PROJECT REPORT

ON

STUDENT MANAGEMENT SYSTEM

SUBMITTED BY :

NAME **:** AISHWARYA VERMA

ROLL.NO. **:** 22051051

DEPARTMENT **:** COMPUTER SCIENCE AND ENGINEERING

SESSION **:** 2022-2026

COLLEGE **:** KALINGA INSTITUTE OF INDUSTRISAL TECHNOLOGY,BHUBANESWAR,ODISHA.

TRAINING CONDUCTED

AT

CENTRAL COALFIELDS LIMITED

SYSTEM DEPARTMENT

DARBHANGA HOUSE

RANCHI

JHARKHAND

DURATION: 1 MONTH (4 WEEKS)

CERTIFICATE

This is to certify that the project on ***STUDENT MANAGEMENT SYSTEM*** with special reference to Central Coalfields Ltd., Headquarter "*Darbhanga House*" has been prepared by AISHWARYA VERMA , Roll No. 22051051 course B.Tech Degree from ***KALINGA INSTITIUTE OF INDUSTRIAL TECHNOLOGY, BHUBANESWAR, ODISHA,*** *under our supervision and guidance.* I appreciate his skill, hard work and sense of commitment in preparation of the project.

Gaurav Kumar Singh
Asst. Manager (system)
CCL Headquarter
Dharbhanga House

G.M(HRD)
CCL Headquarter
Dharbhanga House
Ranchi

AIM OF THE PROJECT

The aim of the Student Management System (SMS) is to provide an efficient, user-friendly and reliable software design of managing student details with an educational institution. The system is designed to simplify the task of adding, viewing, searching and removing student records, ensuring accurate and up-to-date information is always. By streamlining these administrative process, the SMS aims to enhance productivity and reduce errors and facilitate better decision-making for educators and administrators. Additionally, the system seeks to improve the data security and accessibility, providing a robust platform for managing essential student information.

STUDENT MANAGEMENT SYSTEM

ABSTRACT:

Requirements definition and management is recognized as a necessary step in the delivery of successful system and software projects. Discipline is also required by standards, regulations, and quality improvement initiatives. Creating and managing requirements is a challenge for IT, systems, and product development projects or indeed for any activity where you have to manage a contractual relationship. Organizations need to effectively define and manage requirements to ensure they are meeting the needs of the customer while proving compliance and staying on schedule and within budget. The impact of poorly expressed requirements can bring a business out of compliance or even cause significant operational disruptions. Requirements definition and management is an activity that can deliver a high, fast return on investment.

The "STUDENT MANAGEMENT SYSTEM" undertaken as a project is based on relevant technologies. The main aim of this project is to develop an overview of software for student management within an educational institution. This project aims to develop software that simplifies the management of student information. It has been developed to carry out processes easily and quickly, which is not possible with manual systems. This project is developed using C language, and hence it provides a comprehensive solution for the current student management system.

REQUIREMENT ANALYSIS :

Requirement are prone to issues of ambiguity, incompleteness,and inconsistency .Techniques such as rigorous inspection have been show to help deal with these issues. Ambiguities, incompleteness, and inconsistenceies that can be resolved in the requirements phase typically cost orders of magnitude less to corrent than when these same issues are found in later stages of product development. Requiret the ments analysis strives to address these issues.

Take a long time to produce Begin to limit the implementation option available are costly to produce Requirements for both the system and the software are documented and reviewed with the customer.

PROCESS FLOW DIARAM FOR THE BANK MANAGEMENT SYSTEM:

This diagram, represents a student management process, which maintains students record. In this example, a person can add student in the record, remove student, search for student and view all students in the record.

MODULAR DESCRIPTION:

- Add a new Student
- View All Students
- Show details of a Student
- Delete Student details

Module 1 – Add a new Student :

A module new account is literally the form for the administrator to add a new Student. A new student is added with the following details, with the given student ID.

- Student ID
- Student's Name
- Student's Course
- Student's enrollment year
- Email
- Address

Code:

```
static void addStudent() {

    JFrame frame = new JFrame("Add Student");

    frame.setSize(400, 400);
```

```
JLabel label1 = new JLabel("Enter Student ID:");
JTextField studentIdField = new JTextField();
JPanel panel1 = new JPanel(new GridLayout(1, 2));
panel1.add(label1);
panel1.add(studentIdField);

JLabel label2 = new JLabel("Enter Student Name:");
JTextField studentNameField = new JTextField();
JPanel panel2 = new JPanel(new GridLayout(1, 2));
panel2.add(label2);
panel2.add(studentNameField);

JLabel label3 = new JLabel("Enter Course:");
JTextField courseField = new JTextField();
JPanel panel3 = new JPanel(new GridLayout(1, 2));
panel3.add(label3);
panel3.add(courseField);

JLabel label4 = new JLabel("Enter Year of Enrollment:");
JTextField yearField = new JTextField();
JPanel panel4 = new JPanel(new GridLayout(1, 2));
panel4.add(label4);
panel4.add(yearField);
```

```
JLabel label5 = new JLabel("Enter Email:");
JTextField emailField = new JTextField();
JPanel panel5 = new JPanel(new GridLayout(1, 2));
panel5.add(label5);
panel5.add(emailField);

JLabel label6 = new JLabel("Enter Address:");
JTextField addressField = new JTextField();
JPanel panel6 = new JPanel(new GridLayout(1, 2));
panel6.add(label6);
panel6.add(addressField);

JButton submitButton = new JButton("Submit");
submitButton.addActionListener(e -> {
    String id = studentIdField.getText();
    String name = studentNameField.getText();
    String course = courseField.getText();
    String year = yearField.getText();
    String email = emailField.getText();
    String address = addressField.getText();

    try (BufferedReader reader = new BufferedReader(new FileReader("Student.xls"));
         BufferedWriter writer = new BufferedWriter(new FileWriter("NewRec.xls"))) {
        String line;
        while ((line = reader.readLine()) != null) {
```

```
                writer.write(line + "\n");
            }
            writer.write(String.join("\t", id, name, course, year, email, address) + "\n");
        } catch (IOException ex) {
            ex.printStackTrace();
        }

        deleteFile();
        renameFile();
        JOptionPane.showMessageDialog(null, "Details have been updated!");
        frame.dispose();
    });

    frame.setLayout(new GridLayout(7, 1));
    frame.add(panel1);
    frame.add(panel2);
    frame.add(panel3);
    frame.add(panel4);
    frame.add(panel5);
    frame.add(panel6);
    frame.add(submitButton);

    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}
```

Module 2 – View all students:

In this module the manager can see all the employees and their details at one place.

Code:

```
public static void viewAllStudents() {
    JFrame frame = new JFrame("View All Students");
    frame.setSize(500, 400);

    JTextArea textArea = new JTextArea();
    textArea.setEditable(false);
    frame.add(new JScrollPane(textArea), BorderLayout.CENTER);

    try (BufferedReader reader = new BufferedReader(new FileReader("Student.xls"))) {
       textArea.read(reader, null);
    } catch (IOException ex) {
       ex.printStackTrace();
    }

    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  }
```

Module 3 – Show details of all students:

Using this view student module the user can search for an student using its student id and can see the student details.

Code:

```
public static void viewStudent() {
```

```
JFrame frame = new JFrame("View Student");
frame.setSize(500, 400);
frame.setLayout(new BorderLayout());

JTextField studentIdField = new JTextField();
JPanel inputPanel = new JPanel(new GridLayout(1, 2));
JLabel inputLabel = new JLabel("Enter Student ID:");
inputPanel.add(inputLabel);
inputPanel.add(studentIdField);
frame.add(inputPanel, BorderLayout.NORTH);

JTextArea resultArea = new JTextArea();
resultArea.setEditable(false);
frame.add(new JScrollPane(resultArea), BorderLayout.CENTER);

JButton searchButton = new JButton("Search");
frame.add(searchButton, BorderLayout.SOUTH);

searchButton.addActionListener(e -> {
    String studentId = studentIdField.getText();
    try (BufferedReader reader = new BufferedReader(new FileReader("Student.xls"))) {
        String line;
        boolean found = false;
        while ((line = reader.readLine()) != null) {
            String[] records = line.split("\t");
```

```
            if (records[0].equals(studentId)) {
                resultArea.setText(String.join("\n", "Student ID: " + records[0],
                        "Student Name: " + records[1],
                        "Course: " + records[2],
                        "Year of Enrollment: " + records[3],
                        "Email: " + records[4],
                        "Address: " + records[5]));
                found = true;
                break;
            }
        }
        if (!found) {
            resultArea.setText("Student not found.");
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
});

frame.setVisible(true);
frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}
```

Module 4 – Delete student details:

Using this module the user can remove an student with its details. It can be done by searching the student id.

Code:

```
public static void removeStudent() {
        JFrame frame = new JFrame("Remove Student");
        frame.setSize(500, 400);
        frame.setLayout(new BorderLayout());

        JTextField studentIdField = new JTextField();
        JPanel inputPanel = new JPanel(new GridLayout(1, 2));
        JLabel inputLabel = new JLabel("Enter Student ID:");
        inputPanel.add(inputLabel);
        inputPanel.add(studentIdField);
        frame.add(inputPanel, BorderLayout.NORTH);

        JTextArea resultArea = new JTextArea();
        resultArea.setEditable(false);
        frame.add(new JScrollPane(resultArea), BorderLayout.CENTER);

        JButton deleteButton = new JButton("Delete");
        frame.add(deleteButton, BorderLayout.SOUTH);
       deleteButton.addActionListener(e -> {
           String studentId = studentIdField.getText();
           try (BufferedReader reader = new BufferedReader(new FileReader("Student.xls"));
                BufferedWriter writer = new BufferedWriter(new FileWriter("NewRec.xls"))) {
               String line;
               boolean found = false;
```

```
            while ((line = reader.readLine()) != null) {
                if (!line.split("\t")[0].equals(studentId)) {
                    writer.write(line + "\n");
                } else {
                    found = true;
                }
            }
            if (found) {
                JOptionPane.showMessageDialog(null, "Student record deleted.");
            } else {
                JOptionPane.showMessageDialog(null, "Student not found.");
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        deleteFile();
        renameFile();
    });


    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}}
```

FUNCTION POINT ANALYSIS:

Function Point (FP) Analysis is used to estimate the size and complexity of a software system. It involves identifying and categorizing the system's functions into five main components: External

Inputs (EI), External Outputs (EO), External Inquiries (EQ), Internal Logical Files (ILF), and External Interface Files (EIF).

1. External Inputs (EI):

- Add Student: 1 FP
- Remove Student: 1 FP

2. External Outputs (EO):

- View All Students: 1 FP

3. External Inquiries (EQ):

- Search Student: 1 FP

4. Internal Logical Files (ILF):

- Student Data File: 1 FP

5. External Interface Files (EIF):

- None

Total Function Points

Total FP = EI + EO + EQ + ILF + EIF = 1 + 1 + 1 + 1 + 0 = 4 FPs

Maintenance:

Regular Updates

1. Bug Fixes:

- Regularly review and fix any reported bugs to ensure smooth operation.

2. Feature Enhancements:

- Implement new features based on user feedback and changing requirements.

Data Backup

1. Automated Backups:

- Implement regular automated backups of student data to prevent data loss.

2. Manual Backups:

- Encourage manual backups before performing major updates or changes.

Performance Monitoring

1. Logging:

- Implement logging to monitor application performance and identify bottlenecks.

2. Optimization:

- Regularly review and optimize code to improve performance and response times.

User Training

1. Documentation:

- Provide comprehensive user manuals and documentation for reference.

2. Training Sessions:

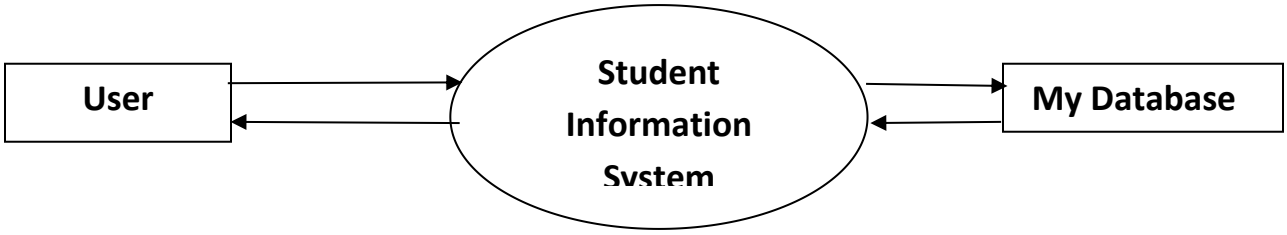
- Conduct training sessions for administrator and users to familiarize them with the system.

By following these steps, the SMS can be effectively maintained, ensuring its reliability and usability over time.

DFD DIAGRAM FOR BANK MANAGEMENT SYSYTEM

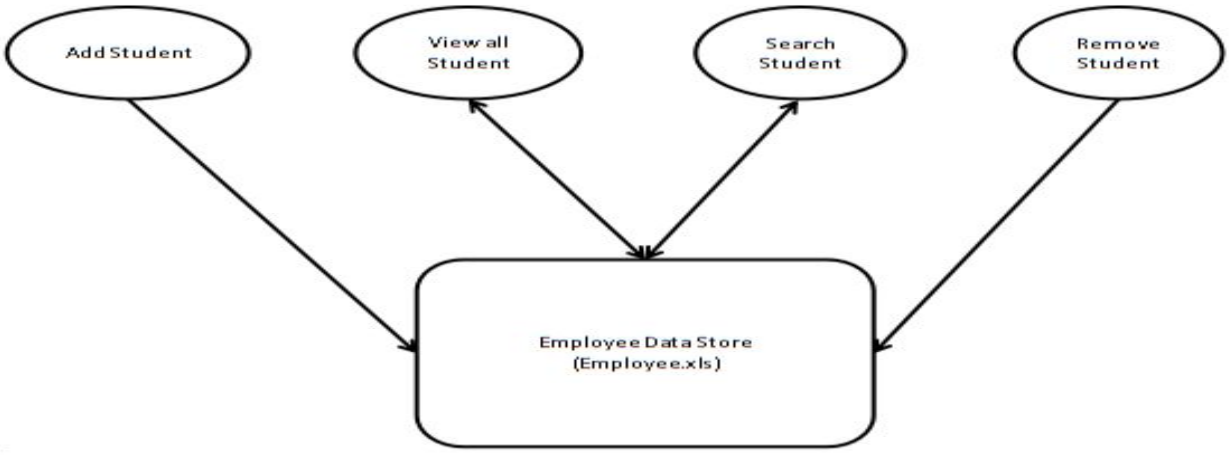
Level 0: Context Diagram

The context diagram shows the system as a single process and its interactions with external entities.



Level 1: Detailed DFD

The Level 1 DFD provides more detail, showing the internal processes of the EMS and how data flows between them.



Descriptions of Processes

Add Student (1.0):

- The manager inputs new student details.
- The system updates the student Data Store with the new details.

View all Students (2.0):

- The administrator requests a list of all students.
- The system retrieves and displays all student details from the student Data Store.

Search Student (3.0):

- The administrator enters an student ID.
- The system searches the Student Data Store for the matching student and displays the details.

Remove Student (4.0):

- The administrator inputs the ID of the student to be removed.
- The system deletes the student's details from the Student Data Store.

.

This DFD provides a comprehensive overview of the Students Management System's functionality, illustrating how data flows between the administrator, the system's processes, and the Student Data Store.

Source Code

```
import java.io.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class StudentManagementSystem {
    static String USER_ID = "aishwaryaVer";
    static String PASSWORD = "student123";

    public static void main(String[] args) {
        JFrame loginFrame = new JFrame("Login");
        loginFrame.setSize(400, 350);
        loginFrame.setLayout(null);

        JLabel userLabel = new JLabel("User ID:");
        JTextField userField = new JTextField();
        userLabel.setBounds(40, 40, 100, 30);
        userField.setBounds(150, 40, 100, 30);

        JLabel passwordLabel = new JLabel("Password:");
        JPasswordField passwordField = new JPasswordField();
        passwordLabel.setBounds(40, 80, 100, 30);
        passwordField.setBounds(150, 80, 100, 30);
```

```
        JButton loginButton = new JButton("Login");
        loginButton.setBounds(150, 120, 100, 30);
        loginButton.addActionListener(e -> {
            String userId = userField.getText();
            String password = new String(passwordField.getPassword());

            if (USER_ID.equals(userId) && PASSWORD.equals(password)) {
                loginFrame.dispose();
                showMainWindow();
            } else {
                JOptionPane.showMessageDialog(loginFrame, "Invalid credentials", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        });

        loginFrame.add(userLabel);
        loginFrame.add(userField);
        loginFrame.add(passwordLabel);
        loginFrame.add(passwordField);
        loginFrame.add(loginButton);

        loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        loginFrame.setVisible(true);
    }
```

```
    static void showMainWindow() {
        JFrame frame = new JFrame("Student Management System");
        frame.setSize(700, 500);
        frame.setLayout(new GridLayout(3, 1));

        JLabel display = new JLabel("Welcome to Student Management System",
SwingConstants.CENTER);
        display.setFont(new Font("Times New Roman", Font.BOLD, 30));
        display.setForeground(Color.BLUE);
        frame.add(display);

        JPanel panel = new JPanel();
        frame.add(panel);

        addButton(panel, "Add a new Student", e -> addStudent());
        addButton(panel, "View all Students", e -> viewAllStudents());
        addButton(panel, "Show details of a Student", e -> viewStudent());
        addButton(panel, "Delete Student details", e -> removeStudent());

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }

    static void addButton(JPanel panel, String text, ActionListener listener) {
        JButton button = new JButton(text);
```

```
    button.setBackground(Color.ORANGE);
    button.setForeground(Color.DARK_GRAY);
    button.addActionListener(listener);
    panel.add(button);
}

static void addStudent() {
    JFrame frame = new JFrame("Add Student");
    frame.setSize(400, 400);

    JLabel label1 = new JLabel("Enter Student ID:");
    JTextField studentIdField = new JTextField();
    JPanel panel1 = new JPanel(new GridLayout(1, 2));
    panel1.add(label1);
    panel1.add(studentIdField);

    JLabel label2 = new JLabel("Enter Student Name:");
    JTextField studentNameField = new JTextField();
    JPanel panel2 = new JPanel(new GridLayout(1, 2));
    panel2.add(label2);
    panel2.add(studentNameField);

    JLabel label3 = new JLabel("Enter Course:");
    JTextField courseField = new JTextField();
    JPanel panel3 = new JPanel(new GridLayout(1, 2));
```

```
panel3.add(label3);
panel3.add(courseField);

JLabel label4 = new JLabel("Enter Year of Enrollment:");
JTextField yearField = new JTextField();
JPanel panel4 = new JPanel(new GridLayout(1, 2));
panel4.add(label4);
panel4.add(yearField);

JLabel label5 = new JLabel("Enter Email:");
JTextField emailField = new JTextField();
JPanel panel5 = new JPanel(new GridLayout(1, 2));
panel5.add(label5);
panel5.add(emailField);

JLabel label6 = new JLabel("Enter Address:");
JTextField addressField = new JTextField();
JPanel panel6 = new JPanel(new GridLayout(1, 2));
panel6.add(label6);
panel6.add(addressField);

JButton submitButton = new JButton("Submit");
submitButton.addActionListener(e -> {
    String id = studentIdField.getText();
    String name = studentNameField.getText();
```

```
    String course = courseField.getText();
    String year = yearField.getText();
    String email = emailField.getText();
    String address = addressField.getText();

    try (BufferedReader reader = new BufferedReader(new FileReader("Student.xls"));
         BufferedWriter writer = new BufferedWriter(new FileWriter("NewRec.xls"))) {
        String line;
        while ((line = reader.readLine()) != null) {
            writer.write(line + "\n");
        }
        writer.write(String.join("\t", id, name, course, year, email, address) + "\n");
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    deleteFile();
    renameFile();
    JOptionPane.showMessageDialog(null, "Details have been updated!");
    frame.dispose();
});

frame.setLayout(new GridLayout(7, 1));
frame.add(panel1);
frame.add(panel2);
```

```
    frame.add(panel3);
    frame.add(panel4);
    frame.add(panel5);
    frame.add(panel6);
    frame.add(submitButton);

    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}

static void deleteFile() {
    File oldFile = new File("Student.xls");
    oldFile.delete();
}

static void renameFile() {
    File newFile = new File("NewRec.xls");
    File oldFile = new File("Student.xls");
    newFile.renameTo(oldFile);
}

public static void viewAllStudents() {
    JFrame frame = new JFrame("View All Students");
    frame.setSize(500, 400);
```

```
    JTextArea textArea = new JTextArea();
    textArea.setEditable(false);
    frame.add(new JScrollPane(textArea), BorderLayout.CENTER);

    try (BufferedReader reader = new BufferedReader(new FileReader("Student.xls"))) {
        textArea.read(reader, null);
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}

public static void viewStudent() {
    JFrame frame = new JFrame("View Student");
    frame.setSize(500, 400);
    frame.setLayout(new BorderLayout());

    JTextField studentIdField = new JTextField();
    JPanel inputPanel = new JPanel(new GridLayout(1, 2));
    JLabel inputLabel = new JLabel("Enter Student ID:");
    inputPanel.add(inputLabel);
    inputPanel.add(studentIdField);
    frame.add(inputPanel, BorderLayout.NORTH);
```

```
JTextArea resultArea = new JTextArea();
resultArea.setEditable(false);
frame.add(new JScrollPane(resultArea), BorderLayout.CENTER);

JButton searchButton = new JButton("Search");
frame.add(searchButton, BorderLayout.SOUTH);

searchButton.addActionListener(e -> {
    String studentId = studentIdField.getText();
    try (BufferedReader reader = new BufferedReader(new FileReader("Student.xls"))) {
        String line;
        boolean found = false;
        while ((line = reader.readLine()) != null) {
            String[] records = line.split("\t");
            if (records[0].equals(studentId)) {
                resultArea.setText(String.join("\n", "Student ID: " + records[0],
                        "Student Name: " + records[1],
                        "Course: " + records[2],
                        "Year of Enrollment: " + records[3],
                        "Email: " + records[4],
                        "Address: " + records[5]));
                found = true;
                break;
            }
```

```
            }
            if (!found) {
                resultArea.setText("Student not found.");
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    });

    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}

public static void removeStudent() {
    JFrame frame = new JFrame("Remove Student");
    frame.setSize(500, 400);
    frame.setLayout(new BorderLayout());

    JTextField studentIdField = new JTextField();
    JPanel inputPanel = new JPanel(new GridLayout(1, 2));
    JLabel inputLabel = new JLabel("Enter Student ID:");
    inputPanel.add(inputLabel);
    inputPanel.add(studentIdField);
    frame.add(inputPanel, BorderLayout.NORTH);
```

```
JTextArea resultArea = new JTextArea();
resultArea.setEditable(false);
frame.add(new JScrollPane(resultArea), BorderLayout.CENTER);

JButton deleteButton = new JButton("Delete");
frame.add(deleteButton, BorderLayout.SOUTH);

deleteButton.addActionListener(e -> {
    String studentId = studentIdField.getText();
    try (BufferedReader reader = new BufferedReader(new FileReader("Student.xls"));
         BufferedWriter writer = new BufferedWriter(new FileWriter("NewRec.xls"))) {
        String line;
        boolean found = false;
        while ((line = reader.readLine()) != null) {
            if (!line.split("\t")[0].equals(studentId)) {
                writer.write(line + "\n");
            } else {
                found = true;
            }
        }
        if (found) {
            JOptionPane.showMessageDialog(null, "Student record deleted.");
        } else {
            JOptionPane.showMessageDialog(null, "Student not found.");
        }
```

```
            } catch (IOException ex) {
                ex.printStackTrace();
            }
            deleteFile();
            renameFile();
        });

        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }
}
```

References

- https://youtube.com/playlist?list=PLqleLpAMfxGDVu5tUmUg9jSQUUB8_5DB0&si=vCCi8RZ5xKQ5hD6h
- https://youtu.be/b35mlSPOlJg?si=syoYM51CbTTU0UlC
- https://chat.openai.com/