CP476 Software Design Document

Group 2

Jessie Newman, Phoebe Schulman, Arzekeil-Abel Garcia De Leon, Yasmine Ali

Table of Contents

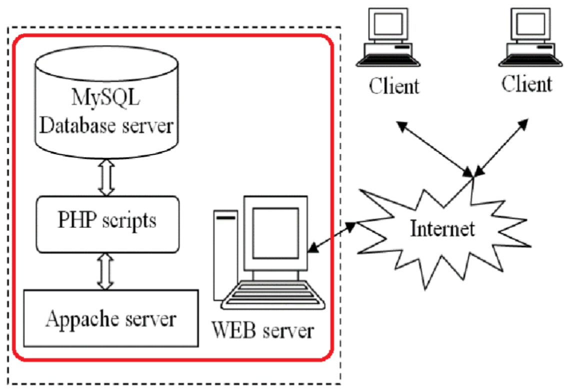
Introduction

The Student Grades Application has been designed and developed to allow for educators and other faculty to view metrics on student success and update grades, as well as calculate the final grades of the students. This document will cover the overall design of this software system, how the backend was developed, and the frontend experience for the user.

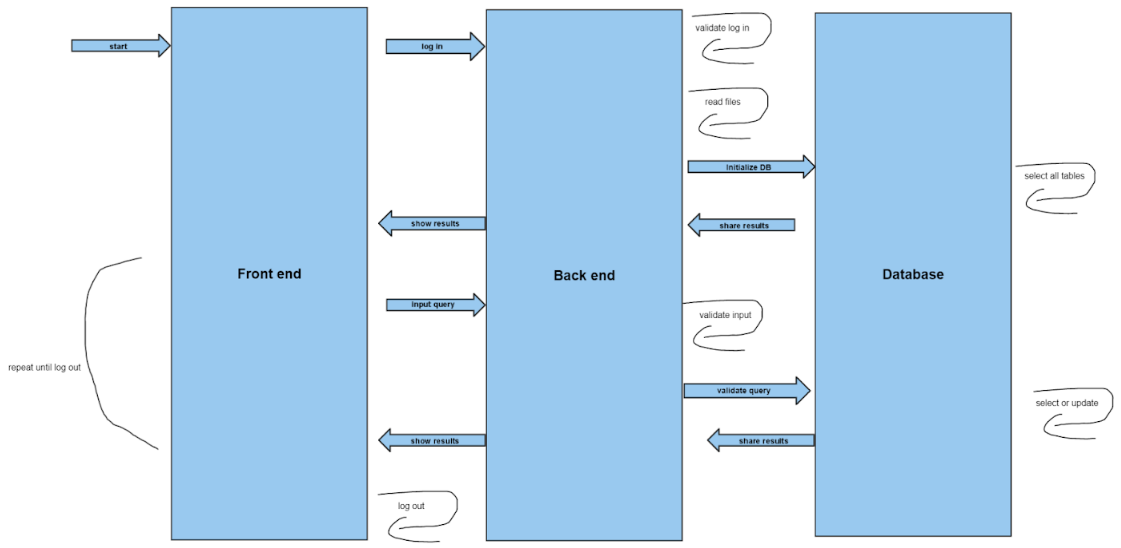
The main purpose of the project was to develop a web server where users can interact with a database server. This project was programmed with SQL, HTML, and PHP, and utilized Apache as the web server and MySQL as the database server to facilitate the program's operation.

Design

The design of this project follows the general architecture used for web applications worldwide. The PHP scripts interface with the MySQL database in order to store, retrieve, and update information that is needed for the application. The Apache server is what serves the application to users when the webpage is requested by various clients across the internet.

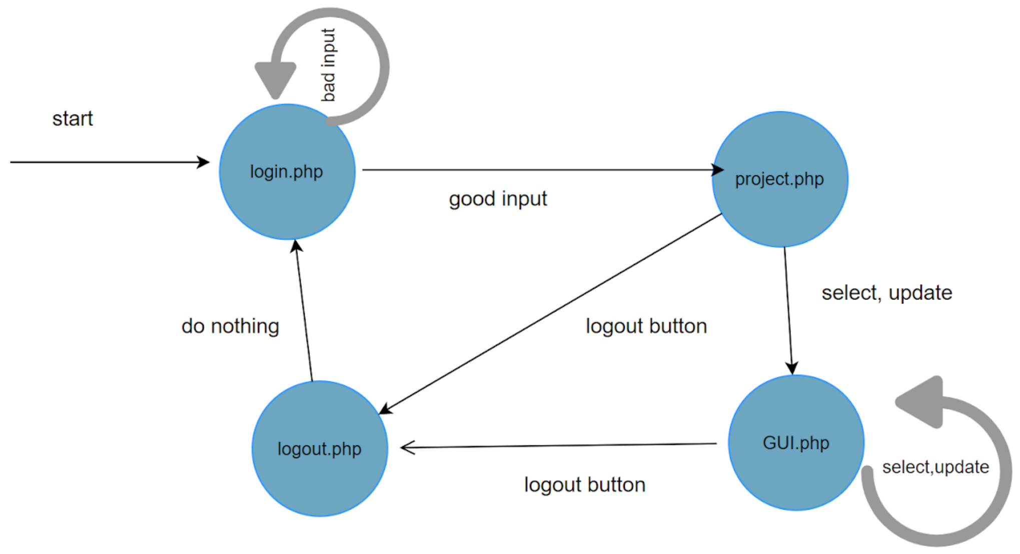


With this in mind, the system can be broken up into 3 components, being the database, the frontend, and the backend (PHP files). The frontend consists of the HTML page the user is presented with, which will send requests back to the PHP files that will send responses, as well as interact with the MySQL database.



Backend

The backend consists of multiple PHP files to facilitate the program.



Within login.php is the login page, where the user must input a valid username and password combination. Until a set of valid login information is provided, the application will remain on this page. This is a security measure so that only authorized users can access our database information.

The main file, project.php, initializes the database and tables, using the NameFile.txt and CourseFile.txt as input. This initialization of the database and tables to a fresh state is a development feature to allow for easier and more consistent testing. It would not be present within the final product, as it is inefficient and impractical to reset the database upon running the application each time. From this file, it also displays all tables and waits for the first input.

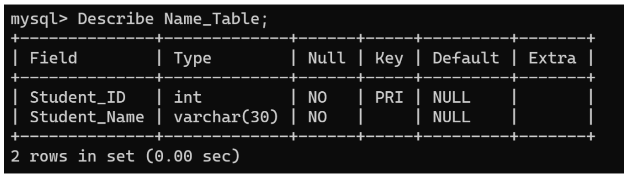
The next file processes the user's input and displays the results, before awaiting the next input. The user's input must be an SQL "select" or "update" statement. The type of SQL statement plus validity of the statement is verified by the program.

During either of the two previous views, if the logout button is clicked then the logout process will be handled by logout.php. After logging out, the user is returned to the login page.

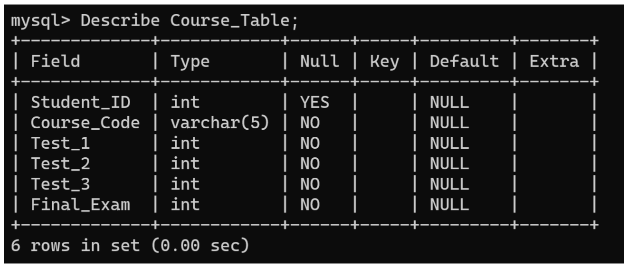
Database

The program creates and displays 3 database tables. The Name Table and Course Table are created from files, as mentioned previously. The Final Grade Table is based off of those two tables. When viewing their schemas, it can be seen that the primary key for all of them is a StudentID.

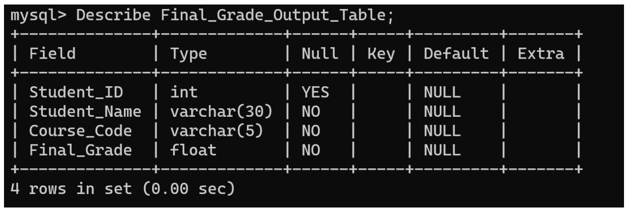
The schemas of the 3 tables:



```
mysql> Describe Name_Table;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Student_ID   | int         | NO   | PRI | NULL    |       |
| Student_Name | varchar(30) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```



```
mysql> Describe Course_Table;
+-------------+------------+------+-----+---------+-------+
| Field       | Type       | Null | Key | Default | Extra |
+-------------+------------+------+-----+---------+-------+
| Student_ID  | int        | YES  |     | NULL    |       |
| Course_Code | varchar(5) | NO   |     | NULL    |       |
| Test_1      | int        | NO   |     | NULL    |       |
| Test_2      | int        | NO   |     | NULL    |       |
| Test_3      | int        | NO   |     | NULL    |       |
| Final_Exam  | int        | NO   |     | NULL    |       |
+-------------+------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```



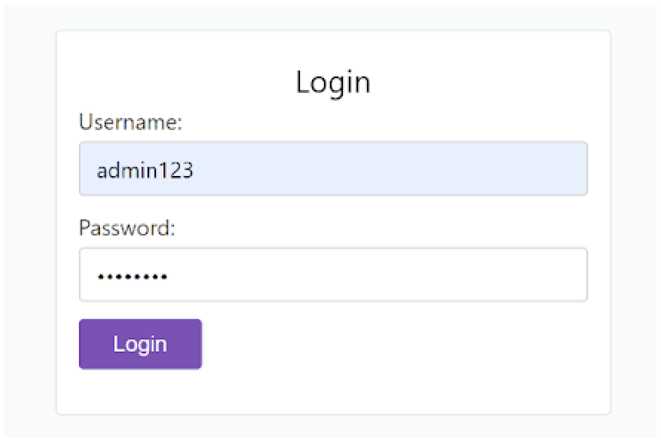
```
mysql> Describe Final_Grade_Output_Table;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Student_ID   | int         | YES  |     | NULL    |       |
| Student_Name | varchar(30) | NO   |     | NULL    |       |
| Course_Code  | varchar(5)  | NO   |     | NULL    |       |
| Final_Grade  | float       | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

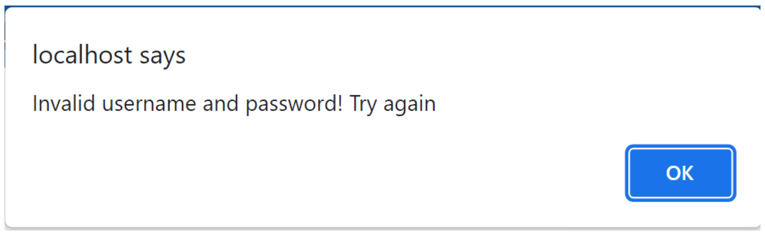
The Name_Table pairs student IDs with the name of the student, the Course_Table stores a student’s individual grades within a course, and the Final_Grade_Output_Table stores the calculated final grade for a student within a course.

The resulting Final_Grade_Output_Table is built by selecting everything from the Name_Table and Course_Table and joining them on the primary key (Student_ID). Then the final grade values are calculated using a formula similar to finalGrade = test1 * 0.20 + test2 * 0.20 + test3 * 0.20 + finalExam * 0.40.

Frontend

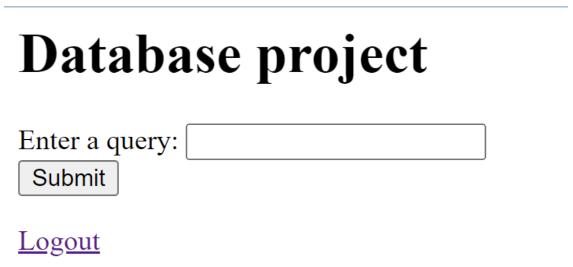
The front end allows for user interactions. The program starts with a login page, to securely access the database information. Entering the wrong login credentials would throw an error and return the user to the login page (PHP Tutorial, 2021).




localhost says

Invalid username and password! Try again

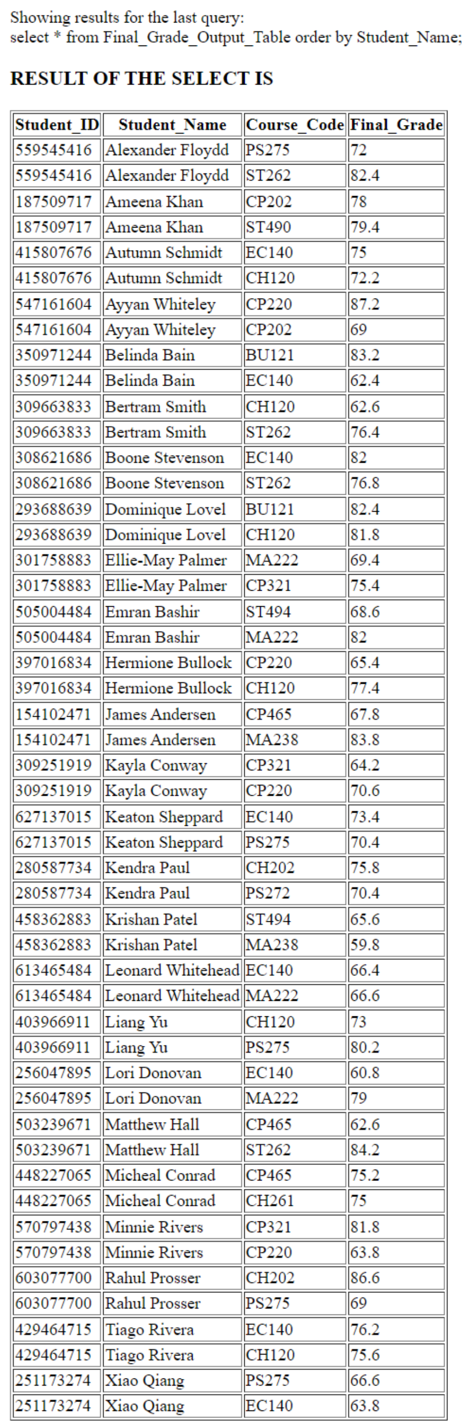
On the main page, users can input SQL queries using a Textbox for SELECT and UPDATE statements. A submit button takes them to the next page to display the results. Clicking the logout button will return them to the login page.


Database project

Enter a query:

Results

The resulting Final_Grade_Output_Table (when sorted by name) would appear such as:



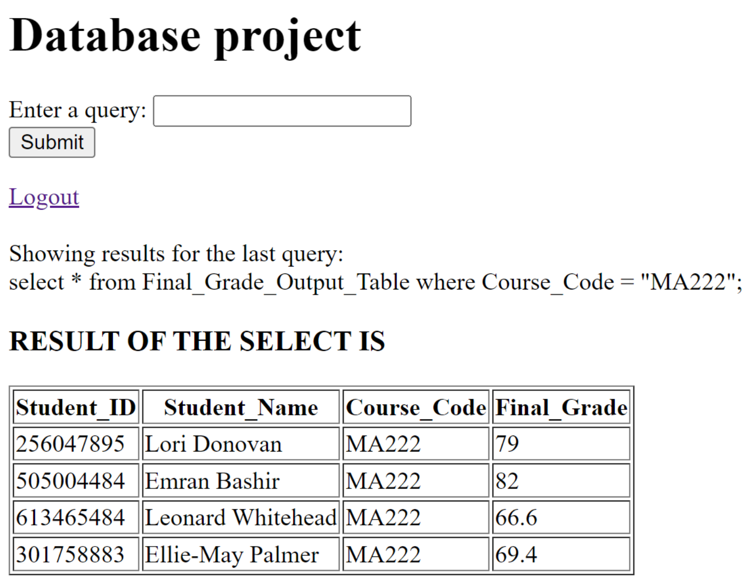
Showing results for the last query:
select * from Final_Grade_Output_Table order by Student_Name;

RESULT OF THE SELECT IS

Student_ID	Student_Name	Course_Code	Final_Grade
559545416	Alexander Floydd	PS275	72
559545416	Alexander Floydd	ST262	82.4
187509717	Ameena Khan	CP202	78
187509717	Ameena Khan	ST490	79.4
415807676	Autumn Schmidt	EC140	75
415807676	Autumn Schmidt	CH120	72.2
547161604	Ayyan Whiteley	CP220	87.2
547161604	Ayyan Whiteley	CP202	69
350971244	Belinda Bain	BU121	83.2
350971244	Belinda Bain	EC140	62.4
309663833	Bertram Smith	CH120	62.6
309663833	Bertram Smith	ST262	76.4
308621686	Boone Stevenson	EC140	82
308621686	Boone Stevenson	ST262	76.8
293688639	Dominique Lovel	BU121	82.4
293688639	Dominique Lovel	CH120	81.8
301758883	Ellie-May Palmer	MA222	69.4
301758883	Ellie-May Palmer	CP321	75.4
505004484	Emran Bashir	ST494	68.6
505004484	Emran Bashir	MA222	82
397016834	Hermione Bullock	CP220	65.4
397016834	Hermione Bullock	CH120	77.4
154102471	James Andersen	CP465	67.8
154102471	James Andersen	MA238	83.8
309251919	Kayla Conway	CP321	64.2
309251919	Kayla Conway	CP220	70.6
627137015	Keaton Sheppard	EC140	73.4
627137015	Keaton Sheppard	PS275	70.4
280587734	Kendra Paul	CH202	75.8
280587734	Kendra Paul	PS272	70.4
458362883	Krishan Patel	ST494	65.6
458362883	Krishan Patel	MA238	59.8
613465484	Leonard Whitehead	EC140	66.4
613465484	Leonard Whitehead	MA222	66.6
403966911	Liang Yu	CH120	73
403966911	Liang Yu	PS275	80.2
256047895	Lori Donovan	EC140	60.8
256047895	Lori Donovan	MA222	79
503239671	Matthew Hall	CP465	62.6
503239671	Matthew Hall	ST262	84.2
448227065	Micheal Conrad	CP465	75.2
448227065	Micheal Conrad	CH261	75
570797438	Minnie Rivers	CP321	81.8
570797438	Minnie Rivers	CP220	63.8
603077700	Rahul Prosser	CH202	86.6
603077700	Rahul Prosser	PS275	69
429464715	Tiago Rivera	EC140	76.2
429464715	Tiago Rivera	CH120	75.6
251173274	Xiao Qiang	PS275	66.6
251173274	Xiao Qiang	EC140	63.8

Here is an example of running the program.

First, the user could query the database to select all of the students in the course MA222.

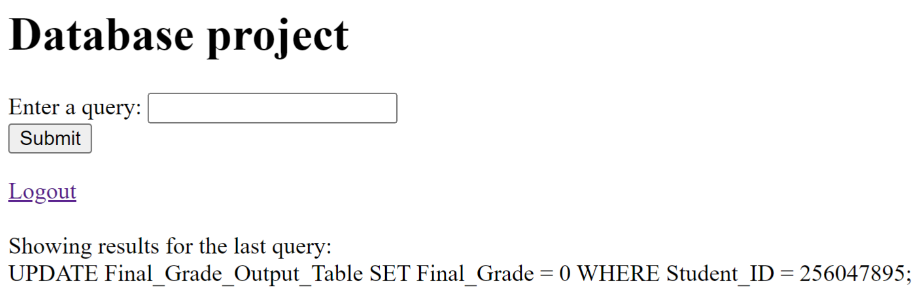


Showing results for the last query:
select * from Final_Grade_Output_Table where Course_Code = "MA222";

RESULT OF THE SELECT IS

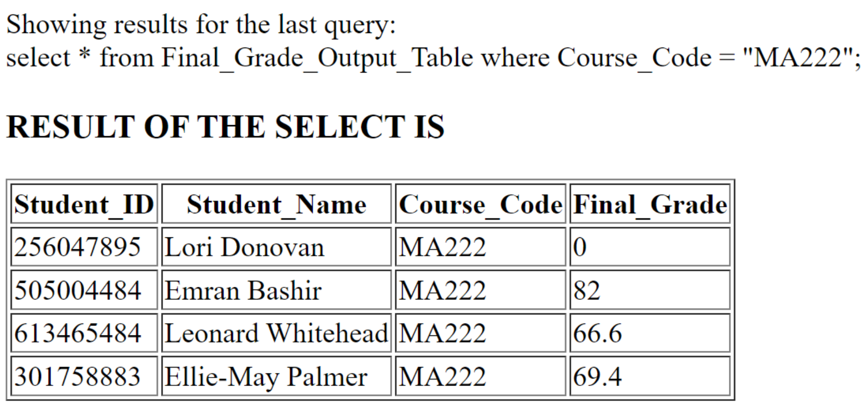
Student_ID	Student_Name	Course_Code	Final_Grade
256047895	Lori Donovan	MA222	79
505004484	Emran Bashir	MA222	82
613465484	Leonard Whitehead	MA222	66.6
301758883	Ellie-May Palmer	MA222	69.4

Then, the user could update a Final Grade of a student.



Showing results for the last query:
UPDATE Final_Grade_Output_Table SET Final_Grade = 0 WHERE Student_ID = 256047895;

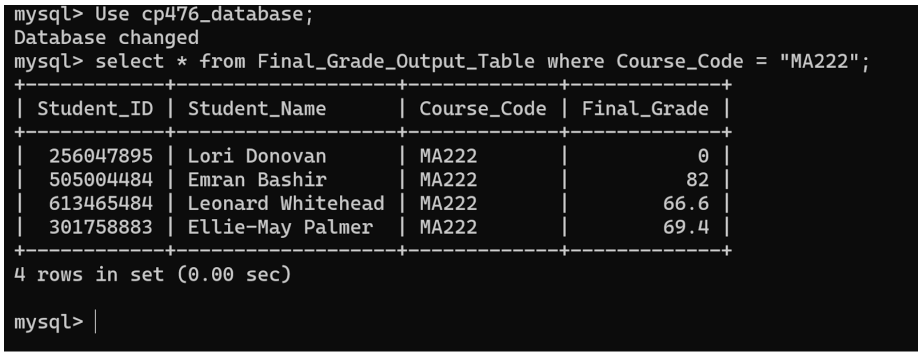
By selecting the students in the course MA222 again, we can see that this change (in the first row) is reflected on both the front end and the server side of the project. This shows that the interactions on the front end are directly connected to the database server.



Showing results for the last query:
select * from Final_Grade_Output_Table where Course_Code = "MA222";

RESULT OF THE SELECT IS

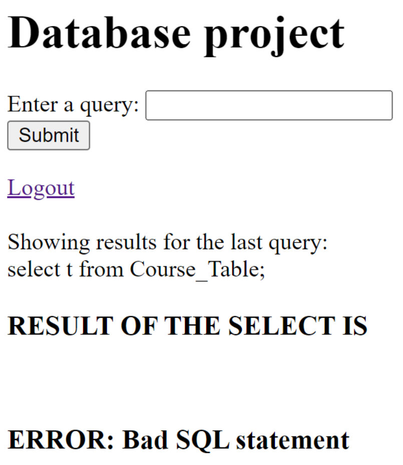
Student_ID	Student_Name	Course_Code	Final_Grade
256047895	Lori Donovan	MA222	0
505004484	Emran Bashir	MA222	82
613465484	Leonard Whitehead	MA222	66.6
301758883	Ellie-May Palmer	MA222	69.4



```
mysql> Use cp476_database;
Database changed
mysql> select * from Final_Grade_Output_Table where Course_Code = "MA222";
+------------+-------------------+-------------+-------------+
| Student_ID | Student_Name      | Course_Code | Final_Grade |
+------------+-------------------+-------------+-------------+
|  256047895 | Lori Donovan      | MA222       |           0 |
|  505004484 | Emran Bashir      | MA222       |          82 |
|  613465484 | Leonard Whitehead | MA222       |        66.6 |
|  301758883 | Ellie-May Palmer  | MA222       |        69.4 |
+------------+-------------------+-------------+-------------+
4 rows in set (0.00 sec)

mysql> 
```

Finally, errors are thrown when users input an invalid SQL statement. Another error occurs if users input a valid SQL statement that's not a select nor an update statement.



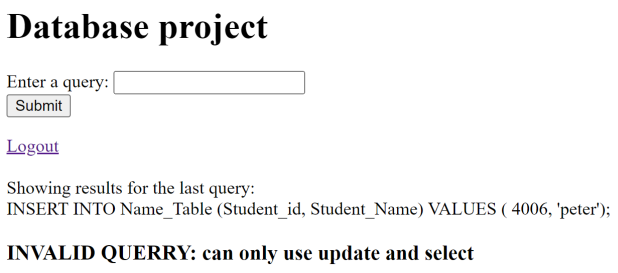
Database project

Enter a query:

Showing results for the last query:
select t from Course_Table;

RESULT OF THE SELECT IS

ERROR: Bad SQL statement



Database project

Enter a query:

Showing results for the last query:
INSERT INTO Name_Table (Student_id, Student_Name) VALUES (4006, 'peter');

INVALID QUERRY: can only use update and select

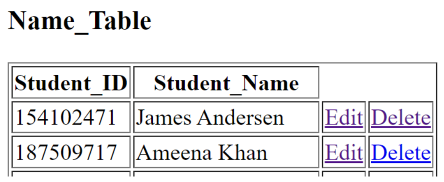
Alternative Solutions and Future Improvements

There is a selection of factors to this software that can be improved in future iterations. One aspect is the manner in which the user indicates what information they wish to view or update. Currently, the application requires the user enter in a 'SELECT' or 'UPDATE' SQL statement by typing the statement into a text box. Because many users may not be familiar with how to compose SQL statements and the naming of columns in the database, this should be changed instead to an interactive UI allowing users to click buttons and drop-down options to filter the output/select what to update instead.

Originally, the requirement to "execute at least two (SELECT and UPDATE) SQL statements" had been interpreted as executing any possible select statement as well as any possible update statement. Thus, the text box implementation gave this basic functionality for the original design. Compared to, for example, creating buttons for every possible permutation of a select statement. For example, users may select ID, Student name, ID and Student name, ID and Test1, etc from a single table. The number of possibilities grows when considering multiple tables together. The text box design also handles the case of adding any possible modifiers to a select statement (JOIN ON, name AS n, ORDER BY, etc) as it verifies if the SQL statement is still valid before running it. Once noted that one of each statement was enough, the design had moved to a button-based implementation. However, further implementations proved challenging.

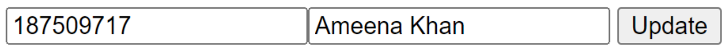
As well, the project was planned to incorporate the use of prepared statements for security purposes. Use of prepared statements is a simple procedure which involves preparing an SQL statement, binding variables to placeholder values, then executing the SQL code (Hanna & Lewis, 2021). These will prevent SQL injections, which are a cyber attack technique of adding malicious code to a database query, which can gain unauthorized access to view or modify the database (CrowdStrike, 2022). One of the project's alternative designs that unfortunately was not fully completed allowed for a mix of buttons and multiple text boxes during the update statements, to better incorporate the prepared statements.

For example, updating a name:

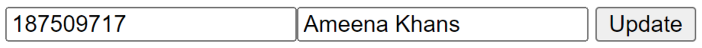


Name_Table

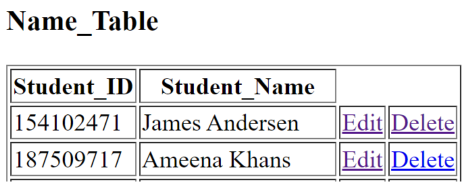
Student_ID	Student_Name		
154102471	James Andersen	Edit	Delete
187509717	Ameena Khan	Edit	Delete



187509717 | Ameena Khan | Update



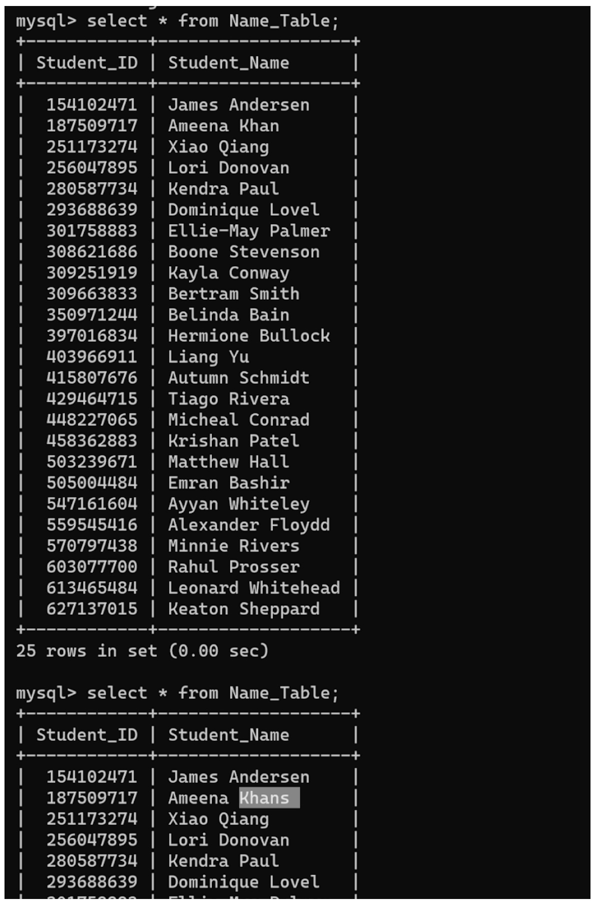
187509717 | Ameena Khans | Update



Name_Table

Student_ID	Student_Name		
154102471	James Andersen	Edit	Delete
187509717	Ameena Khans	Edit	Delete

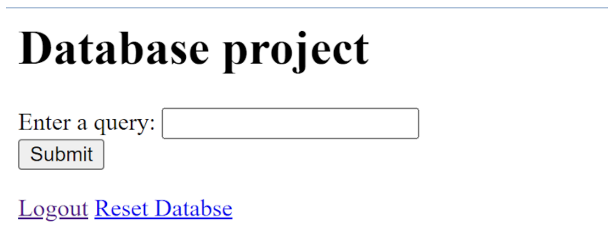
This change is also reflected on the server side.



```
mysql> select * from Name_Table;
+------------+------------------+
| Student_ID | Student_Name     |
+------------+------------------+
|  154102471 | James Andersen   |
|  187509717 | Ameena Khan      |
|  251173274 | Xiao Qiang       |
|  256047895 | Lori Donovan     |
|  280587734 | Kendra Paul      |
|  293688639 | Dominique Lovel  |
|  301758883 | Ellie-May Palmer |
|  308621686 | Boone Stevenson  |
|  309251919 | Kayla Conway     |
|  309663833 | Bertram Smith    |
|  350971244 | Belinda Bain     |
|  397016834 | Hermione Bullock |
|  403966911 | Liang Yu         |
|  415807676 | Autumn Schmidt   |
|  429464715 | Tiago Rivera     |
|  448227065 | Micheal Conrad   |
|  458362883 | Krishan Patel    |
|  503239671 | Matthew Hall     |
|  505004484 | Emran Bashir     |
|  547161604 | Ayyan Whiteley   |
|  559545416 | Alexander Floydd |
|  570797438 | Minnie Rivers    |
|  603077700 | Rahul Prosser    |
|  613465484 | Leonard Whitehead |
|  627137015 | Keaton Sheppard  |
+------------+------------------+
25 rows in set (0.00 sec)

mysql> select * from Name_Table;
+------------+------------------+
| Student_ID | Student_Name     |
+------------+------------------+
|  154102471 | James Andersen   |
|  187509717 | Ameena Khans     |
|  251173274 | Xiao Qiang       |
|  256047895 | Lori Donovan     |
|  280587734 | Kendra Paul      |
|  293688639 | Dominique Lovel  |
```

This new design also allows for the functionally of a “Reset Database” button, to allow for resetting the database on click, rather than automatically.



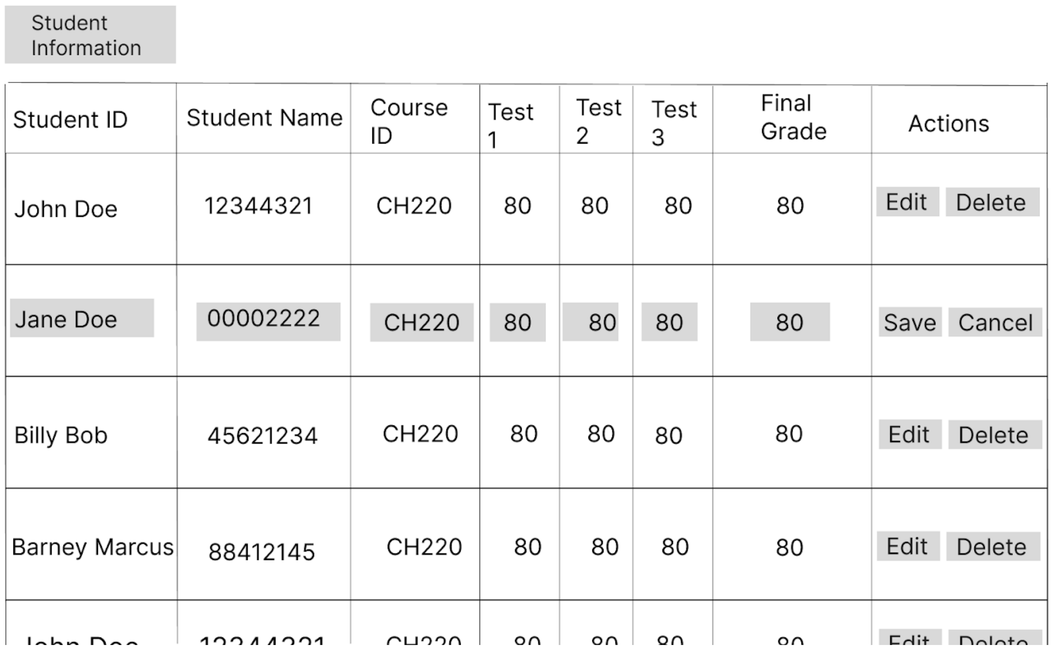
Database project

Enter a query:

Submit

Logout Reset Databse

Furthermore, aesthetic upgrades could have been made to the front end such as:



Student Information

Student ID	Student Name	Course ID	Test 1	Test 2	Test 3	Final Grade	Actions
John Doe	12344321	CH220	80	80	80	80	Edit Delete
Jane Doe	00002222	CH220	80	80	80	80	Save Cancel
Billy Bob	45621234	CH220	80	80	80	80	Edit Delete
Barney Marcus	88412145	CH220	80	80	80	80	Edit Delete
John Doe	12344321	CH220	80	80	80	80	Edit Delete

Aside from that, another area of improvement would be in further verification features being added. Such as testing if updating values fits the schema (for example a Student_ID should remain a 9 digit integer), or if updating a test mark falls in a valid range (from 0 to 100).

In addition, we planned to be able to maintain the database information across all tables. For example, if a grade for Test1 is updated in the Course Table, then this update should be reflected in the Final Grade Table, after recalculating the result for the Final Grade. It would also need to be ensured that changing the Test1 grade will still keep the Final Grade in a valid range (from 0 to 100).

Other changes include a different file structure to better organize and separate the front and back end work. As well as more features overall (insert, delete, etc).

Overall, countless other features could be added or modified in future iterations or updates to this project. As the possibilities for a project such as this are unlimited.

Conclusion

In conclusion, the Student Grades Application allows for web users to be able to interact indirectly with a database server. This project included connecting the database, front end, and back end by using SQL, HTML, and PHP.

Many aesthetic and functional improvements can be made, in particular relating to the use of prepared statements, as originally planned for.

Sources

CrowdStrike. (2022, November 9). *What is a SQL Injection Attack? | CrowdStrike*. crowdstrike.com. https://www.crowdstrike.com/cybersecurity-101/sql-injection/#:~:text=SQL%20injection%20(SQLi)%20is%20a,application%20security%20risk%20in%202021

Hanna, K. T., & Lewis, S. (2021, June 28). *SQL injection*. Software Quality. https://www.techtarget.com/searchsoftwarequality/definition/SQL-injection#:~:text=A%20SQL%20injection%20is%20a,or%20execute%20malicious%20SQL%20statements

Kaplarevic, V. (2021, August 12). *How to Drop a Table in MySQL*. Knowledge Base by phoenixNAP. https://phoenixnap.com/kb/mysql-drop-table

K. (n.d.). *How to get input field value using PHP*. Edureka Community. https://www.edureka.co/community/94514/how-to-get-input-field-value-using-php#:~:text=Use%20PHP's%20%24_POST%20or%20%24_GET,name%20of%20the%20HTML%20tag.&text=To%20show%20the%20value%3A,%24_GET%5B'subject'%5D%3B%20%3F%3E

L. (2019, October 10). *List (Show) Tables in a MySQL Database*. Linuxize. https://linuxize.com/post/show-tables-in-mysql-database/

MySQL DROP DATABASE - How to Delete a Database in MySQL Server. (2021, July 26). MySQL Tutorial. https://www.mysqltutorial.org/mysql-drop-database/

Online Causal Loop Diagram Tool. (n.d.). https://online.visual-paradigm.com/diagrams/features/causal-loop-diagram-tool/

PHP: explode - Manual. (n.d.). https://www.php.net/manual/en/function.explode.php

PHP mysqli fetch_all() Function. (n.d.). https://www.w3schools.com/php/func_mysqli_fetch_all.asp

PHP mysqli fetch_array() Function. (n.d.). https://www.w3schools.com/php/func_mysqli_fetch_array.asp

PHP mysqli query() Function. (n.d.). https://www.w3schools.com/php/func_mysqli_query.asp

PHP MySQL Select Data. (n.d.). https://www.w3schools.com/php/php_mysql_select.asp

PHP mysqli fetch_fields() Function. (n.d.). https://www.w3schools.com/php/func_mysqli_fetch_fields.asp

PHP MySQLi Functions. (n.d.). https://www.w3schools.com/php/php_ref_mysqli.asp

PHP: strtolower - Manual. (n.d.). https://www.php.net/manual/en/function.strtolower.php

PHP Tutorial. (2021, September 29). *PHP Login*. https://www.phptutorial.net/php-tutorial/php-login/

Show values from a MySQL database table inside a HTML table on a webpage. (n.d.). Stack Overflow. https://stackoverflow.com/questions/17902483/show-values-from-a-mysql-database-table-inside-a-html-table-on-a-webpage

SQL INSERT INTO Statement. (n.d.). https://www.w3schools.com/sql/sql_insert.asp

SQL UPDATE Statement. (n.d.). https://www.w3schools.com/sql/sql_update.asp

What do strict types do in PHP? (n.d.). Stack Overflow. https://stackoverflow.com/questions/48723637/what-do-strict-types-do-in-php