Software Design Document (SWDD)

Group 1 Employee Management
System Software Design Document

Names: Mubashar Mian, Krish Patel, Adnan Khan, Hari Thavittupalayam Manivannan,

Section: T/TR
Date: 07/25/2024

Table of Contents

1.0 Introduction

1.1 Purpose

This Software Design Document (SWDD) provides a comprehensive description of the Employee Management System (EMS) design, focusing on the system's architecture, data design, and components. It is intended to guide developers and stakeholders through the system's design and ensure consistency in implementation.

1.2 Scope

The EMS is designed to manage employee records, including personal details, salaries, and pay statements. It supports functionalities such as adding, updating, and deleting employee records, as well as generating various payroll reports.

1.3 Overview

The document outlines the architecture and design of the EMS, including the data model, system components, and their interactions. It also includes a detailed description of the database schema and the pseudocode for the core system functions.

1.4 Reference Material

- IEEE Std 1016-2009
- MySQL Documentation
- Java SE Documentation
- JDBC API Documentation

1.5 Definitions and Acronyms

- **EMS:** Employee Management System
- **SWDD:** Software Design Document
- **DBMS:** Database Management System ● **JDBC:** Java Database Connectivity

2.0 System Overview

The EMS provides a platform for managing employee data and payroll. The system includes functionalities for employee record management, salary updates, and report generation.

3.0 System Architecture

3.1 Architectural Design

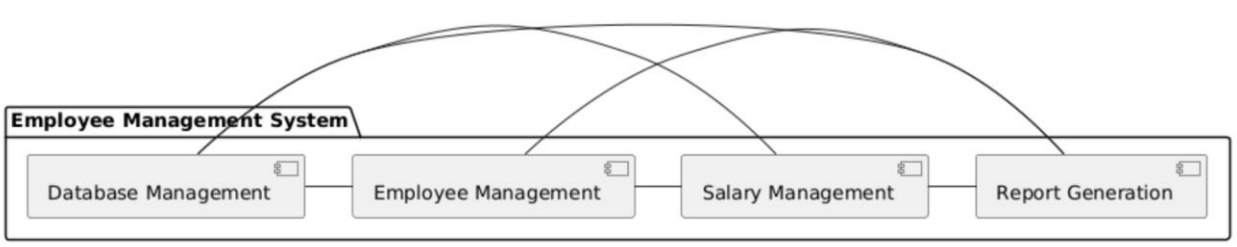
The EMS architecture consists of the following components:

- **Database Management:** Manages database connections and queries.
- **Employee Management:** Handles employee records.
- **Salary Management:** Manages salary updates and pay statements. ● **Report Generation:** Creates various reports.

The system uses a modular approach with well-defined interfaces between components.

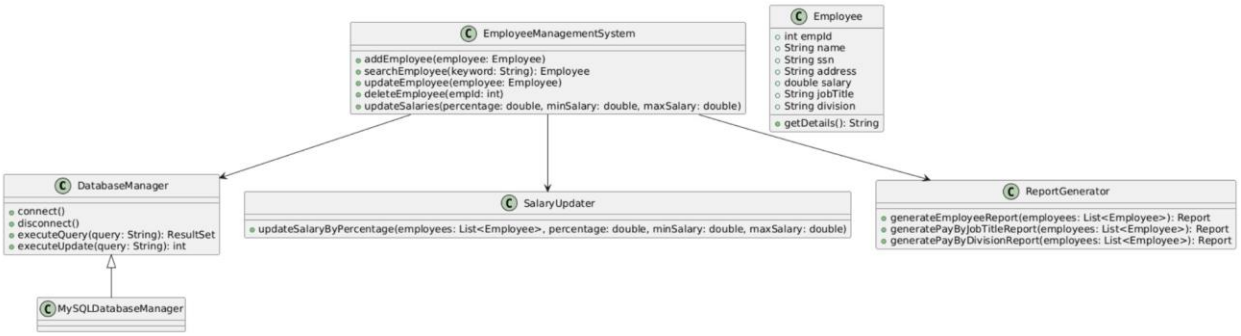
3.2 Decomposition Description Diagrams

1. **Component Diagram**:



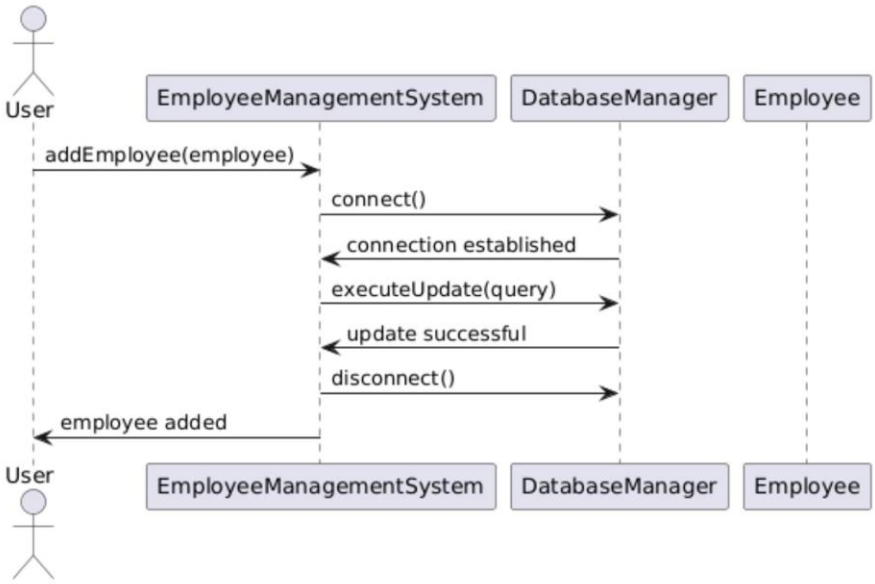
- Identify the main components of your system (e.g., Database Management, Employee Management, Salary Management, Report Generation). ○ Show how these components interact with each other.

2. **Class Diagram**:



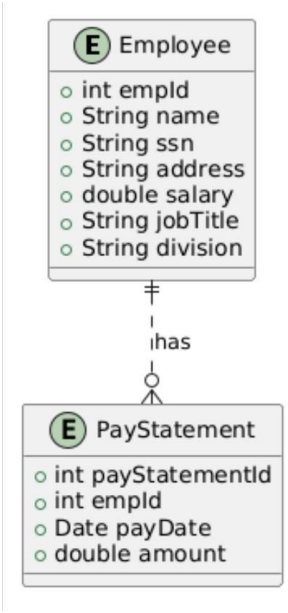
- Define the main classes in your system (e.g., Employee, Salary, Report).
- List the attributes and methods for each class.
- Show the relationships between classes (e.g., inheritance, association).

3. **Sequence Diagram**:



- Choose a key interaction (e.g., adding an employee).
- List the objects involved in this interaction.
- Show the sequence of messages exchanged between these objects to complete the interaction.

4. **ER Diagram**:



- Identify the entities in your database (e.g., Employee, PayStatement).
- Define the attributes for each entity.
- Show the relationships between entities (e.g., one-to-many, many-to-many).

Subsystems

1. **Database Management**
   - Manages connections to the MySQL database.
   - Executes SQL queries.
   - Handles database transactions.
2. **Employee Management**
   - CRUD operations for employee records. ○ Searches and retrieves employee data.
3. **Salary Management**
   - Updates employee salaries based on specified criteria. ○ Processes pay statements.
4. **Report Generation**
   - Generates reports based on employee data and salary information.

Pseudocode

Database Management

- connectToDatabase() {
- // Open connection to MySQL database using JDBC
- }

-
- disconnectFromDatabase() {
- // Close connection to MySQL database
- }
-
- executeQuery(query) { ● // Create a statement
- // Execute the query
- // Return the result set
- }
-
- executeUpdate(query) {
- // Create a statement
- // Execute the update
- // Return the number of rows affected
- }

Employee Management

- addEmployee(employee) {
- connectToDatabase();
- // Prepare SQL INSERT query
- executeUpdate(query);

```
    disconnectFromDatabase();
    // Add employee to local list
}

searchEmployee(keyword) {
    connectToDatabase();
    // Prepare SQL SELECT query with keyword
    executeQuery(query);
    // Retrieve and return employee record
    disconnectFromDatabase();
    }

    updateEmployee(employee) {
    connectToDatabase();
    // Prepare SQL UPDATE query
    executeUpdate(query);
    disconnectFromDatabase();
    }

    deleteEmployee(empId) {
    connectToDatabase();
    // Prepare SQL DELETE query
    executeUpdate(query);
    disconnectFromDatabase();
    // Remove employee from local list
    }
```

Salary Management

```
    updateSalaries(percentage, minSalary, maxSalary) {
    connectToDatabase();
    // Iterate through employee list
```

```
for (each employee within salary range) {
// Calculate new salary
// Update employee record
}
disconnectFromDatabase();
}
```

Report Generation

```
generateEmployeeReport(employees) {
  // Create a new report for
  (each employee in list) {
    // Add employee details to report
  } return
  report;
}
```

```
generatePayByJobTitleReport(employees) {
// Create a new report
// Calculate total pay by job title
// Add results to report
return report;
}

generatePayByDivisionReport(employees) {
// Create a new report
// Calculate total pay by division
// Add results to report
return report;
}
```

-
-
-
-
-
-
-

3.3 Design Rationale

The design prioritizes modularity, allowing for separate development and testing of components. Interfaces facilitate flexibility, enabling changes to the underlying implementations without affecting the rest of the system. This approach supports scalability and maintainability.

4.0 Data Design

4.1 Data Description

The system uses a MySQL database to store employee and pay statement data. The primary tables are:

- **employees:** Stores detailed employee records.
- **pay_statements:** Stores payment records associated with employees.

4.2 Data Dictionary

Tables

employees empId

(INT, Primary Key,

Auto Increment)

name

(VARCHAR(100),

Not Null) ssn

(VARCHAR(9), Not

Null) address

(VARCHAR(255),

-
-
-
-
-
-
-

Not Null) salary

(DOUBLE, Not Null)

jobTitle

(VARCHAR(50))

division

(VARCHAR(50))

pay_statements

- payStatementId (INT, Primary Key, Auto Increment)
- empId (INT, Foreign Key)
- payDate (DATE)
- amount (DOUBLE)

5.0 Component Design

Component Specifications

DatabaseManager

- Interface for database operations.
- Methods: connect(), disconnect(), executeQuery(query), executeUpdate(query)

MySQLDatabaseManager

- Implements DatabaseManager for MySQL.
- Uses JDBC to interact with MySQL databases.

EmployeeManagementSystem

- Manages employee records and interacts with the database.

-
-
-
-
-
-
-
- Methods: addEmployee(employee), searchEmployee(keyword), updateEmployee(employee), deleteEmployee(empId), updateSalaries(percentage, minSalary, maxSalary)

SalaryUpdater

- Handles salary updates.
- Method: updateSalaryByPercentage(employees, percentage, minSalary, maxSalary)

ReportGenerator

- Interface for generating reports.
- Methods: generateEmployeeReport(employees), generatePayByJobTitleReport(employees), generatePayByDivisionReport(employees)

SimpleReportGenerator

- Implements ReportGenerator.
- Generates employee and payroll reports.

6.0 Human Interface Design

6.1 Overview of User Interface

System Overview

The Employee Management System provides a minimalistic yet effective graphical user interface (GUI) using JavaFX. This interface allows users to manage employee data and generate reports through various functionalities. The user interactions are designed to be intuitive and straightforward, enabling efficient management of employee records without the need for complex operations.

User Functionality and Interaction

Main Window

- **Screen Objects:**
  - Title Bar: Displays the name of the application.
  - Tabs/Sections: Organized areas for various functionalities such as Add Employee, Search Employee, Update Employee, Delete Employee, and Update Salaries.
- **Actions:**
  - Navigating Sections: Users can switch between different functionalities by selecting the corresponding tab or section.

Add Employee Section

- **Screen Objects:**
  - Text Fields:
    - Name: Input field for the employee's name.
    - SSN: Input field for the employee's Social Security Number.
    - Address: Input field for the employee's address.
    - Salary: Input field for the employee's salary.
    - Job Title: Input field for the employee's job title.
    - Division: Input field for the employee's division.
  - Add Employee Button: Button to submit the new employee information.
- **Actions:**
  - Enter Data: Users input employee details into the respective fields.

    - Add Employee: Clicking the button sends the data to the system for adding the employee to the database.

Search Employee Section

- **Screen Objects:**
    - Search Field: Input field for entering keywords to search for an employee.
    - Search Button: Button to initiate the search operation.
    - Search Results: Display area for showing the list of matching employee records.
- **Actions:**
    - Enter Search Criteria: Users input search keywords.
    - Execute Search: Clicking the button retrieves and displays matching employee records.

Update Employee Section

- **Screen Objects:**
    - Employee List: Dropdown or list of existing employees for selection.
    - Text Fields:
        - Name: Editable field for the employee's name.
        - SSN: Editable field for the employee's Social Security Number.
        - Address: Editable field for the employee's address.
        - Salary: Editable field for the employee's salary.
        - Job Title: Editable field for the employee's job title.
        - Division: Editable field for the employee's division.
    - Update Employee Button: Button to submit the updated employee information.
- **Actions:**
    - Select Employee: Users select the employee to be updated.
    - Edit Data: Users modify the employee details.
    - Update Employee: Clicking the button sends the updated data to the system for updating the employee record.

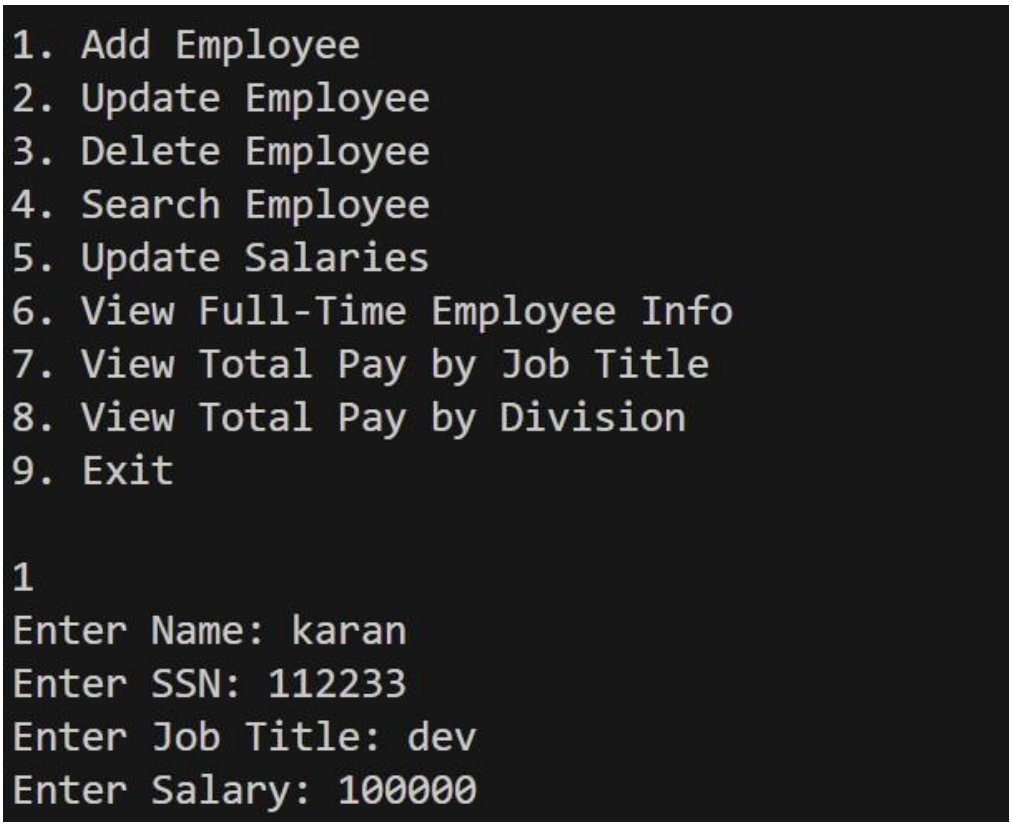
Delete Employee Section

- **Screen Objects:**
    - Employee List: Dropdown or list of existing employees for selection.
    - Delete Employee Button: Button to confirm the deletion of the selected employee.
- **Actions:**
    - Select Employee: Users select the employee to be deleted.
    - Confirm Deletion: Clicking the button removes the selected employee from the database.

Update Salaries Section

- **Screen Objects:**
    - Percentage Field: Input field for entering the salary update percentage.
    - Minimum Salary Field: Input field for specifying the minimum salary range.
    - Maximum Salary Field: Input field for specifying the maximum salary range.
    - Update Salaries Button: Button to apply the salary updates.
- **Actions:**
    - Enter Criteria: Users input the percentage and salary range.
    - Update Salaries: Clicking the button updates the salaries of employees within the specified range.

6.2 Screen Objects and Actions



7.0 Requirements Matrix

Functional Requirement ID	Description	System Component	Satisfaction criteria

R1	Add new employees records	Employ Management System	Employee data is added to the database and retrievable
R2	Search for employees	Employ Management System	Search results match the query criteria.
R3	Update existing employees records	Employ Management System	Employee data is updated and changes persist.

R4	Delete employees records	Employ Management Employ Management System	Employee data is removed from the database.
R5	Update salaries within specified range	Salary Updater	Salaries are updated based on the specified criteria
R6	Generate employed report	Report Generator	Report accurately reflects current employee data.
R7	Generate pay by job title report	Report Generator	Report accurately reflects pay distribution by job title
R8	Generate pay by division report	Report Generator	Report accurately reflects pay distribution by division.