**Student Record System**

**Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of**

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**K Venkata Sudheer – 24KB1A05P9**

**K Rohan Balaji – 24KB1A05T3**

**K Guru Raja – 24KB1A05W3**

**K Manendra – 24KB1A05W2**

**Under the Guidance of**

**P Suneetha**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NBKR Institute of Science and Technology**

**Vidyanagar, Tirupathi(D.T)**

**Andhra Pradesh, 524413**

**Student Record System**

**Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of**

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**K Venkata Sudheer – 24KB1A05P9**

**K Rohan Balaji – 24KB1A05T3**

**K Guru Raja – 24KB1A05W3**

**K Manendra – 24KB1A05W2**

**Under the Guidance of**

**P Suneetha**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NBKR Institute of Science and Technology**

**Vidyanagar, Tirupathi(D.T)**

**Andhra Pradesh, 524413**

**NBKR Institute of Science and Technology (AUTONOMOUS)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



###### CERTIFICATE

This is to certify that the project report entitled Student Record System being submitted by

K Venkata Sudheer 24KB1A05P9

K Rohan Balaji 24KB1A05T3

K Guru Raja 24KB1A05W3

K Manendra 24KB1A05W2

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the NBKR Institute of Science and Technology is a record of bonafied work carried out under my guidance and supervision.

|  |  |
| --- | --- |
| **P Suneetha** | **DR. A. Raja Sekhar Reddy**  **Head of the Department** |

**DECLARATION**

I hereby declare that the dissertation entitled **Student Record System** submitted for the B.Tech Degree is my original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Place: Vidyanagar

Date: 07-05-2025

K Venkata Sudheer – 24KB1A05P9

K Rohan Balaji – 24KB1A05T3

K Guru Raja – 24KB1A05W3

K Manendra – 24KB1A05W2

Acknowledgment:

I would like to express my sincere gratitude to all those who supported and guided me throughout the development of the **Student Record System** project.

First and foremost, I am thankful to my [Instructor's Name / Professor / Guide] for their valuable insights, continuous encouragement, and constructive feedback, which greatly contributed to the success of this project.

This project implements a structured approach to managing student data, using **arrays** to efficiently store student records and a **linked list** to update and manage specific subject marks dynamically. The combination of these data structures has provided a flexible and efficient solution to handle academic records.

I also extend my thanks to my peers, friends, and family for their moral support and motivation during this endeavor.

Finally, I acknowledge the learning resources and documentation that helped me understand the technical concepts needed for this implementation.

Abstract:

The Student Record System is a data management application designed to efficiently store and update academic information for students. This system employs an array structure to store and manage basic student records such as ID, name, and overall subjects. To allow flexible and efficient updates of specific subject marks without altering the entire dataset, a linked list is integrated for each student, representing individual subject marks. This hybrid data structure approach enhances performance by enabling fast access through arrays and dynamic updates via linked lists. The system supports key functionalities such as adding new students, updating subject marks, and displaying academic performance, making it a practical tool for educational institutions aiming to manage student data effectively.

Introduction:

Managing student academic records is a fundamental requirement in educational institutions. An efficient system not only facilitates the storage of student data but also ensures smooth and dynamic updates as students progress through their academic journey. The **Student Record System** aims to address these needs by implementing a simple yet effective data structure-based solution.

In this project, an **array** is used to store the core information of each student, such as their ID, name, and a reference to their academic performance. Arrays are chosen for their ability to provide fast and direct access to student records based on index values, making them ideal for handling a fixed number of students efficiently.

To allow **dynamic and frequent updates** to students' subject marks—such as updating a single subject score without affecting the entire dataset—a **linked list** is employed for each student's subject marks. Linked lists provide the flexibility to insert, delete, or modify individual subject marks with minimal overhead, making them particularly suitable for managing subject-level data that may change over time.

This combination of arrays and linked lists leverages the strengths of both data structures: fast access and dynamic updating. The system is designed to be scalable, easy to understand, and adaptable for use in school or college-level applications where managing academic records accurately and efficiently is crucial

**Software Requirements Analysis**

The **Student Record System** is designed to manage student data efficiently, utilizing arrays for storing student profiles and linked lists for managing subject-specific marks. This section outlines the functional and non-functional requirements, along with the system's constraints and assumptions.

### 1. ****Functional Requirements****

These describe the core operations the system must support:

* **FR1:** The system shall allow the addition of new student records.
* **FR2:** The system shall store student details (e.g., name, roll number, ID) in an array.
* **FR3:** The system shall maintain a linked list for each student to store subject-wise marks.
* **FR4:** The system shall allow insertion, deletion, or updating of subject marks in the linked list.
* **FR5:** The system shall display all students and their subject-wise marks.
* **FR6:** The system shall search for a student by ID or name.
* **FR7:** The system shall provide functionality to calculate average marks or total marks for a student.

### 2. ****Non-Functional Requirements****

These define the quality and performance expectations:

* **NFR1:** The system should respond to updates and queries within a short time frame.
* **NFR2:** The code should be modular and maintainable, using well-defined data structures.
* **NFR3:** The system should be capable of handling at least 100 student records efficiently.
* **NFR4:** The system should have a simple text-based interface for input/output (for CLI version).
* **NFR5:** The system should ensure data consistency during record updates.

### 3. ****System Requirements****

These specify the software and hardware needed to run the system:

* **Operating System:** Windows, Linux, or macOS
* **Compiler/IDE:** GCC for C/C++ or any suitable development environment
* **Programming Language:** C or C++
* **Memory Requirements:** Minimum 4 GB RAM
* **Storage:** Minimum 100 MB free disk space

### 4. ****Assumptions and Constraints****

* The number of students is fixed or limited for simplicity in array implementation.
* Each student can have a varying number of subjects, hence the use of a dynamic structure (linked list).
* Data is stored in memory only; no database or file persistence is implemented unless extended

Proposed System:

**Proposed System**

The proposed **Student Record System** is a lightweight, structured application that aims to simplify the management of student academic data through the effective use of data structures. The system integrates an **array** to store and manage static student information and a **linked list** to handle dynamic updates of subject-specific marks.

### Key Features of the Proposed System:

1. **Student Information Storage (Array-Based):**
   * The system uses a fixed-size array to store student records.
   * Each element in the array represents a student and contains fields such as Student ID, Name, and a pointer/reference to their subject marks.
2. **Dynamic Subject Marks Management (Linked List-Based):**
   * For each student, a linked list is used to store their subject-wise marks.
   * Each node in the linked list contains a subject name/code and the corresponding mark.
   * This structure allows flexible insertion, deletion, and updating of marks without affecting the overall data structure.
3. **Efficient Data Access and Modification:**
   * Arrays allow fast, indexed access to student records.
   * Linked lists provide dynamic memory allocation for subjects, making the system scalable in terms of subject management per student.
4. **User Functionalities:**
   * Add new student records.
   * Add, update, or delete subject marks for individual students.
   * Search for students by ID or name.
   * Display complete academic records of students.
   * Calculate total or average marks.
5. **Scalability and Simplicity:**
   * The system is designed for educational institutions that need a simple and memory-efficient solution for student record handling.
   * It serves as a foundational system that can be extended in the future (e.g., file storage, graphical UI, or database integration).

### Benefits:

* Efficient memory use by combining static (array) and dynamic (linked list) structures.
* Easy maintenance and updates to specific subject marks.
* Good performance for small to medium

Coding:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#define MAX\_STUDENTS 100

#define NAME\_LENGTH 50

// Linked list for subjects and marks

typedef struct Subject {

char name[30];

int marks;

struct Subject \*next;

} Subject;

// Student structure

typedef struct {

int id;

char name[NAME\_LENGTH];

Subject \*subjects; // linked list head

} Student;

Student students[MAX\_STUDENTS];

int studentCount = 0;

// Function to add a student

void addStudent(int id, const char \*name) {

if (studentCount >= MAX\_STUDENTS) {

printf("Max student limit reached.\n");

return;

}

students[studentCount].id = id;

strcpy(students[studentCount].name, name);

students[studentCount].subjects = NULL;

studentCount++;

}

// Function to add/update subject marks for a student

void updateSubjectMarks(int studentId, const char \*subjectName, int marks) {

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

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

Subject \*current = students[i].subjects;

// Search for the subject

while (current != NULL) {

if (strcmp(current->name, subjectName) == 0) {

current->marks = marks; // update marks

return;

}

current = current->next;

}

// Add new subject

Subject \*newSubject = (Subject \*)malloc(sizeof(Subject));

strcpy(newSubject->name, subjectName);

newSubject->marks = marks;

newSubject->next = students[i].subjects;

students[i].subjects = newSubject;

return;

}

}

printf("Student with ID %d not found.\n", studentId);

}

// Display all student records

void displayStudents() {

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

printf("ID: %d, Name: %s\n", students[i].id, students[i].name);

Subject \*current = students[i].subjects;

while (current != NULL) {

printf(" Subject: %s, Marks: %d\n", current->name, current->marks);

current = current->next;

}

}

}

int main() {

int n;

int id,marks;

char name[NAME\_LENGTH],subject[NAME\_LENGTH];

bool choice = true;

while(choice){

printf("enter your choice\n");

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

printf("2. Update Subject Marks\n");

printf("3. Dispaly Students\n");

printf("4. Exit\n");

scanf("%d",&n);

switch(n){

case 1:

printf("enter the Student ID and Name\n");

scanf("%d",&id);

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

addStudent(id,name);

break;

case 2:

printf("enter the student ID, subject and marks\n");

scanf("%d",&id);

scanf(" %[^\n]", subject);

scanf("%d",&marks);

updateSubjectMarks(id,subject,marks);

break;

case 3:

displayStudents();

break;

case 4:

printf("Code Exit from the loop\n");

choice = false;

break;

default:

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

}

}

/\*addStudent(1, "Alice");

addStudent(2, "Bob");

updateSubjectMarks(1, "Math", 90);

updateSubjectMarks(1, "Science", 85);

updateSubjectMarks(2, "DS", 75);

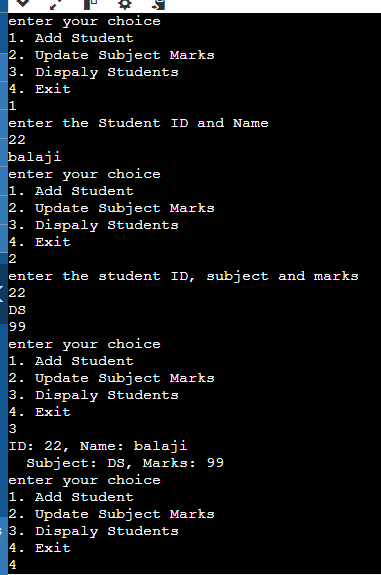
updateSubjectMarks(1, "physics", 95); // update marks

displayStudents();\*/

return 0;

}

OUTPUT:



Conclusion:

The **Student Record System** successfully demonstrates the practical application of fundamental data structures—**arrays** and **linked lists**—in managing academic records. By storing student profiles in an array, the system ensures fast and direct access to individual student data. The use of linked lists for handling subject-specific marks allows for dynamic, flexible updates without restructuring the entire record, which is particularly beneficial in real-world academic scenarios where subject data may frequently change.

This hybrid approach provides an effective balance between performance and flexibility, making the system suitable for small to medium-sized educational institutions. The modular design also lays a strong foundation for future enhancements such as file handling, GUI development, or database integration.

Overall, the project meets its goals of efficient data storage, ease of updates, and user-friendly access, serving as a solid model for student information management using fundamental programming concepts

References:

1. Yashavant Kanetkar, Let Us C, BPB Publications – for understanding C programming concepts and implementation techniques.
2. E. Horowitz, S. Sahni, and D. Mehta, Fundamentals of Data Structures in C, Universities Press – for detailed explanation and application of arrays and linked lists.
3. Herbert Schildt, C: The Complete Reference, McGraw-Hill – for in-depth understanding of the C language and standard libraries.
4. GeeksforGeeks. <https://www.geeksforgeeks.org> – for practical examples and tutorials on arrays, linked lists, and student record management systems.
5. TutorialsPoint. <https://www.tutorialspoint.com> – for simplified explanations and reference code on data structures and file handling in C.
6. Stack Overflow. <https://stackoverflow.com> – for community-driven problem-solving related to coding and logic implementation

