**ANNAMACHARYA INSTITUTE OF TECHNOLOGY**

**AND SCIENCES, TIRUPATI**

**Department of Artificial Intelligence and Data Science**

**PROJECT REPORT**

# On

**Student Record System using Java (LinkedList /**

**ArrayList)**

**Submitted by:**

Name: Y. HARITHA

Roll No: 23AK1A3025

Year/Semester: 3rd Year, 1st Semester

# Introduction

A Student Record System is a simplified version of real-world student management applications that demonstrates fundamental operations such as adding, updating, deleting, and displaying student details. The objective of this project is not only to replicate basic functionalities but also to showcase how object-oriented programming concepts and data structures (LinkedList/ArrayList) can be applied in developing real-life systems.

# Literature Survey

Existing student management systems are highly complex, secure, and integrated with real-time databases. They provide attendance, marks management, report generation, and more. However, for educational purposes, a lightweight Student Record System is sufficient to illustrate core principles such as record management, searching, and data storage. Unlike enterprise-level systems, this project does not use external databases but can be extended for real-world applications.

# Objectives

* Implement core student record operations: add, search, update, delete.
* Maintain academic records such as marks and grades.
* Provide a user-friendly console interface.
* Demonstrate object-oriented programming and use of LinkedList/ArrayList.

# System Requirements

Hardware Requirements:

* Processor: Intel i3 or above
* RAM: 4GB minimum
* Storage: 250 MB free disk space

Software Requirements:

* Operating System: Windows/Linux/MacOS
* JDK (Java Development Kit) 8 or higher
* IDE: Eclipse, IntelliJ IDEA, NetBeans, or Visual Studio Code

# System Analysis

Problem Statement: Manual student record management is slow, error-prone, and lacks automation. This project automates basic student record functions.

Scope: Limited to console operations with no persistent storage. Suitable for learning and small-scale demos.

Feasibility: Technically feasible using Java OOP concepts and data structures. Can be extended with GUI and databases in the future.

# System Design

The Student Record System is designed using Object-Oriented Programming principles. The system consists of two main classes: Student and StudentManager.

* Student: Represents individual student records with ID, name, course, and marks.
* StudentManager: Manages multiple students using LinkedList/ArrayList.- Console Interface: Provides a menu-driven interaction for the user.

# Implementation

Example Java Code:

import java.util.ArrayList;

import java.util.Scanner;

// Class representing each student

class Student {

String id;

String name;

String course;

double marks;

// Constructor

Student(String id, String name, String course, double marks) {

this.id = id;

this.name = name;

this.course = course;

this.marks = marks;

}

// Display student details

public void display() {

System.out.println("ID: " + id + ", Name: " + name +

", Course: " + course + ", Marks: " + marks);

}

}

// Class managing student operations

class StudentManager {

ArrayList<Student> students = new ArrayList<>();

Scanner sc = new Scanner(System.in);

// Add new student

public void addStudent() {

System.out.print("Enter ID: ");

String id = sc.next();

System.out.print("Enter Name: ");

String name = sc.next();

System.out.print("Enter Course: ");

String course = sc.next();

System.out.print("Enter Marks: ");

double marks = sc.nextDouble();

students.add(new Student(id, name, course, marks));

System.out.println("✅ Student added successfully!");

}

// Display all students

public void displayStudents() {

if (students.isEmpty()) {

System.out.println("No students to display.");

} else {

System.out.println("\n--- Student List ---");

for (Student s : students) {

s.display();

}

}

}

// Delete student by ID

public void deleteStudent(String id) {

boolean removed = students.removeIf(s -> s.id.equals(id));

if (removed) {

System.out.println("✅ Student removed successfully.");

} else {

System.out.println("⚠️ No student found with ID: " + id);

}

}

}

// Main class

public class StudentRecordSystem {

public static void main(String[] args) {

StudentManager manager = new StudentManager();

Scanner sc = new Scanner(System.in);

boolean loop = true;

while (loop) {

System.out.println("\n--- Student Record System Menu ---");

System.out.println("1. Add Student");

Sys

# Sample Output

--- Student Record System Menu --1. Add Student

1. Display Students
2. Delete Student
3. Exit

Enter choice: 1

Enter ID: S101

Enter Name: John

Enter Course: CSE Enter Marks: 88.5 Student added successfully!

Enter choice: 2

ID: S101, Name: John, Course: CSE, Marks: 88.5

# Applications

* Educational use for demonstrating Java OOP and data structures.
* Useful for training students in software development.
* Can be extended into a full-fledged student management system.
* Helpful as a base for learning GUI/Database integration.

# Conclusion

This Student Record System is a robust educational project showcasing object-oriented programming and data structures in Java. It helps students bridge the gap between theory and practice and provides a foundation to develop advanced systems with persistence, GUI, and networking.