

- **Introduction**
- **Feasibility Study**
- **Requirement Specification**
- **System Design**
- **UML Diagrams**
- **Implementation Details**
- **Testing**
- **Limitations of the System**
- **Future Scope and Enhancements**
- **Conclusion**

*Full Source Code

2. Introduction

2.1 Problem Statement

This system aims to automate the storage, retrieval, and management of student information using simple data structures.

2.2 Project Objectives

- Store and manage student records
- Allow searching, viewing, and deletion of records
- Implement basic data structures like linked lists
- Provide a console-based interface

2.3 Scope of the System

The system is designed for academic purposes and handles student details like roll number, name, department and marks. It is a command-line tool meant for small-scale record keeping.

3. Feasibility Study

3.1 Technical Feasibility

Built using Java (console-based), relying on arrays or linked lists. No advanced libraries needed.

3.2 Operational Feasibility

Simple menu-driven interface. No prior technical knowledge required to operate.

3.3 Economic Feasibility

No cost. All tools (e.g., Java, IDE) are freely available.

3.4 Time Feasibility

Can be developed within 5–7 days with proper planning.

4. Requirement Specification

4.1 Functional Requirements

- Add new student
- View all students
- Search student by roll number
- Delete student by roll number
- Exit system

4.2 Non-Functional Requirements

- Fast execution
- Simple UI
- Proper data handling and validation

System Design

5.1 Data Structures Used

- **Linked List** to store student records
- **Nodes** represent individual student objects

5.2 System Architecture / Control Flow

- Main menu handles user input
- Functions are called based on user choice
- Data flows between the menu, operations, and linked list nodes.

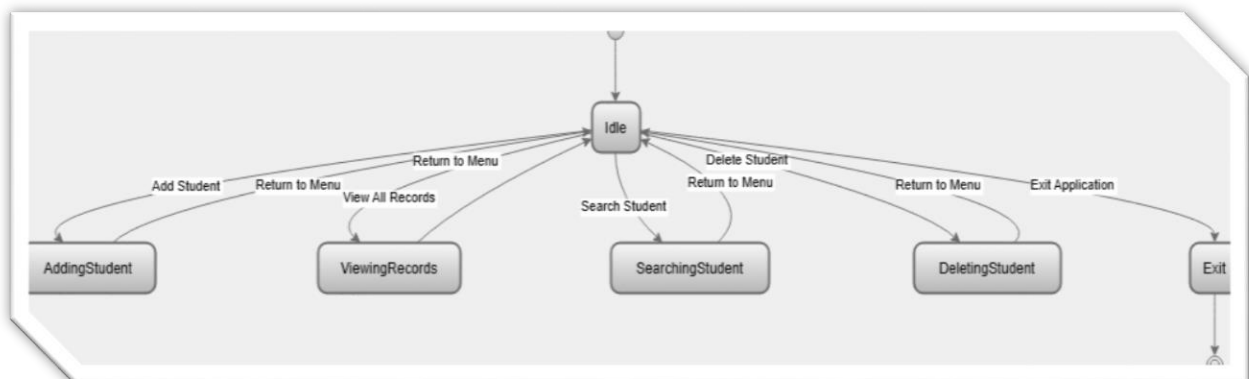
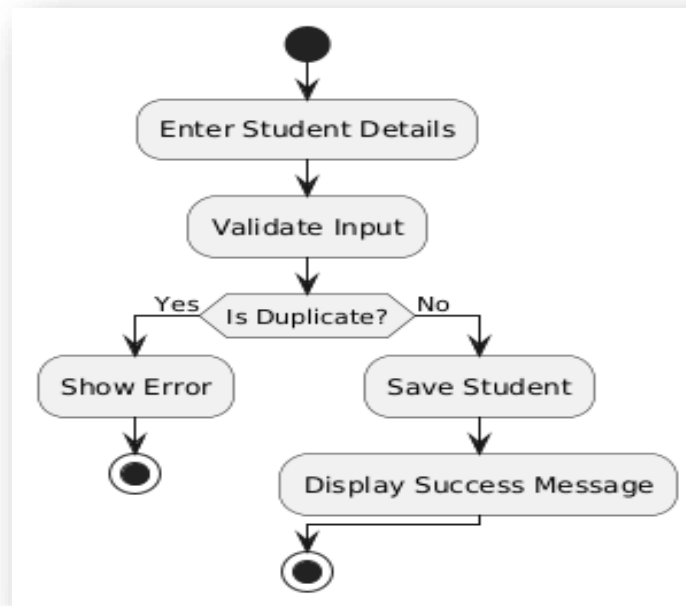
UML Diagrams

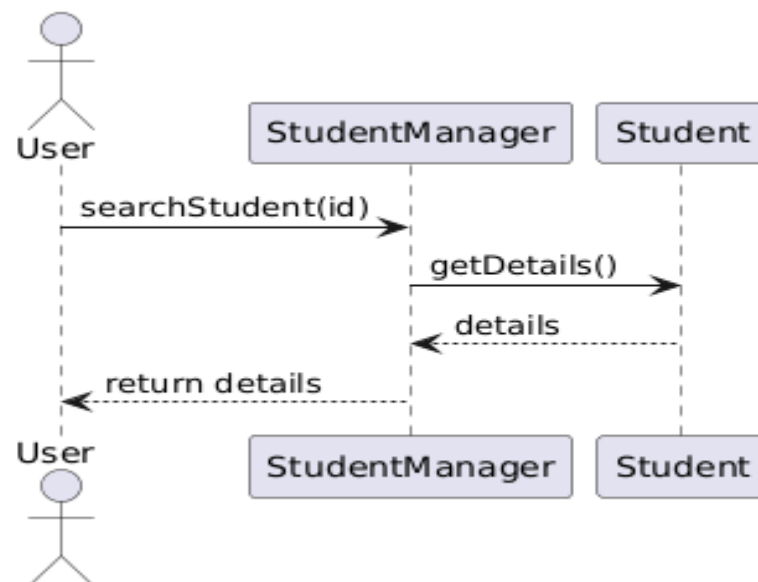
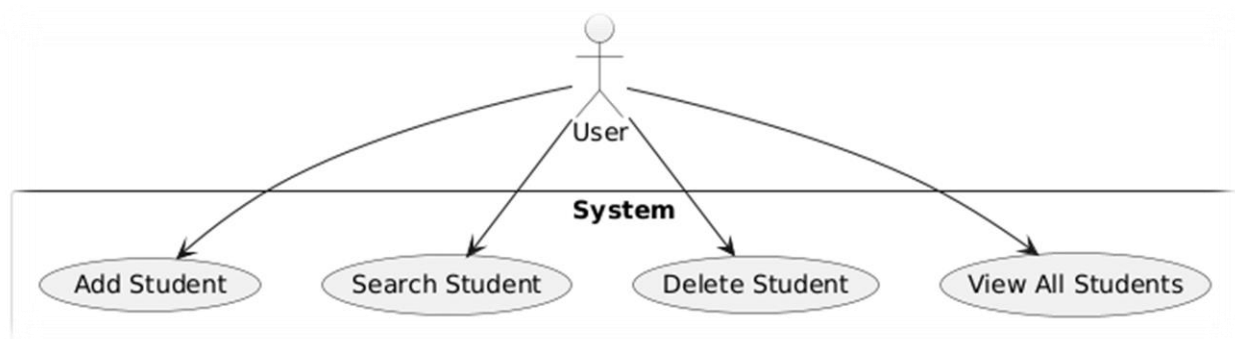
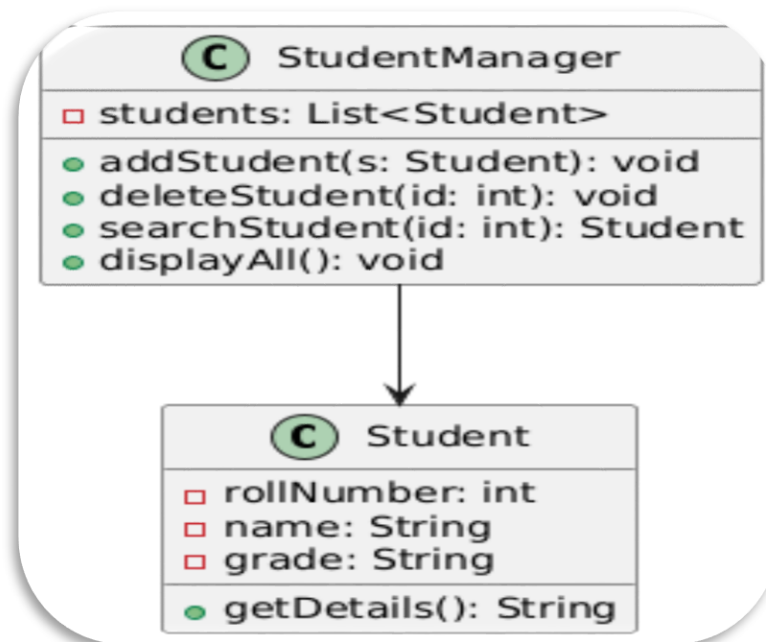
6.1 Use Case Diagram

Shows interactions between user (Admin) and system features like add, search, view, delete.

6.2 Activity Diagram

Visualizes the step-by-step flow of user choices and operations performed.





6.3 Sequence Diagram

Illustrates the order of method calls and object interactions (e.g., main → insert → display).

7. Implementation Details

7.1 Tools and Technologies

- Java
- Any Java IDE (e.g., Eclipse, IntelliJ IDEA)

7.2 Key Code Snippets and Explanations

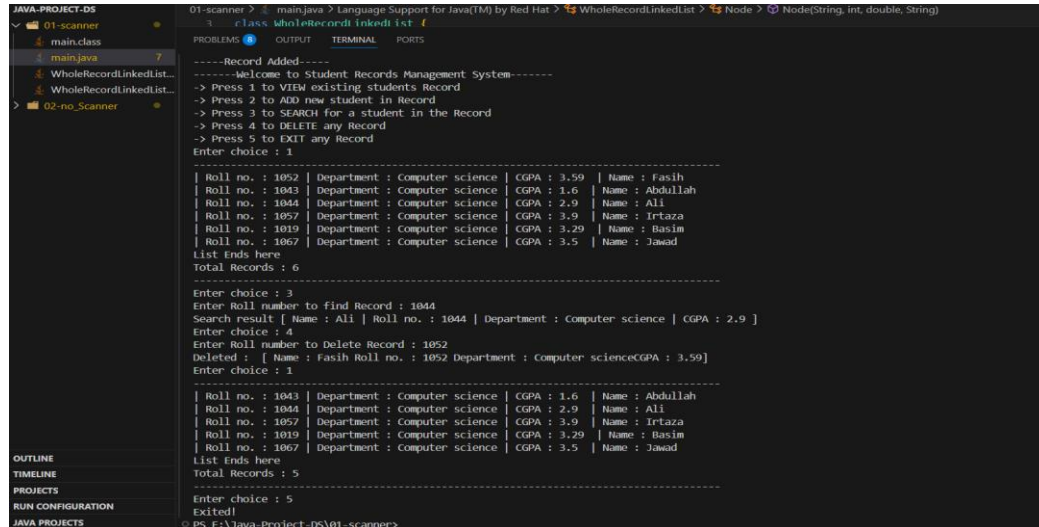
- **addNewStudent()** – adds new student to list
- **searchingStudentByRollnoMethod()** – finds student by roll
- **DeletingStudentByRollno()** -Deleted particular Node(Record)
- **printingList()** – prints all records

7.3 Program Flow Explanation

User selects an option → corresponding function executes → list is updated/displayed

8. Testing

8.1 Test Cases and Results



```
01-scanner > 1. main.java > Language Support for Java(TM) by Red Hat > WholeRecordLinkedList > Node > Node(String, int, double, String)
class WholeRecordLinkedList {
    PROBLEMS OUTPUT TERMINAL PORTS

    -----Record Added-----
    -----Welcome to Student Records Management System-----
    -> Press 1 to VIEW existing students Record
    -> Press 2 to ADD new student in Record
    -> Press 3 to SEARCH for a student in the Record
    -> Press 4 to DELETE any Record
    -> Press 5 to EXIT any Record
    Enter choice : 1

    | Roll no. : 1052 | Department : Computer science | CGPA : 3.59 | Name : Fasih
    | Roll no. : 1043 | Department : Computer science | CGPA : 1.6 | Name : Abdullah
    | Roll no. : 1044 | Department : Computer science | CGPA : 2.9 | Name : Ali
    | Roll no. : 1057 | Department : Computer science | CGPA : 3.9 | Name : Irtaza
    | Roll no. : 1019 | Department : Computer science | CGPA : 3.29 | Name : Basim
    | Roll no. : 1067 | Department : Computer science | CGPA : 3.5 | Name : Jawad
    List Ends here
    Total Records : 6

    Enter choice : 3
    Enter Roll number to find Record : 1044
    Search result [ Name : Ali | Roll no. : 1044 | Department : Computer science | CGPA : 2.9 ]
    Enter choice : 4
    Enter Roll number to Delete Record : 1052
    Deleted : [ Name : Fasih Roll no. : 1052 Department : Computer scienceCGPA : 3.59]
    Enter choice : 1

    | Roll no. : 1043 | Department : Computer science | CGPA : 1.6 | Name : Abdullah
    | Roll no. : 1044 | Department : Computer science | CGPA : 2.9 | Name : Ali
    | Roll no. : 1057 | Department : Computer science | CGPA : 3.9 | Name : Irtaza
    | Roll no. : 1019 | Department : Computer science | CGPA : 3.29 | Name : Basim
    | Roll no. : 1067 | Department : Computer science | CGPA : 3.5 | Name : Jawad
    List Ends here
    Total Records : 5

    Enter choice : 5
    Exited!
    © PS: E:\Java-Project-DS\01-scanner>
```

```
int searchingRollNumber = 0;
int deletingRollno = 0;

Scanner sc = new Scanner(System.in);
WholeRecordLinkedList studentRecordList = new WholeRecordLinkedList();
studentRecordList.insertingStudentAtStart(studentName:"Fasih",rollNumber:1052,cgpa:3.59,department:"Computer science");
studentRecordList.insertingStudentAtStart(studentName:"Abdullah",rollNumber:1043,cgpa:1.6,department:"Computer science");
studentRecordList.insertingStudentAtStart(studentName:"Ali",rollNumber:1044,cgpa:2.9,department:"Computer science");
studentRecordList.insertingStudentAtStart(studentName:"Irtaza",rollNumber:1057,cgpa:3.9,department:"Computer science");
studentRecordList.insertingStudentAtStart(studentName:"Basim",rollNumber:1019,cgpa:3.29,department:"Computer science");
studentRecordList.insertingStudentAtStart(studentName:"Jawad",rollNumber:1067,cgpa:3.5,department:"Computer science");

// studentRecordList.traversingMethod();
int userChose=0;
System.out.println(x:"-----Welcome to Student Records Management System-----");
System.out.println(x:"-> Press 1 to VIEW existing students Record");
System.out.println(x:"-> Press 2 to ADD new student in Record");
```

9. Limitations of the System

- No GUI
- Data not stored permanently (lost after closing)
- No Database integrated yet
- No sorting or advanced filtering

Future Scope and Enhancements

- Make frontend for this in Tailwind & JavaScript
- Use database for persistent storage
- Add features like update, edit, etc.

11. Conclusion

This project served as a great learning experience for applying Java concepts and understanding how data structures like linked lists work in real applications.

**Full source code : <https://github.com/fasihhhh/Student-Record-Management-System>*