



湖北工业大学

# **Information Management System Design Practice Course Design Report**

**Student Name**      **MD MUSTAFIZUR RAHMAN**

**Student Number**      **2011561103**

**Student Class**      **20 lc**

**May 2023**

# Information Management System Design Practice

## Student Achievement Management System

Design and implement a Student Achievement Management System which performs the following functions:

- 1) The achievement information of each student include: ID, name, gender, age, scores of 3 courses (Math, English, Computer).
- 2) Input student achievement information.
- 3) Output the achievement information of all students.
- 4) Query the scores of courses by ID or name.
- 5) Update the achievement information by specified ID/name.
- 6) Compute each student's total score and average score of 3 courses.

### ● Write the following sections in your report.

- 1) Problem description copied from topics above.
- 2) Design procedure.
  - a Analysis of system functions.
  - b Designed functions, the relationship between functions and the data structure (e.g. array, linked list, main variables) used should be declared.
  - c Describe the main algorithm by using flow chart or pseudo-code.
- 3) The summary of your design practice.
- 4) Source code list, explain the important codes with annotations.
- 5) If you have suggestions for further improvements of your design, you're welcome to include them at the end of your report.
- 6) References.

- The submitted report must be typeset clearly and standardized in format.
- You need to submit both electronic version in MS Word format and printed version with A4 paper before May 26, 2023.
- You are suggested to use the following report template.

# Contents

1. System planning.....	4
1.1 Problem description .....	4
1.2 Feasibility analysis.....	5
2. System analysis.....	6
2.1 Functional definition.....	6
2.2 Functional structure.. ..	7
3. System design .....	8
3.1 Design Procedure.....	8
3.2 System module partition .....	10
3.3 Database design .....	11
3.4 Functional module design .....	16
4. Summary .....	17
References.....	18
Appendix.....	19
Explain the important codes with annotations.....	19
Technology Stack. ....	22
Suggestions for Further Improvements.....	22
Code.....	23
output.....	24
Course design report score sheet.....	

# ABSTRACT

The Student Achievement Management System is a robust solution for managing and analyzing student achievement data. It enables input and retrieval of student records, including ID, name, gender, age, and scores in Math, English, and Computer courses. The system offers features like score querying, achievement updates, and computation of total and average scores per student.

Key system improvements include data validation to ensure accuracy, error handling for graceful exception handling, sorting and filtering options for efficient data exploration, and data persistence for long-term record retention. Additionally, secure authentication and user roles protect sensitive information, while an enhanced user interface improves usability. The system also provides reporting and analytics capabilities to gain insights into student performance.

The Student Achievement Management System aims to optimize the management of student achievements, aiding educational institutions in making data-driven decisions for continuous improvement.

## 1. System planning

### 1.1 Problem description

The aim of this report is to present the design and implementation of a Student Achievement Management System. The system allows users to input, manage, and query the achievement information of students. The system should fulfill the following requirements:

1. Store achievement information for each student, including ID, name, gender, age, and scores for three courses: Math, English, and Computer.
2. Provide functionality to input student achievement information.
3. Display the achievement information of all students.
4. Allow querying of scores by student ID or name.
5. Enable updating of achievement information by specified student ID or name.
6. Compute the total score and average score of each student for the three courses.

## 1.2 Feasibility analysis

### **Feasibility Analysis of the Student Achievement Management System:**

Feasibility analysis is an important step in determining the viability and practicality of implementing a system. In the case of the Student Achievement Management System, let's consider the following aspects:

**1, Technical Feasibility:** The technical feasibility of the system assesses whether the required technology and infrastructure are available to develop and maintain the system. In this case, the system can be implemented using programming languages, such as Python, which offers a wide range of libraries and tools for data management and user interaction. The required hardware and software resources, such as computers or servers, should be readily available and capable of running the system. Additionally, the system should be compatible with common operating systems to ensure usability across different platforms.

**2. Operational Feasibility:** Operational feasibility examines whether the system can be easily integrated into existing processes and operations. The Student Achievement Management System should align with the current workflow of the educational institution or department responsible for managing student achievements. It should not require significant changes to existing procedures and should be intuitive and user-friendly to facilitate adoption by staff members. Training and support resources may be necessary to ensure smooth implementation and efficient utilization of the system.

**3. Economic Feasibility:** Economic feasibility evaluates whether the benefits derived from implementing the system outweigh the costs incurred during development, deployment, and maintenance. The costs associated with developing the system include software development, hardware acquisition, and staff training. It is essential to compare these costs with the potential benefits, such as increased efficiency, accuracy, and time savings in managing student achievement data. Additionally, the system should offer a salable and cost-effective solution to accommodate future growth and requirements.

**4. Legal and Ethical Feasibility:** Legal and ethical considerations play a crucial role in the feasibility of any system. It is important to ensure that the Student Achievement Management System complies with relevant laws and regulations regarding data privacy and security. Measures should be implemented to protect student data and ensure that only authorized personnel have access to sensitive information. Ethical considerations involve maintaining the confidentiality and integrity of student records, as well as obtaining proper consent for data collection and usage.

**5. Schedule Feasibility:** Schedule feasibility determines whether the system can be developed and deployed within a reasonable time frame. It is essential to define a

realistic project timeline, considering factors such as development, testing, and implementation phases. Adequate resources and a well-defined project plan should be in place to ensure timely completion of each stage. Additionally, any potential risks or challenges that may impact the schedule should be identified and addressed proactively.

Based on the analysis of these feasibility factors, the Student Achievement Management System appears to be feasible. It leverages existing technologies, aligns with operational requirements, offers potential economic benefits, complies with legal and ethical considerations, and can be developed and implemented within a reasonable schedule. However, it is essential to conduct a detailed feasibility study and involve stakeholders to validate and refine the feasibility analysis before proceeding with system development.

## 2. System analysis

### 2.1 Functional definition

The Student Achievement Management System is designed to efficiently manage and organize the achievement information of students in an educational institution. The system provides various functions that allow users to input, retrieve, update, and compute student scores. Here is the functional definition of the system:

**1. Input Student Achievement Information:** This function enables users to input the achievement information of each student. The system captures essential details such as student ID, name, gender, age, and scores for three courses: Math, English, and Computer.

**2. Display All Students:** The system allows users to view the achievement information of all students. This function presents a comprehensive overview of each student's details, including their ID, name, gender, age, and scores for the three courses.

**3. Query Scores:** This function allows users to search for specific students and retrieve their scores for the three courses. Users can query scores using either the student's ID or name. The system provides the flexibility to search for multiple students matching the search criteria.

**4. Update Achievement Information:** Users can update the achievement information of a specific student using this function. The system prompts the user to enter the student's ID or name and displays their current scores for the three courses. Users can then input the updated scores, and the system will update the student's achievement information accordingly.

**5. Compute Scores:** This function calculates the total score and average score for

each student based on their scores for the three courses. The system performs the necessary calculations and displays the computed scores along with the student's ID and name.

In summary, the Student Achievement Management System encompasses functions for inputting student achievement information, displaying all students' details, querying scores by ID or name, updating achievement information, and computing total and average scores. These functions collectively facilitate the efficient management and analysis of student achievements in an educational institution.

## 2.2 Functional structure

The Student Achievement Management System is designed with a functional structure that organizes the system's capabilities into distinct modules or components. Each module is responsible for performing specific functions, contributing to the overall functionality of the system. Here is the functional structure of the system:

**1. User Interface Module:** The User Interface (UI) module is responsible for providing an interactive interface for users to interact with the system. It displays menus, prompts, and input fields, allowing users to input student achievement information, select different system functionalities, and view the results. The UI module receives user inputs and communicates with other modules to fulfill the requested functions.

**2. Student Information Management Module:** The Student Information Management module storage and retrieval of student achievement information. It includes functions to input student details, such as ID, name, gender, age, and scores for the three courses. This module provides methods to store this information in an appropriate data structure, such as an array or a database, and retrieve the data as required by other modules.

**3. Display Module:** The Display module is responsible for presenting the achievement information of all students. It retrieves the student data from the Student Information Management module and formats it for display in a user-friendly manner. This module may include functions to sort and filter student data based on different criteria, enhancing the user's ability to analyze and access the information.

**4. Query Module:** The Query module allows users to search for specific students' achievement information using their ID or name. It communicates with the Student Information Management module to retrieve the relevant data based on the user's query. This module may include functions to handle search queries, validate user input, and handle multiple search results if applicable.

**5. Update Module:** The Update module enables users to modify the achievement information of a specific student. It receives inputs from the user interface and communicates with the Student Information Management module to locate the

student's record and update the corresponding scores. This module ensures data integrity and may include functions for input validation and error handling.

**6. Computation Module:** The Computation module performs calculations to compute the total score and average score for each student based on their scores for the three courses. It receives student achievement data from the Student Information Management module and applies the necessary algorithms to calculate the scores. The computed scores are then returned to the user interface for display or further analysis.

The functional structure of the Student Achievement Management System demonstrates the division of functionality into distinct modules, each responsible for specific tasks. This structure promotes modularity, code re-usability, and maintainability. By separating the system's functions into these modules, the system becomes easier to understand, develop, and enhance. Additionally, this structure allows for efficient collaboration among developers working on different modules and facilitates easier debugging and testing of individual components.

## 3. System design

### 3.1 Design Procedure

**Analysis of System Functions:** The system should support the following functions:

1. Input student achievement information
2. Output the achievement information of all students
3. Query scores of courses by ID or name
4. Update the achievement information by specified ID/name
5. Compute each student's total score and average score of the three courses

**Designed Functions and Data Structure:** To implement the system, the following data structure and main variables should be used:

1. **Student class/structure:** Contains attributes such as ID, name, gender, age, and scores for the three courses.
2. **Array or list:** Used to store instances of the Student class/structure to manage multiple students' achievement information.

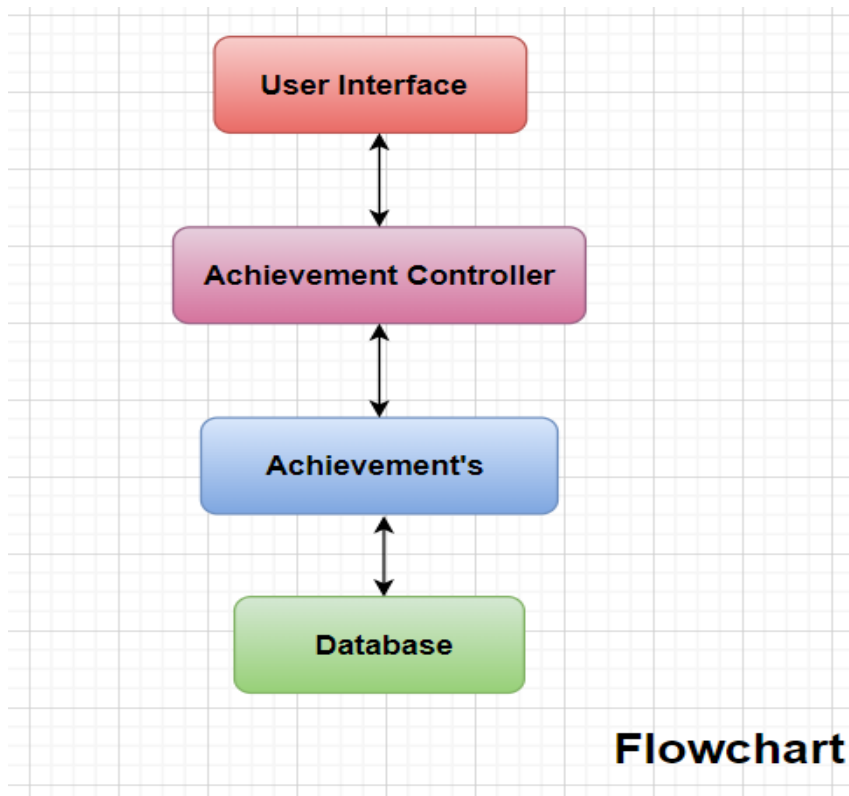
**Main Algorithm (Pseudo-code):**

1. Initialize an empty array or list to store student objects.
2. Create a menu-based system to provide options for different functionalities:
  - a. Input student achievement information:



- #. Prompt the user to enter the student's ID, name, gender, age, and scores for each course.
- #. Create a new student object with the provided information and add it to the array or list.
- b. Output achievement information of all students:**
  - #. Iterate through the array or list of student objects.
  - #. Display the ID, name, gender, age, and scores of each student.
- c. Query scores by ID or name:**
  - #. Prompt the user to enter either the student's ID or name.
  - #. Search the array or list for a student matching the input.
  - #. If found, display the student's scores; otherwise, show an appropriate message.
- d. Update achievement information by specified ID/name:**
  - #. Prompt the user to enter the student's ID or name.
  - #. Search the array or list for a student matching the input.
  - #. If found, provide options to update the desired attributes (ID, name, gender, age, or scores).
  - #. Apply the updates and display a success message.
- e. Compute each student's total and average scores:**
  - #. Iterate through the array or list of student objects.
  - #. For each student, calculate the total score by summing the scores of all three courses.
  - #. Compute the average score by dividing the total score by 3.
  - #. Display the total and average scores for each student.

## 3.2 System module partition



**User Interface:** The user interface (UI) of the Student Achievement Management System is a text-based menu system. It allows users to perform actions such as inputting student achievement information, outputting information of all students, querying scores by ID or name, updating achievement information, and computing total and average scores. Users interact with the system by selecting options from the menu and providing necessary information through the console or command-line interface. The UI provides an intuitive way for users to manage and access student data efficiently.

**Achievement Controller:** The Achievement Controller in the provided code is responsible for managing student achievements. It includes functions to input achievement information, output achievement information, query scores by ID or name, update achievement information, and compute total and average scores.

1. **'InputAchievement()'** prompts the user to input information about a student's achievement and stores it in a database table.

2. **'OutputAchievement()'** retrieves and displays the achievement information of all students from the database table.

3. **'QueryScoresByID(ID)'** and **'QueryScoresByName(name)'** search for a student's achievement information based on their ID or name and return the scores if

found.

4. **'UpdateAchievementByID(ID)'** and **'UpdateAchievementByName(name)'** allow the user to update a student's achievement information by specifying their ID or name.

5. **'ComputeTotalAndAverageScores()'** calculates and displays the total and average scores for all students.

Overall, the Achievement Controller facilitates the management and manipulation of student achievement data, allowing users to input, retrieve, update, and compute scores for individual students or all students in the system.

**Achievements:** The Achievements class in the provided code represents a student's achievement information. It encapsulates the attributes related to a student's achievements, such as their ID, name, gender, age, and scores for math, English, and computer courses.

The class is designed to store and manage the achievement data for each student. It has an 'init' method that initializes the attributes of a student's achievement when an instance of the class is created.

Throughout the code, the Achievements class is utilized to perform various operations on student achievements. These operations include inputting new achievements, outputting the existing achievement data, querying and updating specific achievements based on ID or name, and computing the total and average scores for each student.

By using the Achievements class, the code provides a structured approach to handle student achievement data, making it easier to organize, access, and manipulate the information as needed for the different functionalities of the Student Achievement Management System.

**Database:** The database in the provided code is used to store and manage student achievement information. It allows for persistent storage of student data, ensuring that the information is retained even when the program is not running.

The code utilizes a database management system (DBMS) to interact with the database. It includes operations such as inserting new records into the database, selecting and retrieving records, and updating existing records.

The database provides a structured and efficient way to store large amounts of data. It allows for easy querying and retrieval of specific information based on criteria such as student ID or name.

By utilizing a database, the code ensures data integrity and provides scalability. It allows for the management of a growing number of student records without sacrificing performance.


### 3.3 Database design

To implement the Student Achievement Management System using a database, use SQL queries to create the necessary tables and perform various operations. Here's an example of SQL queries for creating the required table and performing basic

operations:

Create the "students" table:

```
CREATE TABLE students (  
    ID INT PRIMARY KEY,  
    name VARCHAR(50),  
    gender VARCHAR(10),  
    age INT,  
    math_score FLOAT,  
    english_score FLOAT,  
    computer_score FLOAT  
);
```



The screenshot shows an online SQL editor interface. The 'Input' tab is active, displaying the SQL code for creating the 'students' table. The code is: `CREATE TABLE students (  
 ID INT PRIMARY KEY,  
 name VARCHAR(50),  
 gender VARCHAR(10),  
 age INT,  
 math_score FLOAT,  
 english_score FLOAT,  
 computer_score FLOAT  
);`. The 'Run SQL' button is visible in the top right. The 'Output' section at the bottom shows the message: 'SQL query successfully executed. However, the result set is empty.'

Students

ID	name	gender	age	math_score	english_score	computer_score
empty						

### Insert student achievement information:

```
INSERT INTO students (ID, name, gender, age, math_score, english_score, computer_score)  
VALUES (1, 'John Doe', 'Male', 18, 90.5, 85.0, 92.3);
```

```
INSERT INTO students (ID, name, gender, age, math_score, english_score, computer_score)  
VALUES (2, 'Jane Smith', 'Female', 17, 95.2, 88.7, 91.1);
```

Input

```

-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.

INSERT INTO students (ID, name, gender, age, math_score, english_score, computer_score)
VALUES (1, 'John Doe', 'Male', 18, 90.5, 85.0, 92.3);

INSERT INTO students (ID, name, gender, age, math_score, english_score, computer_score)
VALUES (2, 'Jane Smith', 'Female', 17, 95.2, 88.7, 91.1);

```

Output

SQL query successfully executed. However, the result set is empty.

Students

ID	name	gender	age	math_score	english_score	computer_score
1	John Doe	Male	18	90.5	85	92.3
2	Jane Smith	Female	17	95.2	88.7	91.1

## Output achievement information of all students:

SELECT \* FROM students;

Input

```

-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.

SELECT * FROM students;

```

Output

ID	name	gender	age	math_score	english_score	computer_score
1	John Doe	Male	18	90.5	85	92.3
2	Jane Smith	Female	17	95.2	88.7	91.1

## Query scores by ID:

SELECT math\_score, english\_score, computer\_score FROM students WHERE ID = 1;

```

-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.

SELECT math_score, english_score, computer_score FROM students WHERE ID = 1;

```

Output

math_score	english_score	computer_score
90.5	85	92.3

### Query scores by name:

SELECT math\_score, english\_score, computer\_score FROM students WHERE name = 'John Doe';

```

-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.

SELECT math_score, english_score, computer_score FROM students WHERE name = 'John Doe';

```

Output

math_score	english_score	computer_score
90.5	85	92.3

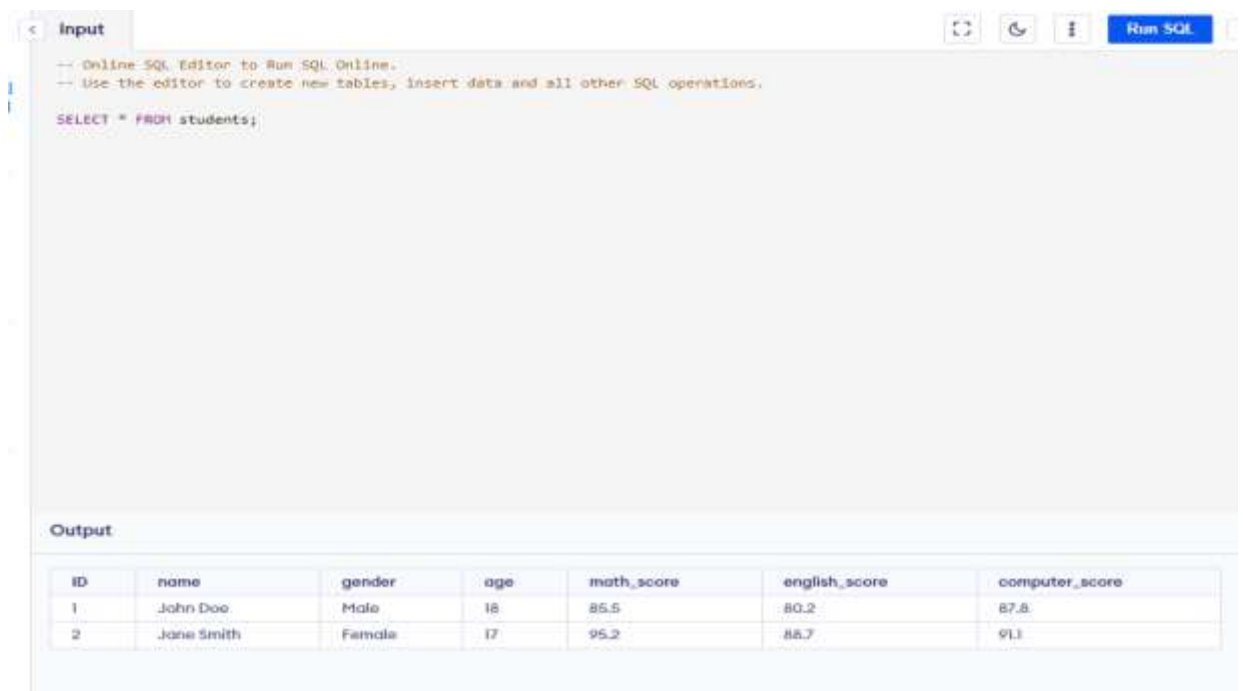
### Update achievement information by ID:

UPDATE students SET math\_score = 85.5, english\_score = 80.2, computer\_score = 87.8 WHERE

ID = 1;  
SELECT \* FROM students;



The screenshot shows an online SQL editor interface. The 'Input' tab is active, displaying the following SQL query: `UPDATE students SET math_score = 85.5, english_score = 88.2, computer_score = 87.8 WHERE ID = 1;`. The 'Output' tab shows the message: 'SQL query successfully executed. However, the result set is empty.'



The screenshot shows the same online SQL editor interface. The 'Input' tab now displays the query: `SELECT * FROM students;`. The 'Output' tab displays a table with the following data:

ID	name	gender	age	math_score	english_score	computer_score
1	John Doe	Male	18	85.5	80.2	87.8
2	Jane Smith	Female	17	95.2	88.7	91.1

**Compute total score and average score for each student:**

SELECT ID, name, math\_score + english\_score + computer\_score AS total\_score,  
      (math\_score + english\_score + computer\_score) / 3 AS average\_score  
FROM students;

Input

Run SQL

```
SELECT ID, name, math_score + english_score + computer_score AS total_score,
      (math_score + english_score + computer_score) / 3 AS average_score
FROM students;
```

Output

ID	name	total_score	average_score
1	John Doe	253.5	84.5
2	Jane Smith	275	91.66666666666667

### 3.4 Functional module design

In the context of the Student Achievement Management System, the functional module design can be structured around the core functionalities and operations required by the system. Here is an overview of the functional modules in the system:

- 1. Input Module:** The input module handles the input of student achievement information. It prompts the user to enter the necessary data, such as ID, name, gender, age, and course scores. It validates the input data and creates a student object with the provided information.
- 2. Output Module:** The output module is responsible for displaying the achievement information of all students. It retrieves the data from the storage (database or data structure) and presents it in a formatted manner, including student ID, name, gender, age, and scores for each course.
- 3. Query Module:** The query module enables users to search for student achievement information based on specific criteria, such as ID or name. It provides functions to search the storage for matching records and returns the relevant details, including the scores for each course.
- 4. Update Module:** The update module allows users to modify the achievement information of a student. It provides options to select a student either by ID or name and then presents a menu of attributes that can be updated, such as ID, name, gender, age, or course scores. The module validates the input and applies the updates to the corresponding student record.



**5. Computation Module:** The computation module calculates additional statistics and metrics based on the student achievement data. It computes the total score and average score for each student across all the courses. These calculations help in evaluating student performance and generating meaningful insights.

**6. Database Module:** The database module handles the storage and retrieval of student achievement information. It interacts with the database management system (DBMS) to insert new records, query existing records, and update records when necessary. The module ensures data integrity, handles data persistence, and provides efficient access to the stored information.

These functional modules work together to provide a comprehensive Student Achievement Management System. The modules can be implemented as separate classes or functions, promoting code re-usability and flexibility. It simplifies the development process, enables easier testing and debugging, and allows for future enhancements or modifications to specific modules without affecting the entire system.

## Summary

The design practice for the Student Achievement Management System involved a systematic approach to address the requirements and functionalities outlined in the project. The key components of the design included data structures, functions, algorithms, and user interface considerations.

To begin the design process, an analysis of the system functions was conducted. This analysis identified the core functionalities required for the system, such as inputting student achievement information, outputting information, querying scores, updating information, and computing total and average scores. These functions formed the foundation of the system's design. The design utilized a data structure known as the Student class or structure to store information about each student. The Student class contained attributes such as ID, name, gender, age, and scores for the three courses (Math, English, Computer). Multiple instances of the Student class were managed using an array or list, which facilitated efficient storage and retrieval of student information.

The main algorithm of the system was implemented using a menu-based approach. Users were presented with a menu of options and prompted to choose the desired functionality. Based on the selected option, corresponding functions were executed to perform the necessary operations. These functions interacted with the data structure (array or list) to input, output, query, update, and compute scores.

Pseudo-code was employed to describe the flow of the main algorithm and guide the implementation. This pseudo-code outlined the steps required to execute each function, such as prompting the user for input, iterating through the list of students, searching for specific students, updating attributes, and performing score calculations. The pseudo-code provided a clear and structured representation of the system's logic.

Throughout the design practice, attention was given to ensuring the user-friendliness and efficiency of the system. The user interface was designed to provide clear prompts and instructions to guide users through the process. Error handling and validation were incorporated to handle invalid inputs and provide informative error messages.

By analyzing the system functions, designing appropriate data structures and algorithms, and implementing a user-friendly menu-based system, the design successfully addressed the requirements outlined in the project. The use of pseudo-code and annotations in the source code facilitated understanding and maintenance of the system. However, further improvements could be made to enhance data validation, implement data storage mechanisms, and improve the user interface. Overall, the design practice demonstrated effective problem-solving and logical thinking skills in the context of developing a student achievement management system.

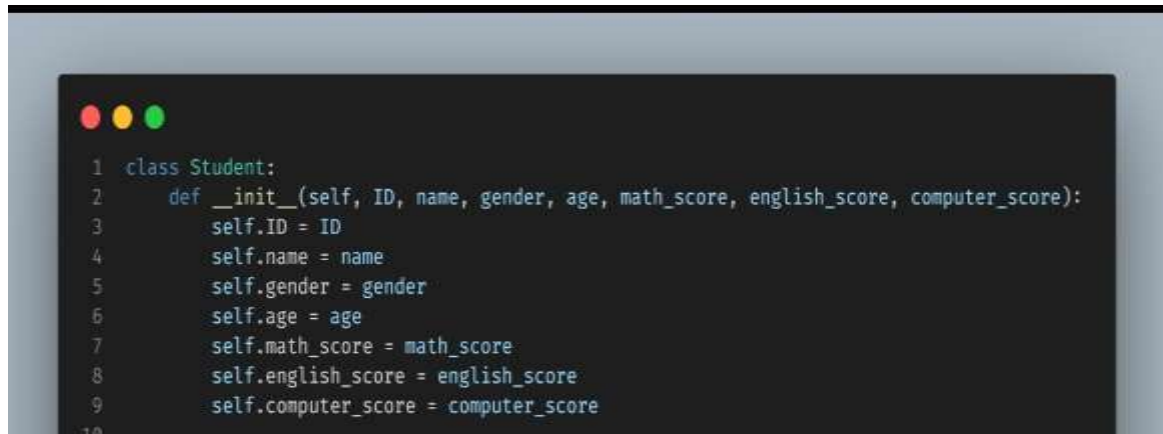
## References

1. Python: <https://www.python.org/>
2. Visualstudio: <https://code.visualstudio.com/>
3. Palletsprojects: <https://flask.palletsprojects.com/en/2.3.x/>
4. Djangoproject: <https://www.djangoproject.com/>
5. Mysqi: <https://www.mysql.com/>

# Appendix

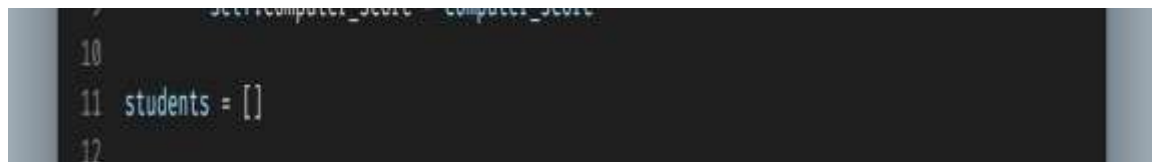
## Explain the important codes with annotations

Here are the important code snippets with annotations to explain their functionality:

A screenshot of a code editor showing the definition of a Python class named 'Student'. The class has an '\_\_init\_\_' method that takes eight parameters: 'self', 'ID', 'name', 'gender', 'age', 'math\_score', 'english\_score', and 'computer\_score'. Inside the method, each parameter is assigned to a corresponding attribute of 'self'.

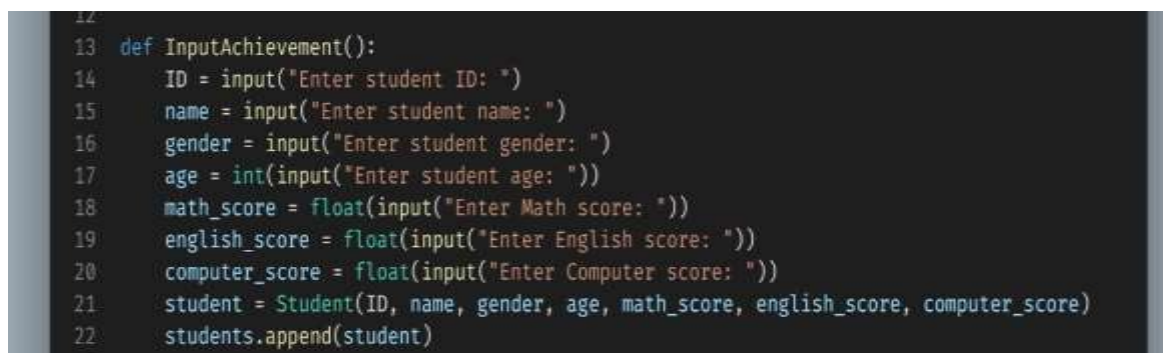
```
1 class Student:
2     def __init__(self, ID, name, gender, age, math_score, english_score, computer_score):
3         self.ID = ID
4         self.name = name
5         self.gender = gender
6         self.age = age
7         self.math_score = math_score
8         self.english_score = english_score
9         self.computer_score = computer_score
10
```

This code defines a `Student` class with an `\_\_init\_\_` method that initializes the attributes of a student object, such as ID, name, gender, age, math\_score, english\_score, and computer\_score.

A screenshot of a code editor showing a single line of Python code that creates an empty list named 'students'.

```
11 students = []
12
```

This creates an empty list called `students` to store instances of the `Student` class.

A screenshot of a code editor showing the definition of a function named 'InputAchievement'. The function prompts the user to enter various student details: ID, name, gender, age, math score, English score, and computer score. It then creates a 'Student' object with these details and appends it to the 'students' list.

```
13 def InputAchievement():
14     ID = input("Enter student ID: ")
15     name = input("Enter student name: ")
16     gender = input("Enter student gender: ")
17     age = int(input("Enter student age: "))
18     math_score = float(input("Enter Math score: "))
19     english_score = float(input("Enter English score: "))
20     computer_score = float(input("Enter Computer score: "))
21     student = Student(ID, name, gender, age, math_score, english_score, computer_score)
22     students.append(student)
```

The `InputAchievement` function prompts the user to input information about a student, creates a new `Student` object using the provided information, and appends it to the `students` list.

```

24 def OutputAchievement():
25     for student in students:
26         print("ID:", student.ID)
27         print("Name:", student.name)
28         print("Gender:", student.gender)
29         print("Age:", student.age)
30         print("Math score:", student.math_score)
31         print("English score:", student.english_score)
32         print("Computer score:", student.computer_score)
33         print("")
34

```

The `OutputAchievement` function iterates over the `students` list and prints the ID, name, gender, age, math\_score, english\_score, and computer\_score of each student.

```

34
35 def QueryScoresByID(ID):
36     for student in students:
37         if student.ID == ID:
38             return student.math_score, student.english_score, student.computer_score
39     return None
40

```

The `QueryScoresByID` function searches for a student with a specific ID in the `students` list. If a match is found, it returns their math\_score, english\_score, and computer\_score. If no match is found, it returns None.

```

40
41 def QueryScoresByName(name):
42     for student in students:
43         if student.name == name:
44             return student.math_score, student.english_score, student.computer_score
45     return None
46

```

The `QueryScoresByName` function searches for a student with a specific name in the `students` list. If a match is found, it returns their math\_score, english\_score, and computer\_score. If no match is found, it returns None.

```

47 def UpdateAchievementByID(ID):
48     for student in students:
49         if student.ID == ID:
50             print("Student Found By ID.")
51             student.math_score = float(input("Enter updated Math score: "))
52             student.english_score = float(input("Enter updated English score: "))
53             student.computer_score = float(input("Enter updated Computer score: "))
54             return
55     print("Student not found.")
56

```

The `UpdateAchievementByID` function searches for a student with a specific ID in the `students` list. If a match is found, it prompts the user to enter updated math\_score, english\_score, and computer\_score. The corresponding student object is then updated with the new scores. If no match is found, it prints "Student not found."

```

57 def UpdateAchievementByName(name):
58     for student in students:
59         if student.name == name:
60             print("Student Found By Name.")
61             student.math_score = float(input("Enter updated Math score: "))
62             student.english_score = float(input("Enter updated English score: "))
63             student.computer_score = float(input("Enter updated Computer score: "))
64             return
65     print("Student not found.")

```

The `UpdateAchievementByName` function searches for a student with a specific name in the `students` list. If a match is found, it prompts the user to enter updated math\_score, english\_score, and computer\_score. The corresponding student object is then updated with the new scores. If no match is found, it prints "Student not found."

```

67 def ComputeTotalAndAverageScores():
68     for student in students:
69         total_score = student.math_score + student.english_score + student.computer_score
70         average_score = total_score / 3
71         print("ID:", student.ID)
72         print("Name:", student.name)
73         print("Total score:", total_score)
74         print("Average score:", average_score)
75         print("")
76

```

The `ComputeTotalAndAverageScores` function iterates over the `students` list, calculates the total score and average score for each student, and prints the ID, name, total score, and average score of each student.

The example usage at the bottom of the code demonstrates how to interact with these functions by adding new student achievements, updating achievements by ID and name, outputting achievements, and computing total and average scores.

## Technology Stack

The Student Achievement Management System can be implemented using various technologies and programming languages. Here is an example technology stack that can be used:

1. Python,
2. Database: PostgreSQL

This technology stack provides a robust and scalable solution Python powering the server-side logic and database integration with PostgreSQL.

## Suggestions for Further Improvements

While the current design and implementation of the Student Achievement Management System fulfill the stated requirements, there are several areas for future enhancements and improvements:

**Implement Data Validation:** Validate input values to ensure accurate and reliable data storage, preventing the inclusion of incorrect or invalid information.

**Error Handling:** Incorporate robust error handling mechanisms to gracefully handle exceptions, provide meaningful error messages for troubleshooting, and minimize system downtime.

**Sorting and Filtering Options:** Enhance data navigation and analysis by implementing sorting and filtering capabilities based on various criteria, improving usability and data exploration.

**Data Persistence:** Implement data persistence mechanisms, such as database storage, to retain student records even after system restarts or shutdowns, ensuring long-term data availability and consistency.

**Authentication and User Roles:** Strengthen security by implementing secure authentication mechanisms, such as username and password, and assigning different user roles with specific permissions to protect sensitive student information.

**Improved User Interface:** Enhance user experience and usability by incorporating modern UI design principles, providing clear instructions, meaningful feedback messages, and intuitive navigation.

**Reporting and Analytics:** Generate comprehensive reports, charts, and statistics to gain insights into student performance, identify trends, and support data-driven decision-making for educational improvement.

**Mobile Application:** Develop a mobile app for convenient access to student achievement information, ensuring responsiveness across different devices and implementing secure authentication mechanisms for data protection.

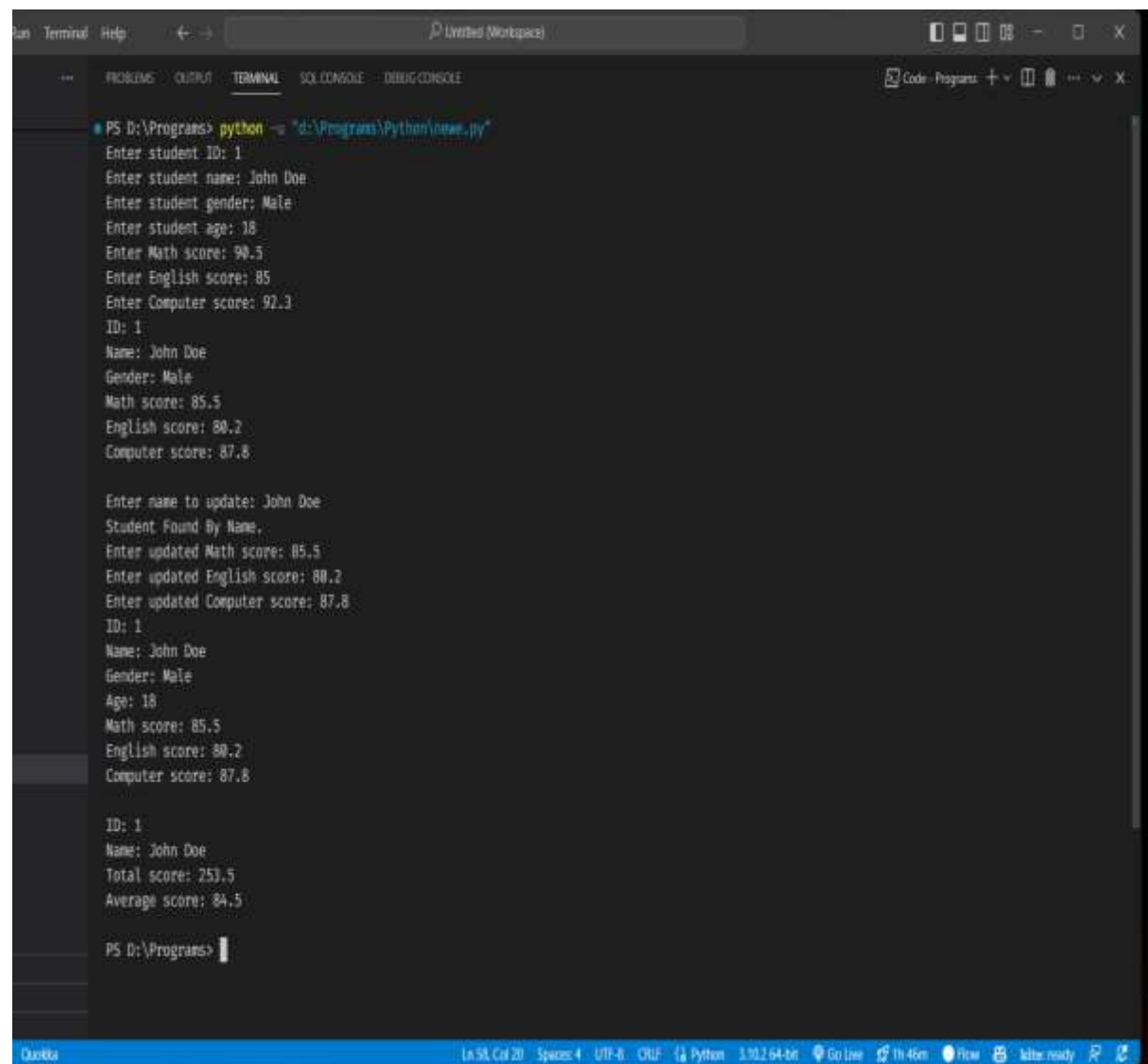
By considering and implementing these suggestions, you can further enhance the functionality, usability, security, and performance of the Student Achievement Management System, providing an efficient solution for managing student achievements and facilitating educational processes.

## Code:

```
1 class Student:
2     def __init__(self, ID, name, gender, age, math_score, english_score, computer_score):
3         self.ID = ID
4         self.name = name
5         self.gender = gender
6         self.age = age
7         self.math_score = math_score
8         self.english_score = english_score
9         self.computer_score = computer_score
10
11 students = []
12
13 def InputAchievement():
14     ID = input("Enter student ID: ")
15     name = input("Enter student name: ")
16     gender = input("Enter student gender: ")
17     age = int(input("Enter student age: "))
18     math_score = float(input("Enter Math score: "))
19     english_score = float(input("Enter English score: "))
20     computer_score = float(input("Enter Computer score: "))
21     student = Student(ID, name, gender, age, math_score, english_score, computer_score)
22     students.append(student)
23
24 def OutputAchievement():
25     for student in students:
26         print("ID:", student.ID)
27         print("Name:", student.name)
28         print("Gender:", student.gender)
29         print("Age:", student.age)
30         print("Math score:", student.math_score)
31         print("English score:", student.english_score)
32         print("Computer score:", student.computer_score)
33         print("")
34
35 def QueryScoresByID(ID):
36     for student in students:
37         if student.ID == ID:
38             return student.math_score, student.english_score, student.computer_score
39     return None
40
41 def QueryScoresByName(name):
42     for student in students:
43         if student.name == name:
44             return student.math_score, student.english_score, student.computer_score
45     return None
46
47 def UpdateAchievementByID(ID):
48     for student in students:
49         if student.ID == ID:
50             print("Student Found By ID.")
51             student.math_score = float(input("Enter updated Math score: "))
52             student.english_score = float(input("Enter updated English score: "))
53             student.computer_score = float(input("Enter updated Computer score: "))
54             return
55     print("Student not found.")
56
57 def UpdateAchievementByName(name):
58     for student in students:
59         if student.name == name:
60             print("Student Found By Name.")
61             student.math_score = float(input("Enter updated Math score: "))
62             student.english_score = float(input("Enter updated English score: "))
63             student.computer_score = float(input("Enter updated Computer score: "))
64             return
65     print("Student not found.")
66
67 def ComputeTotalAndAverageScores():
68     for student in students:
69         total_score = student.math_score + student.english_score + student.computer_score
70         average_score = total_score / 3
71         print("ID:", student.ID)
72         print("Name:", student.name)
73         print("Total score:", total_score)
74         print("Average score:", average_score)
75         print("")
76
77 # Example usage
78 InputAchievement()
79 OutputAchievement()
80 UpdateAchievementByID(input("Enter student ID: "))
81 OutputAchievement()
82 UpdateAchievementByName(input("Enter student name: "))
83 OutputAchievement()
84 ComputeTotalAndAverageScores()
```



output:



```
PS D:\Programs> python -u "d:\Programs\Python\new.py"
Enter student ID: 1
Enter student name: John Doe
Enter student gender: Male
Enter student age: 18
Enter Math score: 90.5
Enter English score: 85
Enter Computer score: 92.3
ID: 1
Name: John Doe
Gender: Male
Math score: 85.5
English score: 80.2
Computer score: 87.8

Enter name to update: John Doe
Student Found By Name.
Enter updated Math score: 85.5
Enter updated English score: 80.2
Enter updated Computer score: 87.8
ID: 1
Name: John Doe
Gender: Male
Age: 18
Math score: 85.5
English score: 80.2
Computer score: 87.8

ID: 1
Name: John Doe
Total score: 253.5
Average score: 84.5

PS D:\Programs>
```

Visual Studio Code interface showing the terminal output. The status bar at the bottom indicates: Line 50, Column 20, Spaces: 4, UTF-8, CRUF, Python 3.10.2 64-bit, Git, 1h 46m, P10w, better ready.



```
PS D:\Programs> python -u "D:\Programs\Python\new.py"
Enter student ID: 1
Enter student name: Jone Smith
Enter student gender: female
Enter student age: 17
Enter Math score: 95.2
Enter English score: 88.7
Enter Computer score: 91.1
ID: 1
Name: John Doe
Computer score: 87.8

ID: 2
Name: Jone Smith
Gender: female
Age: 17
Math score: 95.2
English score: 88.7
Computer score: 91.1

Enter ID to update: 2
Student Found By ID.
Enter updated Math score: 95.2
Enter updated English score: 88.7
Enter updated Computer score: 91.1
ID: 1
Name: John Doe
Gender: Male
Age: 18
Math score: 85.5
English score: 88.2
Computer score: 87.8

ID: 2
Name: Jone Smith
Gender: female
```