MINI PROJECT

Problem Statement

Title: Student Record Management System Using Linked List

Problem:
Colleges and educational institutions maintain large amounts of student information such as name, roll number, CGPA, and year of study. Managing this data manually is inefficient, prone to errors, and time-consuming.

The system should allow administrators to:

1. Add new student records efficiently.
2. Remove student records when a student leaves or graduates.
3. Search for a specific student record by roll number.
4. Display all student records for easy viewing.

Constraints:

- Each student has a unique roll number.
- The system should handle dynamic data -- i.e., the number of students is not fixed.
- Operations should be performed in a way that preserves the order of insertion.

MINI PROJECT

IMPLEMENTATION OF STUDENT RECORD SYSTEM

```
import java.util.*;
class Student{
   String Name;
   int RollNo;
   Float CGPA;
   int year;
   Student next;
   Student(String Name , int RollNo , Float CGPA , int year){
      this.Name = Name;
      this.RollNo = RollNo;
      this.CGPA = CGPA;
      this.year = year;
      this.next = null;
   }
}
public class Student_Record_System {
   Student head;
   public void addStudent(String Name , int RollNo , Float CGPA , int year){
      Student newStudent = new Student(Name , RollNo , CGPA , year);
      if(head == null){
         head = newStudent;
      }
      else{
         Student current = head;
         while(current.next != null){
            current = current.next;
         }
         current.next = newStudent;
```

```
    }
    System.out.println("Student Added: " + Name);
}
    public void removeStudent(int RollNo){
        if(head == null){
            System.out.println("No available student data");
            return;
        }
        if(head.RollNo == RollNo){
            System.out.println("Deleted Student: " + head.Name);
            head = head.next;
            return;
        }
        Student current = head;
        while(current.next != null && current.next.RollNo != RollNo){
            current = current.next;
        }
        if(current.next == null){
            System.out.println("Student Not Found");
        }
        else{
            System.out.println("Deleted Student: " + current.next.Name);
            current.next = current.next.next;
        }
    }

    public void Search(int RollNo){
        Student current = head;
        while(current != null){
            if(current.RollNo == RollNo){
                System.out.println("Record Found:");
```

```
            System.out.println("RollNo: " + current.RollNo + ", Name: " + current.Name + ", CGPA: " + current.CGPA + ", Year: " + current.year);
            return;
          }
          current = current.next;
        }
        System.out.println("Student record not found");
    }

    public void Display(){
        if(head == null){
            System.out.println("No students record available");
            return;
        }
        Student current = head;
        System.out.println("Student Records: ");
        while(current != null){
            System.out.println("RollNo: " + current.RollNo + ", Name: " + current.Name + ", CGPA: " + current.CGPA + ", Year: " + current.year);
            current = current.next;
        }
    }

    public static void main(String[]args){
        Student_Record_System student = new Student_Record_System();
        Scanner sc = new Scanner(System.in);
        while(true){
            System.out.println("Enter the operation to be performed: ");
            String [] str = sc.nextLine().split(" ");
            switch(str[0].toLowerCase()){
                case "addstudent":
                    if(str.length != 5){
```

```
        System.out.println("Incorrect usage");
        System.out.println("Usage: addStudent <Name> <RollNo> <CGPA> <year>");
        break;
    }
    String Name = str[1];
    int RollNo = Integer.parseInt(str[2]);
    Float CGPA = Float.parseFloat(str[3]);
    int year =Integer.parseInt(str[4]);
    student.addStudent(Name , RollNo , CGPA , year);
    break;
case "removestudent":
if(str.length != 2){
    System.out.println("Incorrect usage");
    System.out.println("Usage : removeStudent <RollNo>");
    break;
}
    int remove = Integer.parseInt(str[1]);
    student.removeStudent(remove);
    break;
case "search":
    int search = Integer.parseInt(str[1]);
    student.Search(search);
    break;
case "display":
    student.Display();
    break;
case "exit":
    System.out.println("Thank-You");
    return;
default:
    System.out.println("Invalid operation");
```

```
            break;
        }
      }
    }
}
```

Solution Using the Program

The **Student_Record_System** program solves this problem effectively:

1. **Data Structure Used:**
    - Singly Linked List
        - Each node stores student details (Name, RollNo, CGPA, Year) and a pointer to the next node.
        - Linked list allows dynamic addition and deletion without needing to resize arrays.
2. **Operations Supported:**

Operation	How It Works
addStudent	Adds a new node at the end of the linked list, preserving insertion order.
removeStudent	Traverses the list, finds the node with the given roll number, and removes it by adjusting the pointers.
search	Traverses the list and prints the details of the student with the given roll number.
display	Iterates through the entire linked list and prints all student records.
exit	Terminates the program gracefully.

3. **Input Handling:**

- Commands are entered in **simple CLI format**:
- addStudent Karthik 3063 8.59 3
- removeStudent 3063
- search 3063

- display
- exit
- Validates input length to avoid errors.
- Case-insensitive commands for user convenience.

4. **Benefits of the Solution:**
   - Efficient management of dynamic student data.
   - Easy addition, deletion, and search operations.
   - Clear, organized display of all student records.
   - Simple CLI interface — easy for administrators to use.