Employee Management System for Company 2

Group 12
Mohamed Taha
Zahmad Shields
Joumana Elhagany
Lian Bik
Bribenna Oyinnimiebi

Date: 07/29/2025

TABLE OF CONTENTS

1.0 **INTRODUCTION**

1.1 **Purpose**

This Software Design Documentation shows the structure and components of the Employee Management System made for Corporation ‘2’. It outlines the framework’s composition, design, user interface, and means of execution.

1.2 **Scope**

The Employee Management System backs fundamental worker administration capacities for an organization with approximately (but not limited to) 20 full-time workers. Its backend communicates with a MySQL database and allows for employee record management and reporting. The key functions include: Adding and editing worker information, searching by SSN, name, or ID, calculating salaries, and creating pay reports. The framework utilizes a work area based interface.

1.3 **Overview**

This report is organized into areas that portray the general framework diagram, models illustrating functionality and communication between components, admin interface, and test cases.

1.4 **Reference Material**

Project scenario on iCollege
Class lecture notes
Java Fundamentals video on iCollege
Java Connect to MySQL Database Step by Step → www.youtube.com/watch?v=duEkh8ZsFGs
Java.sql package interface summary from Oracle
Java.util package interface summary from Oracle

1.5 **Definitions and Acronyms**

SSN → Social Security Number
UX → User Experience
DB → Database
JDBC → Java Database Connectivity

2.0 **SYSTEM OVERVIEW**

The Employee Management System is a minimalistic, console-based program that is designed to manage and maintain employee records kept in a SQL DB. The system allows for admins to enter and manipulate data via interaction with the console that directly communicates with the database.

Core Functions:

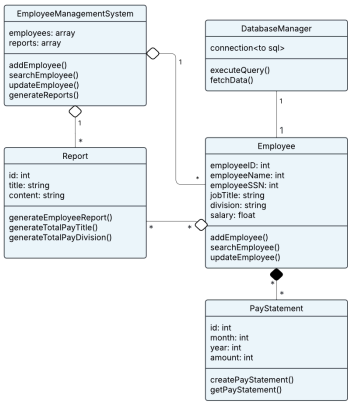
Search for employees by name, SSN, or ID
Update general employee information (division, title, etc.)
Retrieve employee pay history
Generate reports by division and job title

3.0 **SYSTEM ARCHITECTURE**

3.1 **Architectural Design**

The system has clear separation of the:
Model: Java classes representing the employee data
Controller: Classes that handle the user input and output
Database Layer: classes managing MySQL interactions via JDBC
View: Interaction with the user via the console through the controller

Major Components:
Main.java: Entry
Employee.java: Managing employee data to be stored via the management system
Database: Handles all DB queries received from the employeemanager
EmployeeManager.java: Applies business rules and logic
ConsoleUI: minimal UX to output information and take user input

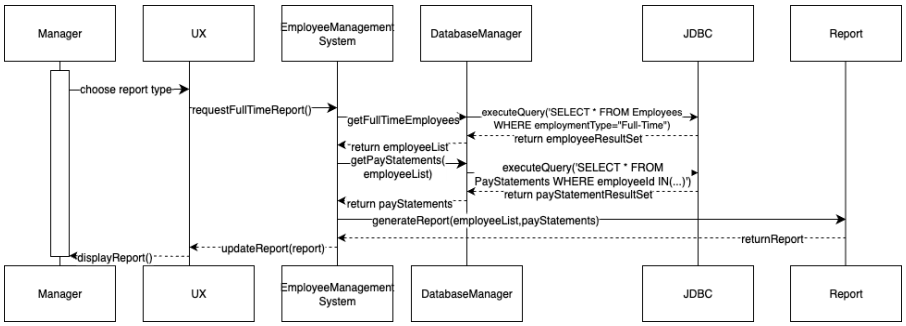


3.2 **Decomposition Description**

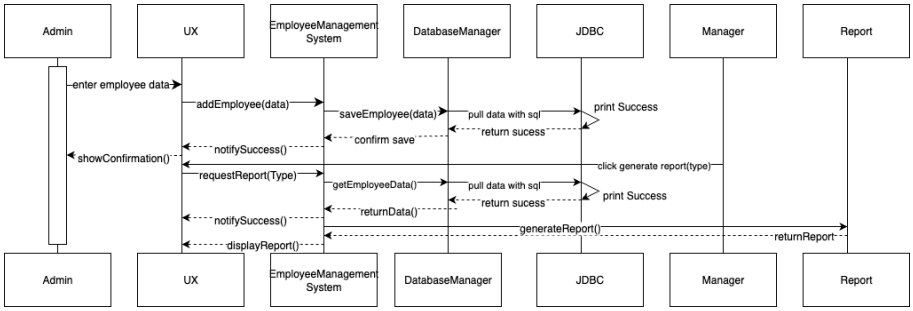
This system uses OOP. Major classes and their relationships are captured in the UML class diagram above and submitted separately

Classes include:
Employee parameters: id, name, title, division, salary, ssn, etc.
Employee methods: getEmployeeById(), updateEmployee(), applySalaryIncrease(), etc.
EmployeeManagementSystem: code behind the ConsoleUI
Report: Generates reports by total pay given a division or job title
DataBaseManager: Intermediary between SQL DB and the EmployeeManagementSystem class
ConsoleUI: Handles client interaction
Sequence diagrams (shown below) include:
Data retrieval by ID (search),Salary update by range and percentage, Report generation

Reporting Sequence Diagram



Overall System Sequence Diagram:



3.3 **Design Rationale**

This program is made to meet expectations of a minimal UX component along with all of the functionality requirements. The simplistic design makes its use easy for the client and the class structure is adaptable such that it is easy to upgrade (to a GUI or website) all the while maintaining organized code.

4.0 **DATA DESIGN**

4.1 **Data Description**

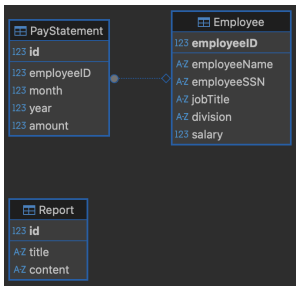
Information is initially stored in primitive data types, strings, lists, integers, etc. in an object but is immediately pushed to a database explained below:
The system uses “employeeData”, a MySQL database named employeeData that is connected to via JDBC. The tables consist of Employee, PayStatement, and Report. It is accessed primarily through the EmployeeManagementSystem.java class.
Major data operations include:
SELECT queries for employee search, UPDATE for record modifications and salary adjustments, INSERT for new employees, More complex queries for reporting such as GROUP BY division/jobTitle

4.2 **Data Dictionary**

**Note that attributes are included in section 3.2 and will not be reiterated in the table.*

Component	**Type**	**Description**
ConsoleUI	UI	takes user input
Employee	Class	Blueprint for Employee Objects Containing information to push to or receive from the DB
EmployeeManagementSystem	Class	Contains entirety of user input handling and calls corresponding methods from the DataBaseManager
Report	Class	Utility class used to retrieve salaries and make salary-related calculations
DataBaseManager	Class	Uses java.sql library to create a database manager and edits and retrieves data from DB based on input from the console that is retrieved from the EmployeeManagementSystem
Sequence Diagrams	Diagram	Illustrates flow from UX → Controller → DB → Output

Below is a diagram illustrating the tables stored in the MySQL database



5.0 **COMPONENT DESIGN**

This section has a pseudocode description for each of the components and their respective methods.

Employee Class

Attributes: employeeID: int
employeeName: String
employeeSSN: String
jobTitle: String
division: String
salary: float
Methods:
Getters and setters for all attributes
toString method that returns a string containing all employee information

EmployeeManagementSystem

DatabaseManager dbManager: Handles all database interactions
Scanner scanner: For user input
Loop:
    Display menu options(add,search,update salary,report by division, report by title, exit)
    case 1,2,3,4,5,6: call
addEmployee(),searchEmployee(),updateEmployee(),genPayReportD(),genPayReportT(),
Exit loop
addEmployee(): prompt user for employee details, create employee object, call dbManager.insertEmployee(employee)
searchEmployee(): prompt user to search by id or SSN, call the corresponding dbManager method, print employee info or return not found
updateSalary: prompt for employee id and change to salary, call dbManager.updateSalary(Employee,salary)
genPayReportD(): prompt for name of division, call dbManager.getEmployeesbyDivision() and report.generatePayReport(employees)
genPayReportT(): prompt for job title, call dbManager.getEmployeesbyTitle() and report.generatePayReport(employees)

DBManager:

Load SQL driver, initialize connection with database using jdbc url, username, password
insertEmployee(Employee): run SQL insert statement, fill in parameters from employee object
getEmployeebyID(employeeID): run SQL where/like statement query, if found return employee, else return none
getEmployeeByName(name): same as above statements
getEmployeeBySSN(ssn):same as above
updateEmployee(employeeID): run SQL UPDATE statement query with the updated fields, match using employeeID
updateSalary(employeeID,salary): Same as above

getEmployeesbyDivision(division): run SELECT statement query in SQL with given division and return list of employees
getEmployeesByTitle(title): same as above, given title instead of division

Report:

genPayReport(employees): traverse through employees in list, print employee.toString(), pay statement, and calculate and print total salary

PayStatement:

attributes: id,month,year,amount
methods:
getPayStatement: return pay statement: month, year, amount

6.0 **HUMAN INTERFACE DESIGN**

6.1 **Overview of User Interface**

The system utilizes a console based UI that supports the following:
Prompts the user for an action (add, search, update, adjust salary, generate report, exit)
Collects input parameters (such as employeeID when updating salary)
Displays output in a concise, clean manner
Shows confirmation of changes or error messages

Menu:
1. Add Employee → asks for employee information
2. Search → asks for SSN
3. Adjust Salary → Search by employeeID/name included, asks for new salary
4. Generate Division Report → asks for division
5. Generate Job Title Report → asks for job title
6. Exit → ends program
Enter choice: _

6.3 **Screen Objects and Actions**

Input	*Description*
Menu Selection	Triggers a corresponding function
Search Input	Accepts name, SSN, or employeeID
Update Input	Prompts for field updates
Salary Input	Updates salary for given employee
Output	Prints the corresponding table, list, or string to the console

7.0 **REQUIREMENTS MATRIX**

Requirement	Component(s)
Add Employee	EmployeeManagementSystem, DataBaseManager
Search by SSN, name, or ID	EmployeeManagementSystem, DataBaseManager
Update employee data	EmployeeManagementSystem, DataBaseManager
Apply salary increases within a defined salary range	EmployeeManager, EmployeeDAO
Generate monthly pay by division	Report, DataBaseManager
Generate monthly pay by job title	Report, DataBaseManager
Add SSN column to employee table	MySQL Script