Group 6

Project Title: Employee Management System

Software Design Document

Names: Trajuan Smith, Ryan Semancik, Deepak Govindarajan, Tarun Gadhiraju, Deborah Maignan, Aaron Mcleed
Class: CSC 3350
Workstation:

Date: (12/02/2025)

Table of Contents

1. Introduction

1.1 Purpose

This Software Design Document (SWDD) describes the architecture, system design, and detailed data structure of the Employee Management System (EMS). The purpose of this document is to translate the software requirements into a comprehensive representation of software components and interfaces, serving as the primary reference for the implementation and coding phase

1.2 Scope

The Employee Management System is a desktop application designed to manage fundamental human resource (HR) data and operations within a small-sized enterprise.

- Goals: to provide an intuitive interface for HR personnel to maintain accurate, secure, and easily accessible employee records.
- Objectives: Implement core functionality, including creating, reading, updating, and deleting (CRUD) employee records.
- Benefits: Reduce manual data entry errors, centralize employee information, and facilitate

1.3 Overviews

Section 2.0 provides a general overview of the system's functionality. Section 3.0 details the high-level system architecture and component decomposition. 4.0 describes the data design and dictionary. Subsequent sections detail the component design, human interface, and the requirements matrix.

1.4 Reference Material

- Software Requirements Specification (SWRS) - [SWRS Document number TBD]
- IEEE Std 1016: IEEE Recommended Practice for Software Design Description.

1.5 Definitions and Acronyms

- CRUD: Create, Read, Update, Delete (The four basic operations of persistent storage).
- DAO: Data Access Object. A design pattern used to abstract and encapsulate all access to the data source.
- EMS: Employee Management System.
- JDBC: Java Database Connectivity. API used to connect and interact with the MySQL database
- MVC: Model-View-Controller. An architectural pattern separating the application into three interconnected parts.

2.0 System Overview

The Employee Management System (EMS) is a data-centric application that acts as a secure intermediary between a user (HR personnel) and the persistent data storage (MySQL Database)

The core functionality includes:

1. Employee Management: allowing authorized users to **add** new employees, **view** existing employee details, **update** employee information ()e.g., salary, department, contact info), and **delete** records.
2. User Authentication: A simple login mechanism to ensure only authorized users can access and modify data.
3. Reporting: Basic functionality to view a list of all current employees.

The system uses Java 21 as the primary development language, Maven for dependency management, and MySQL as the relational database backend.

3.0 System Architecture

3.1 Architectural Design

The EMS will utilize a three-tier Architecture to provide clear separation of concerns, flexibility, and maintainability. The system is partitioned into three distinct subsystems:

1. Presentation Tier (view): the User Interface (UI) that interacts with the user
2. Application Tier (Logic/Controller): contains the business logic, processing user request, and managing data flow.
3. Data Tier (DAO/Database): responsible for persistent storage, retrieval, and management of all application data.

The flow of control starts at the Presentation Tier, moves through the Application Tier for processing, and terminates at the Data Tier for storage or retrieval. The result follows the reverse path back to the user.

3.2 Decomposition Description

The system is decomposed using an Object-Oriented (OO) approach and is expanded to explicitly include the Service Layer and the critical database utility (DatabaseInit). The key subsystems and objects are defined below:

Subsystem/Object	Type	Role and Responsibilities
com.emp_mgmt.model	Subsystem	Holds all data structures (JAVA POJOs) representing real-world entities.
Employee, Payroll, Division, JobTitle	Object (Model)	Holds data attributes for all entities. Used to transfer data between tiers.
com.emp_mgm.db	Subsystem	Handles all low-level communication with the database.
DatabaseConnectionManager	Object (Utility.Singleton)	Manages the singleton instance of the database connection. Responsible for reading .env configuration, connecting via JDBC validation, and closing connections.
DatabaseInit	Object (Utility)	Initialized the database structure and seed data by reading schema.sql

		and sample-data.sql files, ensuring tables are created only if they do not already exist
EmployeeDAO, PayrollDOA, DivisionDOA, JobTitleDOA, EmployeeDivisionDAO, EmployeeJobTitleDAO	Object (DAO)	Implements atomic CRUD operations and specialized queriesfor their respectivetables using JDBC/SQL.
com.employeemgmt.service	Subsystem	The Business Logic Layer. Responsible for coordinating actions across multiple DAOs to fulfill complex business processes
Employee Services	Object (Service)	Manages all employee-related business logic. Coordinates EmployeeDAO, DivisionDAO, and JobTitleDAO to ensure data consistency across multiple tables.
ReportService	Object (Service)	Manages complex data retrieval and calculations for reports. Coordinates all DAOs to gather comprehensive employee data.
com.employeemgmt.ui	Subsystem	Contains the User Interface components
ConsoleUI	Object (View/Controller)	The main application controller. Manages the primary menu and delegates user inout to the specific sub-console objects (EmployeeCoonsole, ReportConsole).
EmployeeConsole	Object (View/Controller)	Handles user interaction for all Employee Managment functions (e.g., displaying forms, handling CRUD input).
ReportConsole	Object (View/Controller)	Handles user interaction for all Reporting functions (e.g., displaying report options, printing results).
com.emp_mgmt.controller	Subsystem	Contains the primary business logic and controls the data flow between the View and DAO.
EmployeeController	Object (Controller)	Validates user input, calls the appropriate method across all DAOs, and enforces business rules.

3.3 Design Rationale

The three-tier architecture was selected for the Employee management system due to its significant advantages in maintainability, flexibility and testability

Key advantages:

- Separation of Concerns: Each layer (presentation, application/controller, data/DAO) has a single, well-defined responsibility. This means that a change in the database is isolated to the DAO layer and does not impact the business logicor the user interface.
- Flexibility and Scalability: By abstracting data access through the DAO layer, the business logic is not tied to MySQL's specific implementation. If the team decided to migrate the database in the future, only the com.emp_mgmt.db package would need to be rewritten, leaving the controller logic intact.

Critical Issues and Trade-offs

While the Three-Tier approach is robust, a trade-off was considered:

- Increased code Overhead: This architecture required more classes. This adds initial complexity compared to a monolithic application where all logic resides in one place
- Mitigation: The team accepted this overhead because the long-term benefits of maintainability and collaboration among team memberssignificantly outweigh the initial cost

The architecture ensures that the project remains manageable, especially as new features are added later, which is why it was chosen over a simpler, less scalable design.

4.0 Data Design

4.1 Data Description

The information domain is transformed into a relational database model composed of five primary tables and two mapping tables to manage relationships.

- Core Entities: The system is centered on the employees entity, which tracks basic personnel data (name, SSN, email).
- One-to-Many Relationships: the payroll table is linked to the employees table, allowing one employee to have multiple historical payroll records
- Many-to-Many Relationships:
    - The relationship between employees and division is managed by the employee_division table, allowing an employee to belong to multiple divisions, or a division to contain multiple employees.
    - The relationship between employees and job_titles is managed by the employee_job_titles table, allowing employees to hold multiple job titles or a title to be held by multiple employees.

This relationship structure ensures data integrity and flexibility for future reporting needs.

4.2 The following tables and their attributes define the system entities:

Entity	Type	Attributes	Description
employees			
payroll			
division			
job_titles			
employee_division			
employee_job_titles			

5.0 Component Design
6.0 Human Interface Design
6.1
6.2
6.3
7.0
7.1