STUDENT MANAGEMENT SYSTEM

Table of Content

1. Overview:

Student Management System (SMS) is a solution tool that is designed to track, maintain and manage all the data generated by a School, including the grades of a student, their attendance, grades etc. I have tried to make a terminal based program facilitating the same.

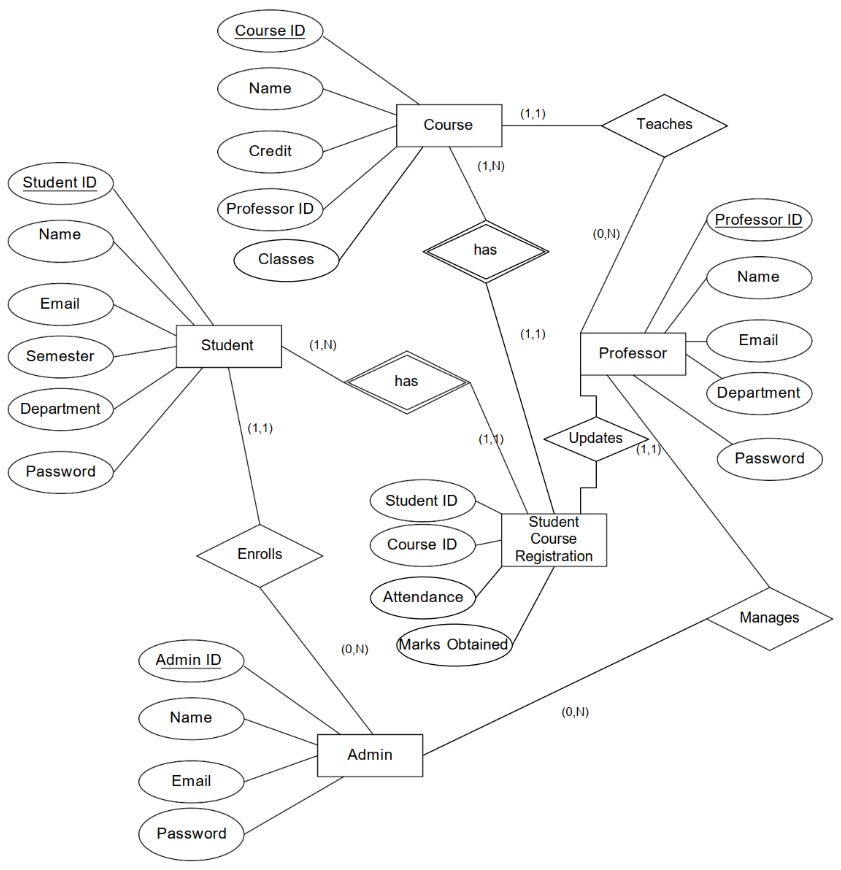
These are entities used in our Student Management System- **Registrar, Student, Professor, Course**.

- **Admin** enrolls and manages students and professors. Each Admin would have an Admin ID, Name, Email, and Password stored in a database.
- A **Student** will have a student id, name, Email, semester, department and a password.
- A **Professor** will have a professor ID, Name, Email, department, and password.
- A **Course** has a course id, name, credit, and Professor ID who is teaching that course.

RELATIONSHIPS:

- A **Course** is taught by a **Professor** and a **Professor** may or may not teach a course.
- A **Student** can be enrolled in one or more **Courses**
- There should be at one and only one **Professor** teaching a **Course .**
- Professor can update attendance and marks of a particular student for a particular course.

I are providing C++ & SQL based program where a user (Registrar, Student and Professors) can login using their ID's and passwords which will be matched from the database. Upon successful login, Admin can onboard students and professors , add a course and register a student to a particular course. Each students can view the offered courses and request registrar to get him registered for the course. Professor updates the attendance of the students for the course they are teaching. Student can view their registered courses, attendance and marks. Professor can also update the attendance .



2. Technology stack used :

i.C++: The project I are delivering is a terminal based project. I are using C++ language to deliver our project.

ii.Sqlite3 : **SQLite** is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine in the world. It is an in-process library and its code is publicly available. It is free for use for any purpose, commercial or private. It is basically an embedded SQL database engine. The following links Ire referred to learn basic of Sqlite3 :

https://www.geeksforgeeks.org/sql-using-c-c-and-sqlite/
https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm
https://www.sqlite.org/cintro.html

iii.SQL : SQL stands for Structured Query Language SQL lets you access and manipulate databases. I am creating a database along with relevant tables and performing SQL operations on the database with the help of C++ language.

3. Project Structure

1.1 Tables

1. **Table Name :** Registrar

   Brief : Creates a registrar table which will store information of all the registrars.

   Attributes : id,name,email,password

```
string regtable = "create table if not exists registrar("
        "id       varchar(20) primary key    not null,"
        "name     varchar(20)         not null,"
        "email    varchar(50) unique not null,"
        "password varchar(30)         not null);";
    int exit = 0;
        char* messaggeError;
        exit = sqlite3_exec(DB, regtable.c_str(), NULL, 0, &messaggeError);

    if (exit != SQLITE_OK)
    {
            cerr << "Error Create Table Registrar" << std::endl;
            sqlite3_free(messaggeError);
    }
```

2. **Table Name :** Professor

   Brief : Creates a professor table which will store information of all the professor.

   Attributes : id,name,email,password,department

```
string proftable = "create table if not exists professor("
    "id       varchar(20) primary key    not null,"
    "name     varchar(20)         not null,"
    "email    varchar(50) unique not null,"
    "password varchar(30)         not null,"
    "department varchar(20) not null);";

    exit = sqlite3_exec(DB, proftable.c_str(), NULL, 0, &messaggeError);

    if (exit != SQLITE_OK)
    {
    cerr << "Error Create Table Student" << std::endl;
    sqlite3_free(messaggeError);
    }
```

3. **Table Name :** Student

Brief : Creates a student table which will store information of all the students.

Attributes : id,name,email,password,semester,department

```
string stutable = "create table if not exists student("
    "id       varchar(20) primary key    not null,"
    "name     varchar(20)        not null,"
    "email    varchar(50) unique not null,"
    "password varchar(30)        not null,"
    "semester int not null ,"
    "department varchar(20) not null);";

    exit = sqlite3_exec(DB, stutable.c_str(), NULL, 0, &messaggeError);

    if (exit != SQLITE_OK)
    {
    cerr << "Error Create Table Student" << std::endl;
    sqlite3_free(messaggeError);
    }
```

4. **Table Name :** Course

Brief : Creates a course table which will store information of all the courses.

Attributes : id,name,credit,prof_id,no.of classes

```
string coursetable = "create table if not exists course("
    "id varchar(20) primary key     not null,"
    "name     varchar(20)        not null,"
    "credit int not null,"
    "prof_id varchar(20) not null,"
    "no_of_classes int not null,"
    "CONSTRAINT fk_prof_id FOREIGN KEY (prof_id) REFERENCES professor(id));";

    exit = sqlite3_exec(DB, coursetable.c_str(), NULL, 0, &messaggeError);

    if (exit != SQLITE_OK)
    {
    cerr << "Error Create Table coursetable" << std::endl;
    sqlite3_free(messaggeError);
    }
```

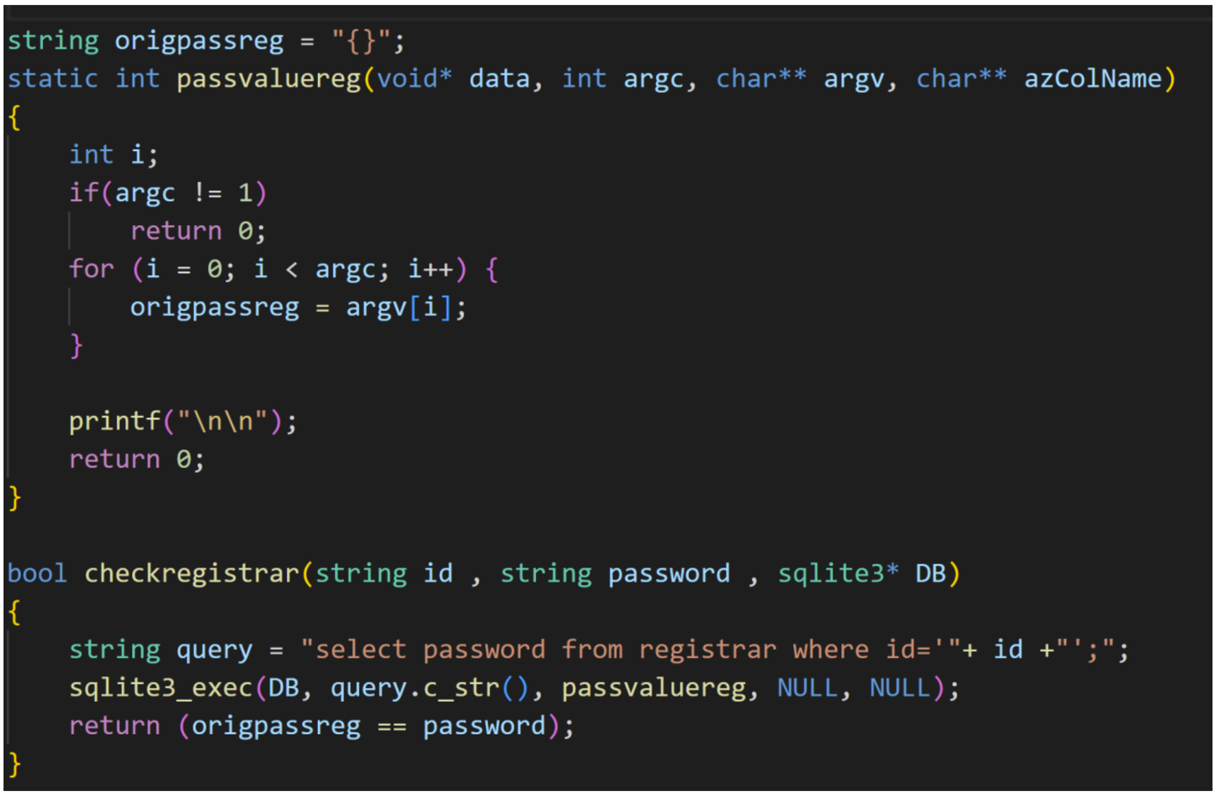
5. **Table Name :** sturegist

Brief : Table which stores the data regarding student registration to different courses , his marks and attendance in the same.

Attributes : courseid,studentid,attendance,marks

```
string sturegtable = "create table if not exists sturegist("
    "courseid varchar(20) not null,"
    "studentid varchar(20) not null,"
    "attendance int not null,"
    "marks int not null,"
    "CONSTRAINT fk_courseid FOREIGN KEY (courseid) REFERENCES  course(id),"
    "CONSTRAINT fk_studentid FOREIGN KEY (studentid) REFERENCES  student(id),"
    "PRIMARY KEY (courseid, studentid));";

    exit = sqlite3_exec(DB, sturegtable.c_str(), NULL, 0, &messaggeError);

    if (exit != SQLITE_OK)
    {
    cerr << "Error Create Table sturegist" << std::endl;
    sqlite3_free(messaggeError);
    }
```

Credentials Check Registrar:



```
string origpassreg = "{}";
static int passvaluereg(void* data, int argc, char** argv, char** azColName)
{
    int i;
    if(argc != 1)
        return 0;
    for (i = 0; i < argc; i++) {
        origpassreg = argv[i];
    }

    printf("\n\n");
    return 0;
}

bool checkregistrar(string id , string password , sqlite3* DB)
{
    string query = "select password from registrar where id='"+ id +"';";
    sqlite3_exec(DB, query.c_str(), passvaluereg, NULL, NULL);
    return (origpassreg == password);
}
```

1.2 Functions

a) Accessed by Registrar

1. Add Registrar

Here, the registrar id, name, email and password are passed as a parameter .This data is being inserted in the table. Here is the code snipet:

```
void addregistrar(string rid ,string rname ,string remail ,string rpass ,sqlite3* DB)
{
   char* messaggeError;
    int exit ;
    string sql = "insert into registrar values('" + rid +"', '" + rname  + "', '"+ remail
+"' , '"+ rpass+"');";

   cout<<"*****************************************************************\n\n";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error Insert" << std::endl;
        sqlite3_free(messaggeError);
    }
    else
        std::cout << "Records created Successfully!" << std::endl;
     cout<<"*****************************************************************\n\n";
}
```

2. Add student

Here, student id, name, email, password, semester and department are passed as a parameter. This data is being inserted in the table.

```
void addstudent(string sid ,string sname ,string semail ,string spass ,string ssem
, string sdepar ,sqlite3* DB)
{
   char* messaggeError;
    int exit ;
    string sql = "insert into student values('" + sid +"', '" + sname  + "', '"+
semail +"' , '"+ spass+"' , "+ssem+" , '" +sdepar +"');";

   cout<<"*****************************************************************\n\n";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error Insert" << std::endl;
        sqlite3_free(messaggeError);
    }
    else
        std::cout << "Records created Successfully!" << std::endl;
     cout<<"*****************************************************************\n\n";
}
```

3. Add Professor

Here, professor id, name, email, password and department is passed as parameter. This data is being inserted in the table. The following code is used to do this process:

```
void addprof(string pid ,string pname ,string pemail ,string ppass , string pdepar
,sqlite3* DB)
{
   char* messaggeError;
    int exit ;
    string sql = "insert into professor values('" + pid +"', '" + pname  + "', '"+
pemail +"' , '"+ ppass+"' , '" +pdepar +"');";

   cout<<"******************************************************************\n\n";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error Insert" << std::endl;
        sqlite3_free(messaggeError);
    }
    else
        std::cout << "Records created Successfully!" << std::endl;
     cout<<"******************************************************************\n\n";

}
```

4. Show all Registrars

This shows all the registrar details to current registrar.

```
void showregistrar(sqlite3* DB)
{
    string query = "select id,name,email from registrar;";
    cout<<"******************************************************************\n\n";
    sqlite3_exec(DB, query.c_str(), callback, NULL, NULL);
    cout<<"******************************************************************\n\n";
}
```

5. Show all students

The current registrar is shown data of all the students:

```
void showstudent(sqlite3* DB)
{
    string query = "select id,name,email,semester,department from student;";
    cout<<"******************************************************************\n\n";
    sqlite3_exec(DB, query.c_str(), callback, NULL, NULL);
    cout<<"******************************************************************\n\n";
}
```

6. **Show all professors**

   Details of all the professors is shown to current registrar.

```
void showprof(sqlite3* DB)
{
    string query = "select id,name,email,department from professor;";
    cout<<"*****************************************************************\n\n";
    sqlite3_exec(DB, query.c_str(), callback, NULL, NULL);
    cout<<"*****************************************************************\n\n";
}
```

7. **Show all courses**

   All the offered courses are shown to the logged in registrar.

```
void showcourses(sqlite3* DB)
{
    string query = "select * from course;";
    cout<<"*****************************************************************\n\n";
    sqlite3_exec(DB, query.c_str(), callback, NULL, NULL);
    cout<<"*****************************************************************\n\n";
}
```

8. **Delete a registrar**

   This function deletes the registrar with a unique rid from the registrar table.

```
void deleteregist(string rid , sqlite3* DB)
{
    int exit;
    char* messaggeError;
    cout<<"*****************************************************************\n\n";
    string sql = "delete from registrar where id = '" + rid+"';";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error DELETE" << std::endl;
        sqlite3_free(messaggeError);
    }
    else
        cout << "Record deleted Successfully!\n";
    cout<<"*****************************************************************\n\n";
}
```

9. **Delete a Student**

Here the registrar deletes a student with a specific sid.

```
void deletestudent(string sid , sqlite3* DB)
{
    int exit;
    char* messaggeError;
    string sql = "delete from student where id = '" + sid+"';";
    cout<<"*****************************************************************\n\n";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error DELETE" << std::endl;

        sqlite3_free(messaggeError);
    }
    else
        std::cout << "Record deleted Successfully!\n";


    cout<<"*****************************************************************\n\n";
}
```

10. **Delete a Professor**

Registrar deletes a professor with a particular pid.

```
void deleteprof(string pid , sqlite3* DB)
{
    int exit;
    char* messaggeError;
    cout<<"*****************************************************************\n\n";
    string sql = "delete from professor where id = '" + pid+"';";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error DELETE" << std::endl;

        sqlite3_free(messaggeError);
    }
    else
        std::cout << "Record deleted Successfully!\n";


    cout<<"*****************************************************************\n\n";
}
```

11. Add course

A new course is added to the course table. Id , name , credit , profid, no_of_classes are passed as parameter.

```
void add_course(string id,string name,string credit,string profid,string nc,sqlite3* DB)
{
    char* messaggeError;
    int exit ;
    string sql = "insert into course values('" + id +"', '" + name+ "', "+ credit +" ,
'"+ profid +"' , " + nc + ");";

   cout<<"*****************************************************************\n\n";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error Insert" << std::endl;
        sqlite3_free(messaggeError);
    }
    else
        std::cout << "Course added Successfully!" << std::endl;
     cout<<"*****************************************************************\n\n";
}
```

12. Add student to a course

A student is enrolled into a course using this function. Initially attendance and marks obtained are set to zero.

```
void add_student_in_course(string cid,string sid,string at,string mr,sqlite3* DB)
{
  char* messaggeError;
    int exit ;

   string sql = "insert into sturegist values('" + cid +"', '" + sid + "', "+ at +" , "+
mr +" );";

   cout<<"*****************************************************************\n\n";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error Insert" << std::endl;
        sqlite3_free(messaggeError);
    }
    else
        std::cout << "Student registered in course Successfully!" << std::endl;
     cout<<"*****************************************************************\n\n";
}
```

b) Accessed by Student

1. Show Attendance

This is used by a student to see his attendance in all the courses.

```
void showattendance(string sid,sqlite3* DB){

        string sql = "select courseid , name as CourseName , attendance , no_of_classes 
from sturegist join course c on c.id = sturegist.courseid where studentid = '"+sid+"';";
        cout<<"*****************************************************************\n\n";
    sqlite3_exec(DB, sql.c_str(), callback, NULL, NULL);
    cout<<"*****************************************************************\n\n";

}
```

2. Show Allotted Courses

The student can see all the courses allocated to him with course as Ill as p

```
void showMycourses(string sid, sqlite3* DB){

      string sql = "select course.id as course_id ,course.name as course_name from 
sturegist inner join course where studentid='" + sid + "';";

      cout<<"*****************************************************************\n\n";
    sqlite3_exec(DB, sql.c_str(), callback, NULL, NULL);
    cout<<"*****************************************************************\n\n";
}
```

3. Show Marks

The student can view his marks obtained in a all courses.

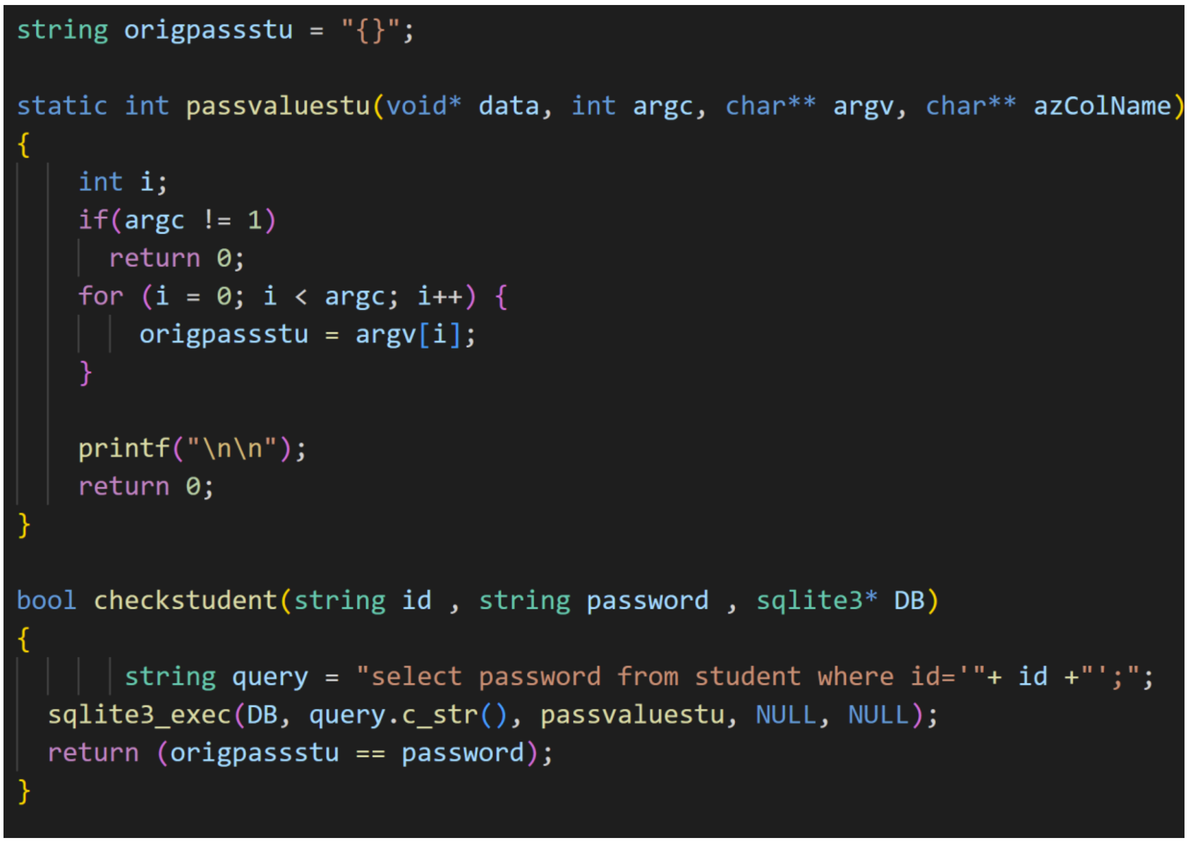
```
void showmarks(string sid,sqlite3* DB){

      string sql = "select courseid , name as CourseName , marks from sturegist join 
course c on c.id = sturegist.courseid where studentid = '"+sid+"';";
      cout<<"*****************************************************************\n\n";
    sqlite3_exec(DB, sql.c_str(), callback, NULL, NULL);
    cout<<"*****************************************************************\n\n";
}
```

4. Show all offered courses

Student can see all the offered courses with details of the professor teaching it.

```
void showOfferedCourses(sqlite3* DB)
{
    string query = "select course.id as CourseID, course.name as CourseName , credit as
CourseCredit , no_of_classes , professor.id as ProfID , professor.name as ProfName ,
professor.email as ProfessorEmail , department  from course join professor on
course.prof_id = professor.id;";
    cout<<"*****************************************************************\n\n";
    sqlite3_exec(DB, query.c_str(), callback, NULL, NULL);
    cout<<"*****************************************************************\n\n";
}
```

Credentials Check Student :



```
string origpassstu = "{}";

static int passvaluestu(void* data, int argc, char** argv, char** azColName)
{
    int i;
    if(argc != 1)
      return 0;
    for (i = 0; i < argc; i++) {
        origpassstu = argv[i];
    }

    printf("\n\n");
    return 0;
}

bool checkstudent(string id , string password , sqlite3* DB)
{
        string query = "select password from student where id='"+ id +"';";
  sqlite3_exec(DB, query.c_str(), passvaluestu, NULL, NULL);
  return (origpassstu == password);
}
```

c) Accessed by Professor

1. Update Attendance

The professor updates the attendance of a particular student in a particular course.

```
void updateattendance(string sid, string cid,string at,sqlite3* DB)
{
       char* messaggeError;
       int exit ;
      string sql = "update sturegist set attendance = "+ at +" where studentid = '" + sid
+ "' and courseid = '" + cid + "';";

      cout<<"*****************************************************************\n\n";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, NULL, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error Update" << std::endl;
        sqlite3_free(messaggeError);
      // throw std::exception();
    }
    else
        std::cout << "Attendance Updated Successfully!" << std::endl;
     cout<<"*****************************************************************\n\n";

}
```

2. Update Marks

Professor updates a students marks in a particular course.

```
void updatemarks(string sid, string cid,string mr,sqlite3* DB)
{
       char* messaggeError;
       int exit ;
      string sql = "update sturegist set marks = " + mr + " where studentid= '" + sid +
"' and courseid= '" + cid + "';";

      cout<<"*****************************************************************\n\n";
    exit = sqlite3_exec(DB, sql.c_str(), 0, NULL, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error Update" << std::endl;
        sqlite3_free(messaggeError);
    }
    else
        std::cout << "Marks Updated Successfully!" << std::endl;
     cout<<"*****************************************************************\n\n";

}
```

3.Finish a Course

Professor can mark a course as finished and remove the course from the sturegist table and course table.

```
 void finishcourse(string cid , sqlite3* DB)
{
    int exit;
    char* messaggeError;
    cout<<"*****************************************************************\n\n";
    string sql = "delete from sturegist where courseid = '" + cid+"';";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error DELETE" << std::endl;
        sqlite3_free(messaggeError);
        return;
    }
    sql = "delete from course where id ='" + cid+"';";
    exit = sqlite3_exec(DB, sql.c_str(), NULL, 0, &messaggeError);
    if (exit != SQLITE_OK) {
        std::cerr << "Error DELETE" << std::endl;
        sqlite3_free(messaggeError);
    }
    else
        std::cout<< "Course finished successfully\n";
    cout<<"*****************************************************************\n\n";
}
```

4.Publish attendance

Professor can publish the final attendance of a particulat course.

```
void publishattendance(string cid , sqlite3* DB)
{
     string sql = "select c.name as CourseName , studentid as StudentID , s.name as
Student_Name , attendance , no_of_classes as TotalClasses from sturegist join course c on
c.id = sturegist.courseid join student s on sturegist.studentid = s.id where c.id = '"+
cid +"';";
        cout<<"*****************************************************************\n\n";
    sqlite3_exec(DB, sql.c_str(), callback, NULL, NULL);
    cout<<"*****************************************************************\n\n";
}
```

5.Publish Marks

Professor publishes the final marks obtained by all the students in a course.

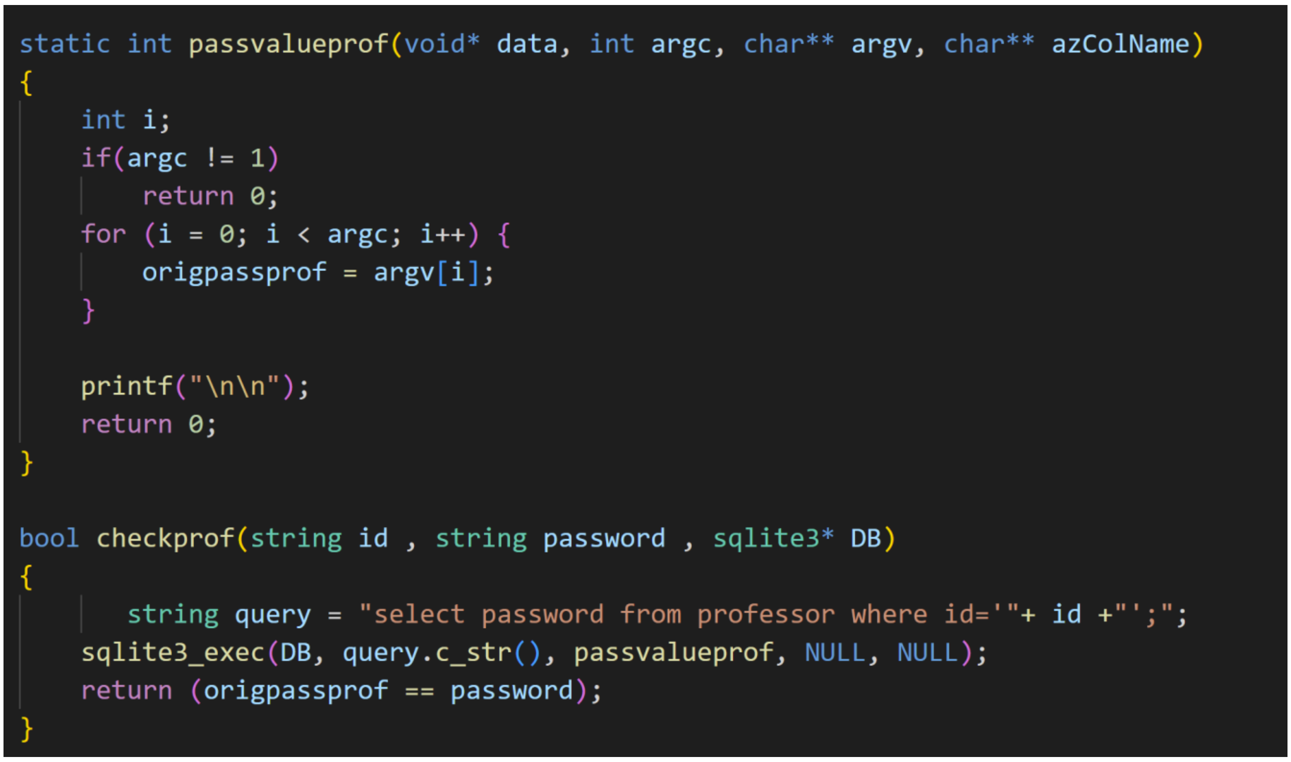
```
void publishmarks(string cid , sqlite3* DB)
{
     string sql = "select c.name as CourseName , studentid as StudentID , s.name as
Student_Name , marks from sturegist join course c on c.id = sturegist.courseid join
student s on sturegist.studentid = s.id where c.id = '"+ cid +"';";
        cout<<"*****************************************************************\n\n";
    sqlite3_exec(DB, sql.c_str(), callback, NULL, NULL);
    cout<<"*****************************************************************\n\n";
}
```

6.Get Enrolled List

Professor gets a list of all the students in his course.

```
void enrolleddlist(string cid , sqlite3* DB)
{
    string sql = "select c.name as CourseName , studentid as StudentID , s.name as
Student_Name from sturegist join course c on c.id = sturegist.courseid join student s on
sturegist.studentid = s.id where c.id = '"+ cid +"';";
        cout<<"*****************************************************************\n\n";
    sqlite3_exec(DB, sql.c_str(), callback, NULL, NULL);
    cout<<"*****************************************************************\n\n";
}
```

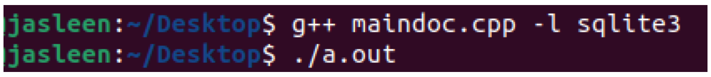
Credentials Check Prof :



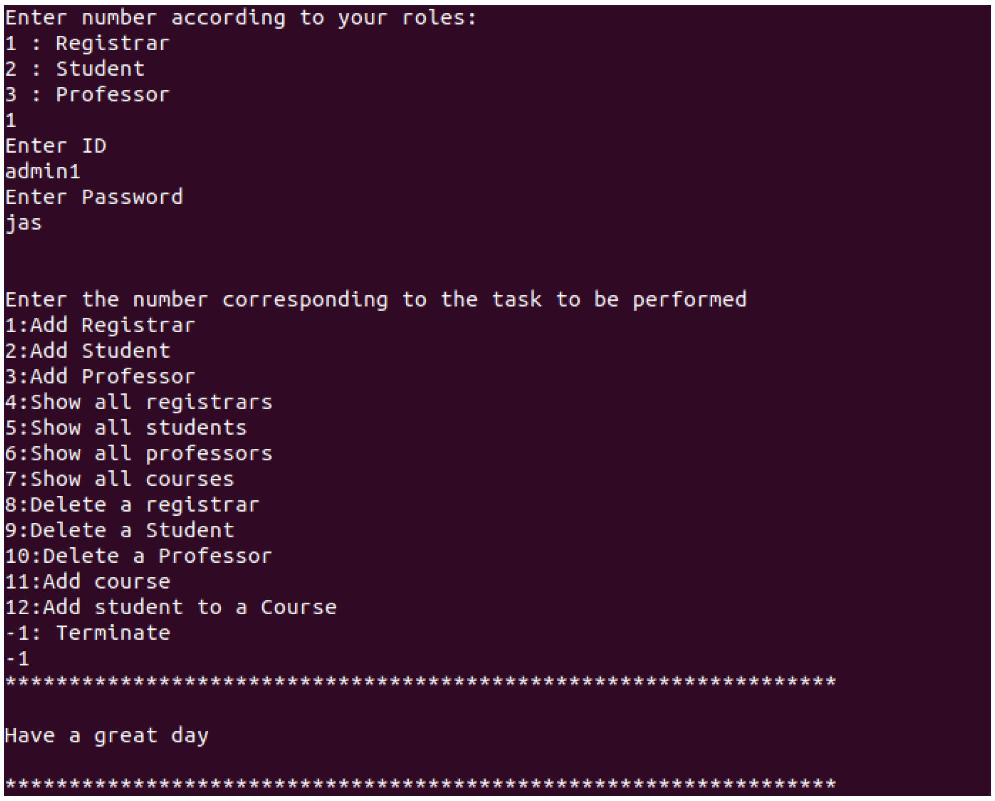
```
static int passvalueprof(void* data, int argc, char** argv, char** azColName)
{
    int i;
    if(argc != 1)
        return 0;
    for (i = 0; i < argc; i++) {
        origpassprof = argv[i];
    }

    printf("\n\n");
    return 0;
}

bool checkprof(string id , string password , sqlite3* DB)
{
        string query = "select password from professor where id='"+ id +"';";
    sqlite3_exec(DB, query.c_str(), passvalueprof, NULL, NULL);
    return (origpassprof == password);
}
```

4. Sample Outputs:

Start the application using:

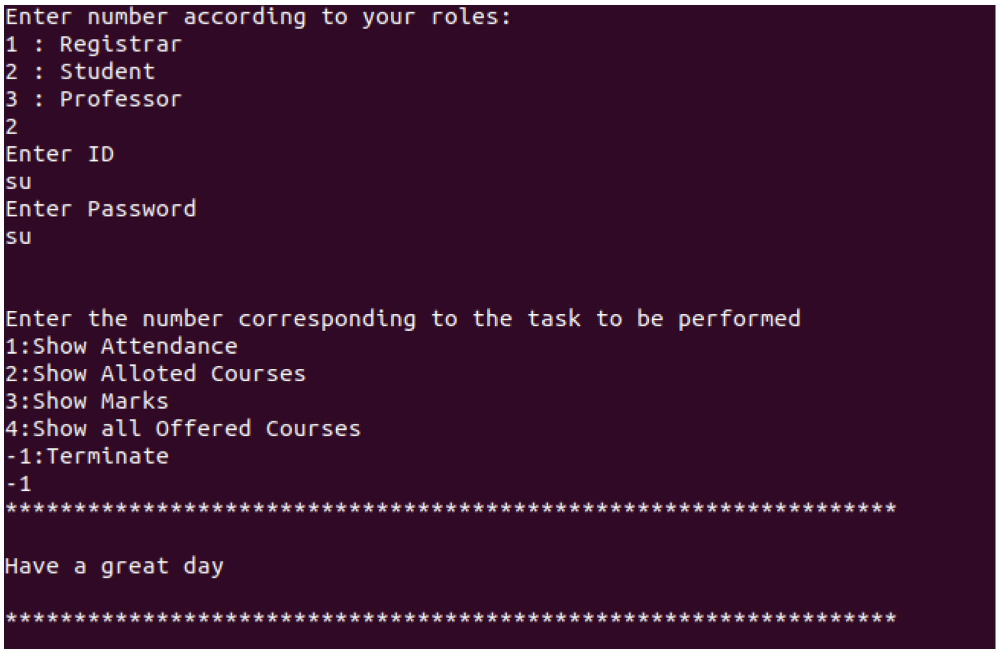


```
jasleen:~/Desktop$ g++ maindoc.cpp -l sqlite3
jasleen:~/Desktop$ ./a.out
```



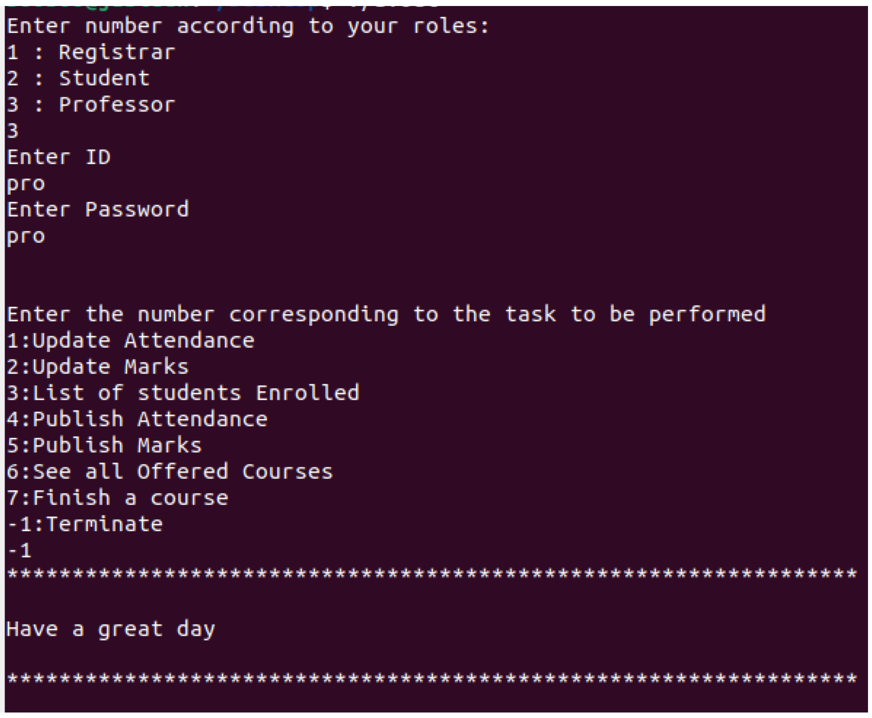
```
Enter number according to your roles:
1 : Registrar
2 : Student
3 : Professor
1
Enter ID
admin1
Enter Password
jas


Enter the number corresponding to the task to be performed
1:Add Registrar
2:Add Student
3:Add Professor
4:Show all registrars
5:Show all students
6:Show all professors
7:Show all courses
8:Delete a registrar
9:Delete a Student
10:Delete a Professor
11:Add course
12:Add student to a Course
-1: Terminate
-1
*****************************************************************

Have a great day

*****************************************************************
```

Registrar Login and Interface :

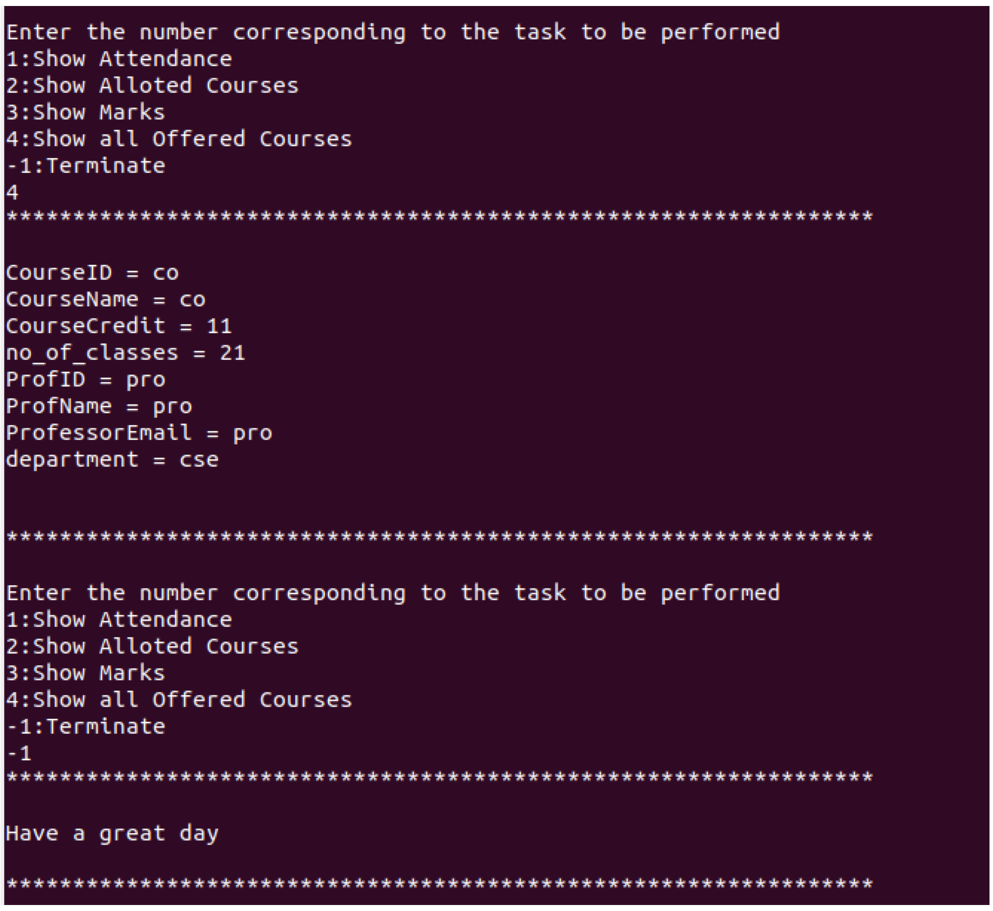


```
Enter number according to your roles:
1 : Registrar
2 : Student
3 : Professor
2
Enter ID
su
Enter Password
su


Enter the number corresponding to the task to be performed
1:Show Attendance
2:Show Alloted Courses
3:Show Marks
4:Show all Offered Courses
-1:Terminate
-1
*****************************************************************

Have a great day

*****************************************************************
```

Student Login Page and Interface :


Professor Login Page and Interface:

```
Enter number according to your roles:
1 : Registrar
2 : Student
3 : Professor
3
Enter ID
pro
Enter Password
pro


Enter the number corresponding to the task to be performed
1:Update Attendance
2:Update Marks
3:List of students Enrolled
4:Publish Attendance
5:Publish Marks
6:See all Offered Courses
7:Finish a course
-1:Terminate
-1
*****************************************************************

Have a great day

*****************************************************************
```


See all Offered Courses :

```
Enter the number corresponding to the task to be performed
1:Show Attendance
2:Show Alloted Courses
3:Show Marks
4:Show all Offered Courses
-1:Terminate
4
*****************************************************************

CourseID = co
CourseName = co
CourseCredit = 11
no_of_classes = 21
ProfID = pro
ProfName = pro
ProfessorEmail = pro
department = cse


*****************************************************************

Enter the number corresponding to the task to be performed
1:Show Attendance
2:Show Alloted Courses
3:Show Marks
4:Show all Offered Courses
-1:Terminate
-1
*****************************************************************

Have a great day

*****************************************************************
```

5. Conclusion and Future Enhancements:

- Support for individual user to update his details.
- Support add minimum percentage attendance support missing which you will be given zero marks and hence deemed fail.
- Allocation of grades on the basis of marks obtained.
- Support for GPA calculation on the basis of marks and credits.

After completing the project I successfully implemented terminal based interface for registrar , professor and students who are the entities of our model. Have also added all the basic functionalities of all the entities with support of error catching. I used SQLite3 which is an embedded SQL Library and integrated it with C++ to get the deliver the product required.