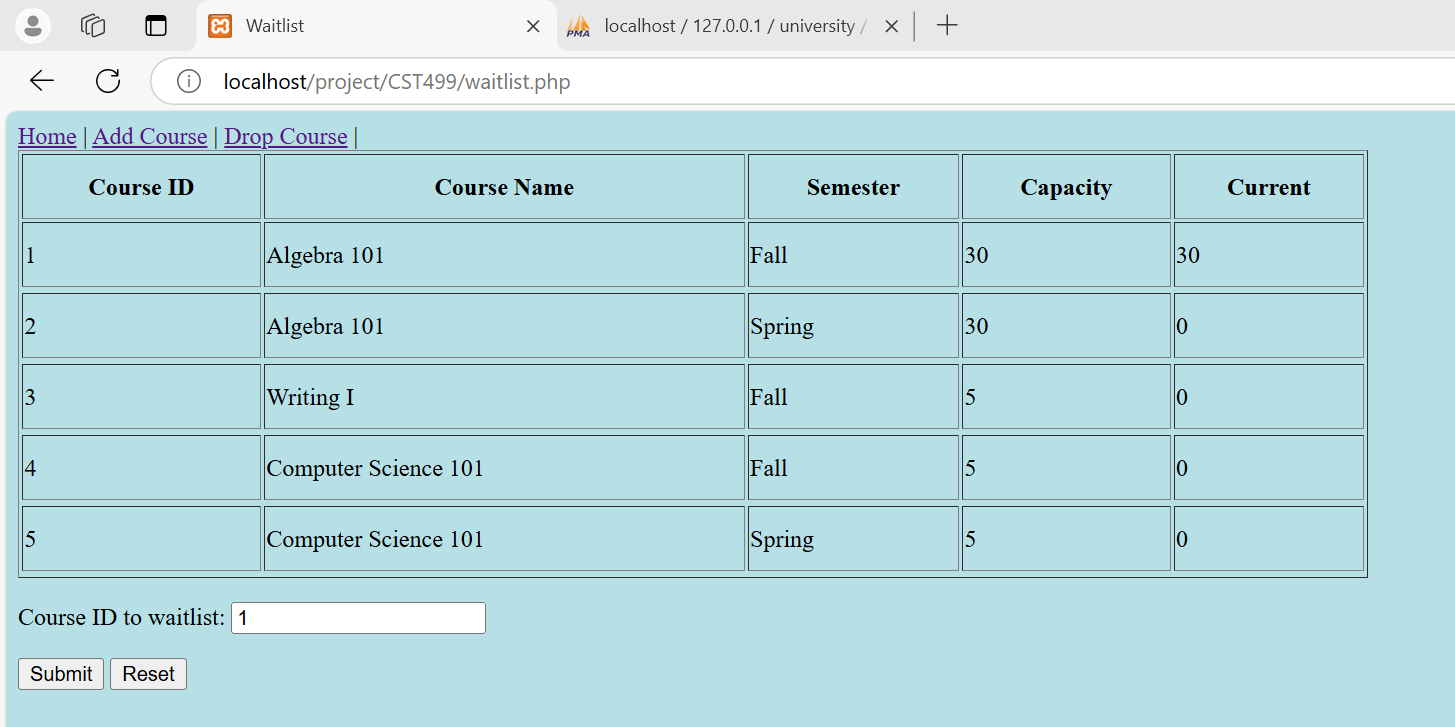
A screenshot of a computer

Description automatically generated

Upon following the waitlist link, the student is taken to the waitlist request page, where they can enter the course ID for the requested waitlist as shown in figure 20.

**Figure 20:**

*Waitlist request form*



Once submitted, a confirmation is given in figure 21. Figure 22 shows the associated twaitlist table has been updated to shown the student ID and course ID being waitlisted.

**Figure 21:**

Waitlist confirmation

A screenshot of a computer

Description automatically generated

**Figure 22:**

*Updated twaitlist*

A screenshot of a computer

Description automatically generated

Lastly, the student should have the ability to logout. From the homepage, if logout is selected the student is directed to the logout page as shown in figure 23.

**Figure 23:**

*logout page.*

A screenshot of a computer

Description automatically generated

If the student chooses logout, the page resets to the original login page with no access to enrollment as shown in figure 7.

**Generate screenshots of the PHP code to implement your logic.**

Including the addition of the logout function not shown in the site map of figure 4, at this time there are 15 php scripts, with 1,305 lines of code. It is beyond the scope of this paper to show all of the php code. It is recommend the reader follow the link to the github site (Maggs, 2024) in the references to view the full code following the site map. However, some excerpts will be shown of interesting implementations.

When a user registers for the website, a unique user id is assigned into tbluser. In the confirmation, the table must be read again after the insert to show the user their ID, which would be the last row of the table (Stack overflow, 2021, How to select the last record of a table in SQL?). The snippet is shown in figure 24. A message is given to remember the student ID.

**Figure 24:**

*Displaying a newly assigned user ID*

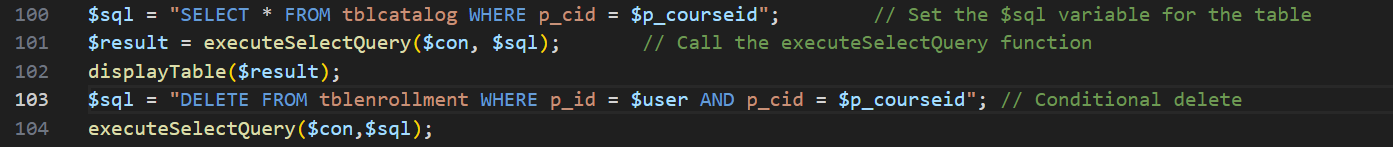
A screen shot of a computer code

Description automatically generated

When a student drops a course, the tblenrollment would potentially have the same student ID in multiple rows, associated with a different course ID. The delete a row from the tblenrollment, a “conditional” delete query needed to be constructed as shown in figure 25 (Stack overflow, 2011, Delete rows conditionally). This also required an updated use of the WHERE clause in SQL (w3schools, 2024, SQL WHERE clause).

**Figure 25:**

*Conditional row delete in php*



**Summarize your experience going through the implementation phase.**

Note: Since this question requires a subjective personal response, I will switch from academic third party voice to business tone.

As with any software development project, time is the enemy. Even though some of the work was adapted from previous courses, all of the webpages and php scripts required editing or completely new coding, with some new techniques and commands requiring a quick learning curve. In a statement, 100% of this code was hand-generated without the use of code generation tools or AI. With all of the fast-paced courses in between CST310 where the original code base was written and CST499, there was a considerable relearning phase. I am glad my original code was well documented.

There is also consideration for future enhancements that were not covered in the original SRS or rubric. Given two more weeks much of this could be implemented:

1. Suppose a student forgets their user ID, how do they recover it?
2. For security, there is no password restriction or limit to the number of times a student could try logging in, leaving the database open to probing attacks.
3. There is no testing on the data entered by the user. For example, a field could be left blank in the registration, and a record is still created. A student may also register for a course multiple times, there is no checking when a student selects a course to see if they are previously enrolled. Though this is actually helpful for testing since a class can be filled by one student, avoiding logging in five different students to fill a class, this would not be desirable in a real system.
4. Admin functions have been left in the flow diagram for viewing complete enrollment and adding courses to the catalog. These will be implemented if time allows as they are not in the rubric but would be interesting.

Finally, there have been changes as the project developed such as original UML classes, flowcharts, and how the site is organized. Though every attempt has been made to go back through and update documentation, I can see the high value of having a review team to assist in catching changes and updating appropriately.

**Conclusion**

In conclusion, the development of these additional features significantly elevates the utility of the university enrollment website. By integrating course registration and schedule management functionalities, the platform now meets core requirements for a fully operational enrollment system. The backend database enhancements and PHP-driven dynamic interactions enable seamless user experiences, reflecting the technical efficacy and practical benefits of the project. These updates transform the website from a basic enrollment interface into an adaptable, interactive tool that meets the diverse needs of students and administrators alike. This progression underscores the value of incremental functionality and demonstrates the potential for further enhancements to align with evolving academic requirements.

**References:**

Maggs, W. (2024). *CST499 Project Files*. Github. Retrieved from [bmaggsjr/CST499: GITHUB for CST499 Assignments](https://github.com/bmaggsjr/CST499)

Stack overflow (2021) *How to select the last record of a table in SQL?* Stack Overflow. Retrieved from [sql server - How to select the last record of a table in SQL? - Stack Overflow](https://stackoverflow.com/questions/5191503/how-to-select-the-last-record-of-a-table-in-sql)

Stack overflow (2019) *Try/catch in mysqli.* Stack overflow. Retrieved from [php - Try / catch in mysqli - Stack Overflow](https://stackoverflow.com/questions/27345377/try-catch-in-mysqli)

W3schools (2024) *PHP Exception Handling*. W3schools. Retrieved from [PHP Exception Handling](https://www.w3schools.com/php/php_exception.asp)

Dbvis (2024) *How to update a value incrementally (+1) in MySQL*. Dbvis. Retrieved from [Default SEO Title](https://www.dbvis.com/thetable/mysql-incremental-value-update/)

W3schools (2024) *SQL WHERE Clause.* W3schools. Retrieved from [SQL WHERE Clause](https://www.w3schools.com/sql/sql_where.asp)

Stack overflow (2011) *Delete rows conditionally*. Stack overflow. Retrieved from [sql - Delete rows conditionally - Stack Overflow](https://stackoverflow.com/questions/5588946/delete-rows-conditionally)