Implementation of Simple and Efficient Student Database Management System

Srikanth Ramavath[1], Dr.Sambasiva Rao Baragada[2]
[1]*Student, III B.Sc. (MPCs), Department of Computer Science, BJR Government Degree College Hyderabad*
[2]*Assistant Professor of Computer Science, Department of Computer Science, BJR Government Degree College Hyderabad*

Abstract **- Advanced database management systems evolve rapidly, consuming trillions of bytes of data being stored in servers, A simple database system with equally potential could be developed in C++ with text files for storage. This paper focusses in implementing a simple and efficient student database management system.**

Index Terms **- Database, student database.**

INTRODUCTION

An organized and systematic office solution is essential for all educational institutions namely college and universities and organizations. There are many departments of administration for the maintenance of college information and student databases in any institution. All these departments provide various records regarding students. Most of these track records need to maintain information about the students. This information could be the general details like student name, address, performance, attendance etc. or specific information related to departments like collection of data. All the modules in college administration are interdependent. They are maintained manually. The system is needed to be automated and centralized as, Information from one module will be needed by other modules. The paper is aimed to develop a simple and efficient student database management system.

METHODOLOGY

Student databases are more or less have similar schema. A conventional student database is administered through various roles. The proposed system comprised of various views of users according to their roles namely student, faculty, proctor and administrator. The proposed system holds both student details as well as student academic marks. Data is stored in external text files. The system is implemented in C++ and primarily aimed to be simple and efficient. The proposed schema for student database is shown in figure 1.

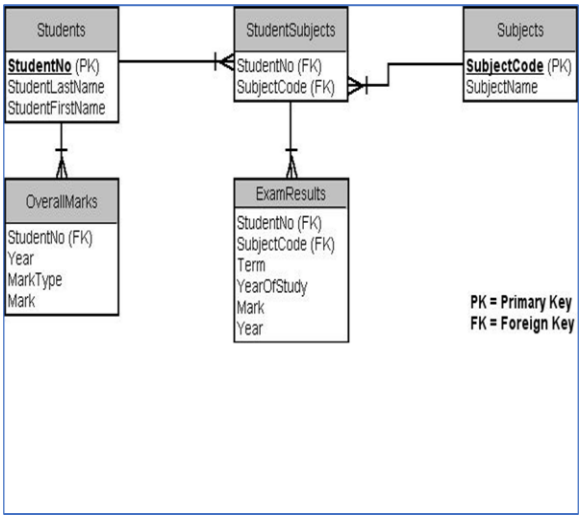

Fig.1: Schema of Student Database System

The system maintains takes student details, faculty details and proctor details along with their marks. Various users have different roles while accessing the data. Class diagram for the proposed system in presented in figure 2.Collaboration diagram of the entities is depicted in figure 3. The numbered sequence of while accessing the system include login, Login, request access, allow access, display, view details, logout, login, request access, allow access, display, enter profile, enter mark, provide data, logout, store data, update data.

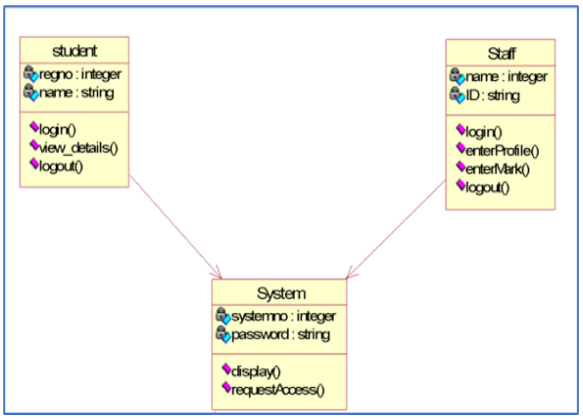

Fig. 2: Class Diagram

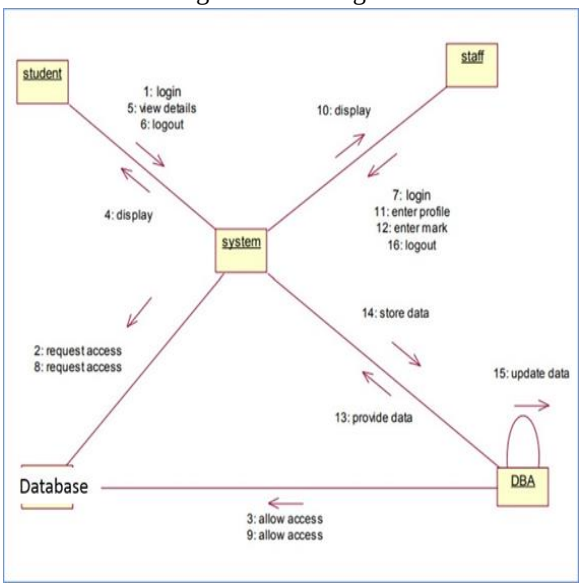

Fig.3: Collaboration Diagram

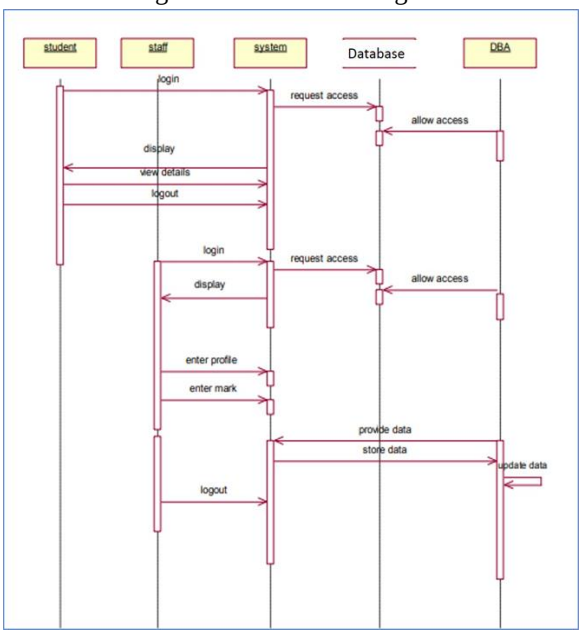

Fig. 4: Sequence Diagram

The sequence of interaction between entities within the system is presented in figure 4. Student and staff details are provided to the system. System takes the responsibility of storing and retrieving the data from the database. DBA or administrator takes supervision of entire system. Use case diagram of the system is illustrated in figure 5.

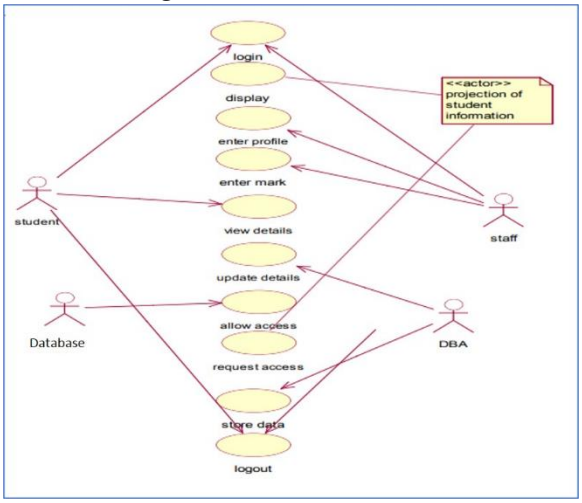

Fig. 5: Use Case Diagram

IMPLEMENTATION

Following code is the simple implementation of Student Management Project written in C++

```
// Include all the necessary libraries.
#include <fstream>
#include <iostream>
#include <stdio.h>
#include <string.h>
usingnamespacestd;
intmain()
{
   // Considering the max length of data entered (name) to
   // be 15.
   chardata[15];
   intn = 0, option = 0, count_n = 0;
   // This is the initial mark alloted to a subject.
   string empty = "00";
   string proctor = "";
   // Name of the file in which DB is stored.
   ifstream f("Example.txt");
   string line;
   // The following for loop counts the total number of
   // lines in the file.
```

```
for(inti = 0; std::getline(f, line); ++i) {
   count_n++;
}
while(option != 6) {
   // This prints out all the available options in the
   // DB
   cout << "\nAvailable operations: \n1. Add New "
         "Students\n2."
      << "Student Login\n3. Faculty Login\n4. "
         "Proctor Login\n5. Admin View\n"
      << "6. Exit\nEnter option: ";
   cin >> option;
   if(option == 1) {
      cout << "Enter the number of students: ";
      cin >> n;
      count_n = count_n + n;
       for(inti = 0; i < n; i++) {
         ofstream outfile;
         outfile.open("Example.txt", ios::app);
         // The entire data of a single student is
         // stored line-by-line.
         cout << "Enter your registration number: ";
         cin >> data;
         outfile << data << "\t";
         cout << "Enter your name: ";
         cin >> data;
         intlen = strlen(data);
         while(len < 15) {
            data[len] = ' ';
            len = len + 1;
         }
         outfile << data << "\t";
         // Inserting empty data initially into the
         // file
         outfile << empty << "\t";
         outfile << empty << "\t";
         cout << "Enter your proctor ID: ";
         cin >> proctor;
         outfile << proctor << endl;
      }
   }
   elseif(option == 2) {
      charregno[9];
      cout << "Enter your registration number: ";
      cin >> regno;
      ifstream infile;
      intcheck = 0;
      infile.open("Example.txt", ios::in);
      // This loop prints out the data according to
      // the registration number specified.
      while(infile >> data) {
         if(strcmp(data, regno) == 0) {
            cout
               << "\nRegistration Number: "<< data
               << endl;
            infile >> data;
            cout << "Name: "<< data << endl;
            infile >> data;
            cout << "CSE1001 mark: "<< data
                 << endl;
            infile >> data;
            cout << "CSE1002 mark: "<< data
                 << endl;
            infile >> data;
            cout << "Proctor ID: "<< data << endl;
            infile.close();
            check = 1;
         }
      }
      if(check == 0) {
         cout << "No such registration number
found!"
              << endl;
      }
   }
   // This loop is used to view and add marks to the
   // database of a student.
   elseif(option == 3) {
      charsubcode[7];
      cout << "Enter your subject code: ";
      cin >> subcode;
      string code1 = "CSE1001", code2 =
"CSE1002",
             mark = "";
      ifstream infile;
      intcheck = 0;

      cout << "\nAvailable operations: \n1. Add data
"
           "about marks\n"
         << "2. View data\nEnter option: ";
      cin >> option;
      if(option == 1) {
         cout
            << "Warning! You would need to add
mark"
            << "details for all the students!"
            << endl;
```

```
    for(inti = 0; i < count_n; i++) {
        fstream file("Example.txt");
        // The seek in file has been done
        // according to the length
        // of the data being inserted. It needs
        // to adjusted accordingly for different
        // lengths of data.
        if(strcmp(subcode, code1.c_str())
            == 0) {
            file.seekp(26 + 37 * i,
                    std::ios_base::beg);
            cout << "Enter the mark of student#"
                << (i + 1) << " : ";
            cin >> mark;
            file.write(mark.c_str(), 2);
        }
        if(strcmp(subcode, code2.c_str())
            == 0) {
            file.seekp(29 + 37 * i,
                    std::ios_base::beg);
            cout << "Enter the mark of student#"
                << (i + 1) << " : ";
            cin >> mark;
            file.write(mark.c_str(), 2);
        }
    }
}
// This loop is used to view marks of a student.
// The extra infile commands have been used to
// get a specific mark only since the data has
// been seperated by a tabspace.
elseif(option == 2) {
    infile.open("Example.txt", ios::in);
    if(strcmp(subcode, code1.c_str()) == 0) {
        cout << "Registration number - Marks\n"
            << endl;
        while(infile >> data) {
            cout << data;
            infile >> data;
            infile >> data;
            cout << " - "<< data << endl;
            infile >> data;
            infile >> data;
            check = 1;
        }
    }
    infile.close();
    infile.open("Example.txt", ios::in);
    if(strcmp(subcode, code2.c_str()) == 0) {
        cout << "Registration number - Marks\n"
            << endl;
        while(infile >> data) {
            cout << data;
            infile >> data;
            infile >> data;
            infile >> data;
            cout << " - "<< data << endl;
            infile >> data;
            check = 1;
        }
    }
    infile.close();
    if(check == 0) {
        cout << "No such subject code found!"
            << endl;
    }
}
// This loop displays all the details of students
// under the same proctor ID.
elseif(option == 4) {
    charprocid[7];
    cout << "Enter your proctor ID: ";
    cin >> procid;
    intcheck = 0;
    chartemp1[100], temp2[100], temp3[100];
    chartemp4[100], id[100];
    ifstream infile;
    infile.open("Example.txt", ios::in);
    while(infile >> temp1) {
        infile >> temp2;
        infile >> temp3;
        infile >> temp4;
        infile >> id;
        if(strcmp(id, procid) == 0) {
            cout << "\nRegistration Number: "
                << temp1 << endl;
            cout << "Name: "<< temp2 << endl;
            cout << "CSE1001 Mark: "<< temp3
                << endl;
            cout << "CSE1002 Mark: "<< temp4
                << endl;
            check = 1;
        }
    }
    if(check == 0) {
        cout << "No such proctor ID found!"<<
endl;
```

```
        }
      }
      // This loop acts as an admin view to see all the
      // data in the file.
       elseif(option == 5) {
        charpassword[25];
        cout << "Enter the admin password: ";
        cin >> password;
         // This variable value can be changed according
        // to your requirement of the administrator
        // password.
         string admin_pass = "admin";
        if(strcmp(password, admin_pass.c_str()) == 0) {
           cout << "Reg No.       "
                "\tName\tCSE1001\tCSE1002\tProctor "
                "ID"
               << endl;
           ifstream infile;
           infile.open("Example.txt", ios::in);
           chardata[20];
            while(infile >> data) {
              cout << data << "\t";
              infile >> data;
              cout << data << "\t";
              infile >> data;
              cout << data << "\t";
              infile >> data;
              cout << data << "\t";
              infile >> data;
              cout << data << endl;
           }
        }
      }
   }
```

Output:

```
Available operations:
1. Add New Students
2. Student Login
3. Faculty Login
4. Proctor Login
5. Admin View
6. Exit
Enter option: 1
Enter the number of students: 2
Enter your registration number: 107018468008
Enter your name: Sreekanth
Enter your proctor ID: 1001
Enter your registration number: 107018468009
Enter your name: Gopal
Enter your proctor ID: 1002

Available operations:
1. Add New Students
2. Student Login
3. Faculty Login
4. Proctor Login
5. Admin View
6. Exit
Enter option: 3
Enter your subject code: CS501

Available operations:
1. Add data about marks
2. View data
Enter option: 1
Warning! You would need to add mark details for all the students!
Enter the mark of student#1 : 52
Enter the mark of student#2 : 89
No such subject code found!

Available operations:
1. Add New Students
2. Student Login
3. Faculty Login
4. Proctor Login
5. Admin View
6. Exit
Enter option: 5
Enter the admin password: admin
Reg No.       Name   CSE1001   CSE1002   Proctor ID
15BCE2083   Dheeraj   52   00   1001
15BCE2082   Rohan   89   00   1002

Available operations:
1. Add New Students
2. Student Login
3. Faculty Login
4. Proctor Login
5. Admin View
6. Exit
Enter option: 6
```

RESULTS AND DISCUSSION

User has 5 kinds of options namely, add new students, student login, faculty login, proctor login and admin view respectively. With the student login, user can provide data pertaining to student name, his proctor id, registration number. Faculty login can provide academic marks for students. Admin has the provision to supervise the entire management system. Following is the sequence of output of the system. The system is simple to build with limited requirements and it takes external text files for storage. The proposed system is effective and efficient conformed with basic functionalities.

REFERENCES

[1] C Projects, Yashwant Kanetkar.
[2] C++ Projects, Reeta Sahoo.