Georgia State University
Software Development Final Project

Background

The task of managing employee files in company '2' is currently being performed manually. The company uses scripts and simple tools such as dBeaver to work with the databases. This method is not efficient since it also reduces the company's ability to scale and generate urgent needed reports on short notice. With less than 20 fulltime employees, the organization needs a simple and light weight approach to data management that will improve the working process of the organization's employees.

To address these needs, the new Employee Management System will provide basic but important capabilities in maintaining employees records. This system will allow for major processes like employee record entry, record modifications, employee records retrieval, and payroll reports maintenance. This means that automation will cut down on manual work, improve accuracy, and provide a single point of access to employee information.

This project focuses on delivering a **minimal working user experience (UX)** to interact with the database efficiently. The UX will be either a console-based interface or a JavaFX GUI, tailored to ensure ease of use for the single administrator responsible for data entry and management. With no need for complex user authentication or multi-user support, the system will emphasize simplicity and reliability.

The Employee Management System will also support additional features like:

- Searching for employees using various identifiers (e.g., name, SSN, employee ID).
- Updating salaries based on specific conditions, such as a percentage increase for employees within a defined salary range.
- Generating essential reports, such as pay statements and monthly payroll summaries by division and job title.

This project aligns with the company's goals of improving data accessibility and operational efficiency, providing a robust foundation for managing employee data effectively.

Group 10

Employee Database Management and Reporting Tool

Software Design Document

Members:

1. Dua Spall
2. Sneha Muppala
3. Naol Seyum
4. Adwaita Boralkar
5. Abhay Deep Singh
6. Jigyasa jha
7. Aryan Sahu

Date:

December 5, 2024

Table of Content:

1.0 INTRODUCTION

1.1 Purpose

In this document, the design and implementation of an Employee Management System for the Company—'2' is described. The goal of this system is to help in the organization of the employees' data, enable efficient payrolls, and provide reporting options that allow for ease of use for the single administrator.

1.2 Scope

The system is designed for a company with less than 20 full-time employees. It will offer the following key functionalities:

- Insert, update, delete, and search employee records.
- Generate reports on pay statements, total pay by job title, and total pay by division.
- Update salary based on specified conditions. This is a console or JavaFX-based application integrated with a MySQL database via JDBC.

1.3 Overview

This project delivers a minimum usable product for the UX of an Employee Management System that is designed to be modular and scalable. It will have a dynamic
interface to interact with the database for seamless, robust, and error-free data management. The final deliverables will include functional code, UML diagrams, and a recorded video demonstration.

1.4 Reference Material

- MySQL Documentation.
- JavaFX API Documentation.

- JDBC Documentation.

1.5 Definitions and Acronyms

- **EMS**: Employee Management System.
- **UX**: User Experience.
- **JDBC**: Java Database Connectivity.
- **SSN**: Social Security Number.
- **SQL**: Structured Query Language.

2.0 SYSTEM OVERVIEW

The Employee Management System is a database driven software solution designed to manage employee data efficiently. It provides functionalities such as data insertion, updates, and reporting. The system interacts with a MySQL database via JDBC and features either a console-based or JavaFX GUI for user interaction.

3.0 SYSTEM ARCHITECTURE

3.1 Architectural Design

The system follows a layered architecture:

1. **Presentation Layer**: Console or JavaFX interface.
2. **Application Layer**: Core logic for employee management.
3. **Database Layer**: MySQL database accessed via JDBC.

3.2 Decomposition Description

The system consists of the following components:

- **UI Component**: Handles user interaction.
- **Controller Component**: Implements business logic.
- **Database Component**: Manages data storage and retrieval.

3.3 Design Rationale

This modular design ensures separation of concerns, allowing ease of maintenance and scalability. It leverages Java's object-oriented principles and MySQL's robust database capabilities.

4.0 DATA DESIGN

4.1 Data Description

The MySQL database contains tables for employees, job titles, divisions, and pay statements. Key attributes include:

- Employee: `emp_id`, `name`, `SSN`, `job_title`, `division`, `salary`.
- Pay Statement: `statement_id`, `emp_id`, `pay_date`, `amount`.

4.2 Data Dictionary

Table	*Field*	*Description*
Employee	*emp_id*	*Unique identifier for employee.*
	name	*Employee full name*
	ssn	*Social Security Number.*
	Job_title	*Job Designation*
	division	*Division name*
	salary	*Monthly Salary*
Pay Statement	*statement_id*	*Unique identifier for pay statement.*
	emp_id	*Employee ID*
	pay_date	*Date of payment*
	amount	*Payment amount*

5.0 COMPONENT DESIGN

Each Java class represents a core functionality:

- **EmployeeManager**: Handles CRUD operations.
- **DatabaseConnector**: Establishes and manages database connections.
- **ReportGenerator**: Generates requested reports.

6.0 HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

The user interface is designed for simplicity, using either a text-based console or a basic JavaFX GUI. Both interfaces facilitate intuitive navigation for the administrator.

6.2 Screen Image

For JavaFX, the main screen includes fields for entering employee data, buttons for CRUD operations, and dropdown menus for generating reports.

6.3 Screen Object and Actions

- **Text Fields**: Input fields for employee attributes.
- **Buttons**: Submit actions for operations (Insert, Update, Delete, Search).
- **Dropdowns**: Selection menus for report generation.

7.0 REQUIREMENTS MATRIX

Requirement ID	***Description***	***Priority***	***Status***
R1	***Insert new employee records***	***High***	***Working***
R2	***Update employee detail***	***High***	***Working***
R3	***Generate payroll report***	***High***	***Working***
R4	***Search employee by ID ,name, or SSN***	***Medium***	***Working***
R5	***Update salary based on percentage increase.***	***Medium***	***Working***

8.0 APPENDICES

- **Appendix A**: MySQL scripts for database setup.
- **Appendix B**: UML diagrams.
- **Appendix C**: Test cases for CRUD operations.
- **Appendix D**: Video demonstration guidelines.

9.0 Test Cases

9.1 Function: Update Employee Data

Test Case : Successful Update of Employee Data

Input:

• Employee ID: "4"

• Changed Data: Job Title = "Senior Developer", Division = "Technology"

Expected Output:

• The database reflects the updated employee information.

• Confirmation message: "Employee data updated successfully."

Initialization:

• Ensure employee "4" exists in the database

Dependencies:

• A working database connection.

• A valid employee ID.

Test Steps:

1. Select employee with ID "4"

2. Update the job title to "Senior Developer" and division to "Technology."

3. Save the changes.

4. Query the database to verify the updates.

Tear Down:

• Return to main menu.

9.2 Test Case Invalid Update of Employee Data

Input:

• Employee ID: "1000"

• Changed Data: Job Title = "Senior Developer", Division = "Technology"

Expected Output:

• The system displays an error message: "Error: No employee found with the given ID."

• No changes are made to the database.

Initialization:

• Ensure employee ID "1000" does not exist in the database.

Dependencies:

• A working database connection.

• A valid but non-existent employee ID.

Test Steps:

1. Attempt to select employee with ID "1000."

2. Attempt to update the job title to "Senior Developer" and division to "Technology."

3. Verify the system displays the error message: "Error: No employee found with the given ID."

4. Query the database to confirm no new rows or changes were made.

Tear Down:

• Return to the main menu.

9.3Function: Search for Employee

Test Case : Valid Search for Employee by ID

Input:

• Employee ID: "1"

Expected Output:

• Employee details are displayed:

o Name: "John Doe"

o All information connected to John Doe.

Initialization:

• Ensure employee ID "1" exists in the database with valid data.

Dependencies:

• A working database connection.

• A valid employee record for ID "1"

Test Steps:

1. Enter employee ID "1" in the search field.

2. Execute the search query.

3. Verify that the retrieved details match the data stored in the database.

Tear Down:

• Return to the main menu.

9.4 Test Case: Invalid Search for Employee by ID

Input:

• Employee ID: "1000"

Expected Output:

• The system displays an error message: "Employee not found."

• No employee details are retrieved.

Initialization:

• Ensure employee ID "1000" does not exist in the database.

Dependencies:

• A working database connection.

• A non-existent employee ID.

Test Steps:

1. Enter employee ID "1000" in the search field.

2. Execute the search query.

3. Verify that no details are retrieved from the database.

4. Verify the error message: "Employee not found" is displayed.

Tear Down:

• Return to the main menu.

9.5 Function: Update Salary for All Employees Less Than a Particular Amount

Test Case 3.1: Valid Update of Salaries Below Threshold

Input:

• Threshold Salary: “0”, "$60,000"

• Percentage Increase: "5%"

Expected Output:

• Employees with salaries less than $60,000 have their salaries updated with a 5% increase.

• Confirmation message: "Salaries updated successfully."

Initialization:

• Ensure at least one employee in the database has a salary less than $60,000.

• Example employee:

o Employee ID: "4"

o Current Salary: "$50,000"

Dependencies:

• A working database connection.

• At least one employee with a salary below the threshold.

Test Steps:

1. Enter the threshold salary as “0”, "$60,000."

2. Enter the percentage increase as "5%."

3. Execute the update query.

4. Query the database to verify that employees with salaries less than $60,000 have been updated.

5. Confirm that other employees (with salaries >= $60,000) remain unaffected.

Tear Down:

• Reset affected employee salaries to their original values.

Test Case 3.2: Invalid Update of Salaries Below Threshold (No Matching Employees)

Input:

• Threshold Salary: "$30,000"

• Percentage Increase: "5%"

Expected Output:

• The system displays an error or warning message: "No employees found with a salary below the threshold."

• No changes are made to the database.

Initialization:

• Ensure no employee in the database has a salary below $30,000.

Dependencies:

• A working database connection.

• No employees meeting the criteria for the update (salary < $30,000).

Test Steps:

1. Enter the threshold salary as "$30,000."

2. Enter the percentage increase as "5%."

3. Execute the update query.

4. Verify that no changes have been made to any employee's salary in the database.

5. Confirm the system displays the error or warning message: "No employees found with a salary below the threshold."

Tear Down:

• Return to the main menu.