PROJECT REPORT

ON

EMPLOYEE MANAGEMENT SYSTEM

SUBMITTED BY :

NAME	**:** KINSHUK
Roll No.	**:** 22051081
DEPARTMENT	**:** COMPUTER SCIENCE AND ENGINEERING
SESSION	**:** 2022 - 2026
COLLEGE	**:** KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT) , BHUBANESWAR ,ODISHA.

Training Conducted

at

CENTRAL COALFIELDS LIMITED

System Department

Darbhanga House

Ranchi

Jharkhand

Duration: 1 Month (4 Weeks)

<u>Certificate</u>

This is to certify that the project on ***EMPLOYEE MANAGEMENT SYSTEM*** with special reference to Central Coalfields Ltd., Headquarter "*Darbhanga House*" has been prepared by *Kinshuk*, Roll No. 22051081 course B.Tech Degree from ***KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT) , 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 (Systems)

CCL Headquarter

Dharbhanga House

Ranchi

G.M. (HRD)

CCL Headquarter

Dharbhanga House

Ranchi

Aim of the Project

The aim of the Employee Management System (EMS) is to provide an efficient, user-friendly, and reliable software solution for managing employee details within a company. The system is designed to simplify the tasks of adding, viewing, searching, and removing employee records, ensuring accurate and up-to-date information is always available. By streamlining these administrative processes, the EMS aims to enhance productivity, reduce errors, and facilitate better decision-making for managers. Additionally, the system seeks to improve data security and accessibility, providing a robust platform for managing essential employee information.

EMPLOYEE MANAGEMENT SYSTEM

ABSTRACT:

The Employee Management System (EMS) is a software application designed to maintain employee details within a company. It allows for the addition, removal, search, and viewing of employee information. The EMS simplifies employee record management, providing a user-friendly interface for managers to handle employee data effectively. The system consists of four main modules: Add Employee, View All Employees, Search Employee, and Remove Employee, each designed to handle specific tasks related to employee management.

REQUIREMENT ANALYSIS :

Functional Requirements

1. Add Employee:

- Input fields for employee details: ID, Name, Department, Salary, Hire-date, and Address.
- Store the new employee's information in the system.

2. View All Employees:

- Display a list of all employees with their details.
- Provide a scroll-able and readable format.

3. Search Employee:

- Input field to enter the employee ID.
- Display details of the employee matching the entered ID.

4. Remove Employee:

- Input field to enter the employee ID.
- Remove the employee record matching the entered ID from the system.

Non-Functional Requirements

1. Usability:

- User-friendly interface for easy navigation and interaction.

2. Performance:

- Quick response time for searching and displaying employee details.

3. Reliability:

- Accurate addition, retrieval, and deletion of employee records.

4. Security:

- Secure handling of employee data to prevent unauthorized access.

System Requirements

1. Hardware:

- A computer system with basic input-output peripherals.

2. Software:

- Java Development Kit (JDK) for running the Java application.
- A text file ("Employee.xls") for storing employee data.

Process Flow for The Employee Management System:

This diagram, which maintains employee details . In this example, a person can add an employee to the company , remove an employee search for employee details and view all employees in the company.

Modular Description:

Add Employee

View All Employee

Search Employee

Delete Employee

Module 1 – Add Employee :

A module add employee is literally the form for the manager to add new employees data. A new employee is added with the following details, with the given employee id.

- Employee ID
- Employee Name
- Department
- Salary
- Hire-date
- Address

Code :-

```
public static void addEmployee() {
JFrame frame = new JFrame("Add Employee");
frame.setSize(400, 400);

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

JLabel label2 = new JLabel("Enter Employee Name:");
```

```
JTextField empNameField = new JTextField();

JPanel panel2 = new JPanel(new GridLayout(1, 2));

panel2.add(label2);

panel2.add(empNameField);


JLabel label3 = new JLabel("Enter Department:");

JTextField depField = new JTextField();

JPanel panel3 = new JPanel(new GridLayout(1, 2));

panel3.add(label3);

panel3.add(depField);


JLabel label4 = new JLabel("Enter Salary:");

JTextField salaryField = new JTextField();

JPanel panel4 = new JPanel(new GridLayout(1, 2));

panel4.add(label4);

panel4.add(salaryField);


JLabel label5 = new JLabel("Enter Hire-date:");

JTextField hireDateField = new JTextField();

JPanel panel5 = new JPanel(new GridLayout(1, 2));

panel5.add(label5);

panel5.add(hireDateField);


JLabel label6 = new JLabel("Enter Address of the Employee:");

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 = empIdField.getText();
String name = empNameField.getText();
String dept = depField.getText();
String salary = salaryField.getText();
String hireDate = hireDateField.getText();
String address = addressField.getText();

try (BufferedReader reader = new BufferedReader(new FileReader("Employee.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, dept, salary, hireDate, 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);
}
 public static void deleteFile()
   {
       File oldFile = new File("Employee.xls");
       oldFile.delete();
   }
   public static void renameFile()
   {
       File newFile = new File("NewRec.xls");
       File oldFile = new File("Employee.xls");
       newFile.renameTo(oldFile);
   }
```

Module 2 – View All Employees:

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

Code :-

```
public static void viewAllEmployees() {
       JFrame frame = new JFrame("View All Employees");
```

```
        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("C:\\Users\\KIIT\\Desktop\\Internship\\Employee.xls"))) {

            textArea.read(reader, null);

        } catch (IOException ex) {

            ex.printStackTrace();

        }


        frame.setVisible(true);

        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    }
```

Module 3 – View Employee:

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

Code :-

```
public static void viewEmployee() {

        JFrame frame = new JFrame("View Employee");

        frame.setSize(500, 400);

        frame.setLayout(new BorderLayout());


        JTextField empIdField = new JTextField();

        JPanel inputPanel = new JPanel(new GridLayout(1, 2));

        JLabel inputLabel = new JLabel("Enter Employee ID:");
        inputPanel.add(inputLabel);

        inputPanel.add(empIdField);

        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 empId = empIdField.getText();

            try (BufferedReader reader = new BufferedReader(new
    FileReader("C:\\Users\\KIIT\\Desktop\\Internship\\Employee.xls"))) {

                String line;

                boolean found = false;

                while ((line = reader.readLine()) != null) {

                    String[] records = line.split("\t");

                    if (records[0].equals(empId)) {

                        resultArea.setText(String.join("\n", "Employee ID: " + records[0],

                                "Employee Name: " + records[1],

                                "Department: " + records[2],

                                "Salary: " + records[3],

                                "Hire-date: " + records[4],

                                "Address: " + records[5]));

                        found = true;

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

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

Module 4 – Remove an Employee:

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

Code :-

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

    JTextField empIdField = new JTextField();
    JPanel inputPanel = new JPanel(new GridLayout(1, 2));
    JLabel inputLabel = new JLabel("Enter Employee ID:");
    inputPanel.add(inputLabel);
    inputPanel.add(empIdField);
    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 empId = empIdField.getText();
            try (BufferedReader reader = new BufferedReader(new
FileReader("C:\\Users\\KIIT\\Desktop\\Internship\\Employee.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(empId)) {
                        writer.write(line + "\n");
                    } else {
                        found = true;
                    }
                }
                if (found) {
                    JOptionPane.showMessageDialog(null, "Employee record deleted.");
                } else {
                    JOptionPane.showMessageDialog(null, "Employee 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 Employee: 1 FP
- Remove Employee: 1 FP

2. External Outputs (EO):

- View All Employees: 1 FP

3. External Inquiries (EQ):

- Search Employee: 1 FP

4. Internal Logical Files (ILF):

- Employee 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 employee 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 managers and users to familiarize them with the system.

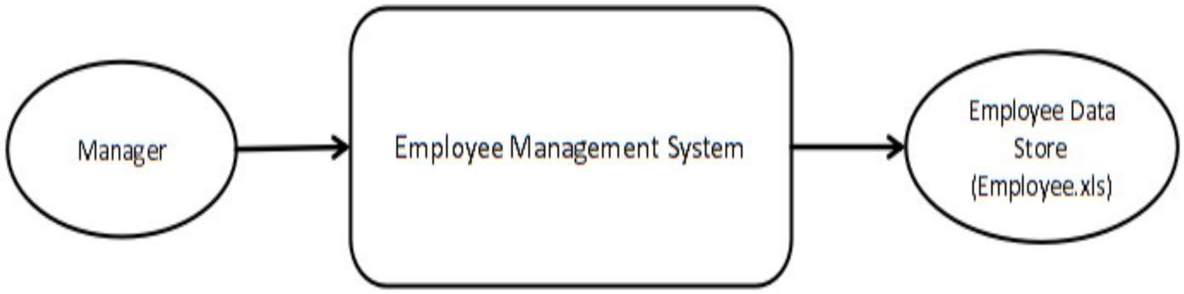
By following these steps, the EMS can be effectively maintained, ensuring its reliability and usability over time.

Data Flow Diagram (DFD) for Employee Management System (EMS)

A Data Flow Diagram (DFD) visually represents the flow of data within a system. Here, we'll create a Level 0 (Context Diagram) and Level 1 DFD for the Employee Management System.

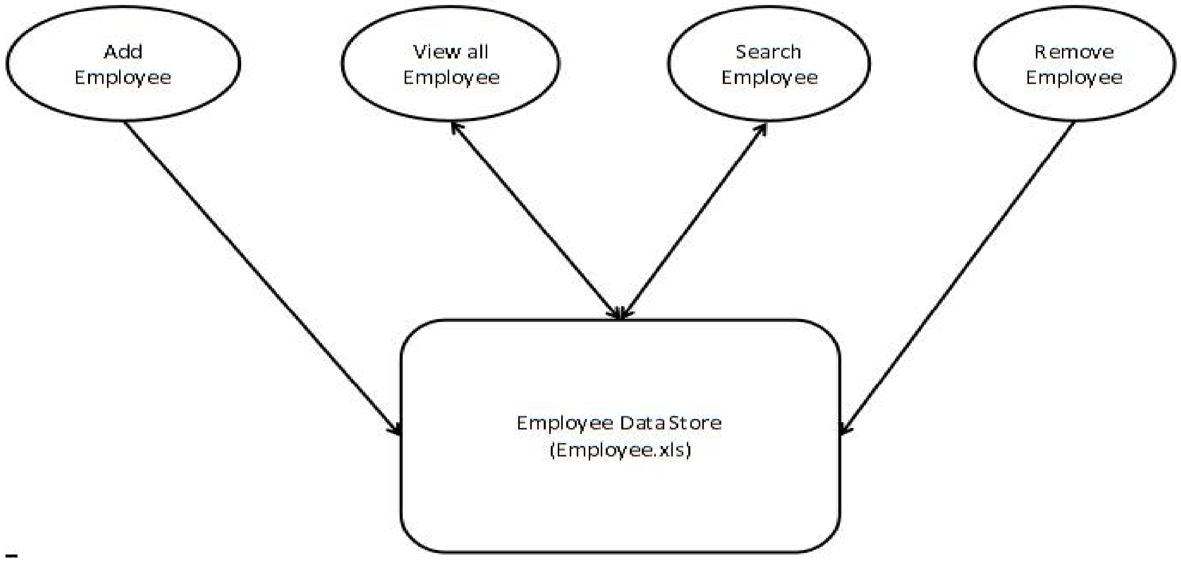
Level 0: Context Diagram

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


Manager
Employee Management System
Employee Data Store (Employee.xls)

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 Employee (1.0):

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

View All Employees (2.0):

- The manager requests a list of all employees.
- The system retrieves and displays all employee details from the Employee Data Store.

Search Employee (3.0):

- The manager enters an employee ID.
- The system searches the Employee Data Store for the matching employee and displays the details.

Remove Employee (4.0):

- The manager inputs the ID of the employee to be removed.
- The system deletes the employee's details from the Employee Data Store.

View Employee (5.0):

- The manager requests details of a specific employee by entering the employee ID.
- The system retrieves and displays the details of the specified employee from the Employee Data Store.

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

Source Code

```
import java.io.*;

import javax.swing.*;

import java.awt.*;

import java.awt.event.*;


class EmployeeManagementGUI

{

static String USER_ID = "kinshuk2903";

static String PASSWORD = "kinshuk095";


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

```
JLabel display = new JLabel("Welcome to Employee 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 Employee", e -> addEmployee());
addButton(panel, "View all Employees", e -> viewAllEmployees());
addButton(panel, "Show details of an Employee", e -> viewEmployee());
addButton(panel, "Delete Employee details", e -> removeEmployee());

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

public 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);
}

public static void addEmployee() {
JFrame frame = new JFrame("Add Employee");
frame.setSize(400, 400);
```

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

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

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

JLabel label4 = new JLabel("Enter Salary:");
JTextField salaryField = new JTextField();
JPanel panel4 = new JPanel(new GridLayout(1, 2));
panel4.add(label4);
panel4.add(salaryField);

JLabel label5 = new JLabel("Enter Hire-date:");
JTextField hireDateField = new JTextField();
JPanel panel5 = new JPanel(new GridLayout(1, 2));
panel5.add(label5);
panel5.add(hireDateField);
```

```
JLabel label6 = new JLabel("Enter Address of the Employee:");
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 = empIdField.getText();
String name = empNameField.getText();
String dept = depField.getText();
String salary = salaryField.getText();
String hireDate = hireDateField.getText();
String address = addressField.getText();

try (BufferedReader reader = new BufferedReader(new FileReader("Employee.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, dept, salary, hireDate, 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);
}

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

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

public static void viewAllEmployees() {
JFrame frame = new JFrame("View All Employees");
```

```
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("C:\\Users\\KIIT\\Desktop\\Internship\\Employee.xls"))) {

textArea.read(reader, null);

} catch (IOException ex) {

ex.printStackTrace();

}


frame.setVisible(true);

frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

}


public static void viewEmployee() {

JFrame frame = new JFrame("View Employee");

frame.setSize(500, 400);

frame.setLayout(new BorderLayout());


JTextField empIdField = new JTextField();

JPanel inputPanel = new JPanel(new GridLayout(1, 2));

JLabel inputLabel = new JLabel("Enter Employee ID:");

inputPanel.add(inputLabel);

inputPanel.add(empIdField);

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 empId = empIdField.getText();

try (BufferedReader reader = new BufferedReader(new
FileReader("C:\\Users\\KIIT\\Desktop\\Internship\\Employee.xls"))) {

String line;

boolean found = false;

while ((line = reader.readLine()) != null) {

String[] records = line.split("\t");

if (records[0].equals(empId)) {

resultArea.setText(String.join("\n", "Employee ID: " + records[0],

"Employee Name: " + records[1],

"Department: " + records[2],

"Salary: " + records[3],

"Hire-date: " + records[4],

"Address: " + records[5]));

found = true;

break;

}

}

if (!found) {

resultArea.setText("Employee not found.");

}

} catch (IOException ex) {

ex.printStackTrace();

}

});
```

```
frame.setVisible(true);

frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

}


public static void removeEmployee() {

JFrame frame = new JFrame("Remove Employee");

frame.setSize(500, 400);

frame.setLayout(new BorderLayout());


JTextField empIdField = new JTextField();

JPanel inputPanel = new JPanel(new GridLayout(1, 2));

JLabel inputLabel = new JLabel("Enter Employee ID:");

inputPanel.add(inputLabel);

inputPanel.add(empIdField);

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 empId = empIdField.getText();

try (BufferedReader reader = new BufferedReader(new
FileReader("C:\\Users\\KIIT\\Desktop\\Internship\\Employee.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(empId)) {
writer.write(line + "\n");
} else {
found = true;
}
}
if (found) {
JOptionPane.showMessageDialog(null, "Employee record deleted.");
} else {
JOptionPane.showMessageDialog(null, "Employee 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/