**STUDENT DATABASE MANAGEMENT SYSTEM**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfilment of the requirements to award the degree of

**Bachelor of Technology/Master of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**K. Venu (AP23110011447)**

**V. Sai Pavan (AP23110011453)**

**Y. Venu Gopal (AP23110011440)**

**N. Scanny** **(AP23110011414)**



Under the Guidance of

**Mrs. Karnena Kavitha Rani**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**November,2025**

# Certificate

Date: 16-Nov-22

This is to certify that the work present in this Project entitled “**Student Database Management System**” has been carried out by group 14 **under** my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

**Supervisor**

(Signature)

Prof. / Dr. **Karnena Kavitha Rani**

Designation, Affiliation.

**Co-supervisor**

(Signature)

Prof. / Dr. [Name]

Designation, Affiliation.

**Acknowledgements**

We would like to express our gratitude **Mrs.** **Karnena Kavitha Rani** for assisting us with the research at every step. Without the advice and assistance of our adviser, this study effort would not be feasible. The suggestions, feedbacks, and encouragement from our advisors were extremely helpful in assisting us in finishing this research.

**Abstract**

Abstract: Student Database Management System in C++

**1. Introduction**

This program is a Student Database Management System (SDMS) written in C++. It is designed to manage student records efficiently, including their ID, name, age, and marks. The program supports two types of users: Admin and Student, each with specific roles and permissions.

**2. Features**

* **For Admins**:
* Add Student: Add new student records.
* Display All Students: View all student details in a table format.
* Search Student by ID: Find a specific student using their ID.
* Delete Student by ID: Remove student records.
* Update Student by ID: Edit student age and marks.
* Save and Exit: Save all changes to a file for future use.
* **For Students**:
* Display All Students: View a list of all student records.
* Search Student by ID: Check individual student details.
* Exit: Leave the system without making changes.

**3. Key Concepts in the Program**

* **Object-Oriented Programming (OOP)**: The Student class manages student data, encapsulating attributes and methods.
* **File Handling**: Reads and writes student records to a file (students.txt) to ensure data persistence.
* **Role-Based Access Control**: Admins have advanced features like adding, deleting, and updating records, while students have limited access.
* **Error Handling**: Prevents duplicate student IDs and handles invalid inputs gracefully.
* **Dynamic Data Storage**: Uses the vector container to store student records in memory.

**4. User Roles and Security**

* Admin Role: Requires a password (admin123) for secure access to all features.
* Student Role: Provides read-only access to the database.

**5. Conclusion**

This system simplifies the management of student records by automating common tasks. It also showcases important C++ concepts like classes, file handling, and vectors, making it an excellent example for learning practical programming and data management techniques.

**Methodology**

Developing the Student Database Management System

The following steps describe how the program was developed, tested, and implemented:

**1. Understanding Requirements**

Identify key tasks for managing student data, such as adding, deleting, updating, and viewing records.

Design separate roles for Admin and Student users, each with specific permissions.

**2. Designing the System**

* Class Design: Create a student class to hold attributes like ID, name, age, and marks, with methods for displaying and updating data.
* Menu System: Design user-friendly menus for Admins and Students, offering options tailored to their roles.
* File Handling: Plan for saving and loading data to ensure that records are persistent between sessions.

**3. Implementing Features**

**Core Functionalities:**

* Add a student with unique ID validation.
* View all student records in a tabular format.
* Search for a student using their ID.
* Delete or update student details based on Admin input.
* Role-Based Access: Secure Admin features with a password and limit Student access to view-only functionalities.
* File Operations: Use file streams to read and write student data to a text file (students.txt).

**4. Coding and Development**

Use C++ programming to implement the system:

Use vectors to store student records dynamically.

Employ functions for modularity and code reuse.

Include input validation to prevent errors, like duplicate IDs or invalid inputs.

**5. Testing the System**

Test each feature individually to ensure it works as intended.

Validate edge cases, such as:

Adding a duplicate student ID.

Searching for non-existent student IDs.

Handling empty records gracefully.

Test role-based access to ensure Admin and Student permissions work correctly.

**6. Deployment and Usage**

Provide the program in an executable format.

Ensure the system loads existing data from the file on startup and saves changes before exiting.

Offer clear instructions for Admins and Students to interact with the system.

This methodology ensures a structured approach to developing and maintaining the Student Database Management System.

**DISCUSSION**

1. **INPUT OF CODE**

#include<iostream>

#include<iomanip>

#include<fstream>

#include<string>

#include<vector>

#include<limits>

#include<sstream>

using namespace std;

class student{

private:

int age;

string id,name,father\_name,contact\_No,address;

double cgpa; //Student Details and their attributes.

public:

student(){} //default Constructor.

student(int ag,string i,string n,string f,string y,double z,string a):age(ag),id(i),name(n),father\_name(f),contact\_No(y),cgpa(z),address(a){} //Parameterized Constructor.

//Method functions.

string getid(){

return id;

}

double get\_result(){

return cgpa;

}

string getname(){

return name;

}

string getaddress(){

return address;

}

int getage(){

return age;

}

string getcontact(){

return contact\_No;

}

string getfathername(){

return father\_name;

}

void display(){

cout<<setw(10)<<id<<" "<<setw(20)<<name<<setw(5)<<age<<setw(20)<<father\_name<<setw(15)<<contact\_No<<setw(5)<<cgpa<<setw(30)<<address<<endl;

}

void setage(int x){

age=x;

}

void setcgpa(double x){

cgpa=x;

}

};

void menu(bool isadmin){

cout<<endl<<endl;

cout<<setw(15)<<"Student Database Management System"<<endl;

cout<<string(50,'-')<<endl;

if(isadmin){

cout<<"1-Add student"<<endl;

cout<<"2-display student details"<<endl; //admin user to prefer

cout<<"3-search student using ID"<<endl;

cout<<"4-delete student using ID"<<endl;

cout<<"5-Update student"<<endl;

cout<<"6-Clear the data from Saved file"<<endl;

cout<<"7-Save the file and exit"<<endl<<endl;

}else{

cout<<"1-display student details"<<endl; //student user to prefer

cout<<"2-exit"<<endl<<endl;

}

cout<<"Enter the choice:"<<endl;

}

void savetofile(vector<student> &students,string file){

ofstream obj(file,ios::out);

for(int i=0;i<students.size();i++){

obj<<students[i].getid()<<" , "<<students[i].getname()<<" , "<<students[i].getage()<<" , "<<students[i].get\_result()<<" , "<<students[i].getfathername()<<" , "<<students[i].getcontact()<<" , "

<<students[i].getaddress();

}

cout<<"Data saved successfully.."<<endl;

}

void loadfromfile(vector<student> &students,string file){

ifstream obj(file);

if(!obj){

cout<<endl<<"No existing records are found..!"<<endl;

return;

}else{

int age;

string id,name,father\_name,contact\_No,address,line;

double cgpa;

while(getline(obj,line)){

stringstream ss(line);

string convert;

getline(ss,id,',');

getline(ss,name,',');

getline(ss,convert,',');age=stoi(convert);

getline(ss,convert,',');cgpa=stod(convert);

getline(ss,father\_name,',');

getline(ss,contact\_No,',');

getline(ss,address,',');

students.emplace\_back(age,id,name,father\_name,contact\_No,cgpa,address);

}

cout<<"Data loaded successfully..!"<<endl;

}

}

void clearfile(string file, vector<student> &students) {

ofstream obj(file, ios::trunc); // Open file in truncate mode

if (obj) {

students.clear(); // Clear the in-memory student list

cout << "All data deleted successfully from the file and memory." << endl;

} else {

cout << "Error: Unable to clear the file." << endl;

}

}

int main(){

vector<student> students;

string pass,id;

int age;

double cgpa;

cout<<"Enter the role admin (or) Student: "<<endl;

string role;

cin>>role;

bool isadmin=false;

if(role=="admin"){

cout<<"Enter password:";

cin>>pass;

if(pass=="Srmap123"){

cout<<endl<<"Admin login successful!"<<endl;

isadmin=true;

}else{

cout<<"Incorrect password"<<endl;

}

}else if(role=="student"){

cout<<"Student login succesful!"<<endl;

}else{

cout<<"Login failed"<<endl;

}

string file="students.txt";

loadfromfile(students,file);

int choice;

do{

menu(isadmin);

if(isadmin){

cin>>choice;

switch(choice){

case 1:{

string name,father\_name,contact\_No,address;

cout<<"Enter student id: ";

cin>>id;

for(int i=0;i<students.size();i++){

if(students[i].getid()==id){

cout<<"ID already exits.Try again!"<<endl;

break;

}

}

cin.ignore();

cout<<endl<<"Enter Student Name: ";

getline(cin,name);

cout<<endl<<"Enter Student age: ";

cin>>age;

cout<<endl<<"Enter student's Father name: ";

cin.ignore();

getline(cin,father\_name);

cout<<endl<<"Enter student's contact info: ";

cin>>contact\_No;

cout<<endl<<"Enter cgpa: ";

cin>>cgpa;

cin.ignore();

cout<<endl<<"Enter address: ";

getline(cin,address);

students.emplace\_back(age,id,name,father\_name,contact\_No,cgpa,address);

cout<<"Student added successfully!"<<endl;

break;

}

case 2:{

if(students.empty()){

cout<<"No records are found."<<endl;

}else{

cout<<endl;

cout<<left<<setw(5)<<"Sl.NO "<<setw(10)<<"ID"<<setw(20)<<"Name"<<setw(5)<<"Age"<<setw(20)<<"Father Name"<<setw(15)<<"Contact info"<<setw(5)<<" CGPA "<<setw(30)<<"Student's Address"<<endl;

cout<<string(110,'-')<<endl;

for(int i=0;i<students.size();i++){

cout<<left<<setw(5)<<i+1;

students[i].display();

}

}

break;

}

case 3:

{

cout<<"Enter the student id to search: ";

cin>>id;

cout<<endl;

bool found=false;

for(int i=0;i<students.size();i++){

if(students[i].getid()==id){

found=true;

cout<<" Student details Found: "<<endl;

cout<<left<<setw(5)<<"Sl.NO "<<setw(10)<<"ID"<<setw(20)<<"Name"<<setw(5)<<"Age"<<setw(20)<<"Father Name"<<setw(15)<<"Contact info"<<setw(5)<<" CGPA "<<setw(30)<<"Student's Address"<<endl;

students[i].display();

}

}

if(!found){

cout<<"student with the id doesn't Exist."<<endl;

}

break;

}

case 4:

{

cout<<"Enter the student id to delete: ";

cin>>id;

cout<<endl;

bool found=false;

for(int i=0;i<students.size();i++){

if(students[i].getid()==id){

found=true;

for(int j=i;j<students.size()-1;j++){

students[j]=students[j+1];

}

students.pop\_back(); //Remove the last duplicate element.

cout<<"Student Record deleted successfully."<<endl;

break;

}

}

if(!found){

cout<<"student with the id doesn't Exist."<<endl;

}

break;

}

case 5:

{

cout<<"enter the id to update: ";

cin>>id;

cout<<endl;

bool found=false;

for(int i=0;i<students.size();i++){

if(students[i].getid()==id){

cout<<"Update age: ";

cin>>age;

cout<<endl;

cout<<"Update cgpa: ";

cin>>cgpa;

cout<<endl;

students[i].setage(age);

students[i].setcgpa(cgpa);

found=true;

cout<<"Student details updated successfully!"<<endl;

}

}

if(!found){

cout<<"student with the id doesn't Exist."<<endl;

}

break;

}

case 6:

clearfile(file,students);

break;

case 7:

savetofile(students,file);

break;

default:

cout<<"Invalid choice....Please try again!"<<endl;

}

}else{

cin>>choice;

switch(choice){

case 1:

{

cout<<left<<setw(5)<<"Sl.NO "<<setw(10)<<"ID"<<setw(20)<<"Name"<<setw(5)<<"Age"<<setw(20)<<"Father Name"<<setw(15)<<"Contact info"<<setw(5)<<" CGPA "<<setw(30)<<"Student's Address"<<endl;

cout<<string(110,'-')<<endl;

for(int i=0;i<students.size();i++){

cout<<left<<setw(5)<<i+1;

students[i].display();

}

break;

}

case 2:

cout<<"Exited..."<<endl;

break;

default:

cout<<"Invalid choice....Please try again!"<<endl;

}

}

}while((isadmin &&choice!=7) || (!isadmin && choice!=2));

return 0;

}

1. OUTPUT OF CODE

Enter the role admin (or) Student:

admin

Enter password:Srmap123

Admin login successful!

Data loaded successfully..!

Student Database Management System

--------------------------------------------------

1-Add student

2-display student details

3-search student using ID

4-delete student using ID

5-Update student

6-Clear the data from Saved file

7-Save the file and exit

Enter the choice:

6

All data deleted successfully from the file and memory.

Student Database Management System

--------------------------------------------------

1-Add student

2-display student details

3-search student using ID

4-delete student using ID

5-Update student

6-Clear the data from Saved file

7-Save the file and exit

Enter the choice:

1

Enter student id: AP23110011447

Enter Student Name: k.venu

Enter Student age: 19

Enter student's Father name: k.nagarju

Enter student's contact info: 9390388035

Enter cgpa: 8.84

Enter address: bethamcherla

Student added successfully!

Student Database Management System

--------------------------------------------------

1-Add student

2-display student details

3-search student using ID

4-delete student using ID

5-Update student

6-Clear the data from Saved file

7-Save the file and exit

Enter the choice:

1

Enter student id: AP231100114XX

Enter Student Name: student2

Enter Student age: 22

Enter student's Father name: father2

Enter student's contact info: 9345XXX673

Enter cgpa: 8.83

Enter address: kurnool

Student added successfully!

Student Database Management System

--------------------------------------------------

1-Add student

2-display student details

3-search student using ID

4-delete student using ID

5-Update student

6-Clear the data from Saved file

7-Save the file and exit

Enter the choice:

2

Sl.NO ID Name Age Father Name Contact info CGPA Student's Address

--------------------------------------------------------------------------------------------------------------

1 AP23110011447 k.venu 19 k.nagarju 9390388035 8.84 bethamcherla

2 AP231100114XX student2 22 father2 9345XXX673 8.83 kurnool

Student Database Management System

--------------------------------------------------

1-Add student

2-display student details

3-search student using ID

4-delete student using ID

5-Update student

6-Clear the data from Saved file

7-Save the file and exit

Enter the choice:

3

Enter the student id to search: AP231100114XX

Student details Found:

Sl.NO ID Name Age Father Name Contact info CGPA Student's Address

AP231100114XX student2 22 father2 9345XXX673 8.83 kurnool

Student Database Management System

--------------------------------------------------

1-Add student

2-display student details

3-search student using ID

4-delete student using ID

5-Update student

6-Clear the data from Saved file

7-Save the file and exit

Enter the choice:

4

Enter the student id to delete: AP23110011447

Student Record deleted successfully.

Student Database Management System

--------------------------------------------------

1-Add student

2-display student details

3-search student using ID

4-delete student using ID

5-Update student

6-Clear the data from Saved file

7-Save the file and exit

Enter the choice:

2

Sl.NO ID Name Age Father Name Contact info CGPA Student's Address

--------------------------------------------------------------------------------------------------------------

1 AP231100114XX student2 22 father2 9345XXX673 8.83 kurnool

Student Database Management System

--------------------------------------------------

1-Add student

2-display student details

3-search student using ID

4-delete student using ID

5-Update student

6-Clear the data from Saved file

7-Save the file and exit

Enter the choice:

5

enter the id to update: AP231100114XX

Update age: 23

Update cgpa: 8.85

Student details updated successfully!

Student Database Management System

--------------------------------------------------

1-Add student

2-display student details

3-search student using ID

4-delete student using ID

5-Update student

6-Clear the data from Saved file

7-Save the file and exit

Enter the choice:

2

Sl.NO ID Name Age Father Name Contact info CGPA Student's Address

--------------------------------------------------------------------------------------------------------------

1 AP231100114XX student2 23 father2 9345XXX673 8.85 kurnool

Student Database Management System

--------------------------------------------------

1-Add student

2-display student details

3-search student using ID

4-delete student using ID

5-Update student

6-Clear the data from Saved file

7-Save the file and exit

Enter the choice:

7

Data saved successfully..

Enter the role admin (or) Student:

student

Student login succesful!

Data loaded successfully..!

Student Database Management System

--------------------------------------------------

1-display student details

2-exit

Enter the choice:

1

Sl.NO ID Name Age Father Name Contact info CGPA Student's Address

--------------------------------------------------------------------------------------------------------------

1 AP231100114XX student2 23 father2 9345XXX673 8.85 kurnool

Student Database Management System

--------------------------------------------------

1-display student details

2-exit

Enter the choice:

2

Exited...

--------------------------------

Process exited after 145.9 seconds with return value 0

Press any key to continue . . .

**CONCLUDING REMARKS**

The Student Database Management System is a user-friendly and efficient solution for managing student records. It allows Admins to perform critical tasks like adding, updating, and deleting student information while ensuring data security and role-based access. Students can easily view and search for their records, making the system useful for all users.

The program emphasizes simplicity, accuracy, and persistence by using file handling to save data. With its structured design and robust testing, the system is reliable and effective. In the future, additional features like data sorting, improved security, or a graphical interface can be introduced to enhance functionality and usability further.

Overall, this system demonstrates how programming can simplify complex tasks, making it a valuable tool for educational institutions.

**FUTURE WORKS**

The Student Database Management System can be improved and expanded in the future with the following features:

**1. Data Sorting and Filtering**

Add options to sort student records by ID, name, age, or marks.

Allow filtering based on specific criteria, like students above a certain mark’s threshold.

**2. Enhanced Security**

Use encrypted passwords for Admin login to ensure better security.

Implement user authentication for students to access their specific data.

**3. Graphical User Interface (GUI)**

Develop a GUI to make the system more interactive and user-friendly.

Use GUI frameworks like Qt or libraries such as SFML for C++.

**4. Data Analysis Tools**

Include features to calculate class averages or highlight top-performing students.

Provide reports summarizing data trends, like age or marks distributions.

**5. Database Integration**

Connect the system to a relational database (e.g., MySQL or SQLite) for better scalability and performance.

Enable real-time updates and data synchronization.

**6. Cloud and Remote Access**

Make the system accessible over a network or the internet to allow remote management.

Use cloud storage for secure, scalable data management

**7. Export and Import Feature**

Allow exporting data to formats like CSV, Excel, or PDF for external use.

Add the ability to import data from existing databases or spreadsheets.

These enhancements can improve the system’s functionality, security, and user experience, making it even more practical for real-world applications**.**