**SRI VENKATESWARA COLLEGE OF ENGINEERING**

**(AUTONOMOUS)**

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING**



**(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu Accredited by NBA, New Delhi & NAAC with ‘A’ grade)**

**Karakambadi Road, TIRUPATI – 517507**

**2023 – 2026**

**Dr. V. Lakshmi Devi Ph.D.**

**Head of the Department**

**PROJECT REPORT**

* **Name of the Trainee**  : K.Thilak
* **Name of the Company**  : ByteXL TechED Pvt. Ltd
* **Name of the Supervisor/Guide** : M.Pranay
* **Title of Report** : Student Record System
* **Field of Training** : Programming Technologies
* **Area of the project** : C Programming

ABSTRACT

The Student Record Management System is developed using the C programming language to streamline the process of maintaining student records in educational institutions. This system is designed to handle various aspects of student information, including personal details, academic records, attendance, and disciplinary actions.

The Student Record Management System leverages the efficiency and performance of the C language to provide a robust and scalable solution. Key features include data entry, data retrieval, updating records, and generating reports. The system uses file handling and data structures to ensure data integrity and security. Additionally, the Student Record Management System incorporates user authentication and role-based access control to protect sensitive information.

By automating routine administrative tasks, the Student Record Management System aims to reduce the workload on administrative staff and improve the accuracy of record-keeping. The implementation of this system in C programming demonstrates its capabilities in managing large datasets and performing complex operations efficiently.

This ab stract outlines the system's objectives core functionalities, and the expected benefits of implementation, emphasizing the importance of an effective student record management solution in enhancing the educational experience for both students and staff.

INTRODUCTION

STUDENT RECORD MANAGEMENT SYSTEM

In today's rapidly evolving educational landscape, the efficient management and maintenance of student records is paramount for the smooth functioning of academic institutions. The Student Record System is a comprehensive software application designed address this critical need by automating and optimizing the administrative tasks associated with student data management. This system serves as a centralized repository for storing and retrieving vital information about students, including their personal details, academic performance, attendance records, and participation in extracurricular activities.

One of the primary benefits of the Student Record System is its ability to enhance the accuracy and reliability of student records. By digitizing the data entry process, the system minimizes the chances of human error that are often associated with record-keeping. This ensures that the information is up-to-date and accurate, allowing educators and administrators to make informed decisions based on real-time data. Additionally, the system provides robust security features to protect sensitive student information, ensuring that access is restricted to authorized personnel only.

The Student Record System also offers a user-friendly interface that simplifies the process of data management for all stakeholders. Teachers can easily input grades and attendance records, while administrators can generate comprehensive reports with just a few clicks. Students and parents, on the other hand, can access their records online, fostering transparency and enabling them to stay informed about academic progress and areas that may require improvement. This seamless communication between all parties involved helps to create a more collaborative and supportive educational environment.

Furthermore, the system's ability to integrate with other educational technologies, such as learning management systems and communication platforms, enhances its functionality and versatility. This interoperability allows more holistic approach to student management, ensuring that all aspects of a student's educational journey are documented and accessible in one place. By streamlining these processes, the Student Record System not only saves time and reduces administrative burdens but also allows educators to focus more on delivering quality education and less on paperwork.

Another notable advantage of the Student Record System is its scalability. Whether it is implemented in a small school or a large university, the system be customized to meet the specific needs and requirements of the institution. This adaptability ensures that the system remains relevant and effective, regardless of the size and scope of the educational establishment.

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

Hardware - intel i3

Speed - 2.1 GHz

RAM - 4GB

Hard Disk - 100 GB SSD

Floppy Drive - 2.88 MB

Key Board - Standard Windows Keyboard

Mouse - Two or Three-Button Mouse

SOFTWARE REQUIREMENTS:

Operating System: Windows 10, 11

Technology: C

Compiler: byteXL TechEd editor (https://bytexl.app/editor).

SOURCE CODE

#include <stdio.h>

#include <string.h>

struct Student {

int rollNo;

char name[100];

float marks;

};

struct Student students[100];

int count = 0;

void addStudent() {

if (count < 100) {

printf("Enter Roll No: ");

scanf("%d", &students[count].rollNo);

printf("Enter Name: ");

scanf(" %[^\n]", students[count].name);

printf("Enter Marks: ");

scanf("%f", &students[count].marks);

count++;

printf("Student added successfully!\n");

} else {

printf("Record is full!\n");

}

}

void viewStudents() {

if (count == 0) {

printf("No students available!\n");

} else {

printf("Students:\n");

for (int i = 0; i < count; i++) {

printf("Roll No: %d, Name: %s, Marks: %.2f\n", students[i].rollNo, students[i].name, students[i].marks);

}

}

}

void searchStudent() {

int rollNo;

printf("Enter Roll No to search: ");

scanf("%d", &rollNo);

for (int i = 0; i < count; i++) {

if (students[i].rollNo == rollNo) {

printf("Student found!\n");

printf("Roll No: %d, Name: %s, Marks: %.2f\n", students[i].rollNo, students[i].name, students[i].marks);

return;

}

}

printf("Student not found!\n");

}

int main() {

int choice;

while (1) {

printf("\nStudent Record System\n");

printf("1. Add Student\n");

printf("2. View Students\n");

printf("3. Search Student\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addStudent();

break;

case 2:

viewStudents();

break;

case 3:

searchStudent();

break;

case 4:

printf("Exiting...\n");

return 0;

default:

printf("Invalid choice! Try again.\n");

}

}

return 0;

}

CODE EXPLANATION

**Step-by-Step Explanation:-**

1. **Include Headers**

#include <stdio.h>

#include <string.h>

These headers include the standard input-output library and the string manipulation library.

1. **Define the** Student **Structure**

struct Student {

int rollNo;

char name[100];

float marks;

};

This structure defines a student with three fields: rollNo (an integer for the roll number), name (a string for the student's name), and marks (a float for the student's marks).

1. **Declare Variables**

struct Student students[100];

int count = 0;

students is an array that can hold up to 100 student records. count keeps track of the number of students added.

1. **Add Student Function**

void addStudent() {

if (count < 100) {

printf("Enter Roll No: ");

scanf("%d", &students[count].rollNo);

printf("Enter Name: ");

scanf(" %[^\n]", students[count].name);

printf("Enter Marks: ");

scanf("%f", &students[count].marks);

count++;

printf("Student added successfully!\n");

} else {

printf("Record is full!\n");

}

}

This function adds a new student record. It checks if the array is not full before allowing the user to input the student's details. It uses scanf to read input for the roll number, name, and marks, and then increments the count.

1. **View Students Function**

void viewStudents() {

if (count == 0) {

printf("No students available!\n");

} else {

printf("Students:\n");

for (int i = 0; i < count; i++) {

printf("Roll No: %d, Name: %s, Marks: %.2f\n", students[i].rollNo, students[i].name, students[i].marks);

}

}

}

This function displays all the student records. If there are no students, it prints a message indicating that.

1. **Search Student Function**

void searchStudent() {

int rollNo;

printf("Enter Roll No to search: ");

scanf("%d", &rollNo);

for (int i = 0; i < count; i++) {

if (students[i].rollNo == rollNo) {

printf("Student found!\n");

printf("Roll No: %d, Name: %s, Marks: %.2f\n", students[i].rollNo, students[i].name, students[i].marks);

return;

}

}

printf("Student not found!\n");

}

This function searches for a student by roll number. It prompts the user to enter the roll number and then checks each record. If a matching roll number is found, it prints the student's details; otherwise, it prints "Student not found."

1. **Main Function**

int main() {

int choice;

while (1) {

printf("\nStudent Record System\n");

printf("1. Add Student\n");

printf("2. View Students\n");

printf("3. Search Student\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addStudent();

break;

case 2:

viewStudents();

break;

case 3:

searchStudent();

break;

case 4:

printf("Exiting...\n");

return 0;

default:

printf("Invalid choice! Try again.\n");

}

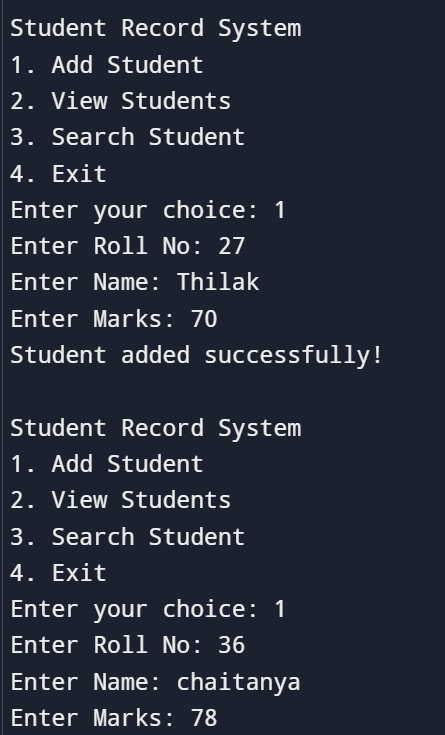
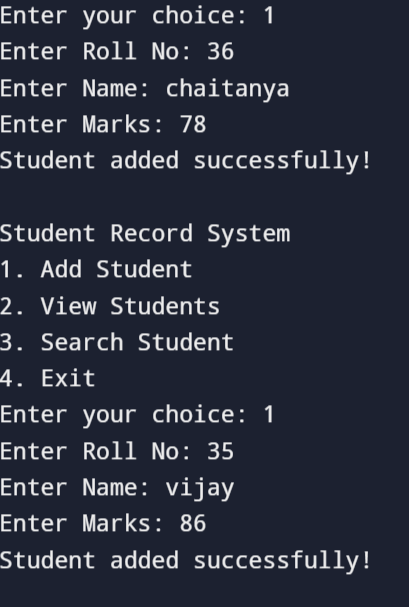
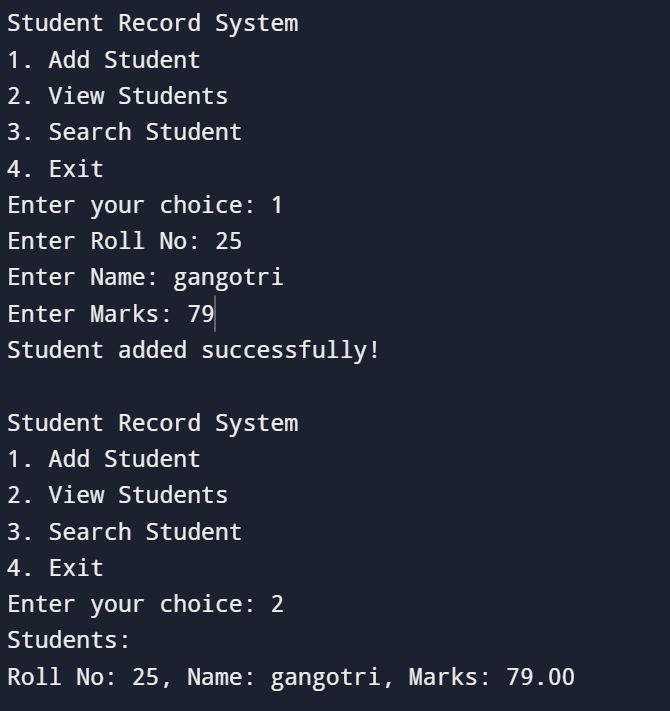
}

return 0;

}

The main function contains a loop that displays a menu and handles user input. Depending on the user's choice, it calls the appropriate function (add, view, or search students) or exits the program.

SCREENSHOTS



THANK YOU