Software Design Documentation

Design and Implementation of a
Student Information Organizational System (SIOS)

Professor Shaun Gao
CP476 Internet Computing
Group Assignment
Thursday, March 25th, 2024

Group 8
Lubna Al Rifaie [200821590]
Gurman Sekhon [181004500]
Adil Nawaz [190956140]
Syed Atif [200679450]

Table of Contents

1 Introduction

1.1 Document Overview

The Student Grades Application: a robust platform engineered to enhance the educational experience by providing educators and faculty members with an advanced suite of tools for tracking student achievement, updating grades, and calculating final scores. This guide will take you through our software system's sophisticated design, the backend's development journey, and the intuitive Frontend experience we've crafted for users.

This project aims to introduce a dynamic web server that facilitates direct interaction with a powerful database server. Developed using SQL, HTML, and PHP, and supported by Apache and MySQL for web and database server functionalities, respectively, our application stands as a testament to modern programming excellence, designed to streamline educational processes and elevate user engagement.

1.2 Purpose of Software

The purpose of this software is to provide a web-based interface that allows the user to access the database, and perform the following queries:

1. Search for course-related data for specific students
2. Delete course-related data for specific students
3. Update course-related data for specific students

2 Software Architecture Overview

This project embodies a design paradigm that ensures both scalability and reliability. The system seamlessly interacts with the MySQL database by integrating PHP scripts, as well as facilitating efficient data storage, retrieval, and updates essential for the application's functionality. Powered by the Apache server, the application is delivered to users upon request, enabling a broad accessibility spectrum across the internet.

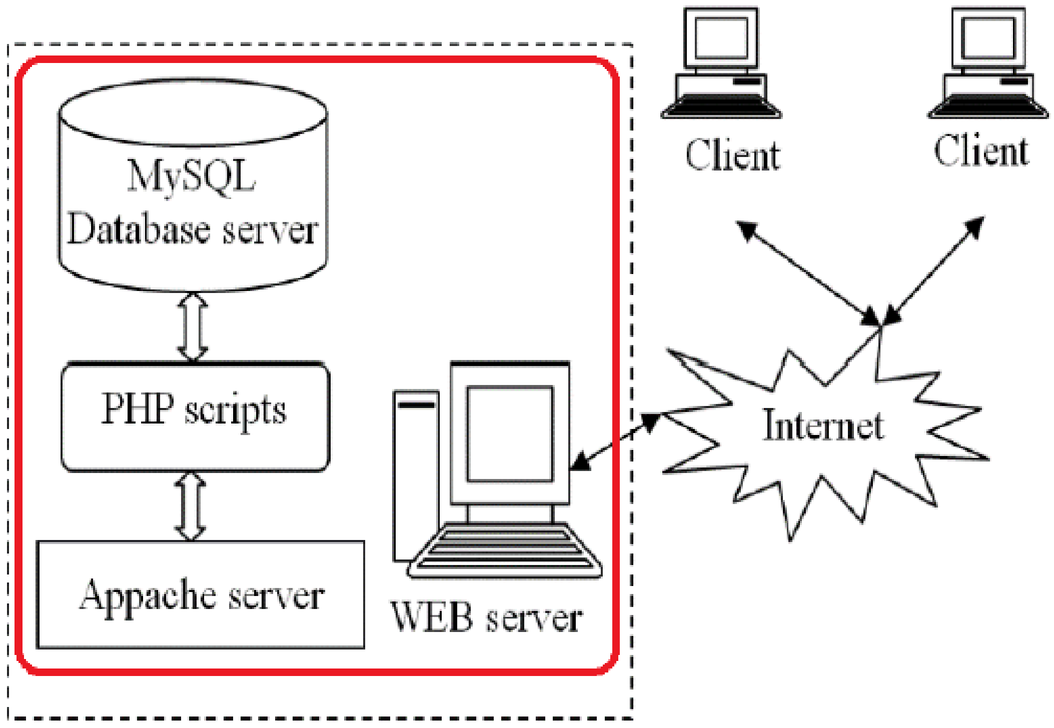
2.1 User Interface

The user interface for the web application can be broken down into three main components: the authentication component, the querying component, and finally the query retrieval component.

The authentication component is what the user first encounters. The user has the option to either log in or sign up and create a new account. If the user already has valid credentials, they may continue to the querying component. Otherwise, they must first create a new set of valid credentials, and then log in.

Once logged in, the user will encounter the querying component. This component contains a series of web pages with different branches (dependent on the user's initial selections). The first webpage the user will encounter will allow them to select which querying function they would like to perform, as well as which table they would like to perform the function upon. The user may select either the "Names", "Test Grades", or "Final Grades" table, and may select either the "Search", "Update" or "Delete" functions. If the user selects the "Search" or "Delete" functions, then they will be taken to a webpage to enter the primary key of the table they selected, separated by commas (ex. 1, A). On the other hand, if the user selects the "Update" function, they will be taken to a webpage where they can enter the primary key of the table they selected as well as the new values they would like to populate the corresponding row.

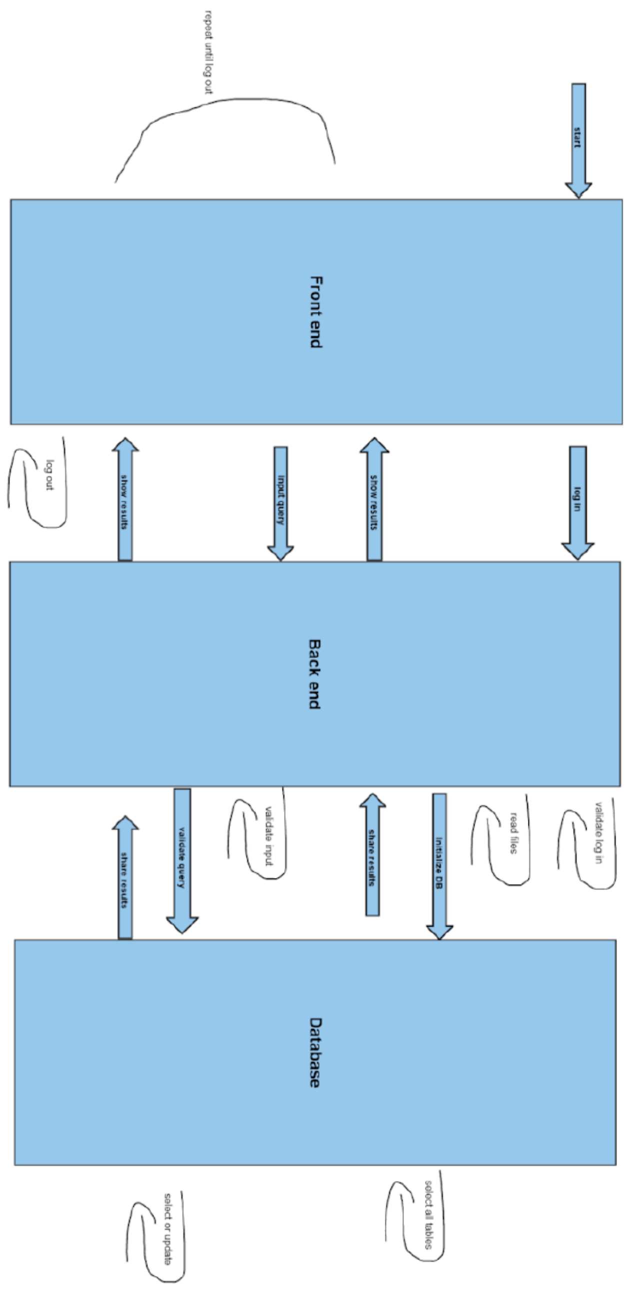
Finally, once the user has provided all the information for the query, they will continue to the query retrieval component, which consists of a single webpage that will display the results of the user's query.



Above: An overview of the application architecture.

2.2 Architectural Design

Bearing this in mind, the architecture of the system is elegantly segmented into three core components: the database, the Frontend, and the Backend. The Frontend is embodied by the webpages displayed to the user, serving as the interactive gateway through which requests are collected. Once a request is collected, the Backend uses the information, connects to the database, and returns the query results. These results are then communicated back to the Frontend and displayed to the user. This meticulously designed interaction framework ensures a seamless flow of information, providing a cohesive and intuitive user experience.



Above: A more detailed view of the application architecture.

3 Software design description

3.1 Frontend

3.1.1 Component Interfaces

The Frontend of the application is expertly designed to facilitate user interactions, beginning with a secure login page. This initial gateway ensures that access to the database information is guarded, requiring correct login credentials for entry. Should a user enter incorrect information, the system gracefully handles this by displaying an error message and redirecting the user back to the login page, maintaining security without compromising user experience (PHP Tutorial, 2021).

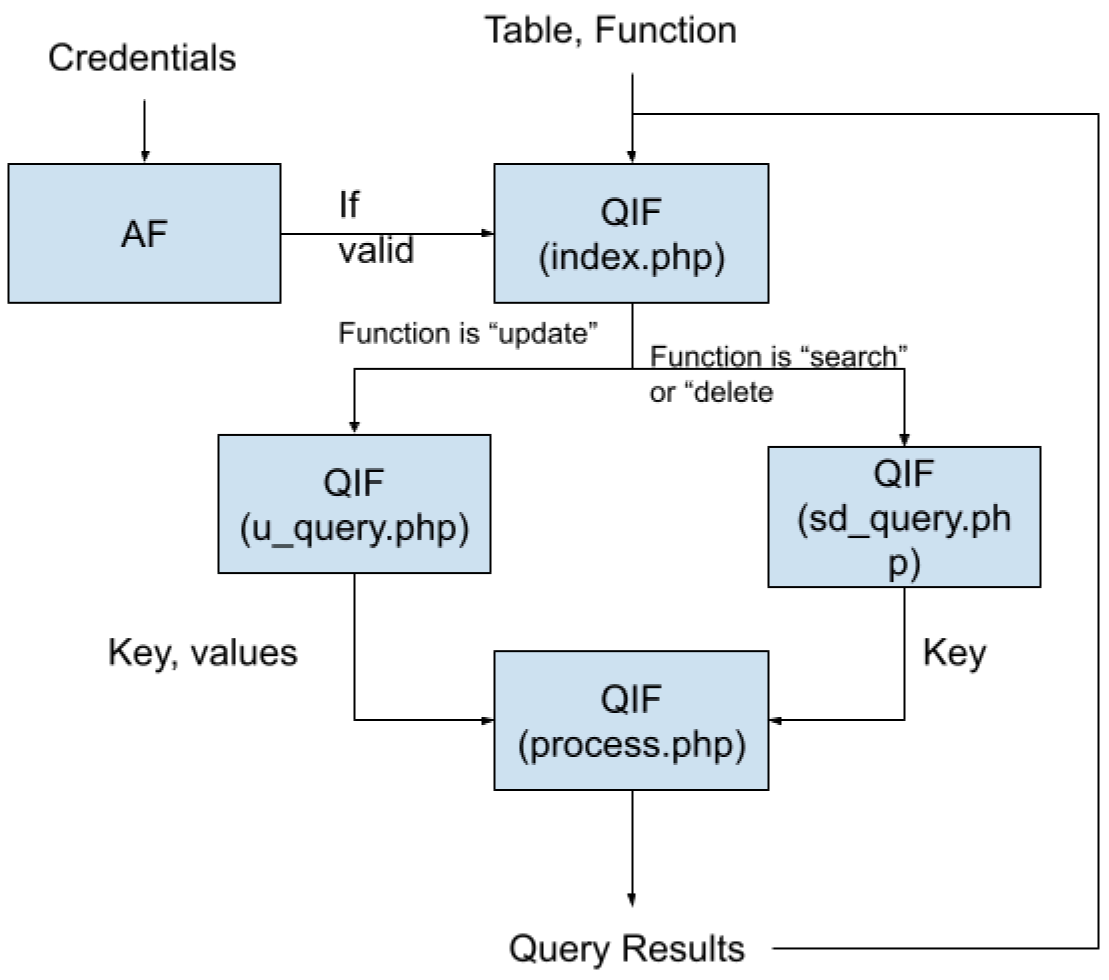
In the frontend, there are 2 major components: the authentication frontend (AF) component, and the querying interface frontend (QIF) component. Each of these is made up of several sub-components, which will be explained in the next section.

The AF component provides a visual interface through which the user can log in. If the user does not have any valid credentials, they also have the option to create a new account with the system. This ensures that only authorized individuals may access the web application, through which the database may be changed.

Once logged in, the authentication component connects the user to the QIF. The frontend portion of the QIF consists of 4 PHP files: index.php, sd_query.php, u_query.php, and process.php.

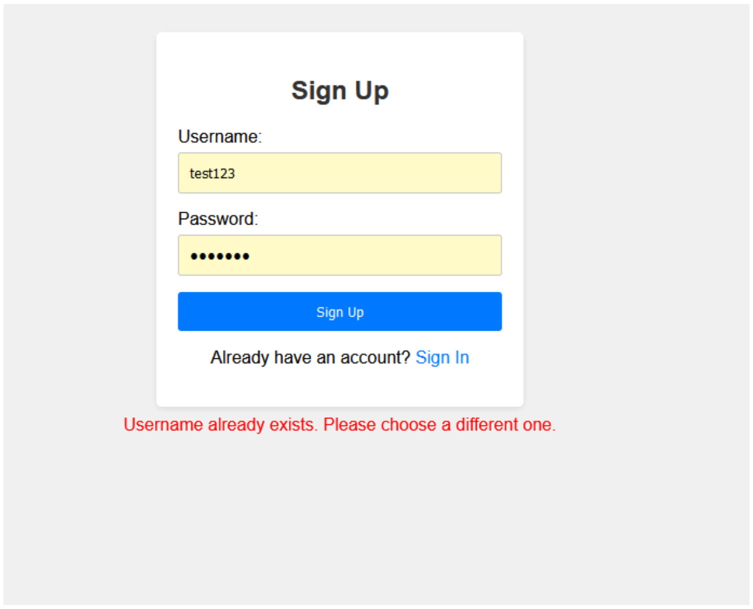
The first file, index.php, is the initial webpage file that the user is shown once logged in. On this page, they must select which table they would like to access, and which of 3 functions (search, update or delete) they would like to perform. Once they have made their selections, the system then (through an intermediary PHP file that will be discussed later) conditionally takes the user to either sd_query.php or u_query.php. The condition is dependent on which function the user selected: if “search” or “delete”, the user is taken to sd_query.php, otherwise, the user is taken to u_query.php.

In sd_query.php, the user may input the key they would like to use for either the “search” or “delete” query. In u_query.php, the user must enter the key-value pair they would like to use to update their selected table. Once all this information has been captured, the user is then taken to process.php, which displays the results of the user’s query on the webpage, as well as provides a button to give the user the option to start another query.



Above: An overview of the frontend components.

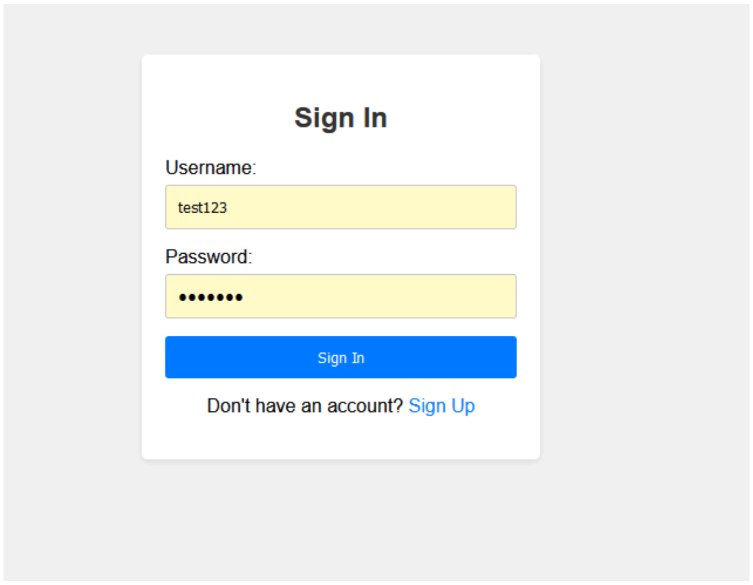
Left: The login page on invalid (above) and valid (below) credentials.



test123

•••••••

Username already exists. Please choose a different one.



test123

•••••••

3.1.2 Component Design Description

The signin.php file consists of HTML and queries. It creates a user_login table and runs a query to verify if the user is in the table and if the password matches. Within the signin.php file, there is an HTML code that redirects users to a signup.php file if the user wishes to create a new account. Signup.php runs queries to check whether a username is already in the table. If the username is not already taken, signup is successful and users will be redirected to the sign-in page to re-enter their information and access the program and data.

Now, on to the QIF component. First, in index.php we start a new session. This is done so that two crucial variables that are needed across all files (table and function) are globally accessible. Once that is done, there is an HTML section that creates 2 dropdowns, the first of which allows the user to select the table they would like to query, and the second allows them to select which function they would like to use for the query. At the bottom, there is a button for the form, which posts the user's selections to reroute.php, which conditionally redirects the user to either sd_query.php or u_query.php, as described above.

*As an aside, it is important to note that although the user only makes one selection for the table, for both ease for the user and to maintain database integrity, the query is run on ALL tables. The table selection is simply used to inform the user of which keys/values to use, and how to enter them.

In sd_query.php, first, we start the session within the file, and then the key format to display to the user is determined, dependent on the selected table. The user is then asked to input the key they wish to use. In u_query.php the setup is the same, though the user is also asked for the values they wish to update the table with. Both files also contain a button at the bottom that says "Submit", which posts the inputted information to process.php.

In process.php, once again the session is started within the file to gain access to the table and function variables. We then construct 2 arrays: an array containing the key, and an array containing the values. Next, depending on which function the user selected, the appropriate function is called, and the output is stored. Finally, the file contains an HTML section, in which the results are displayed, and the user is presented with a button that will allow them to start another query.

3.1.3 Workflows and Algorithms

Upon reaching the main page, users are greeted with an intuitive interface featuring a Textbox, where they can craft SQL queries specifically for SELECT and UPDATE operations. A strategically placed submit button effortlessly guides them to a subsequent page, dedicated to showcasing the results of their queries. For those wishing to conclude their session, a logout button is conveniently available, ensuring a smooth transition back to the login page, thus completing the user journey with elegance and efficiency.

3.2 Backend

3.2.1 Component interfaces

The backend of the web application consists of 3 major components: the authentication backend (AB) component, the querying interface backend (QIB) component, and the querying component. The 3 components do not interact with each other directly.

3.2.2 Component Design Description

The purpose of the AB component is to:

1. Validate credentials provided by the user against the authorized user database
2. Add credentials as necessary to the authorized user's database

The backend queries simply just verify the user's username, and if the password matches what's in the table. If a user wishes to create an account, we use a simple select from statement, and if returns false we use an insert into a statement to append to the table with the new username.

Overall, the AB component for the sign-in and sign-up pages plays a crucial role in ensuring that only authorized users can access the application's features and data. It manages both the verification of existing users and the addition of new users to the system.

The purpose of the QIB component is simply to redirect the user to the correct webpage based on the user's input. As discussed in 3.1, from index.php, the user will either be redirected to sd_query.php or u_query.php, depending on which function the user has selected. This redirection is handled by reroute.php. The second redirect that must be handled is from reset.php, which takes the user back to index.php to start another query.

In reroute.php, first, a new session is started within the file. Then the table and function values that are posted from index.php are saved as session variables, as they will be used across all files. Next, depending on which function the user selects, the user will either be redirected to sd_query.php or u_query.php.

If the user is on process.php and clicks the button at the bottom to start another query, the redirection to index.php is handled by reset.php. Within this file, a new session is started and the table and function session variables are unset. This ensures that there is no "leftover" information from the previous query. Once this is done, the user is then redirected to index.php, where they can start another query.

Finally, the purpose of the querying component is to take the information gathered in the querying interface frontend (QIF) component and supply the information to functions that will perform the query and return the results.

Search, update and delete functions were created for the database access queries from PHP. They all incorporate prepared SQL statements as it is recommended and it is the better approach to use in this case.

searchStudent function helps you find a specific student by their ID. It shows you the student's name and all the courses they're taking, along with their corresponding grades. The function first looks for a student matching with the provided ID, if a student is found, the name is printed. Then, it looks for all the courses that the student is taking and lists the courses as well as the grades the student got in them. If no student matches the ID, the function tells you that no student has been found!

deleteStudent function is used to remove all records of a student from the database. The function takes the ID of the student and ensures that the student's information is deleted from both the CourseTable and the NameTable. The function starts by attempting to delete the student's entry from the NameTable. If successful, it confirms that the student has been removed. If unsuccessful, the function indicates there was an 'error deleting the student.' Then, it proceeds to remove all the student's course information from the CourseTable and informs you whether this operation was successful, similar to the operation for the NameTable.

updateStudent function allows you to update a student's name and add a new course along with its grades to the data. The function requires the student's ID, the new name, the new course name, and the new course grades as input parameters. It begins by updating the student's name in the database with the provided new name. Subsequently, it adds a new course entry for the student, complete with the grades for that course. Throughout the process, the function provides feedback to keep you informed. It notifies you if the student's name was successfully updated with a message like 'Student name updated.' Similarly, it confirms the addition of the new course and grades with a message such as 'Course and grades added.' In case of any errors during the update or addition process, it alerts you with error messages like 'Error updating student name' or 'Error adding course and grades.'

3.2.3 Workflows and Algorithms

Login Mechanism: At the heart of our security protocol is the login.php file, which hosts the login interface. Users are prompted to enter a valid username and password combo to proceed. This page acts as the application's gateway, steadfastly ensuring that only authorized individuals gain access to the database information. It embodies our commitment to security, maintaining the login screen until successful authentication.

Initialization and Display: The core functionality is encapsulated in project.php, responsible for initializing the database and its tables. It ingeniously uses NameFile.txt and CourseFile.txt for input, setting up a pristine database environment ideal for streamlined and consistent testing phases. This auto-initialization feature, designed for development ease, is absent in the final product to avoid the impracticality and inefficiency of resetting the database with every application launch. Furthermore, project.php serves as the nexus for displaying all tables and priming the system for the user's initial input.

Input Processing and Result Display: After initialization, the application transitions to processing user inputs, accepting SQL "select" or "update" commands. This phase involves verifying the input's validity and the appropriateness of the SQL statement, showcasing the program's dynamic and interactive capabilities. After displaying the results, it seamlessly prepares for the next round of user input.

Secure Logout: The logout functionality is adeptly managed by logout.php. A single click on the logout button triggers a secure sign-off process, ultimately redirecting users back to the login page. This ensures a full-circle user experience, from secure entry through to a safe and orderly exit from the application.

4 Database

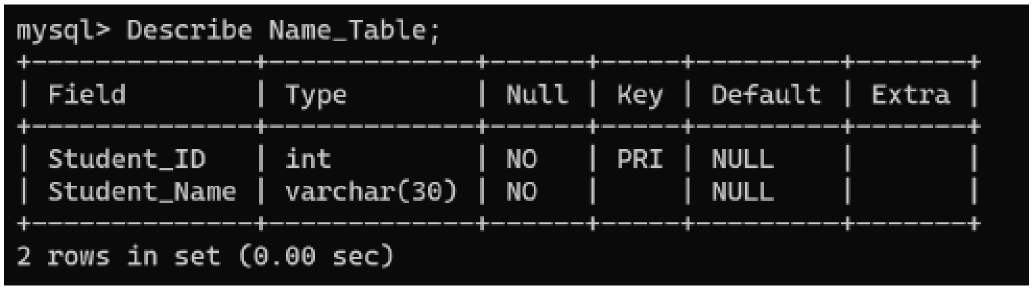
The application elegantly generates and presents three distinct database tables, each playing a pivotal role in its ecosystem. The Name Table and Course Table are meticulously constructed from the previously mentioned files, laying the foundational data structure. Building upon this, the Final Grade Table is derived, intertwining the insights from both the Name and Course Tables to produce a comprehensive assessment of student performance. A closer inspection of their schemas reveals a thoughtfully designed architecture where the StudentID emerges as the cornerstone, serving as the primary key across all tables. This unified identifier system not only streamlines data management but also enhances the integrity and accessibility of the database.

4.1 Database Design Description

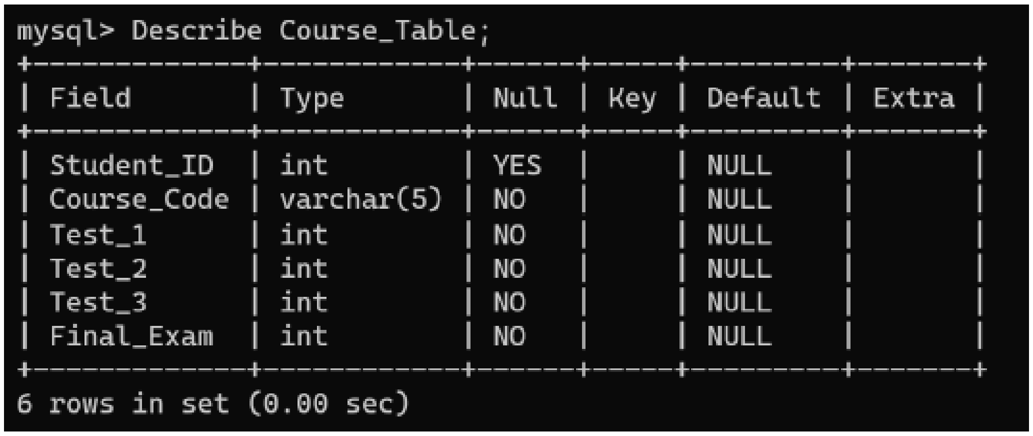
The architecture of the database is thoughtfully designed with three specialized tables to streamline student data management. The Name_Table establishes a critical link between student IDs and their respective names, ensuring a personalized touch to data handling. In parallel, the Course_Table meticulously records each student's grades within specific courses, capturing their academic journey in detail. Building on this foundation, the Final_Grade_Output_Table emerges as a culmination of this data, representing each student's calculated final grade within a course with precision.

This synthesis into the Final_Grade_Output_Table is achieved through a sophisticated process of selecting and merging data from both the Name_Table and Course_Table, based on the unifying Student_ID primary key. The calculation of the final grade is then executed with a carefully balanced formula, akin to finalGrade = test1 * 0.20 + test2 * 0.20 + test3 * 0.20 + finalExam * 0.40`, encapsulating the essence of academic assessment with mathematical elegance.

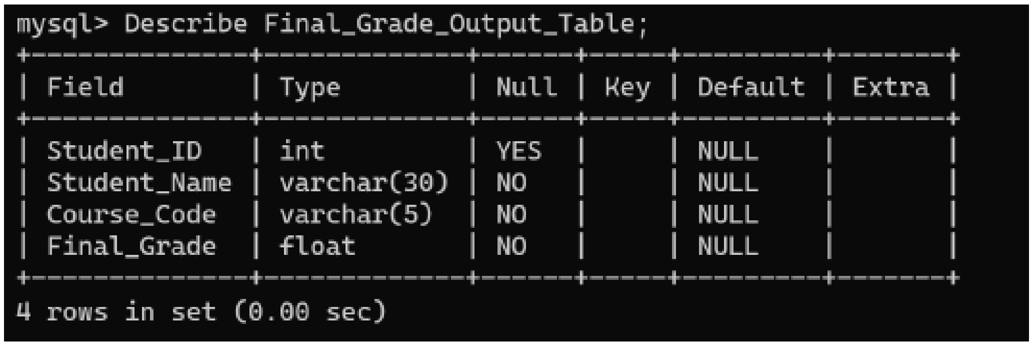
The schemas of the 3 tables:



```
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)
```