# Title: Student Record Management System

**Abstract**

The *Student Record Management System* is a Java-based console application developed to efficiently manage student details such as roll number, name, and marks using the **Linked List** data structure. The system provides essential operations like adding, updating, deleting, searching, and displaying student records dynamically. Unlike static data storage methods such as arrays, linked lists allow flexible memory usage and faster modifications without reallocation. This project demonstrates core **Data Structure** concepts—such as nodes, pointers, and dynamic memory allocation—in a practical and interactive way. It aims to simplify record management for educational institutions while showcasing the real-world application of linked lists in software development.

## Introduction

The Student Record Management System is a console-based project designed to manage student information efficiently using the Linked List data structure in Java. The main objective of this project is to store, modify, and display student details such as roll number, name, and marks in a dynamic way. Unlike arrays, linked lists allow efficient insertion and deletion without wasting memory, making them suitable for managing student data dynamically.  
  
This project demonstrates core concepts of Data Structures, such as nodes, pointers, and dynamic memory allocation, in an easy-to-understand and practical form.

## Existing System

In traditional record management systems, student information is maintained manually in registers or static arrays. This system has several drawbacks:  
  
• Difficult to add or delete records efficiently.  
• Requires fixed memory allocation (in case of arrays).  
• Searching and sorting take more time.  
• Data updates are not managed dynamically.  
  
Hence, a more flexible and efficient approach using Linked Lists is required to overcome these limitations.

## Proposed System

The proposed system is a Java-based program that uses a Singly Linked List to manage student details. Each student record is stored as a node containing:  
  
• Roll Number  
• Name  
• Marks  
  
The system allows users to:  
  
• Add, delete, update, and search records easily.  
• Display all student data in a neat format.  
• Sort records by marks.  
• Count the total number of students.  
  
It is a menu-driven program that helps users interactively perform all operations.

## Software and Hardware Requirements

Software Requirements:  
• Operating System: Windows / Linux / macOS  
• Programming Language: Java  
• IDE (Optional): VS Code / Eclipse / IntelliJ IDEA  
• JDK Version: JDK 8 or above

Hardware Requirements:  
• Processor: Intel Core i3 or higher  
• RAM: Minimum 2 GB  
• Hard Disk: At least 100 MB free space  
• Input Device: Keyboard  
• Output Device: Monitor

## Code Implementation

low is the enhanced version of the Student Record Management System with additional features:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Structure for a Student

struct Student {

int rollNo;

char name[50];

char course[30];

float marks;

struct Student\* next;

};

struct Student\* head = NULL;

// Function to create a new student node

struct Student\* createStudent(int rollNo, char name[], char course[], float marks) {

struct Student\* newStudent = (struct Student\*)malloc(sizeof(struct Student));

newStudent->rollNo = rollNo;

strcpy(newStudent->name, name);

strcpy(newStudent->course, course);

newStudent->marks = marks;

newStudent->next = NULL;

return newStudent;

}

// Function to add a student

void addStudent() {

int rollNo;

char name[50];

char course[30];

float marks;

printf("\nEnter Roll No: ");

scanf("%d", &rollNo);

getchar();

printf("Enter Name: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = 0;

printf("Enter Course: ");

fgets(course, sizeof(course), stdin);

course[strcspn(course, "\n")] = 0;

printf("Enter Marks: ");

scanf("%f", &marks);

struct Student\* newStudent = createStudent(rollNo, name, course, marks);

if (head == NULL) {

head = newStudent;

} else {

struct Student\* temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = newStudent;

}

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

}

// Function to display all students

void displayStudents() {

if (head == NULL) {

printf("\n⚠️ No students found!\n");

return;

}

struct Student\* temp = head;

printf("\n--- Student List ---\n");

while (temp != NULL) {

printf("Roll No: %d | Name: %s | Course: %s | Marks: %.2f\n",

temp->rollNo, temp->name, temp->course, temp->marks);

temp = temp->next;

}

}

// Function to search a student

void searchStudent() {

int rollNo;

printf("\nEnter Roll No to search: ");

scanf("%d", &rollNo);

struct Student\* temp = head;

while (temp != NULL) {

if (temp->rollNo == rollNo) {

printf("✅ Student Found:\n");

printf("Roll No: %d | Name: %s | Course: %s | Marks: %.2f\n",

temp->rollNo, temp->name, temp->course, temp->marks);

return;

}

temp = temp->next;

}

printf("⚠️ Student not found!\n");

}

// Function to update student details

void updateStudent() {

int rollNo;

printf("\nEnter Roll No to update: ");

scanf("%d", &rollNo);

getchar();

struct Student\* temp = head;

while (temp != NULL) {

if (temp->rollNo == rollNo) {

printf("Enter New Name: ");

fgets(temp->name, sizeof(temp->name), stdin);

temp->name[strcspn(temp->name, "\n")] = 0;

printf("Enter New Course: ");

fgets(temp->course, sizeof(temp->course), stdin);

temp->course[strcspn(temp->course, "\n")] = 0;

printf("Enter New Marks: ");

scanf("%f", &temp->marks);

printf("✅ Student details updated!\n");

return;

}

temp = temp->next;

}

printf("⚠️ Student not found!\n");

}

// Function to delete a student

void deleteStudent() {

int rollNo;

printf("\nEnter Roll No to delete: ");

scanf("%d", &rollNo);

struct Student\* temp = head;

struct Student\* prev = NULL;

while (temp != NULL && temp->rollNo != rollNo) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) {

printf("⚠️ Student not found!\n");

return;

}

if (prev == NULL)

head = temp->next;

else

prev->next = temp->next;

free(temp);

printf("🗑️ Student deleted successfully!\n");

}

// Main menu

int main() {

int choice;

do {

printf("\n=== STUDENT MANAGEMENT SYSTEM (Using Linked List) ===\n");

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

printf("2. Display All Students\n");

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

printf("4. Update Student\n");

printf("5. Delete Student\n");

printf("6. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar();

switch (choice) {

case 1: addStudent(); break;

case 2: displayStudents(); break;

case 3: searchStudent(); break;

case 4: updateStudent(); break;

case 5: deleteStudent(); break;

case 6: printf("Thank you! Exiting program...\n"); break;

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

}

} while (choice != 6);

return 0;

}

**OUTPUT**

=== STUDENT MANAGEMENT SYSTEM (Using Linked List) ===

1. Add Student

2. Display All Students

3. Search Student

4. Update Student

5. Delete Student

6. Exit

Enter your choice: 1

Enter Roll No: 101

Enter Name: Sathvika

Enter Course: BCA

Enter Marks: 92

✅ Student added successfully!

=== STUDENT MANAGEMENT SYSTEM (Using Linked List) ===

Enter your choice: 1

Enter Roll No: 102

Enter Name: Arjun

Enter Course: BCA

Enter Marks: 85

✅ Student added successfully!

=== STUDENT MANAGEMENT SYSTEM (Using Linked List) ===

Enter your choice: 2

--- Student List ---

Roll No: 101 | Name: Sathvika | Course: BCA | Marks: 92.00

Roll No: 102 | Name: Arjun | Course: BCA | Marks: 85.00

=== STUDENT MANAGEMENT SYSTEM (Using Linked List) ===

Enter your choice: 3

Enter Roll No to search: 102

✅ Student Found:

Roll No: 102 | Name: Arjun | Course: BCA | Marks: 85.00

=== STUDENT MANAGEMENT SYSTEM (Using Linked List) ===

Enter your choice: 4

Enter Roll No to update: 102

Enter New Name: Arjun Reddy

Enter New Course: BCA

Enter New Marks: 88

✅ Student details updated!

=== STUDENT MANAGEMENT SYSTEM (Using Linked List) ===

Enter your choice: 5

Enter Roll No to delete: 101

🗑️ Student deleted successfully!

=== STUDENT MANAGEMENT SYSTEM (Using Linked List) ===

Enter your choice: 6

Thank you! Exiting program...

**Conclusion**

The Student Record Management System provides an efficient way to handle and manage student data using the Linked List data structure. It allows dynamic memory usage, efficient insertion and deletion, and better organization compared to traditional array-based systems. This project not only demonstrates the practical use of Linked Lists but also helps understand fundamental operations like traversal, searching, sorting, and updating records.  
  
Hence, this system successfully fulfills the goal of developing a simple, interactive, and dynamic data structure–based student information manager.