TASK 4

EMPLOYEE MANAGEMENT SYSTEM

1 Project Objective

The main objective of this project is to design and implement a console-based Employee Management System that efficiently manages employee records using Java Collections and File Handling.

The system allows users to:

Add new employees

View employee records

Search employees

Update employee information

Delete employee records

Generate salary reports

Save and load data from files

This project demonstrates practical implementation of:

Object-Oriented Programming (OOP)

Java Collections Framework

File Handling using Serialization

Exception Handling

Data persistence techniques

System Features

The system includes the following features:

Employee Record Management

- Create employee record
- Read/display all employee records

- Update employee details
- Delete employee records

Search Functionality

- Search employee by ID
- (Optional extension: Search by name or department)

Reporting

- Total number of employees
- Total salary expenditure
- Average salary
- Highest and lowest salary

File Persistence

- Save employee data to file
- Load employee data on program startup

Exception Handling

- Invalid number input handling
- Duplicate ID prevention
- File not found handling
- Input validation

2 Setup and Installation

Software Requirements

- Java JDK 8 or above
- IntelliJ IDEA (or any Java IDE)

Step 1: Install Java

- Download JDK from Oracle website

- Install it
- Verify installation using:

```
java -version
```

Step 2: Create Project in IntelliJ

1. Open IntelliJ IDEA
2. Click New Project
3. Select Java
4. Choose installed JDK
5. Name the project: EmployeeManagementSystem
6. Click **Finish**

Step 3: Add Java File

1. Right-click src
2. Select New → Java Class
3. Name it:
4. EmployeeManagementSystem
5. Paste the full project code
6. Click Run

3 CODE STRUCTURE

```
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.*;

class Employee implements Serializable {
    private static final long serialVersionUID = 1L;

    private String id;
    private String name;
    private String department;
    private String position;
    private double salary;
    private Date joinDate;

    public Employee(String id, String name, String department,
                    String position, double salary) {
        this.id = id;
        this.name = name;
        this.department = department;
        this.position = position;
        this.salary = salary;
        this.joinDate = new Date();
    }

    public String getId() { return id; }
    public String getName() { return name; }
    public String getDepartment() { return department; }
    public String getPosition() { return position; }
    public double getSalary() { return salary; }
    public Date getJoinDate() { return joinDate; }

    public void setName(String name) { this.name = name; }
    public void setDepartment(String department) { this.department = department; }
    public void setPosition(String position) { this.position = position; }
    public void setSalary(double salary) { this.salary = salary; }

    @Override
    public String toString() {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        return String.format("ID: %s | Name: %s | Dept: %s | Position: %s | Salary: ₹%.2f | Joined:
%s",
                id, name, department, position, salary, sdf.format(joinDate));
    }
}
```

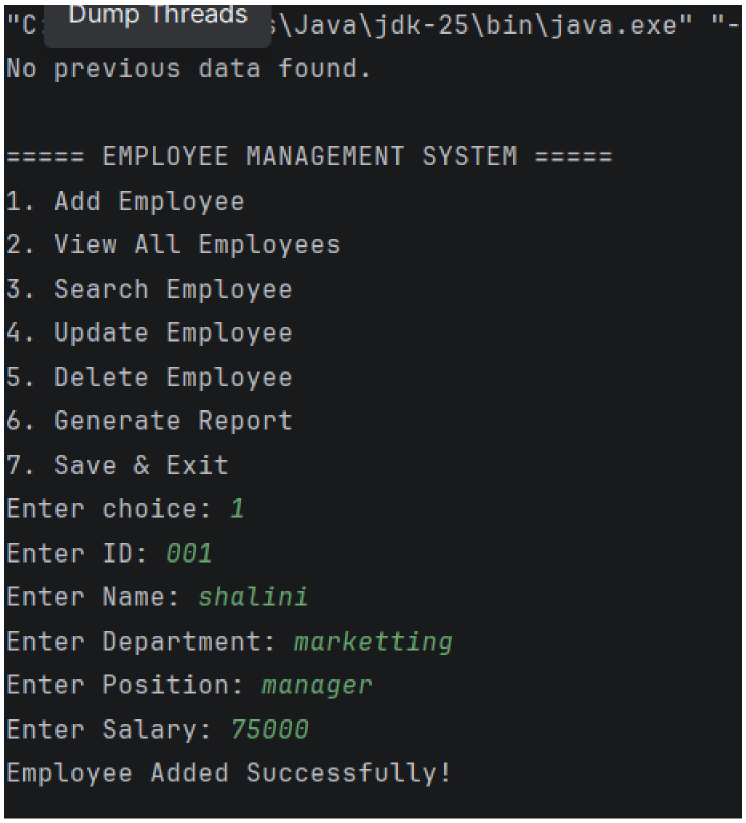
```
public class EmployeeManagementSystem {

    private static ArrayList<Employee> employees = new ArrayList<>();
    private static HashMap<String, Employee> employeeMap = new HashMap<>();
    private static Scanner scanner = new Scanner(System.in);
    private static final String FILE_NAME = "employees.dat";

    public static void main(String[] args) {
        loadFromFile();
        menu();
    }

    public static void menu() {
        while (true) {
            System.out.println("\n===== EMPLOYEE MANAGEMENT SYSTEM =====");
            System.out.println("1. Add Employee");
            System.out.println("2. View All Employees");
            System.out.println("3. Search Employee");
            System.out.println("4. Update Employee");
            System.out.println("5. Delete Employee");
            System.out.println("6. Generate Report");
            System.out.println("7. Save & Exit");
            System.out.print("Enter choice: ");

            int choice = getInt();

            switch (choice) {
                case 1:
                    addEmployee();
                    break;
                case 2:
                    viewEmployees();
                    break;
                case 3:
                    searchEmployee();
                    break;
                case 4:
                    updateEmployee();
                    break;
                case 5:
```

```
                deleteEmployee();
                break;
            case 6:
                generateReport();
                break;
            case 7:
                saveToFile();
                System.out.println("Data Saved. Exiting...");
                return;
            default:
                System.out.println("Invalid choice!");
        }
    }
}

// ================= ADD =================
private static void addEmployee() {
    scanner.nextLine();

    System.out.print("Enter ID: ");
    String id = scanner.nextLine();

    if (employeeMap.containsKey(id)) {
        System.out.println("Employee already exists!");
        return;
    }

    System.out.print("Enter Name: ");
    String name = scanner.nextLine();

    System.out.print("Enter Department: ");
    String dept = scanner.nextLine();

    System.out.print("Enter Position: ");
    String pos = scanner.nextLine();

    System.out.print("Enter Salary: ");
    double salary = getDouble();

    Employee emp = new Employee(id, name, dept, pos, salary);
```

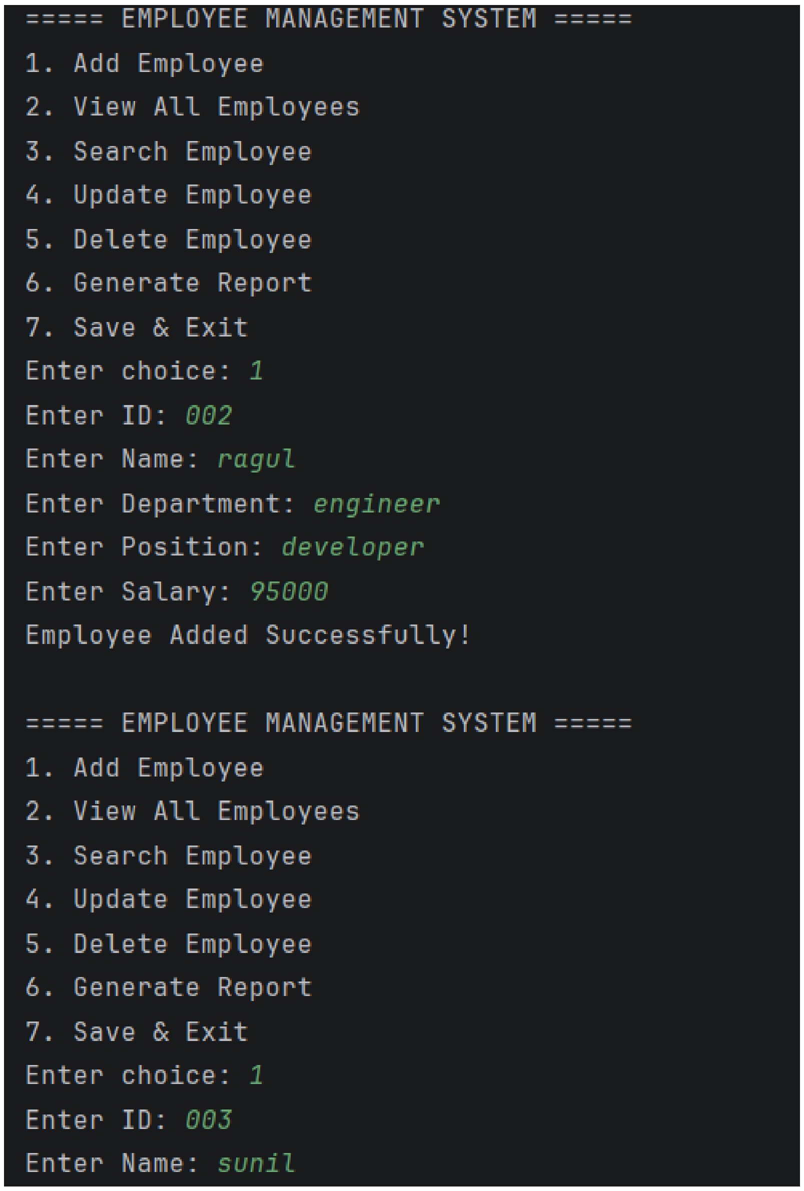
```
        employees.add(emp);
        employeeMap.put(id, emp);

        System.out.println("Employee Added Successfully!");
    }

    // ================= VIEW =================
    private static void viewEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees found!");
            return;
        }

        for (Employee e : employees) {
            System.out.println(e);
        }
    }

    // ================= SEARCH =================
    private static void searchEmployee() {
        scanner.nextLine();
        System.out.print("Enter Employee ID: ");
        String id = scanner.nextLine();

        Employee emp = employeeMap.get(id);

        if (emp != null)
            System.out.println(emp);
        else
            System.out.println("Employee not found!");
    }

    // ================= UPDATE =================
    private static void updateEmployee() {
        scanner.nextLine();
        System.out.print("Enter Employee ID: ");
        String id = scanner.nextLine();

        Employee emp = employeeMap.get(id);
        if (emp == null) {
            System.out.println("Employee not found!");
            return;
        }

        System.out.print("New Name: ");
        emp.setName(scanner.nextLine());

        System.out.print("New Department: ");
        emp.setDepartment(scanner.nextLine());

        System.out.print("New Position: ");
        emp.setPosition(scanner.nextLine());

        System.out.print("New Salary: ");
        emp.setSalary(getDouble());

        System.out.println("Employee Updated Successfully!");
    }

    // ================= DELETE =================
    private static void deleteEmployee() {
        scanner.nextLine();
        System.out.print("Enter Employee ID: ");
        String id = scanner.nextLine();

        Employee emp = employeeMap.remove(id);

        if (emp != null) {
            employees.remove(emp);
            System.out.println("Employee Deleted Successfully!");
        } else {
            System.out.println("Employee not found!");
        }
    }

    // ================= REPORT =================
    private static void generateReport() {
        if (employees.isEmpty()) {
            System.out.println("No employees available.");
            return;
        }

        double total = 0;
        double highest = employees.get(0).getSalary();
        double lowest = employees.get(0).getSalary();

        for (Employee e : employees) {
            total += e.getSalary();
            if (e.getSalary() > highest) highest = e.getSalary();
            if (e.getSalary() < lowest) lowest = e.getSalary();
        }

        System.out.println("\n===== REPORT =====");
        System.out.println("Total Employees: " + employees.size());
        System.out.println("Total Salary: ₹" + total);
        System.out.println("Average Salary: ₹" + (total / employees.size()));
        System.out.println("Highest Salary: ₹" + highest);
        System.out.println("Lowest Salary: ₹" + lowest);
    }

    // ================= FILE SAVE =================
    private static void saveToFile() {
        try (ObjectOutputStream oos =
                     new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {

            oos.writeObject(employees);

        } catch (IOException e) {
            System.out.println("Error saving file!");
        }
    }

    // ================= FILE LOAD =================
    @SuppressWarnings("unchecked")
    private static void loadFromFile() {
        try (ObjectInputStream ois =
                     new ObjectInputStream(new FileInputStream(FILE_NAME))) {
```

```
            employees = (ArrayList<Employee>) ois.readObject();

            for (Employee emp : employees) {
                employeeMap.put(emp.getId(), emp);
            }

        } catch (Exception e) {
            System.out.println("No previous data found.");
        }
    }

    // ================= INPUT VALIDATION =================
    private static int getInt() {
        while (!scanner.hasNextInt()) {
            System.out.println("Enter valid number!");
            scanner.next();
        }
        return scanner.nextInt();
    }

    private static double getDouble() {
        while (!scanner.hasNextDouble()) {
            System.out.println("Enter valid salary!");
            scanner.next();
        }
        return scanner.nextDouble();
    }
}
```

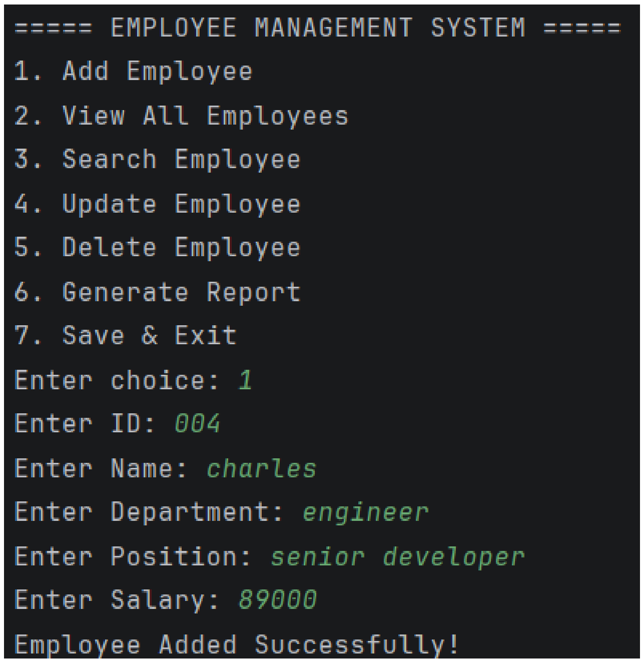
4 VISUAL DOCUMENTATION



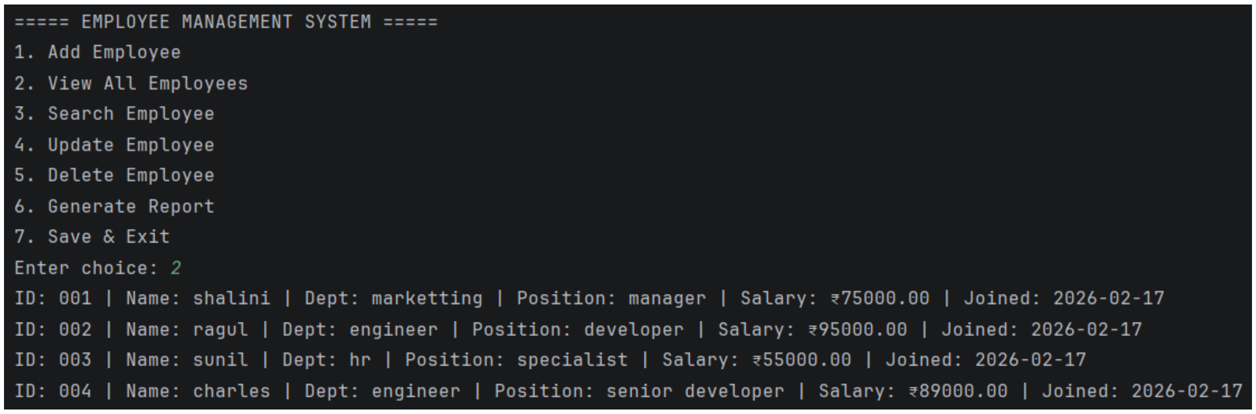
```
"C:          s\Java\jdk-25\bin\java.exe" "-
No previous data found.

===== EMPLOYEE MANAGEMENT SYSTEM =====
1. Add Employee
2. View All Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Generate Report
7. Save & Exit
Enter choice: 1
Enter ID: 001
Enter Name: shalini
Enter Department: marketting
Enter Position: manager
Enter Salary: 75000
Employee Added Successfully!
```



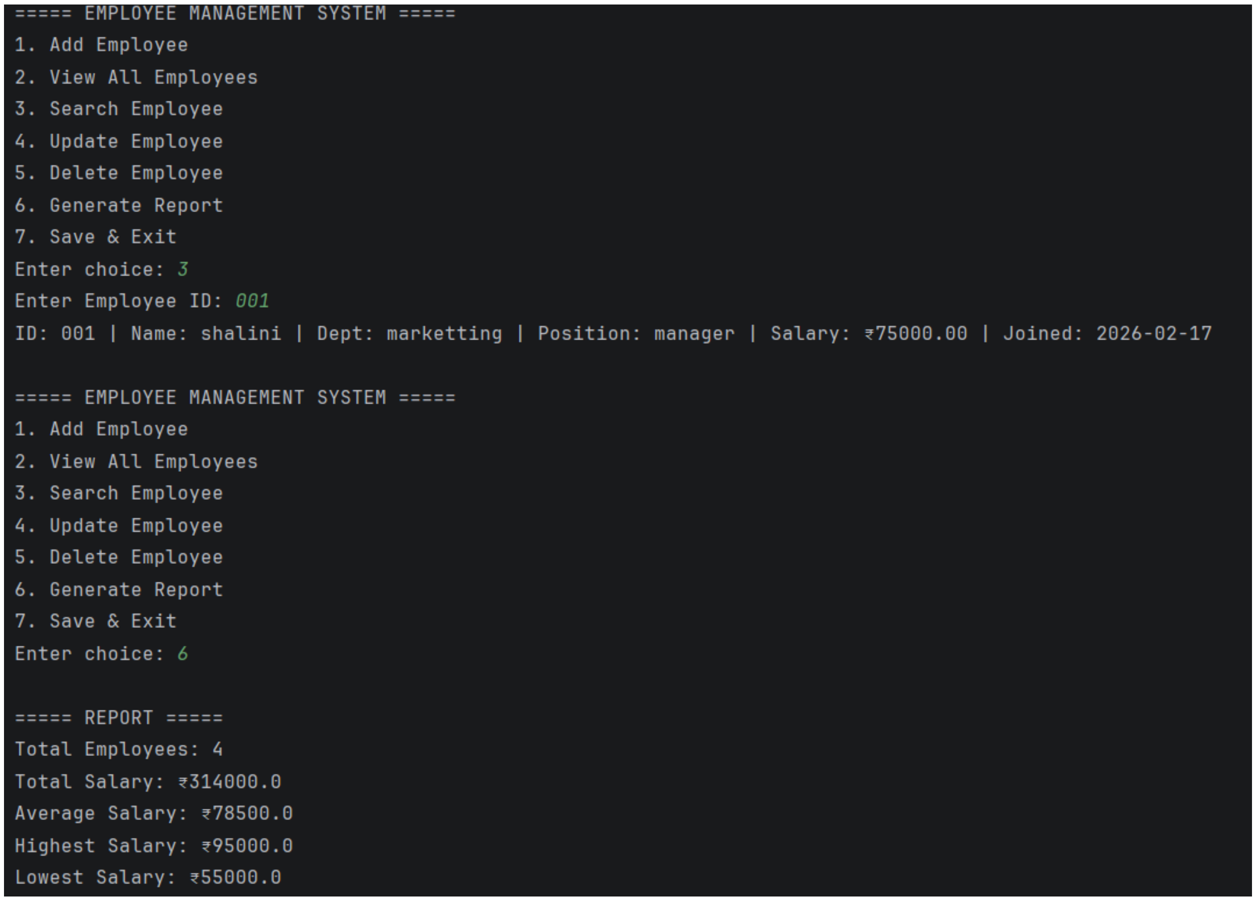
```
===== EMPLOYEE MANAGEMENT SYSTEM =====
1. Add Employee
2. View All Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Generate Report
7. Save & Exit
Enter choice: 1
Enter ID: 002
Enter Name: ragul
Enter Department: engineer
Enter Position: developer
Enter Salary: 95000
Employee Added Successfully!

===== EMPLOYEE MANAGEMENT SYSTEM =====
1. Add Employee
2. View All Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Generate Report
7. Save & Exit
Enter choice: 1
Enter ID: 003
Enter Name: sunil
```



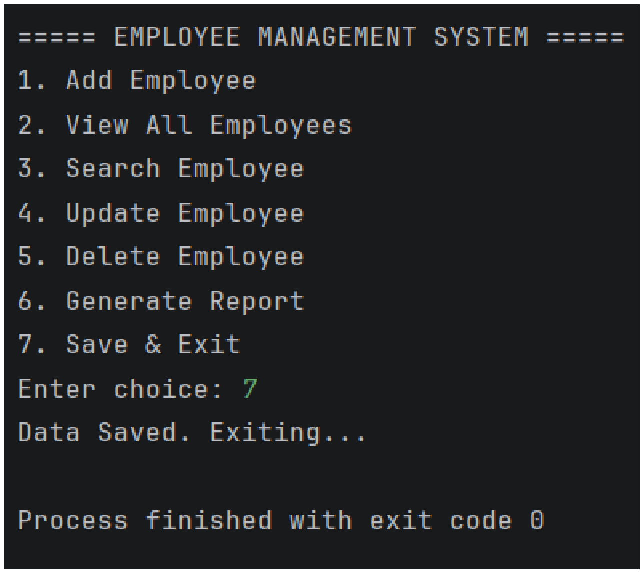
```
===== EMPLOYEE MANAGEMENT SYSTEM =====
1. Add Employee
2. View All Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Generate Report
7. Save & Exit
Enter choice: 1
Enter ID: 004
Enter Name: charles
Enter Department: engineer
Enter Position: senior developer
Enter Salary: 89000
Employee Added Successfully!
```



```
===== EMPLOYEE MANAGEMENT SYSTEM =====
1. Add Employee
2. View All Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Generate Report
7. Save & Exit
Enter choice: 2
ID: 001 | Name: shalini | Dept: marketting | Position: manager | Salary: ₹75000.00 | Joined: 2026-02-17
ID: 002 | Name: ragul | Dept: engineer | Position: developer | Salary: ₹95000.00 | Joined: 2026-02-17
ID: 003 | Name: sunil | Dept: hr | Position: specialist | Salary: ₹55000.00 | Joined: 2026-02-17
ID: 004 | Name: charles | Dept: engineer | Position: senior developer | Salary: ₹89000.00 | Joined: 2026-02-17
```



```
===== EMPLOYEE MANAGEMENT SYSTEM =====
1. Add Employee
2. View All Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Generate Report
7. Save & Exit
Enter choice: 3
Enter Employee ID: 001
ID: 001 | Name: shalini | Dept: marketting | Position: manager | Salary: ₹75000.00 | Joined: 2026-02-17

===== EMPLOYEE MANAGEMENT SYSTEM =====
1. Add Employee
2. View All Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Generate Report
7. Save & Exit
Enter choice: 6

===== REPORT =====
Total Employees: 4
Total Salary: ₹314000.0
Average Salary: ₹78500.0
Highest Salary: ₹95000.0
Lowest Salary: ₹55000.0
```



```
===== EMPLOYEE MANAGEMENT SYSTEM =====
1. Add Employee
2. View All Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Generate Report
7. Save & Exit
Enter choice: 7
Data Saved. Exiting...

Process finished with exit code 0
```

5 Technical Details

ArrayList

- Dynamic array implementation
- Allows indexed access
- Used for iteration and reporting

HashMap

- Key-value structure
- Key = Employee ID
- Value = Employee object
- Fast data retrieval

Application Architecture

Console-based architecture:

User → Menu Interface → Business Logic → Data Collections → File System

---

Exception Handling

The system handles:

Input Validation

- Non-numeric salary input
- Invalid menu choice
- Empty values

File Exceptions

- FileNotFoundException
- IOException
- ClassNotFoundException

Testing and Validation

Test Case 1: Add Employee

Input valid details → Employee added successfully

Test Case 2: Duplicate ID

System prevents duplicate entry

Test Case 3: Search Employee

Correct employee details displayed

Test Case 4: Delete Employee

Employee removed from both collections

Test Case 5: File Persistence