**CSE Department – Faculty of Engineering - MSA**

**Spring 2025**

**GSE122 GSE122i COM265 PROGRAMMING 2**

**Course Project**

**Course Instructor:Dr. Ahmed El Anany**

**Due Date 9/MAY/2025 11:59 PM on E-learning**

**Discussion inside lecture 18/May till 23/May inside lab as per lab slot**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** | **Ahmed Mohamed Shaban** | **Student ID** | **231329** |
| **Student Name** | **mohamed raed** | **Student ID** | **234197** |
| **Student Name** | **Omar Amr** | **Student ID** | **244009** |
| **Student Name** | **Youssef ahmed abdelrhman** | **Student ID** | **234989** |
| **Student Name** | **Mohab mahmoud** | **Student ID** | **235663** |
| **TA Name** | **Eng. Dina Magdy**  **Eng. Youmna Mohamed**  **Eng. Mohamed Khaled**  **Eng. Hussien Mostafa** | **Grade: /** | |

**Staff Management System**

**Table of Contents**

[**Project Overview 3**](#_1qiv9zobnqqy)

[Objectives 3](#_kxya5mh10gqo)

[Roles and Responsibilities 4](#_4hrs94c0szki)

[Algorithm and external libraries 5](#_lbxclr1u8qm6)

[GUI and Database Usage 6](#_ns54jo9tktz)

[**Code explaining 7**](#_sk6m5j3846ai)

[**Output and results 8**](#_m9h15o464mtl)

[**GitHub(optional) 9**](#_u0qg68ra51oo)

[**References 10**](#_1lsyo46b7om8)

# **Project Overview**

## This project involves designing and implementing a Java-based desktop application that manages staff employee data. The system allows the user to perform various operations such as adding, editing, deleting, and searching employee records using simple GUI. This application is developed using Java Swing and is structured using object-oriented programming principles.

## **Objectives**

## The primary objective of this project is to:

## Develop a GUI-based employee management system.

## Apply object-oriented programming techniques.

## Implement data storage using HashMap.

## Ensure validation of user inputs.

## Provide meaningful feedback to users.

## 

## 

## 

## **Roles and Responsibilities**

## Ahmed Mohamed: Research & Report.

## Youssef: libraries

## Raed: code overview

## Mohab: algorithm

## Amr: git hub

## 

## **Algorithm and external libraries**

The core logic of the Staff Management System revolves around implementing basic data management operations (CRUD: Create, Read, Update, Delete) in an object-oriented fashion. The system operates entirely in memory and relies on Java’s standard libraries for both data processing and user interaction.

* **Main Algorithm Structure**:
  + The application starts by launching the main GUI window (JFrame) and displaying six key operations.
  + Each operation (insert, search, edit, delete, filter, display all) is handled by a corresponding method in the main class.
  + Employee records are stored in a HashMap<Integer, Employee> where the employee ID serves as the key, and the Employee object contains personal and job-related information.
* **Insert Operation Algorithm**:
  + Prompt the user using JOptionPane to input all employee details.
  + Validate inputs using the EmployeeValidator class.
  + Add the new employee to the HashMap if the ID is not already in use.
* **Search, Edit, and Delete**:
  + Search: Check if the ID exists in the map; display record if found.
  + Edit: Replace the existing object with a new Employee object after validation.
  + Delete: Remove the record by ID.
* **Display Operations**:
  + Use StringBuilder to aggregate records and display using JOptionPane.
  + Filter logic checks employee salary threshold before displaying.
* **Validation Logic**:
  + A separate static class (EmployeeValidator) is responsible for checking valid ID, name, age (18–65), role, and non-negative salary.
  + Each field is validated with feedback provided through dialog boxes.
* **External Libraries Used**:
  + javax.swing.\*: Used for GUI creation including windows (JFrame), input dialogs (JOptionPane), and buttons (JButton).
  + java.util.\*: Provides the HashMap and other utility functions like iteration over employee records.
  + java.awt.\*: Supports GUI layout management using GridLayout.

This approach provides a structured and efficient means of managing employee records within a simple desktop application, making it ideal for small-scale administrative purposes and educational demonstrations.

## **GUI and Database Usage**

* **Graphical User Interface (GUI)**: The GUI of the Staff Management System is developed using the Java Swing library. The main window (JFrame) presents users with six clearly labeled buttons in a GridLayout to facilitate interaction. Each button corresponds to one of the primary operations: Insert, Search, Edit, Delete, View High Salary Employees, and Display All.

For user interaction, JOptionPane dialogs are employed to gather input and display messages. These dialogs simplify the interface and ensure quick data entry without requiring complex form handling. For example, when inserting a new employee, a series of input dialogs guide the user through ID, name, age, address, role, and salary input.

Error handling is integrated using try-catch blocks and validations, providing the user with immediate feedback in case of incorrect input, such as entering text where a number is expected.

* **GUI Layout Overview**:
  + JFrame: Main window container
  + GridLayout: Organizes the buttons vertically in six rows
  + JButton: Triggers each operation
  + JOptionPane: Used for input collection and result display
* **Database Usage (In-Memory Storage)**: Instead of using an external database like MySQL or SQLite, the project utilizes an in-memory HashMap<Integer, Employee> to store employee records during runtime. This design decision simplifies the project, reduces dependencies, and demonstrates basic data management techniques.
  + **Key (Integer)**: Represents a unique Employee ID
  + **Value (Employee Object)**: Stores name, age, address, role, and salary

While this approach means the data is lost when the application closes, it is appropriate for demonstrating fundamental concepts such as object-oriented programming, data structure use, and GUI interaction. A future enhancement could involve integrating persistent storage through file I/O or a lightweight embedded database.

## 

## **Code explaining**

The code for the Staff Management System is organized into three primary Java classes:

**1. Employee.java**

This class defines the Employee object which stores all relevant data fields:

* int id – unique identifier for the employee
* String name, String address, String role – personal and professional details
* int age – age (restricted between 18 and 65)
* double salary – monthly salary (non-negative)

**Key method:**

* toString() – formats employee details into a readable string for display.

**2. StaffManagementSystem.java**

This is the main class containing the entry point and user interface logic.

**Attributes:**

* HashMap<Integer, Employee> employees – used to store employee data during runtime.

**Key Methods:**

* main(String[] args) – Initializes the application and launches the GUI.
* displayMainMenu() – Builds and displays the GUI with six buttons arranged in a grid.
* insertEmployee() – Collects employee data from user input dialogs, validates them, and inserts them into the HashMap.
* searchEmployeeById() – Prompts for an ID and retrieves the corresponding employee if found.
* editEmployeeDetails() – Allows the user to update information of an existing employee by re-entering their details.
* deleteEmployee() – Removes the employee with the given ID from the HashMap.
* searchEmployeesWithHighWage() – Filters and displays employees earning more than 20,000.
* displayRecordList() – Shows all employees currently stored.

**Dialog Management:**

* JOptionPane.showInputDialog() is used to gather input (ID, name, age, etc.).
* JOptionPane.showMessageDialog() is used to show results, confirmations, and error messages.

**Error Handling:**

* Try-catch blocks are used extensively to catch NumberFormatException and provide error messages without crashing the program.

**3. EmployeeValidator.java**

This utility class performs input validation to ensure data integrity before it is accepted.

**Validation Rules:**

* ID must be positive and unique.
* Name, Address, and Role must not be empty or null.
* Age must be between 18 and 65.
* Salary must be zero or greater.

**Method:**

* validate (int id, String name, int age, String address, String role, double salary) – returns a Boolean indicating whether input is acceptable and shows error dialogs when needed.

**Code Modularity and OOP Principles**

The project applies core object-oriented principles:

* **Encapsulation**: Employee data is grouped into an object.
* **Separation of Concerns**: GUI logic, business logic, and validation are separated into distinct classes.
* **Reusability**: Validation logic is reused for both insertion and editing operations.

This code organization ensures maintainability, clarity, and ease of enhancement for future development.

# 

# Output and results

The program executes successfully and provides a clear, interactive user interface through dialog windows. Below are detailed descriptions of the expected outputs and results for each core operation:

**1. Main Menu Display**

* Upon launching the application, a JFrame displays six buttons in a vertical layout.
* Each button represents a feature (Insert, Search, Edit, Delete, High Salary, Display All).

**2. Inserting a New Employee**

* The system prompts the user for employee details in multiple input dialogs.
* If the data passes validation, a success message confirms the addition.
* If invalid data is entered (e.g., age below 18), an error dialog explains the issue.

**3. Searching by ID**

* The user is prompted to input an employee ID.
* If found, a message dialog displays the full details of the employee.
* If not found, a notification alerts the user accordingly.

**4. Editing an Employee**

* Similar to insertion, but the system first verifies the ID exists.
* If it does, all details are re-entered, validated, and updated.

**5. Deleting an Employee**

* The user inputs an employee ID to delete.
* If the ID exists, the record is removed and confirmation is shown.
* Otherwise, an error message indicates the ID was not found.

**6. Displaying All Employees**

* A list of all employee records is compiled and displayed using JOptionPane.
* If no employees exist, the system displays a message indicating the list is empty.

**7. Employees with Salary > 20,000**

* A filtered list of employees whose salary exceeds the threshold is shown.
* If none qualify, an appropriate message is displayed.

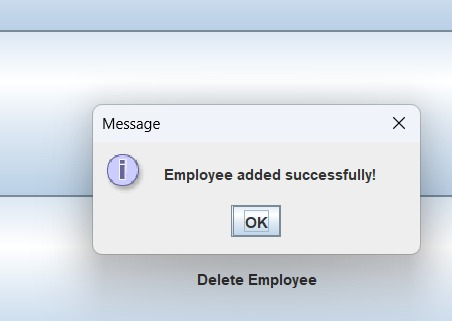
# 

# 

# 

# 

# 





# GitHub(optional)

Include link of github repo containing your project code and report and add screenshot of repo with commit logs

# References