CPP STUDENT DATABASE MANAGEMENT SYSTEM

```cpp
#include <iostream>
#include <cstring>
#include <vector>
#include <string>
#include <stdexcept> // For exception handling
#include <algorithm>



using namespace std;


void access_course_id();
void access_roll_number();
void access_name();
void number_of_students();
void update();
void delete_student();
void display_all_students();


struct Student
{
  string first_name;
  string last_name;
  string roll_number;
  int course_id[5];
  double sgpa[8];
};


vector<Student> students;           // Dynamically store student data
int num_students = 0;               // Track the number of students


// Function to validate string input (replace with more robust validation if
needed)
bool validateString(const string& input) {
  // Check for empty strings and invalid characters (adjust as needed)
  return !input.empty() && all_of(input.begin(), input.end(), [](char c) {
    return isalnum(c) || isspace(c);
  });
}
```

```
// Function to validate numeric input (replace with more robust validation if
needed)
bool validateNumeric(const string& input) {
  // Check if the input string contains only digits and a single decimal point
(optional)
  return all_of(input.begin(), input.end(), [](char c) {
    return isdigit(c) || c == '.';
  });
}


void create_database()
{
  Student new_student;

  // Enter the name of the student

  cout << "Enter the name of the student : " << endl;
  getline(cin, new_student.first_name);
  if (!validateString(new_student.first_name))
  {
    cerr << "Invalid name. Please enter alphanumeric characters and spaces
only." << endl;
    return; // Exit function if name is invalid
  }

  //Enter the last name of the student

  cout << "Enter the last name of the student : " << endl;
  getline(cin, new_student.last_name);
  if (!validateString(new_student.last_name))
  {
    cerr << "Invalid last name. Please enter alphanumeric characters and
spaces only." << endl;
    return;
  }


  //Enter the student roll number
  cout << "Enter the student roll number : " << endl;
  getline(cin, new_student.roll_number);
  if (!validateString(new_student.roll_number))
  {
    cerr << "Invalid roll number. Please enter alphanumeric characters only."
<< endl;
    return;
  }
```

```
  //Enter the sgpa of the student
  cout << "Enter the sgpa (up to 8 SEM): " << endl;
  for (int j = 0; j < 8; j++)
  {
    cout << "SGPA of SEM : " << j + 1<<endl;
    string cgpa_str;
    getline(cin, cgpa_str);
    if (validateNumeric(cgpa_str)) {
      new_student.sgpa[j] = stof(cgpa_str);
    } else {
      new_student.sgpa[j] = 0; // Assume 0 for invalid input
      cerr << "Invalid course ID. Skipping..." << endl;
    }
    if (new_student.sgpa[j] == 0) {
      break;
    }
  }


  //Enter the course id (up to 5):
  cout << "Enter the course id (up to 5): " << endl;
  for (int j = 0; j < 5; j++)
  {
    cout << "Course Id of : " << j + 1 << " (enter 0 to skip): ";
    string course_id_str;
    getline(cin, course_id_str);
    if (validateNumeric(course_id_str)) {
      new_student.course_id[j] = stoi(course_id_str);
    } else {
      new_student.course_id[j] = 0; // Assume 0 for invalid input
      cerr << "Invalid course ID. Skipping..." << endl;
    }
    if (new_student.course_id[j] == 0) {
      break;
    }
  }

  cin.ignore(); // Clear input buffer after numeric input

  students.push_back(new_student);
  num_students++;
  cout << "Student record created successfully!" << endl;
}


// Implement similar validation and error handling for other functions
(access_name, access_roll_number, access_course_id)
int main()
{
```

```
  int choice;

  while (1)
  {
    cout << "1. Create database" << endl;
    cout << "2. Access With name of the student" << endl;
    cout << "3. Access With roll number" << endl;
    cout << "4. Access with Course ID" << endl;
    cout << "5. Find number of students"<<endl;
    cout << "6. update student database"<<endl;
    cout << "7. Delete student database"<<endl;
    cout << "8. Display all Students "<<endl;
    cout << "9. Exit" << endl;
    cout << "Enter your choice: ";
    cin >> choice;

    cin.ignore(); // Clear input buffer

    switch (choice)
    {
      case 1:     //Implement create_database function with validation and
error handling
                create_database();
                break;
      case 2:     // Implement access_name function with validation and error
handling
                access_name();
                break;
      case 3:     // Implement access_roll_number function with validation and
error handling
                access_roll_number();
                break;
      case 4:     // Implement access_course_id function with validation and
error handling
                access_course_id();
                break;
      case 5:     // Implement number_of_students function
              number_of_students();
              break;
      case 6:     // Implement update function with validation and error
handling
              update();
              break;
      case 7:     // Implement delete_student function
              delete_student();
              break;
      case 8:     // Implement display_all_students function
              display_all_students();
```

```
            break;
      case 9:     //Exit from the Program
              exit(0);
              break;
      default:
              cout << "Invalid choice" << endl << endl;
    }
  }
}


// Placeholder for access_name function
void access_name()
{
  double total_sgpa=0.0;
  int count=0;
  string name;
  cout << "Enter the name of the student : " << endl;
  getline(cin, name);
  int found = 0;
  for (int j = 0; j < num_students; j++)
  {
    if (name == students[j].first_name)
    {
      cout << "Student name : " << students[j].first_name << endl;
      cout << "Student last name : " << students[j].last_name << endl;
      cout << "Student roll number : " << students[j].roll_number << endl;
      cout << "Student sgpa"<< endl;
      for(int s=0;s<8;s++)
      {
        if(students[j].sgpa[s]!=0)
        {
          cout<<"SGPA of the student "<<s+1<<" : "<<students[j].sgpa[s]<<endl;
          total_sgpa+=students[j].sgpa[s];
          count++;
        }
      }

      cout << "Student course id : " << endl;
      for (int c = 0; c < 5; c++) {
        if (students[j].course_id[c] != 0)
        {
          cout << "Course Id Of the student " << c + 1 << ":" <<
students[j].course_id[c] << endl;
        }
      }
      if(count>0)
      {
        double cgpa=total_sgpa/count;
```

```
        cout<<"Student CGPA : "<<cgpa<<endl;
        double percentage=(cgpa-0.75)*10;
        cout<<"Student Percentage : "<<percentage<<endl;
      }
      else{
        cout<<"No valid data found"<<endl;
      }
      found = 1;
      break;
    }
  }
  if (!found)
  {
    cout << "Student not found" << endl;
  }
}


// Placeholder for access_roll_number function
void access_roll_number()
{
  double total_sgpa=0.0;
  int count=0;
  string roll;
  cout << "Enter roll number : " << endl;
  getline(cin, roll);
  int found = 0;
  for (int j = 0; j < num_students; j++)
  {
    if (roll == students[j].roll_number)
    {
      cout << "Student name : " << students[j].first_name << endl;
      cout << "Student last name : " << students[j].last_name << endl;
      cout << "Student roll number : " << students[j].roll_number << endl;
      cout << "Student sgpa"<< endl;
      for(int s=0;s<8;s++)
      {
        if(students[j].sgpa[s]!=0)
        {
          cout<<"SGPA of the student "<<s+1<<" : "<<students[j].sgpa[s]<<endl;
          total_sgpa+=students[j].sgpa[s];
          count++;
        }
      }
      cout << "Student course id : " << endl;
      for (int c = 0; c < 5; c++) {
        if (students[j].course_id[c] != 0)
        {
```

```
          cout << "Course Id of the Student " << c + 1 << ": "<<
students[j].course_id[c] << endl;

        }
      }

      if(count>0)
      {
        double cgpa=total_sgpa/count;
        cout<<"Student CGPA : "<<cgpa<<endl;
        double percentage=(cgpa-0.75)*10;
        cout<<"Student Percentage : "<<percentage<<endl;
      }
      else{
        cout<<"No valid data found"<<endl;
      }

      found = 1;
      break;
    }
  }
  if (!found)
  {
    cout << "Student not found" << endl;
  }
}


// Placeholder for access_course_id function
void access_course_id()
{
  double total_sgpa=0.0;
  int count=0;
  int course;
  cout << "Enter the course id of the student : " << endl;
  cin >> course;
  cin.ignore(); // Clear input buffer
  int found = 0;
  for (int j = 0; j < num_students; j++)
   {
    for (int c = 0; c < 5; c++)
    {
      if (course == students[j].course_id[c])
      {
        cout << "Student name : " << students[j].first_name << endl;
        cout << "Student last name : " << students[j].last_name << endl;
        cout << "Student Roll Numbr : " << students[j].roll_number << endl;
        cout << "Student sgpa"<< endl;
      for(int s=0;s<8;s++)
```

```
      {
        if(students[j].sgpa[s]!=0)
        {
          cout<<"SGPA of the student "<<s+1<<" : "<<students[j].sgpa[s]<<endl;
          total_sgpa+=students[j].sgpa[s];
          count++;
        }
      }
        cout << "Student course id : " << endl;
        for(int k=0;k<5;k++)
        {
          cout<<"The course id of student
"<<k+1<<":"<<students[j].course_id[k]<<endl;
        }
        if(count>0)
      {
        double cgpa=total_sgpa/count;
        cout<<"Student CGPA : "<<cgpa<<endl;
        double percentage=(cgpa-0.75)*10;
        cout<<"Student Percentage : "<<percentage<<endl;
      }
      else{
        cout<<"No valid data found"<<endl;
      }

      found = 1;
      }
      if(found)
      {
        break;
      }
    }
   }
   if(!found)
   {
    cout<<"Student not found"<<endl;
   }
}


// Placeholder for number_of_students function
void number_of_students()
{
  cout<<"The total number of students are in database is :
"<<num_students<<endl;
}


// Placeholder for update function (implement validation and error handling)
```

```
void update()
{
  string roll;
  cout<<"Enter the roll number of the student whose details you want to update
: "<<endl;
  getline(cin,roll);
  for(int j=0;j<num_students;j++)
  {
    if(roll==students[j].roll_number)
    {
      cout<<"Want to update"<<endl;
      cout<<"1. First name"<<endl;
      cout<<"2. Last name"<<endl;
      cout<<"3. Roll number"<<endl;
      cout<<"4. SGPA"<<endl;
      cout<<"5. Course id"<<endl;
      int choise;
      cin>>choise;

      cin.ignore();

      switch(choise)
      {
        case 1:
              cout<<"Enter the first name you want to update"<<endl;
              getline(cin,students[j].first_name);
              cout<<"Updated successfully"<<endl;
              break;
        case 2:
              cout<<"Enter the last name you want to update"<<endl;
              getline(cin,students[j].last_name);
              cout<<"Updated successfully"<<endl;
              break;
        case 3:
              cout<<"Enter the roll number you want to update"<<endl;
              getline(cin,students[j].roll_number);
              cout<<"Updated successfully"<<endl;
              break;
        case 4:
              cout<<"Enter the sgpa you want to update"<<endl;
              for(int s=0;s<8;s++)
              {
                cout<<"Enter the SGPA of SEM "<<" : "<<s+1<<endl;
                string sgpa_st;
                getline(cin,sgpa_st);
                if(validateNumeric(sgpa_st))
                {
                  students[j].sgpa[s]=stof(sgpa_st);
```

```
                cout<<"Updated successfully"<<endl;
              }
              else
              {
                students[j].sgpa[s]=0;
                cout<<"Invalid input...."<<endl;
              }
              if(students[j].sgpa[s]==0)
              {
                break;
              }
            }
            break;

        case 5:
            cout<<"Enter the course code that you want to update"<<endl;

            for(int c=0;c<5;c++)
            {
              cout<<"Enter the course code os course "<<" : "<<c+1<<endl;
              string course_id_str;
              getline(cin,course_id_str);
              if(validateNumeric(course_id_str))
              {
                students[j].course_id[c]=stoi(course_id_str);
                cout<<"Updated successfully"<<endl;
              }
              else
              {
                students[j].course_id[c]=0;
                cout<<"Invalid input...."<<endl;
              }
              if(students[j].course_id[c]==0)
              {
                break;
              }
            }
            break;
        default:
            cout<<"Invalid input"<<endl;
      }
    }
  }
}


// Placeholder for delete function
void delete_student()
{
```

```
  string roll;
  cout<<"Enter the roll number of the student whose details you want to delete
: "<<endl;
  getline(cin,roll);

  auto it = find_if(students.begin(), students.end(), [&roll](const Student&
s) {
    return s.roll_number == roll;
  });
  if(it !=students.end())
  {
    students.erase(it);
    num_students--;
    cout<<"Student data deleted successfully"<<endl;
  }
  else{
    cout<<"Student not found"<<endl;
  }
}


// Placeholder for display_all_students function
void display_all_students()
{
    double total_sgpa=0.0;
    int count =0;
    cout << "Displaying all students: " << endl;
    for (const auto& student : students)
    {
        cout << "Name: " << student.first_name << " " << student.last_name <<
endl;
        cout << "Roll number: " << student.roll_number << endl;
        for (int i = 0; i < 8; ++i)
        {
            if (student.sgpa[i] != 0)
            {
                cout << "SGPA SEM " << i + 1 << ": " << student.sgpa[i] <<
endl;
                total_sgpa+=student.sgpa[i];
                count++;
            }
        }
        if(count>0)
        {
          double cgpa=total_sgpa/count;
          cout<<"Student cgpa : "<<cgpa<<endl;
          double percentage=(cgpa-0.75)*10;
          cout<<"Student percentage : "<<percentage<<endl;
        }
```

```
        for (int i = 0; i < 5; ++i)
        {
            if (student.course_id[i] != 0)
            {
                cout << "Course ID " << i + 1 << ": " << student.course_id[i]
<< endl;
            }
        }
        cout << "----------------------" << endl;
    }
}
```

C BASIC BANKING SYSTEM

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct Account {
    char username[20];
    char password[20];
    float balance;
    int transaction_count;
    float* transactions;
    struct Account* next;
} Account;

Account* head = NULL;

void create_account() {
    Account* new_account = (Account*) malloc(sizeof(Account));
    printf("Enter username: ");
    scanf("%s", new_account->username);
    printf("Enter password: ");
    scanf("%s", new_account->password);
    new_account->balance = 0.0;
    new_account->transaction_count = 0;
    new_account->transactions = (float*) malloc(sizeof(float) * 10);
    new_account->next = head;
    head = new_account;
    printf("Account created successfully!\n");
}

void login() {
    char username[20];
    char password[20];
    printf("Enter username: ");
    scanf("%s", username);
    printf("Enter password: ");
    scanf("%s", password);
    Account* current = head;
    while (current != NULL) {
        if (strcmp(current->username, username) == 0 && strcmp(current-
>password, password) == 0) {
            printf("Login successful!\n");
            // Perform operations for the logged-in user
            int choice;
            printf("1. Deposit\n");
            printf("2. Withdraw\n");
            printf("3. Check Balance\n");
```

```
            printf("4. Transaction History\n");
            printf("5. Logout\n");
            scanf("%d", &choice);
            switch (choice) {
                case 1:
                    float amount;
                    printf("Enter amount to deposit: ");
                    scanf("%f", &amount);
                    current->balance += amount;
                    current->transactions[current->transaction_count++] =
amount;
                    printf("Deposit successful!\n");
                    break;
                case 2:
                    float amount_withdraw;
                    printf("Enter amount to withdraw: ");
                    scanf("%f", &amount_withdraw);
                    if (current->balance >= amount_withdraw) {
                        current->balance -= amount_withdraw;
                        current->transactions[current->transaction_count++] =
-amount_withdraw;
                        printf("Withdrawal successful!\n");
                    } else {
                        printf("Insufficient balance!\n");
                    }
                    break;
                case 3:
                    printf("Balance: %.2f\n", current->balance);
                    break;
                case 4:
                    printf("Transaction History:\n");
                    for (int i = 0; i < current->transaction_count; i++) {
                        printf("%f\n", current->transactions[i]);
                    }
                    break;
                case 5:
                    printf("Logging out...\n");
                    return;
                default:
                    printf("Invalid choice!\n");
            }
        }
        current = current->next;
    }


}
```

```c
int main() {
    int choice;
    while(1)
    {
        printf("1. Create Account\n");
    printf("2. Login\n");
    printf("3. Exit\n");
    scanf("%d", &choice);
    switch (choice) {
        case 1:
            create_account();
            break;
        case 2:
            login();
            break;
        case 3:
            printf("Exiting...\n");
            return 0;
        default:
            printf("Invalid choice!\n");
    }
    }

    return 0;
}
```