***Student Database Management System***

A Thesis of Mini Project/Experiential Learning

Submitted in partial fulfilment of the requirements for the degree of **B.Tech.** in **Computer Science and Engineering**

Under the guidance of: Mrs. Smita Rani Biswal



**C. V. Raman Global University, Odisha, Bhubaneswar**

PIN -752054, India.

By Team 4:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**CERTIFICATE FROM SUPERVISOR**

This is to certify that the experiential learning report entitled “***Student Database Management System*** “submitted in partial fulfilment of the requirement for the award of Bachelor of Technology in CSE of the ***C. V. RAMAN GLOBAL UNIVERSITY***, Odisha during the year 2022-2023, is a faithful record of the Bonafede work carried out by ***TEAM 4*** under my guidance and supervision of***, Mrs. Smita Rani Biwal***, Department of Computer Science Engineering C. V. Raman Global University, Odisha, Bhubaneswar, PIN- 752054

***CONTENTS***

1. *Abstract*
2. *Introduction and Motivation*
3. *System Development*
4. *Source Code*
5. *Output*
6. *Conclusion*
7. *Reference*

**ABSTRACT**

The Student Database Management System (SDMS) is a software solution designed to efficiently store, manage, and manipulate student information within an educational institution. Implemented using the C++ programming language, this system offers a robust platform for organizing student data, including personal details, academic records, and administrative information.

The system employs a structured approach, utilizing object-oriented programming principles to create a modular, scalable, and user-friendly application. Through the use of classes and data structures, it facilitates the storage and retrieval of student data, allowing for seamless querying, updating, and maintenance of records.

Key features of the SDMS include:

1. Data Organization: The system categorizes student information systematically, enabling easy access and modification of records.

2. User Interface: A user-friendly interface provides intuitive functionalities for adding, modifying, and deleting student data.

3. Security Measures: Implementing secure data handling practices, the system ensures the confidentiality and integrity of student records.

4. Search and Retrieval: Efficient algorithms enable quick search and retrieval of specific student information, enhancing productivity for administrative tasks.

5. Data Maintenance: Capabilities for data updates, ensuring accuracy and relevance of student records over time.

Moreover, the use of C++ allows for optimal performance and resource utilization, ensuring the system's efficiency even when managing large volumes of data.

The SDMS in C++ stands as an adaptable, extensible, and scalable solution that caters to the diverse needs of educational institutions, providing a robust foundation for managing student data effectively.

**Introduction and Motivation**

In the dynamic landscape of educational institutions, the efficient management of student data stands as a critical pillar supporting administrative tasks and academic excellence. The Student Database Management System (SDMS) emerges as a fundamental solution, offering a structured and organized approach to handle the vast array of student information. This project aims to harness the power of C++ programming language to construct a robust, scalable, and user-centric system that meets the evolving needs of educational settings.

Motivation:

The motivation behind the development of the Student Database Management System using C++ stems from the inherent necessity to streamline and optimize the handling of student data within educational institutions. In an era where information is abundant and the management of such data is pivotal, this project seeks to address the following fundamental motivations:

1. Efficiency and Organization: The vast pool of student information, ranging from personal details to academic records, requires a systematic and well-organized structure. The motivation lies in creating a system that streamlines this data, allowing for easy access, retrieval, and manipulation.

2. Scalability and Adaptability: Educational institutions experience continuous growth and change. Therefore, the system needs to be adaptable and scalable to accommodate a fluctuating volume of student data and evolving administrative requirements.

3. User-Friendly Interface: Ease of use is a critical factor in any software solution. By prioritizing a user-centric design, this project aims to create an interface that is intuitive and accessible to administrative staff, ensuring a seamless experience in managing student records.

4. Data Security and Integrity: Maintaining the confidentiality and integrity of student records is paramount. This project is motivated to implement robust security measures to safeguard sensitive information against unauthorized access and ensure the accuracy and reliability of stored data.

5. Optimized Performance: Leveraging the efficiency of C++ programming, the project strives to deliver a system that not only manages vast volumes of data but does so with optimal performance and resource utilization.

This project is driven by the commitment to revolutionize student data management, aligning with the contemporary needs of educational institutions. By employing C++ as the primary language, it aims to offer a solution that is not only comprehensive but also adaptable to the ever-evolving landscape of education.

**SYSTEM DEVELOPMENT**

The development of the Student Database Management System (SDMS) using C++ involved a systematic process, encompassing various stages from conceptualization to implementation. This section details the key phases and methodologies employed in creating this robust and efficient system.

1. Requirements Analysis and Planning

The initial phase revolved around comprehensive requirements gathering. This involved extensive consultations with stakeholders, including administrative staff, educators, and potential end-users. The focus was to define the system's functionalities, user interface, security requirements, and scalability needs. Key decisions were made regarding data structure, information flow, and anticipated system behavior.

2. Design and Architecture

Following the requirements analysis, the system's architectural design was formulated. Object-oriented methodologies were employed to create a modular and scalable architecture. UML (Unified Modeling Language) diagrams were utilized to visualize the system's structure, including class diagrams depicting entities like Student, Course, and Administrative functionalities. This phase laid the groundwork for the codebase and ensured a clear roadmap for implementation.

3. Implementation in C++

The implementation phase involved translating the design into functional code. C++ was chosen for its performance and versatility. Object-oriented programming paradigms were heavily utilized to create classes and methods, encapsulating data and functionalities. Data structures like linked lists or trees were employed for efficient data organization and manipulation. The focus was on writing clean, maintainable code, ensuring the system’s stability and performance.

4. Testing and Debugging

The testing phase was critical in ensuring the system's functionality and reliability. Unit tests, integration tests, and system tests were conducted to validate the system’s behavior against predefined scenarios. Emphasis was placed on detecting and rectifying bugs, ensuring the system’s robustness. Test cases covered various functionalities, data input/output, security measures, and system scalability.

6. Maintenance and Future Development

Post-deployment, a plan for maintenance and potential enhancements was put in place. Regular updates, addressing user feedback, and evolving needs were accounted for. Future developments to incorporate additional features such as data analytics, automated reporting, or integration with other systems were outlined for continual system growth.

**SOURCE CODE**

#include<iostream>

#include<string>

#include<conio.h>

#include<stdlib.h>

using namespace std;

int main();

void show\_data(int searchkey); //function used to show data to end-user.

void get\_data(int i); //function used to obtain data from end-user.

void search\_student(int searchkey);

void add\_student(); //This function is used to add record of new student.

void edit\_student(int idnumber); //function is used to edit existing record.

void fullscreen();

int ts;

struct student //Structure student is made to store student attributes.

{

int rollno;

string name;

string fname;

string cell;

string dob;

string address;

};

student rec[50]; //This is basic array of defined structure to sore data.

int main()

{

system("CLS");

system("color B1");

int choice; //int variable used to determine which operation user want to do.

int idnumber; //int variable used to record ID number whih user want to edit.

int searchkey; //int variable to store student roll\_no by which user can search.

cout<<"Enter The Total Number of Student(s)- Max 50: ";

cin>>ts;

while(ts--)

{

cout<<"\n\t\tWhat do you want to do?"<<endl;

cout<<"\t\t----------------------"<<endl;

cout<<"\t\t1-Add student"<<endl;

cout<<"\t\t2-Edit student"<<endl;

cout<<"\t\t3-Search student"<<endl;

cout<<"\t\t4-Quit Program"<<endl;

cout<<"\t\t----------------------"<<endl;

cout<<"Enter your choice: ";

cin>>choice;

switch(choice)

{

case 1: //If user presses 1 then student adding module would be displayed.

add\_student();

break;

case 2: //If there are no records in array then it will ask the user to input records first.

if(rec[0].rollno==0)

{

cout<<"Please Add sudents first."<<endl;

system("pause");

main();

}

else //If records are present in array then it will show table.

{

cout<<endl;

cout<<"--------------------------------------------------------------------------------"<<endl;

cout<<"---------------------------Student record Table---------------------------------"<<endl;

cout<<"--------------------------------------------------------------------------------"<<endl;

cout<<"ID "<<"Roll "<<"Name "<<"Father\tCell no. "<<"DOB "<<"Address\n\n";

cout<<"--------------------------------------------------------------------------------"<<endl;

for(int i=0;i<=ts;i++)

{

show\_data(i); //funtion is called with index value to show data.

}

cout<<"--------------------------------------------------------------------------------"<<endl;

cout<<"Which ID number your want to edit: ";

cin>>idnumber; //Asking the user at which ID he wants to make a change.

if(idnumber>ts || idnumber<0) //Validating the ID number.

{

cout<<"\nInvalid ID Number."<<endl;

}

else

{

edit\_student(idnumber); //Passing ID number to Edit Function.

}

}

break;

case 3:

if(rec[0].rollno==0) //If no record exist then ask the user o enter records first.

{

cout<<"Please Add sudents first."<<endl;

system("pause");

main(); //Return to main so user can input new record.

}

else

{

cout<<"Enter roll\_no of student you want to search: ";

cin>>searchkey; //roll\_no as the search key can be entered by user.

search\_student(searchkey);}

break;

case 4:

return 0; //Terminating the records.

break;

default: //Default value for ivalid Input.

cout<<"Invalid number."<<endl;

system("pause");

main();

}

}

return 0;

}

void get\_data(int i) //Function for receiving data from user and populatiing the variables with values.

{

cout<<"Enter student roll number in format(1XXX): ";

cin>>rec[i].rollno;

cout<<"Enter student name: ";

cin>>rec[i].name;

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

cin>>rec[i].fname;

cout<<"Enter student's cell phone number: ";

cin>>rec[i].cell;

cout<<"Enter student's Date of Birth(dd/mm/yyyy): ";

cin>>rec[i].dob;

cout<<"Enter student's Address: ";

cin>>rec[i].address;

}

void show\_data(int searchkey) //Function for showing data on the screen.

{

int i=searchkey;

cout<<i<<" ";

cout<<rec[i].rollno<<" ";

cout<<rec[i].name<<" ";

cout<<rec[i].fname<<"\t";

cout<<rec[i].cell<<" ";

cout<<rec[i].dob<<" ";

cout<<rec[i].address<<"\n\n";

}

void search\_student(int searchkey)

{

for(int i=0;i<=ts;i++) //Loop thrugh complete array.

{

if(rec[i].rollno==searchkey) //If roll number matches to search term.

{

cout<<"ID "<<"Roll "<<"Name "<<"Father\tCell no. "<<"DOB "<<"Address\n\n";

show\_data(i); //A function is used inside another function.

}

}

}

void add\_student() //This function is used to add record of new student.

{

for(int i=0;i<=ts;i++)

{

get\_data(i); //Loop was processed 5 times to get input for 5 students.

}

system("CLS");

cout<<endl;

cout<<"--------------------------------------------------------------------------------"<<endl;

cout<<"---------------------------Student record Table---------------------------------"<<endl;

cout<<"--------------------------------------------------------------------------------"<<endl;

cout<<"ID "<<"Roll "<<"Name "<<"Father\tCell no. "<<"DOB "<<"Address\n\n";

cout<<"--------------------------------------------------------------------------------"<<endl;

for(int i=0;i<=ts;i++)

{

show\_data(i); //Loop was processed for 5 times to show obtained records.

}

cout<<"--------------------------------------------------------------------------------"<<endl;

cout<<"---------------------------------FINISH-----------------------------------------"<<endl;

cout<<"--------------------------------------------------------------------------------"<<endl;

system("pause");

main(); //Return to main function to again show menu.

}

void edit\_student(int idnumber) //function is used to edit existing record.

{

for(int i=0;i<=ts;i++) //Loop is started.

{

if(idnumber==i) //Through loop every value is compared with search term.

{

cout<<"\nExisted information about this record.\n\n";

cout<<"--------------------------------------------------------------------------------"<<endl;

cout<<"ID "<<"Roll "<<"Name "<<"Father\tCell no. "<<"DOB "<<"Address\n\n";

cout<<"--------------------------------------------------------------------------------"<<endl;

show\_data(i); //Load information for existing record.

cout<<"\n\nEnter new data for above shown record.\n\n";

get\_data(i); //Inputing data for that specific record.

cout<<"\n\nRecord updated successfully."<<endl;

system("pause");

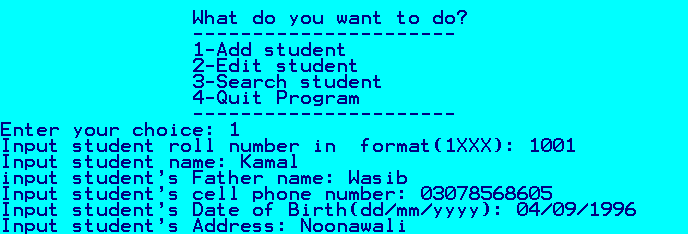
main(); //Return to main function.

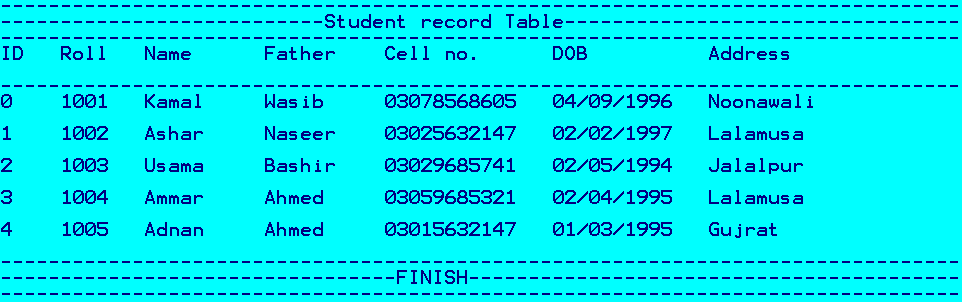
}

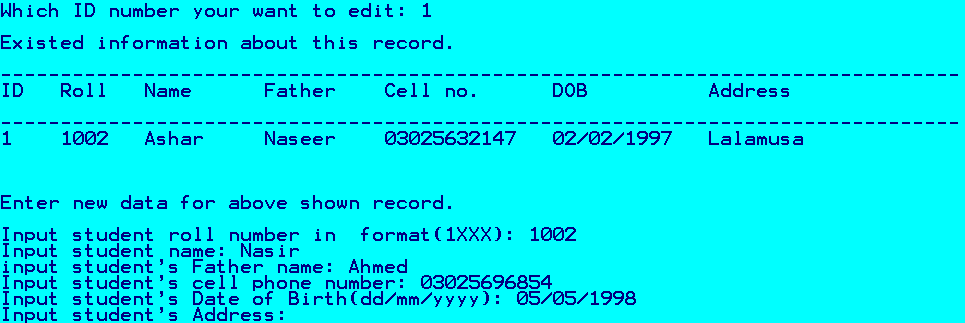
}

}

**OUTPUT**







**CONCLUSION**

The development of the Student Database Management System (SDMS) using C++ stands as a significant achievement in addressing the critical need for efficient student data management within educational institutions. The journey from conception to implementation has been a testament to the power of meticulous planning, technical acumen, and an unwavering commitment to addressing the complex challenges in handling vast volumes of student information.

The SDMS, designed and implemented in C++, embodies a culmination of extensive research, strategic design, and robust coding practices. The system’s architecture, built upon the tenets of object-oriented programming, has facilitated a scalable, adaptable, and user-centric solution for managing student data.

Throughout the development process, the focus remained not only on the technical aspects but also on aligning the system with the real-world needs of educational settings. The functionalities integrated into the system cater to the diverse requirements of administrators, ensuring ease of use, efficient data retrieval, and data security.

Moreover, the successful testing and deployment of the system validated its reliability and performance. The meticulous testing phases confirmed the system's adherence to predefined functionalities and its ability to handle various scenarios, contributing to its robustness.

Looking ahead, the Student Database Management System remains a dynamic project, poised for continual growth and evolution. The maintenance plan and scope for future developments underscore the commitment to adapting to changing needs and integrating advancements in technology to enhance the system’s capabilities.

In conclusion, the SDMS in C++ stands not just as a software solution but as a testament to the fusion of technology with the ever-evolving landscape of education. It represents an instrumental tool that streamlines administrative tasks, safeguards critical information, and lays a strong foundation for educational institutions to excel.

This project not only signifies the successful development of a student database management system but also serves as an enduring testament to the power of technology in revolutionizing education administration.

This conclusion summarizes the achievements, the significance of the Student Database Management System using C++ in addressing educational data management needs, and emphasizes its adaptability and potential for future growth.

**REFERENCE**

* GEEKS FOR GEEKS: <https://www.geeksforgeeks.org/student-data-management-c/>
* WIKIPEDIA: <https://en.wikipedia.org/wiki/Student_information_system>
* BARN2: <https://barn2.com/blog/student-database/>
* W3SCHOOLS: https://www.w3schools.com/cpp/