# **Student Management System**

**Note:** The System is totally consoled based, no session will be saved.

**Planning:**

* We decided to develop a mini version of SMS.
* We have gathered information from internet about our project.
* We have divided the project into portions.
* We used **Waterfall Model** for our project.

The requirements are as follows:

**Requirements:**

We will discuss the requirements of Student Management System in detail. All of these are discussed below in detail.

**User Requirements:**

* User will be able to login to perform different operations.
* User will be able to enter Students information.
* User will be able to add data of 30 students.
* User will be able to view the entered data.
* User will be able to search specific record via (student id, name, class, degree, admission year).
* User will be able to update existing records.
* User will be able to delete all the entered data.
* User will be able to view Fee record.
* User will be able to Submit fee of a specific student through student id.
* User will be able to logout the program.

**Functional Requirements:**

* **Login:**
  + - Users must log in with a username and password.
    - The username is "admin123" and the password is "46465."
    - Incorrect entries for the username or password the system must prompt the user to re-enter the correct username or password.
* **Data Insertion:**
  + - Users must be able to insert information for multiple students.
    - The system must double-check user inputs for each student's information (e.g., Student ID, Name, Class, Degree, Admission Year).
    - The system must allow entering data for up to 30 students.
    - For incorrect entries the system will prompt user to re-enter.
* **Data Display:**
  + - Users should be able to view all the inserted data of students.
    - If no data is entered, the system will show that “**No Data is Entered yet**”.
* **Data Search:**
* Users will be able to search for student data via (student id , Name, Class, Degree, Admission Year).
* If the searched data is found, it will display the data otherwise, a message will appear that “**Data Not Found**".
* **Data Update:**
* Users will be able to update the information for a specific student based on their student ID.
* User will be able to modify the complete data of student.
* The system will display the previous data before updating the record.
  + **Data Deletion:**
    - User will be able to delete all records.
    - A confirmation prompt must be displayed before deleting all records.
* **Fee Management:**
  + - The program includes a fee-related functionality allowing users to view fee records and submit fees.
    - Users can view the remaining fee for each student and submit additional unpaid fees.
    - The program asks for bank name, transaction ID, and the fee amount. It updates the paid and remaining fee accordingly.
* **Logout:**
  + - Users will be able to log out of the system, and the system will display a

logout message.

**Non-functional Requirements:**

1. **Usability:**
   * The system has a user-friendly interface with clear instructions and prompts.
2. **Reliability:**
   * The system is able to handle errors and avoid unexpected crashes.
3. **Security:**
   * The login system is provided to ensure the basic security by requiring a valid username and password.
4. **Accuracy:**

* The system is able to manage fee dues accurately.

1. **Scalability:**

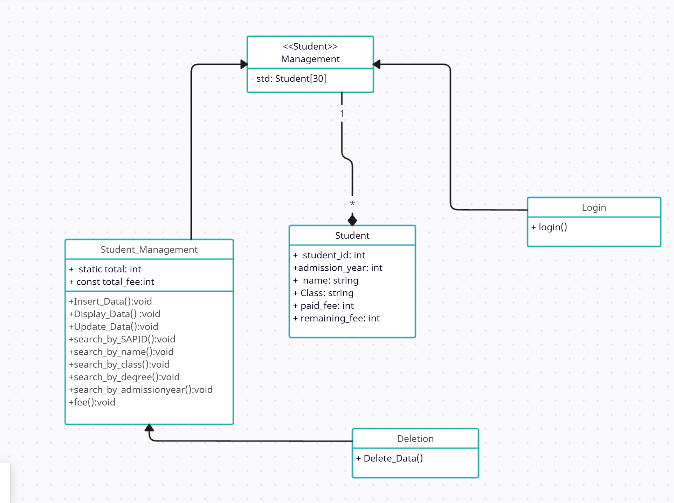
* The system is able to scale and handle the increasing number of student records.

**Constraints:**

* User can store a maximum of 30 students’ data.
* User will Enter Student ID of 6 or fewer digits.
* Name is a string of up to 30 characters.
* Degree is a string of up to 40 characters.
* Admission Year will be between 1990 and 2023.
* Bank name is a string of up to 50 characters.
* User is restricted at various areas to input specific data.

**Design:**

**Class Diagram:**



**Architectural Design:**

A screenshot of a computer

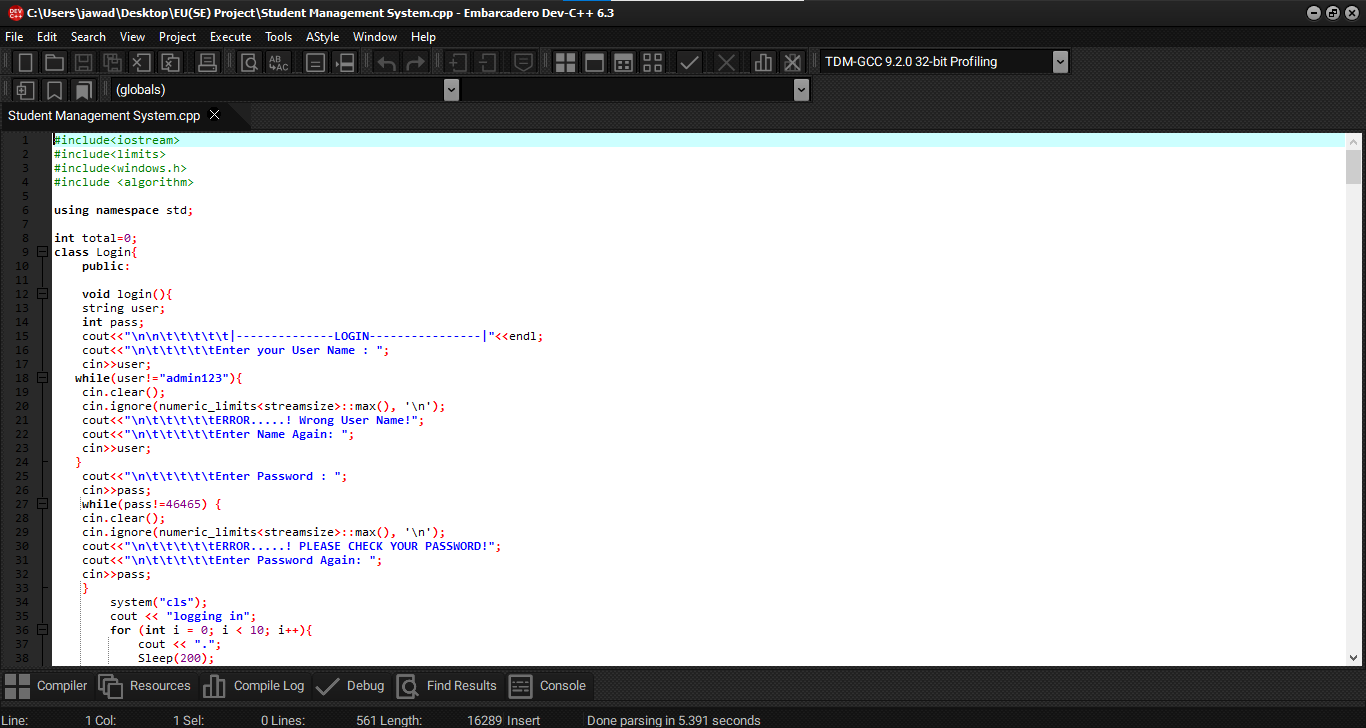
Description automatically generated

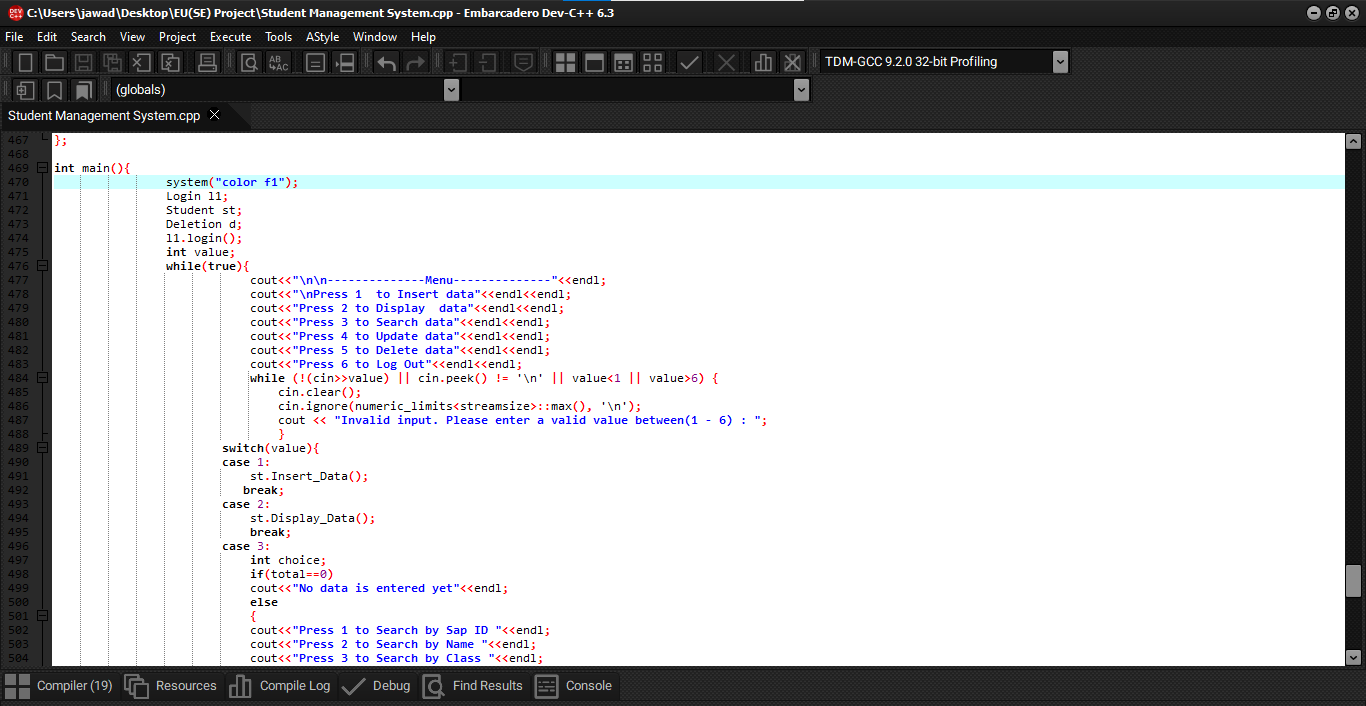
A screenshot of a computer

Description automatically generated

**Implementation:**

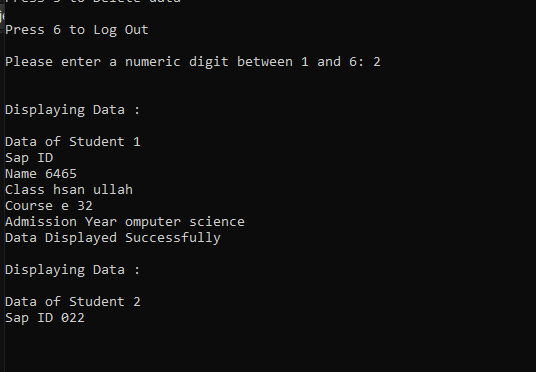
* The Project is completely written in c++.
* Project file is uploaded on Moellim.



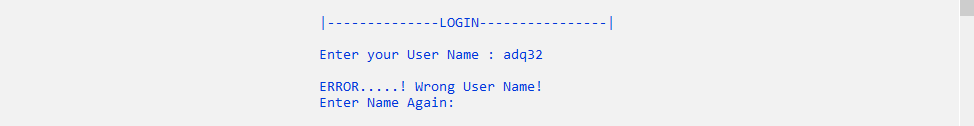


**Testing:**

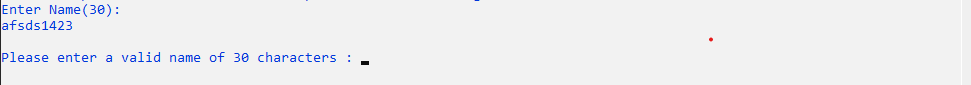
* Error Handling is done at every user input.
* The most challenging error to solve was the use of getline() which wasted a lot of our time. As it was not able to show the correct data entered. As you can see



* After testing the error handling done are;
* At login system will prompt user when wrong username or password is entered.



* At name(string)only Alphabets can be entered.



* At certain areas the user is restricted to enter specific input and if restriction doesn’t work it will prompt and will avoid crashing.

