CSc3350 Software Development
Group Team Project
Spring 2024

Team: Infinite Loopers

Project Title: Employee Management System

Software Design Document

Team Members:

1. Christian Mahary (cmahary1)
2. Nahom Abera (nabera1)
3. Samuel Alemu (salemu3)
4. Samuel Holison (sholison1)
5. Victoria Nwanze (vnwanze1)

Due Date: 04/28/2024

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Purpose

The purpose of this SWDD (Software Design Document) is to outline the design of a basic employee management system for Company Z. It details the architecture and functionalities of the system.

The intended audience is anyone involved in the development and implementation of this system, including Infinite Loopers' team members, Company Z database administrators, Company Z project managers and future developers.

The employee management system is designed to streamline various aspects of managing employee information within an organization. It provides a centralized platform for efficiently handling tasks related to employee data management. The system offers the following functionalities:

1. Add Employee: Allows the addition of new employees to the database, capturing essential details such as employee ID, name, contact information, hire date, salary, address, and date of birth.
2. Remove Employee: Provides the capability to remove employees from the database to ensure up-to-date employee records.
3. Search Employee: Enables users to search for employees and display their information based on the following searching criteria: first name, last name, employee ID or SSN (if the column added already).
4. Update Employee Data: Allows for the modification of employee information, including fields such as name, email, salary, address, and date of birth.
5. Update Employee Division: Allows users to update an employee's division within the organization. Users can select from available divisions and assign employees to the appropriate division.
6. Update Employee Salary Range: Provides functionality to adjust the salary range for employees within specified minimum salary thresholds.
7. Add SSN Column to Database: Offers the ability to add a Social Security Number (SSN) column to the database schema.
8. Full Time Employee Information with past payroll history: Generates reports for full-time employees, including detailed information about their employment status, job titles, salary history, and past payroll records.
9. Part Time Employee Information with past payroll history: Similar to full-time employee reports, this functionality generates detailed reports specifically tailored to part-time employees, including relevant payroll history.
10. Generate Reports by Division or Job Titles: Provides reporting capabilities to generate customized reports based on employee divisions or job titles.

2.0 DATABASE SCHEMA DIAGRAM

The Employee Management System is made upon the “employeedata” database which contains important information of Company Z employees. The “employeedata” have the following structure:

1. “employees” Table: This is the central table containing personal details about each employee such as:
    - Employee ID (“empid”): The primary key for the employees table.
    - First name (“Fname”)
    - Last name (“Lname”)
    - Email address (“email”)
    - Hire date (“HireDate”)
    - Salary (“Salary”)
    - Address (“address”)
    - Date of birth (“date_of_birth”)
2. “division” Table: Stores information about different divisions within the company, such as:
    - Division ID (“id”): The primary key for the division table.
    - Name (“Name”)
    - City (“city”)
    - Address lines (“addressLine1” and “addressLine2”)
    - State (“state”)
    - Country (“country”)
    - Postal code (“postalCode”)
3. “employee_division” Table: A table that links employees to their respective divisions.
    - Employee ID (“empid”)
    - Division ID (“div_id”)
4. “job_titles” Table: Lists different job titles within the company:
    - Job Title ID (“job_title_id”): The primary key for the “job_titles” table.
    - Job Title (“job_title”)
5. “employee_job_titles” Table: Another junction table that connects employees to their job titles.
    - Employee ID (“empid”)
    - Job Title ID (“job_title_id”)
6. “payroll” Table: Contains payment details for full-time employees,

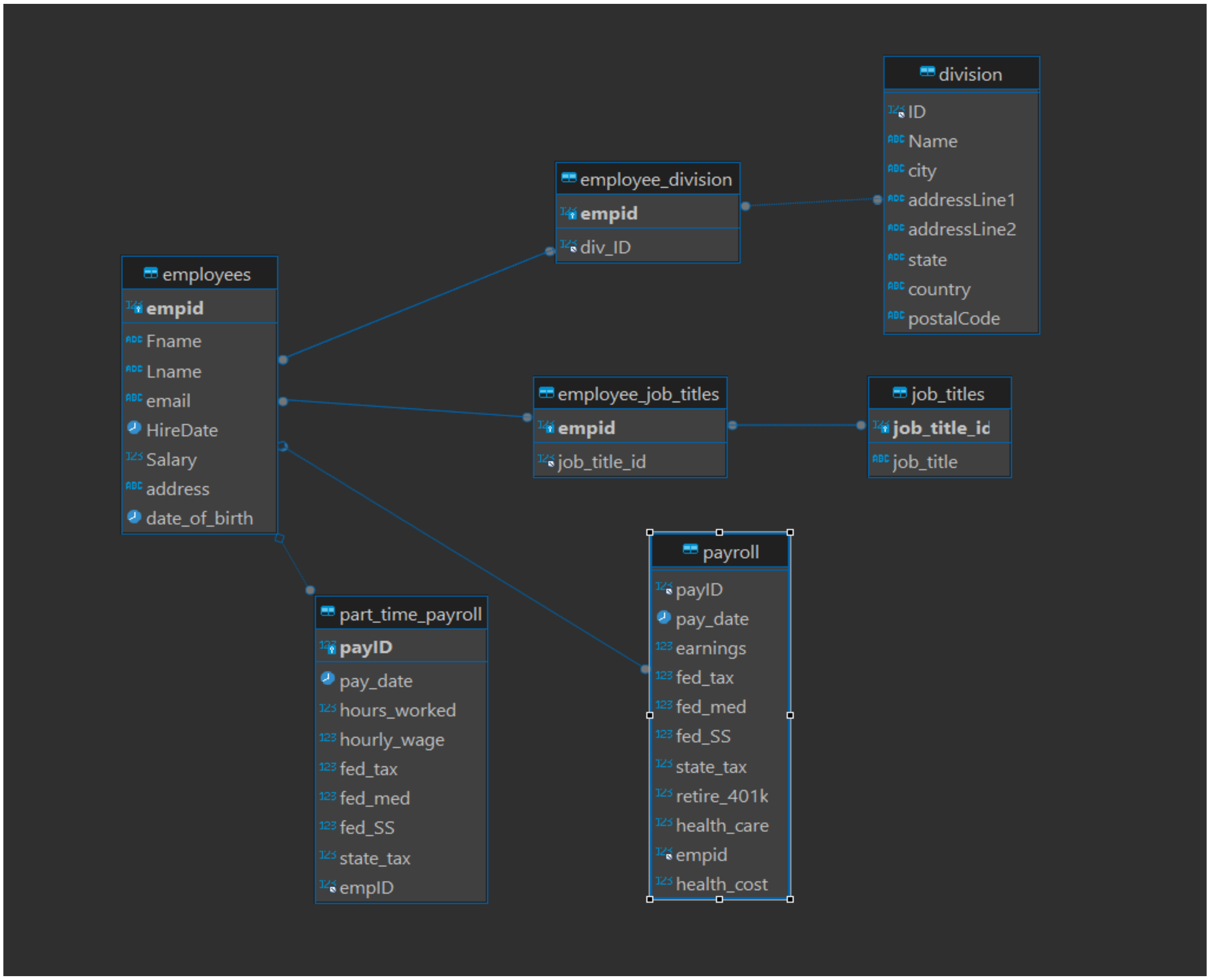
- Pay ID (“payid”): The primary key for the payroll table.
- Pay date (“pay_date”)
- Earnings (“earnings”)
- Federal tax (“fed_tax”)
- Federal Medicare (“fed_med”)
- Federal Social Security (“fed_SS”)
- State tax (“state_tax”)
- Retirement 401(k) contributions (“retire_401k”)
- Health care (“health_care”)
- Health care cost (health_cost)
- Employee ID (“empid”)

7. part_time_payroll Table: Presumably for part-time employees, containing details about their pay:
    - Pay ID (payID): The primary key for the part_time_payroll table.
    - Pay date (pay_date)
    - Hours worked (hours_worked)

- Hourly wage (hourly_wage)
- Federal tax (fed_tax)
- Federal Medicare (fed_med)
- Federal Social Security (fed_SS)
- State tax (state_tax)
- Employee ID (empid)

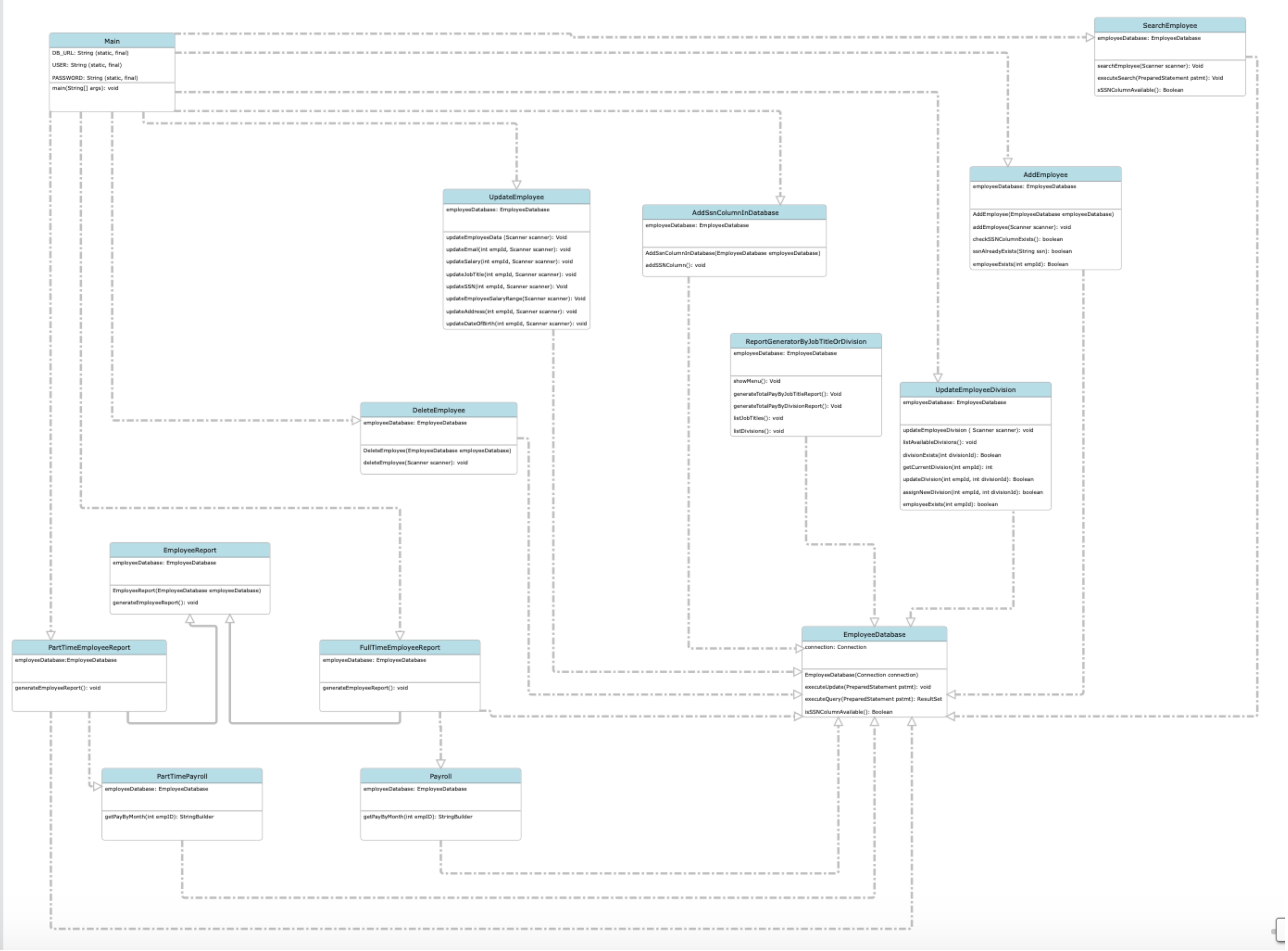
Note:

- The “employees” table primary key “empid” servers as a link between the “employees” table and the “payroll”, “part_time_payroll”, “employee_job_titles”, and “employee_division” tables.
- The “job_title_id” column serves as “job_titles” primary key and “employee_job_titles” table foreign key to create a link between these two tables.
- The “divisions” table primary key “id” serves as a reference point for the “div_id” column in the “employee_division” table to link the employees with their divisions’ detail.



3.0 JAVA CLASS DIAGRAM

This the Java Class Diagram of the System:


Main
DB_URL: String (static, final)
USER: String (static, final)
PASSWORD: String (static, final)
main(String[] args): void
SearchEmployee
employeeDatabase: EmployeeDatabase
searchEmployee(Scanner scanner): Void
executeSearch(PreparedStatement pstmt): Void
sSSNColumnAvailable(): Boolean
AddEmployee
employeeDatabase: EmployeeDatabase
AddEmployee(EmployeeDatabase employeeDatabase)
addEmployee(Scanner scanner): void
checkSSNColumnExists(): boolean
ssnAlreadyExists(String ssn): boolean
employeeExists(int empId): Boolean
UpdateEmployee
employeeDatabase: EmployeeDatabase
updateEmployeeData (Scanner scanner): Void
updateEmail(int empId, Scanner scanner): void
updateSalary(int empId, Scanner scanner): void
updateJobTitle(int empId, Scanner scanner): void
updateSSN(int empId, Scanner scanner): Void
updateEmployeeSalaryRange(Scanner scanner): Void
updateAddress(int empId, Scanner scanner): void
updateDateOfBirth(int empId, Scanner scanner): void
AddSsnColumnInDatabase
employeeDatabase: EmployeeDatabase
AddSsnColumnInDatabase(EmployeeDatabase employeeDatabase)
addSSNColumn(): void
ReportGeneratorByJobTitleOrDivision
employeeDatabase: EmployeeDatabase
showMenu(): Void
generateTotalPayByJobTitleReport(): Void
generateTotalPayByDivisionReport(): Void
listJobTitles(): void
listDivisions(): void
UpdateEmployeeDivision
employeeDatabase: EmployeeDatabase
updateEmployeeDivision (Scanner scanner): void
listAvailableDivisions(): void
divisionExists(int divisionId): Boolean
getCurrentDivision(int empId): int
updateDivision(int empId, int divisionId): Boolean
assignNewDivision(int empId, int divisionId): boolean
employeeExists(int empId): boolean
DeleteEmployee
employeeDatabase: EmployeeDatabase
DeleteEmployee(EmployeeDatabase employeeDatabase)
deleteEmployee(Scanner scanner): void
EmployeeReport
employeeDatabase: EmployeeDatabase
EmployeeReport(EmployeeDatabase employeeDatabase)
generateEmployeeReport(): void
PartTimeEmployeeReport
employeeDatabase:EmployeeDatabase
generateEmployeeReport(): void
FullTimeEmployeeReport
employeeDatabase: EmployeeDatabase
generateEmployeeReport(): void
EmployeeDatabase
connection: Connection
EmployeeDatabase(Connection connection)
executeUpdate(PreparedStatement pstmt): void
executeQuery(PreparedStatement pstmt): ResultSet
isSSNColumnAvailable(): Boolean
PartTimePayroll
employeeDatabase: EmployeeDatabase
getPayByMonth(int empID): StringBuilder
Payroll
employeeDatabase: EmployeeDatabase
getPayByMonth(int empID): StringBuilder

4.0 TEST CASES

a. Add Employee

Task Description: This function allows adding new employees to the database.

Test Cases:

- Test Case 1 (Pass): New Employee added to the database
  - Input: New Employee’s id, Name, Hire date, Address, Salary, Email, DOB
  - Expected Result:
    - The system prompts for the new employee’s id
    - The system checks if the new employee id doesn’t exist to achieve employee id uniqueness
    - Then the user is prompted to enter the new employee’s Name, Hire date, Address, Salary, Email, DOB
    - The system updates the database with the new employee information.
    - Then a success message will be shown upon.
- Test Case 2 (Fail): Add Employee with existing employee id
  - Input: Already existing Employee’s id
  - Expected Result:
    - The system prompts for the new employee’s id
    - The system checks if the new employee id doesn’t exist to achieve employee id uniqueness
    - The system shows an error message showing the provided employee id already exists

b. Add SSN Column

Task Description: This function allows adding the SNN column to the database.

Test Cases:

- Test Case 1 (Pass): Addition of the of the SSN column to the database
  - Input:
  - Expected Result:
    - The user will choose the add SSN option from the main menu
    - The system will check if the SSN column exists or not
    - Then success message will be displayed and the user is notified to select the option to update employee details to add SSN for already existing employees
- Test Case 2 (Fail): SSN Column Already exists
  - Input: Already existing Employee’s id
  - Expected Result:
    - The user will choose the add SSN option from the main menu
    - The system will check if the SSN column exists or not
    - The system shows an error message showing the SSN Column already exists in the database

c. Update Employee Data

Task Description: This function allows modifying any field in an employee's record. It should retrieve the existing employee information, prompt the user for changes, update the database with the modified data, and confirm the update.

Test Cases:

- Test Case 1 (Pass): Update Employee Address
  - Input: Existing employee ID, new employee address
  - Expected Result:
    - The system prompts for employee's id
    - The system checks if an employee with the given employee id exits
    - Then the user is prompted for a new address
    - The system updates the database with the new address.
    - Then a success message will be shown upon successful name update.

Test Case 2 (Fail): Update Employee address with invalid Employee id

a. Input: Non-Existing employee ID, new employee address

b. Expected Result:

i. The system prompts for employee's id

ii. The system checks if an employee with the given employee id exits

iii. The system shows an error message showing there's no registered employee with the given employee id

d. Search Employee

Task Description: This feature will enable finding employees based on name, SSN, or employee ID. It will allow the user to specify the search criteria and return matching employee information.

Test Cases:

- Test Case 1 (Pass): Search by Employee ID
  - Input: Valid employee ID
  - Expected Result:
    - The system will prompt for search criteria (e.g., name, SSN, ID).
    - Then the user enters the employee ID.
    - Then the system retrieves matching employee information and displays it to the user.
- Test Case 2 (Fail): Search by Non-existent Employee Name
  - Input: Name of a person not currently employed in the company('Z')
  - Expected Result:
    - The system will ask for search criteria (e.g., name, SSN, ID).
    - Then the user enters the name.
    - Then the system will try to retrieve the employee information based on the given name and display a message indicating no matching employee found for the input name.

e. Update Salary for Employees Less Than a Particular Amount

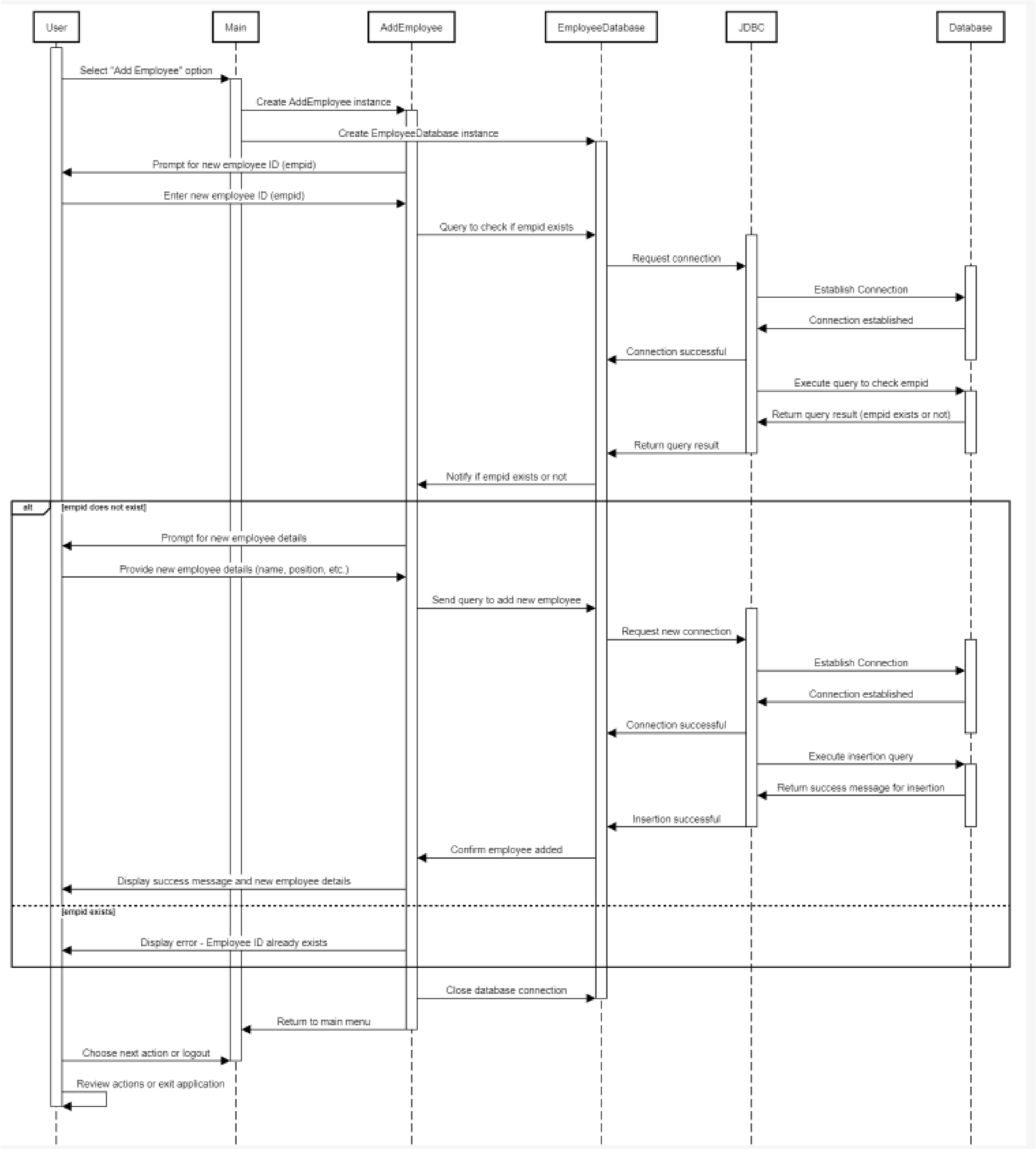
Task Description: This feature will increase salaries by a specific percentage for employees whose current salary is below a certain amount. It should retrieve employee data, calculate the new salary based on the percentage increase, and update the database for qualifying employees.

Test Cases:

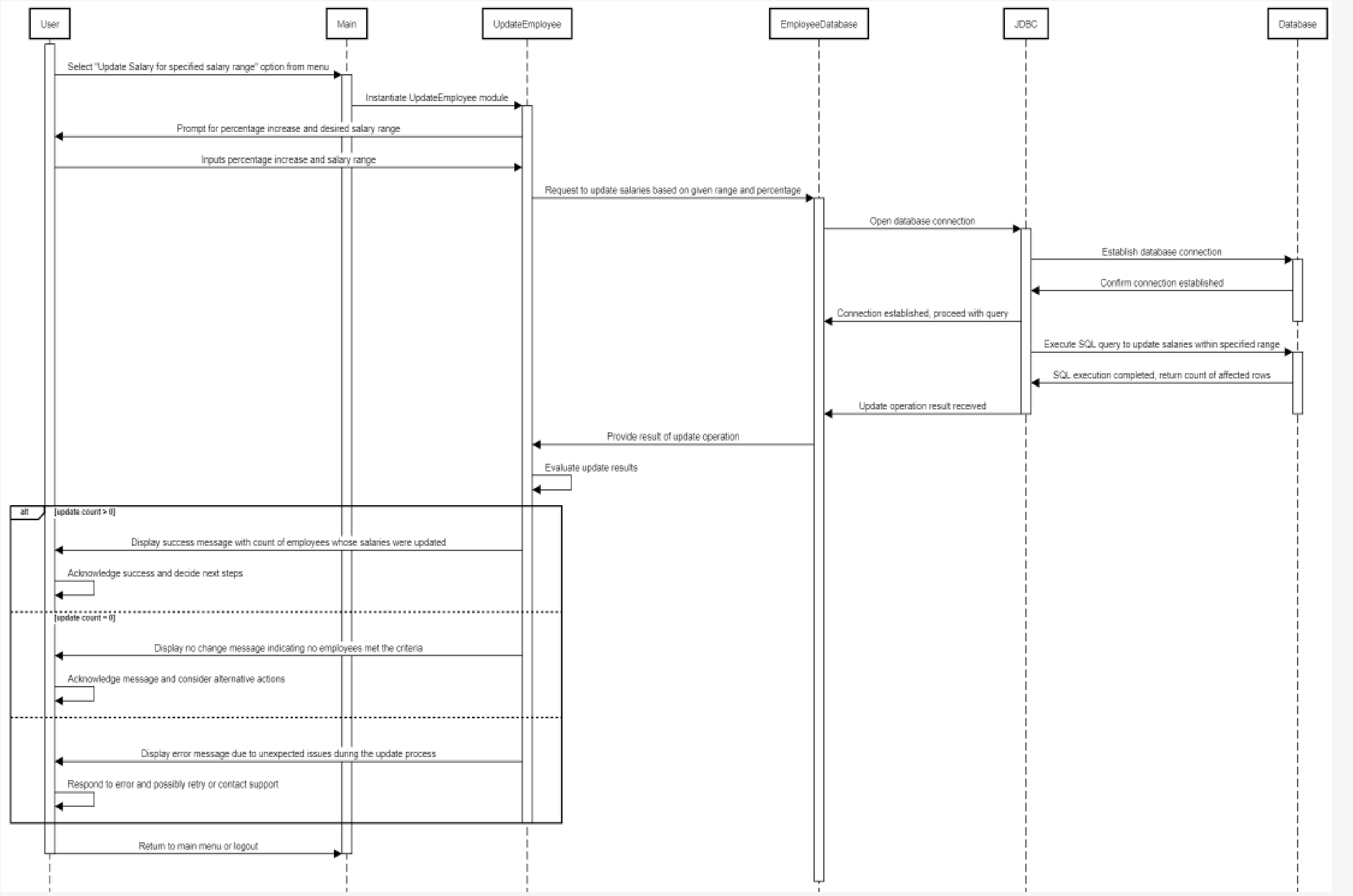
- Test Case 1 (Pass): Update Salary for Eligible Employees
  - Input: Percentage increase and salary range
  - Expected Result:
    - The system retrieves employees within the given salary range.
    - Then it calculates and updates the salary for the employees within the given salary range
    - Then it will display confirmation with the number of employees whose salaries were updated.
- Test Case 2 (Fail): No Salary Update for Employees below given Amount.
  - Input: Percentage increase and a salary amount which is below all employees' salary
  - Expected Result:
    - The system will try to retrieve the employee data.
    - Then it notifies the use that no employee salaries were updated because no employee met the salary range criteria.

5.0 SEQUENCE DIAGRAMS

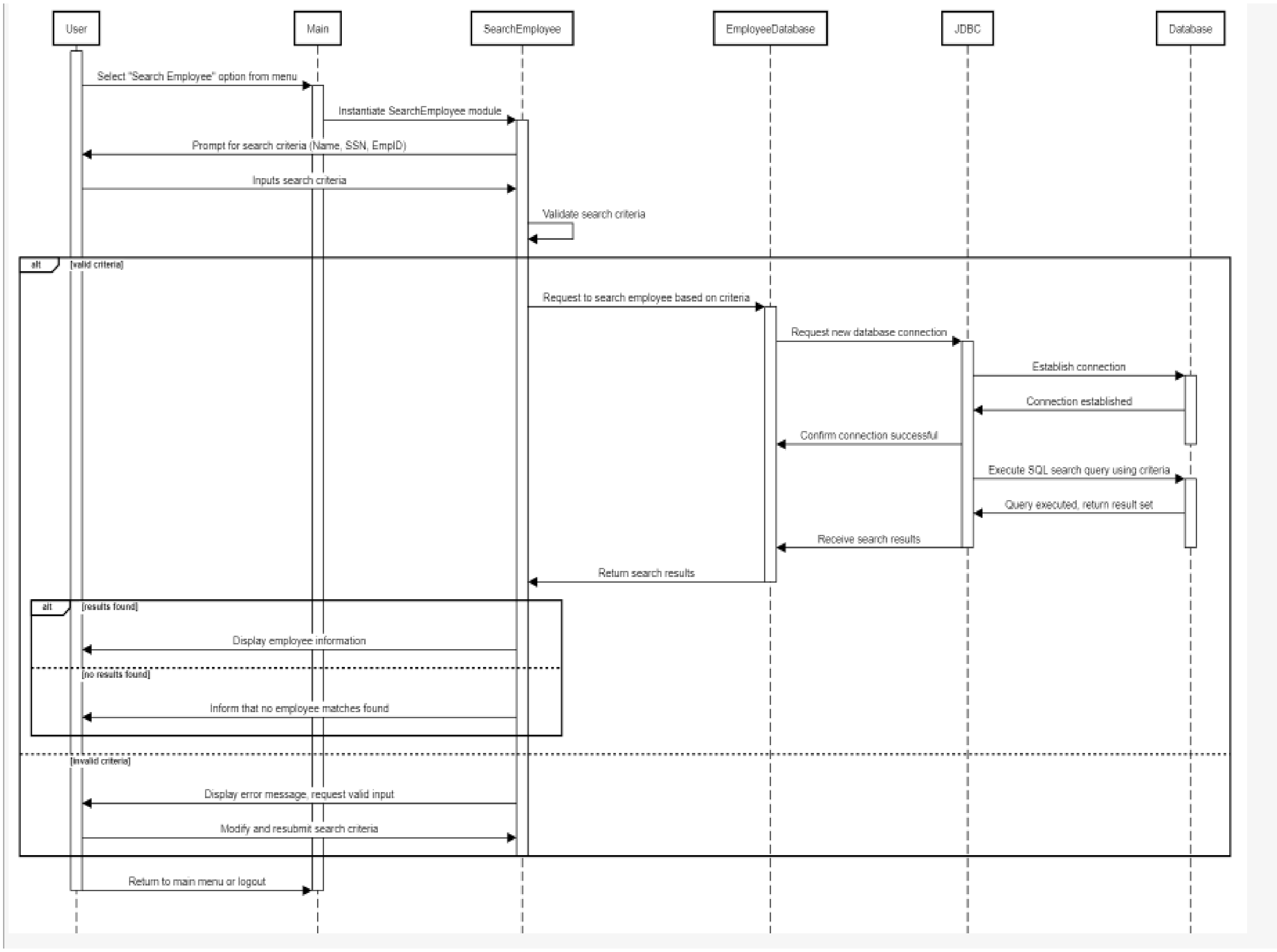
a) Add New Employee


User
Main
AddEmployee
EmployeeDatabase
JDBC
Database
Select "Add Employee" option
Create AddEmployee instance
Create EmployeeDatabase instance
Prompt for new employee ID (empid)
Enter new employee ID (empid)
Query to check if empid exists
Request connection
Establish Connection
Connection established
Connection successful
Execute query to check empid
Return query result (empid exists or not)
Return query result
Notify if empid exists or not
alt
[empid does not exist]
Prompt for new employee details
Provide new employee details (name, position, etc.)
Send query to add new employee
Request new connection
Establish Connection
Connection established
Connection successful
Execute insertion query
Return success message for insertion
Insertion successful
Confirm employee added
Display success message and new employee details
[empid exists]
Display error - Employee ID already exists
Close database connection
Return to main menu
Choose next action or logout
Review actions or exit application

b) Increase salary by % if current base is less than a given amount


User
Main
UpdateEmployee
EmployeeDatabase
JDBC
Database
Select "Update Salary for specified salary range" option from menu
Instantiate UpdateEmployee module
Prompt for percentage increase and desired salary range
Inputs percentage increase and salary range
Request to update salaries based on given range and percentage
Open database connection
Establish database connection
Confirm connection established
Connection established, proceed with query
Execute SQL query to update salaries within specified range
SQL execution completed, return count of affected rows
Update operation result received
Provide result of update operation
Evaluate update results
alt
[update count > 0]
Display success message with count of employees whose salaries were updated
Acknowledge success and decide next steps
[update count = 0]
Display no change message indicating no employees met the criteria
Acknowledge message and consider alternative actions
Display error message due to unexpected issues during the update process
Respond to error and possibly retry or contact support
Return to main menu or logout

c) Search Employee


User
Main
SearchEmployee
EmployeeDatabase
JDBC
Database
Select "Search Employee" option from menu
Instantiate SearchEmployee module
Prompt for search criteria (Name, SSN, EmpID)
Inputs search criteria
Validate search criteria
alt
[valid criteria]
Request to search employee based on criteria
Request new database connection
Establish connection
Connection established
Confirm connection successful
Execute SQL search query using criteria
Query executed, return result set
Receive search results
Return search results
alt
[results found]
Display employee information
[no results found]
Inform that no employee matches found
[invalid criteria]
Display error message, request valid input
Modify and resubmit search criteria
Return to main menu or logout