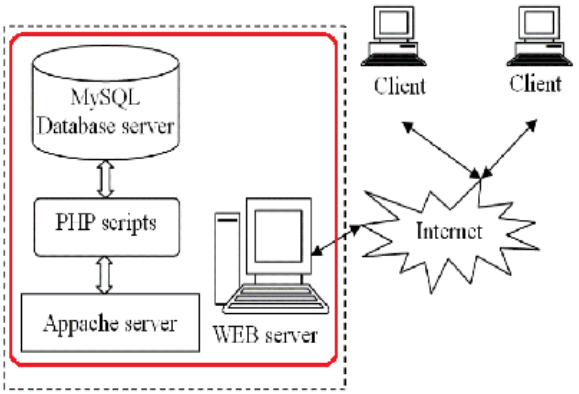
CP476
Internet Computing
Group Final Project
April 3, 2023
Group 7

Beheashta Atchekzai - 190637520
Nicholas Rudowski - 191963270
Scott Richardson - 190655350
Tyler Gillies - 190474580

Introduction

The purpose of this design document is to provide an overview of our web server project and its corresponding database and server-side scripts. The goal of the project was to create a web server that interacts with web users and a locally hosted database, where the web server is Apache and the database is MySQL. The server-side language that was used was PHP, and this was all set up on a Windows device. The project requires the execution of at least two different SQL statements initiated by web users, we chose to do a SELECT and UPDATE statement for this. The database will consist of two tables, Name Table and Course Table. A requirement of the project is to output all of the student's final calculated grades based on the data in the two input tables. The output format is described in Appendix B. This design document will provide an overview of the project, including the architecture, database design, SQL programming, user interface, output format, implementation details, security considerations, and testing and validation.

Architecture Overview



The design and construction of our project architecture includes the utilisation and implementation of a 3-tiered approach. Having a front-end for users to interact with, a back-end to handle all the information, and then our database to store and share our data of the students and courses. This 3-tiered architecture approach utilises PHP scripts embedded with HTML for our users to interact with, hosted on a local Apache Server, and with PHP connected to our MySQL database server.

Our front-end allows us to interact with the users, and this is done through our HTML scripts which greet the user and allow for interactivity through execution of PHP commands implemented through the back-end. All running on an Apache server, allowing for the incoming HTTP request from our client side users to then interact with said HTML content. This whole project reaches our users by connecting our server side information with the client side machines.

Once our HTML content hosted on Apache servers are interacted with, the PHP Scripts start to do their job. Connecting to our MySQL Database server which hosts our site's student and course information. The main tables hosted were given and contain information for all the students, their ID numbers, their courses, and respective course grades. This MySQL part of our architecture is considered the data layer.

The back-end connection between our users and the data is controlled and executed through our PHP scripts. The PHP has connected to our database and uses prepared SQL Statements to prompt our MySQL database for data. These prepared SQL Statements run through our PHP scripts both update and select data from our MySQL database and then work with the other layers of the architecture to output that data then back to our users on the

front-end. The entire implementation of our architecture allows for secure, reliable, and scalable operation for the numerous different scenarios in which this project will operate.

Database Design

Our project uses a MySQL database, named StudentGradesDB, consisting of a NameTable and a CourseTable. No relationships or constraints exist within the database. The database was created through the use of the mySQLi extension in PHP. PHP code was written to execute SQL statements in order to create the database and its tables, as well as to fill the tables with the assigned values.

The NameTable is made up of two columns, the StudentID column and the StudentName column. All columns in this table are of type VARCHAR. The CourseTable is made up of six columns. The first two columns are the StudentID and CourseID columns. The remaining four columns contain grade values and are named Test1, Test2, Test3, Exam. All of the columns in this table are of type VARCHAR.

```
mysql> use StudentGradesDB;
Database changed
mysql> SELECT * FROM CourseTable;
+-----------+----------+-------+-------+-------+------+
| StudentID | CourseID | Test1 | Test2 | Test3 | Exam |
+-----------+----------+-------+-------+-------+------+
| 280587734 | PS272    | 75    | 75    | 75    | 0    |
| 280587734 | CH202    | 66    | 82    | 81    | 88   |
| 256047895 | MA222    | 69    | 80    | 72    | 87   |
| 154102471 | CP465    | 63    | 82    | 58    | 68   |
| 187509717 | CP202    | 58    | 98    | 56    | 89   |
| 503239671 | ST262    | 66    | 84    | 95    | 88   |
| 448227065 | CP465    | 59    | 69    | 56    | 96   |
| 429464715 | CH120    | 54    | 93    | 71    | 80   |
| 627137015 | EC140    | 85    | 56    | 72    | 77   |
| 415807676 | EC140    | 70    | 89    | 90    | 63   |
| 293688639 | CH120    | 85    | 80    | 78    | 83   |
| 256047895 | EC140    | 66    | 53    | 85    | 50   |
| 350971244 | BU121    | 55    | 96    | 75    | 95   |
| 308621686 | EC140    | 90    | 80    | 70    | 60   |
| 570797438 | CP321    | 81    | 71    | 65    | 96   |
| 505004484 | ST494    | 70    | 95    | 54    | 62   |
| 251173274 | PS275    | 86    | 64    | 65    | 59   |
| 397016834 | CP220    | 84    | 78    | 55    | 55   |
```

```
mysql> SELECT * FROM NameTable;
+-----------+-------------------+
| StudentID | StudentName       |
+-----------+-------------------+
| 308621686 | Boone Stevenson   |
| 448227065 | Micheal Conrad    |
| 309251919 | Kayla Conway      |
| 350971244 | Belinda Bain      |
| 415807676 | Autumn Schmidt    |
| 603077700 | Rahul Prosser     |
| 547161604 | Ayyan Whiteley    |
| 187509717 | Ameena Khan       |
| 309663833 | Bertram Smith     |
| 293688639 | Dominique Lovel   |
| 570797438 | Minnie Rivers     |
| 403966911 | Liang Yu          |
| 559545416 | Alexander Floydd  |
| 503239671 | Matthew Hall      |
| 256047895 | Lori Donovan      |
| 301758883 | Ellie-May Palmer  |
| 627137015 | Keaton Sheppard   |
| 429464715 | Tiago Rivera      |
| 458362883 | Krishan Patel     |
| 280587734 | Kendra Paul       |
| 613465484 | Leonard Whitehead |
| 154102471 | James Andersen    |
| 397016834 | Hermione Bullock  |
| 505004484 | Emran Bashir      |
```

SQL Programming

Our Project utilises several prepared SQL Statements to interact with our MySQL database and to retrieve and modify data related to the students and courses. We use prepared statements over direct run SQL statements because prepared statements offer greater security and reliability through separating the actual SQL code and the input data, binding placeholder variables to the input data – This helps prevent SQL injection attack.

Here is an example of one of many SELECT statements in our program. This one is used to get data from merged tables – the TestCourseTable and TestNameTable – this represents Appendix B:

```php
//Grab all the grades for this course
$sql = $conn->prepare("SELECT CourseTable.StudentID, CourseTable.CourseID, CourseTable.Test1,
CourseTable.Test2, CourseTable.Test3, CourseTable.Exam, NameTable.StudentName FROM CourseTable
INNER JOIN NameTable ON CourseTable.StudentID=NameTable.StudentID WHERE CourseID = ?");
$sql->bind_param("s", $courseID);
$sql->execute();

$result = $sql->get_result();
```

This SELECT statement retrieves the variables StudentID, CourseID, Test1, Test2, Test3, Exam, and StudentName columns from the TestCourseTable and TestNameTable in the MySQL database. The INNER JOIN word is to join the two tables based on their common key StudentID, and then WHERE is used to filter and only grab the specified data in which the CourseID matches the inputted CourseID.

In addition to SELECT statements, we will also be using UPDATE statements to update grade related data in the database. Here is an example of an UPDATE statement that we will be using to update a student's grade in the TestCourseTable:

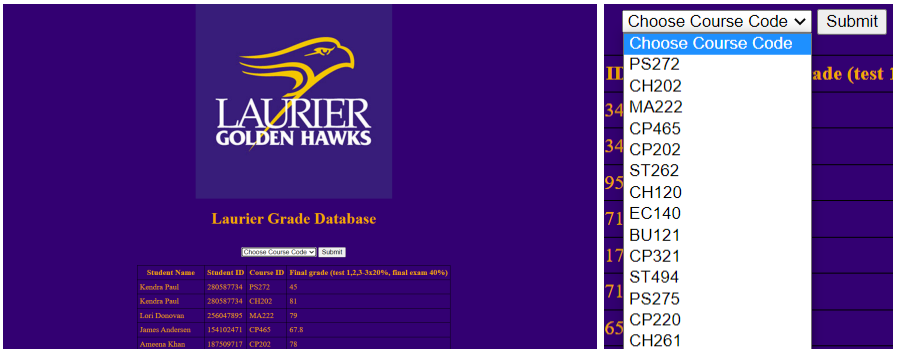
```php
$sql = $conn->prepare("UPDATE CourseTable
SET Test1 = COALESCE(NULLIF(?, ''), Test1),
    Test2 = COALESCE(NULLIF(?, ''), Test2),
    Test3 = COALESCE(NULLIF(?, ''), Test3),
    Exam = COALESCE(NULLIF(?, ''), Exam)
WHERE StudentID = ? AND CourseID = ?");
$sql->bind_param("ssssss", $test1, $test2, $test3, $exam, $studentID, $course);
$sql->execute();

$result = $sql->get_result();
```

This statement updates the Test1, Test2, Test3, and Exam columns in the TestCourseTable for the specified StudentID and CourseID values. The new values for these columns are specified using PHP variables ($test1, $test2, $test3, and $exam), which are passed into the statement using a prepared approach. We are also using the COALESCE and NULLIF functions to handle cases where the new value for a column is an empty string, so that the existing value for that column is preserved.

User Interface

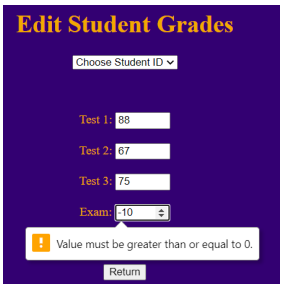
Use and interaction of forms and input fields with web users and the web server
WelcomeForm.php:
This file serves as the main page for the Laurier Grade Database system. It displays the school logo and a dropdown menu for selecting a course code. The user can select a course code and submit the form to view the student grades for the selected course. The PHP code in this file retrieves unique course IDs from the CourseTable and populates the dropdown menu.

This page also contains the full Appendix B output table defined in the requirements. It shows all of the student's final calculated grades.



InputGrades.php:
The WelcomeForm.php file includes an HTML form where the user can select a course code from a drop-down menu. Once the user selects a course and submits the form, they are taken to InputGrades.php. This file retrieves and presents all of the student test grades for the selected course in a table format. The table includes the student's name, student ID, course ID, and their grades for Test 1, Test 2, Test 3, and the final exam. Underneath this table, there is a HTML form where the user can select a student ID from a drop-down menu and input updated grade values for the selected student. The grades in the database will be updated with the newly entered values upon submission of the form.

UpdateGrades.php:
When the user submits the form in InputGrades.php, they are directed to UpdateGrades.php, which processes the form data and updates the student's grades in the CourseTable of the database. After the grades have been updated in the database, the user is then redirected back to InputGrades.php, where they can view the updated grades in the table.



The design uses a consistent colour scheme and style across all pages. Dropdown menus are used to select course codes and student IDs, minimising the chance of user input errors. The system automatically calculates the final grade based on the individual test and exam scores. Error handling and connection management are employed to ensure a smooth user experience. The table in InputGrades.php displays the student grades in a clear and organised manner, making it easy for users to view and understand the information.

Output Format

The output of the Laurier Grade Database consists of a table displaying the final grades of all current students. The table is generated using PHP and HTML, and it is presented in a clear and structured format.

Table Structure:

The table is structured with columns for Student Name, Student ID, Course ID, and Final Grade. The table headers are styled with a border and padding for better readability. The table rows are generated dynamically using PHP, based on the result of the SQL query.

Output Explanation:
Student Name: The full name of the student.
Student ID: The unique identifier for each student.
Course ID: The unique identifier for the course the student is enrolled in.
Final Grade: The calculated final grade for the student in the selected course. It is based on the student's scores in Test 1, Test 2, Test 3, and the final exam.
Calculation:
The final grade for each student is calculated using the formula provided in the requirements document:

Final Grade = (Test1 * 0.2) + (Test2 * 0.2) + (Test3 * 0.2) + (Exam * 0.4)

This formula calculates the weighted average of the student's test scores and the final exam score. The tests have a combined weight of 60% (20% for each test), and the final exam has a weight of 40%. The calculated final grade is rounded to one decimal place.

The SQL query in the PHP code retrieves the relevant student and grade data from the database, performs the above calculation, and outputs the result as a new column in the table named "Final_Grade." The table is generated by iterating through the results of the SQL query and outputting the corresponding values for each student.

Implementation Details

The web server and database for this project are hosted locally on a windows device using an Apache web server. The MySQL database used to house the Name and Course tables was set up using SQL statements within PHP using the mySQLi extension. To do this, Apache was installed onto a windows machine and configured to host the web server. Once it was set up, mySQL was also installed onto this windows machine so that the database could be created. Then using PHP code, SQL statements, and the mySQLi extension for PHP, the database and its two tables were created by reading the provided csv files and inserting the data into the database.

The files containing the PHP and HTML code for the interface and functionality of the project are located within a folder in the Apache directory. This allows them to be accessed and run from a browser. Because of this, when the web user interacts with the web page and manipulates entries in the database, the locally hosted MySQL database will be affected by these changes. Each HTML form links to a PHP script that is run server-side to handle the user inputs and retrieve data from the database.

Security Considerations

Security concerns were an important factor to consider when designing the project

since the database can be altered by web user actions. To protect our project from security vulnerabilities, we implemented a variety of safeguards including the use of SQL prepared statements, user input validation, and the use of drop-down selection menus rather than text box inputs.

When a user submits an HTML form there are multiple user-inputted values which are used in SQL queries. Thus, it is important to ensure that users input the expected data types. To ensure this is the case, all user input fields are type restricted. For example, when a user is prompted to input an updated grade value, the input is restricted to only number values. Users will not be able to submit strings or other data types in these fields. Our group took this one step further by adding further restrictions on the actual data values. When a user updates a grade value, the inputted value must be a number and be within the range of 0 to 100.

Another method we used to try to minimise security concerns was through the use of drop-down selections rather than text field inputs. When a user is prompted to input a course code, a list of possible selections is populated within a drop-down menu, rather than a blank text field where the user would enter a course code as a string. By doing so, the web user is limited to only selecting a course code that exists within the database. This ensures that a web user does not intentionally or unintentionally select a course which does not exist. It also minimises risk of other malicious forms of data being submitted in the HTML forms.

All SQL queries that can be triggered by user actions are implemented with the use of SQL prepared statements. The risk of SQL injection attacks is minimised through the use of prepared statements. The prepared statements tell MySQL what types of data to expect from the parameters bound to the query. Malicious attempts to use incorrect data in queries will be blocked.

Testing and Validation

The project was tested with an extensive variety of different possible inputs that a web user could give. Many different types of inputs and numbers were entered into the grade editing fields, and because of the use of dropdown lists and the text fields restricting inputs to only numbers between 0 and 100, the project was able to handle all possible inputs from the user. In addition, testing was done on the mySQL database where changes were made to the data from both the browser as well as the terminal. This showed that changes made on either end would reflect in the other. Since our final grade table is not stored in the database, but is generated every time the page is accessed, we were also able to see that the final grade calculation was updated dynamically when the user edits test grades.

References

Gao, L. (2023). CP-476-A Lecture Slideshows [Powerpoint slides]. https://mylearningspace.wlu.ca/d2l/home/468503.

HTML Forms. W3Schools. Retrieved March 15, 2023, from https://www.w3schools.com/html/html_forms.asp

PHP: Hypertext Preprocessor. PHP.net. Retrieved March 31, 2023, from https://www.php.net/manual/en

PHP MySQL Prepared Statements. W3Schools. (2023, April 2). Retrieved April 2, 2023, from https://www.w3schools.com/php/php_mysql_prepared_statements.asp#:~:text=A%20prepared%20statement%20is%20a,(labeled%20%22%3F%22).