Documentation of Employee Management System project

Name	ID
Mohamed Ahmed Hassan	2101219
Mohamed Ali Elsayed	2101224
Mohammed Hussien Aly sakr	2101281
Mohammed ebrahem farghly	2101279
Mohamed Atef Diab Mohamed	2101248
Omar Youssef Abdallah	2101415

Employee Management System

Introduction

This project demonstrates a **Employee Management System** integrating an SQL database with a Java application. The system handles **departments**, **employees**, and **payroll records** using **Builder Pattern, Singleton Pattern, Factory Pattern, Proxy Pattern, Repository and Adapter Pattern**.

The system involves:

- Managing **employees** (Full-Time, Part-Time, Contractor) using different patterns.
- Integrating payroll processing and legacy payroll systems.
- Connecting the system to a relational database (**Microsoft SQL server**) for data storage.

Database integration in java

To connect the Java code to the MySQL server database, we will use the **JDBC (Java Database Connectivity)** API.

Design Patterns

The system is scalable, well-structured, and demonstrates practical use of:

- **Singleton Pattern:** Ensures a single, shared instance of the database connection manager.
- **Proxy Pattern:** Adds a layer of control and security for the database connection.
- **Builder Pattern**: Simplifies object creation for Employees.
- **Factory Pattern:** Provides a centralized way to create different departments dynamically.
- **Adapter Pattern:** Allows the integration of legacy payroll systems with new components.
- **Repository Pattern** :Encapsulates database access logic, providing CRUD operations for Employee objects.

Use Case Senarios

- **Scenario 1: Adding a New Employee**
  - A user can dynamically create a new employee of any type (Full-Time, Part-Time, Contractor) and associate them with a department.
  - The new employee's details are stored in the database, and their salary is updated in the payroll table.
- **Scenario 2: Processing Payroll**
  - The system retrieves employee salary details from the database and processes payroll.
  - For legacy compatibility, the system also updates the **Legacy Payroll** table via the Adapter.
- **Scenario 3: Real-Time Data Retrieval**
  - A user can view all employees with their department names and salaries.
  - performance management.
- **Scenario 4: Ensuring Secure Database Access**
  - The system establishes a database connection through the **Proxy** to ensure that the connection is secure and monitored.
  - Connection logs can be saved for further auditing.

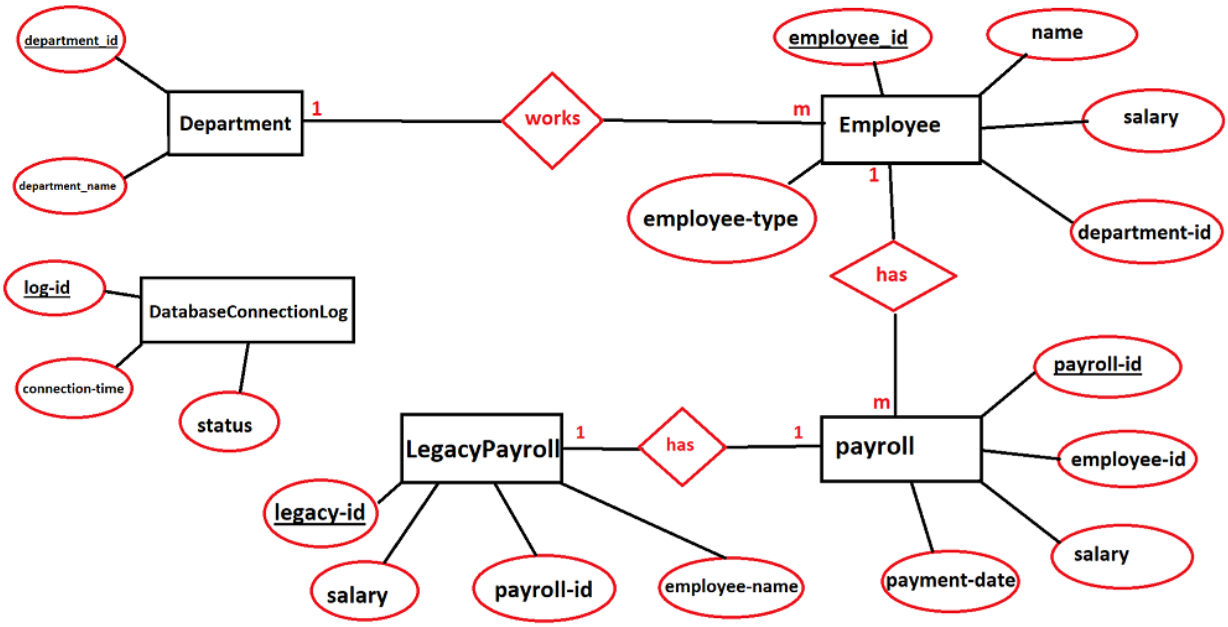
Conclusion

The **Employee Management System** integrates a **Microsoft SQL server** database with a Java application to manage employees, departments, and payroll records efficiently. Using well-known design patterns like **Singleton**, **Factory**, **Builder**, and **Proxy**, the system ensures clean architecture, flexibility, and easy maintenance.

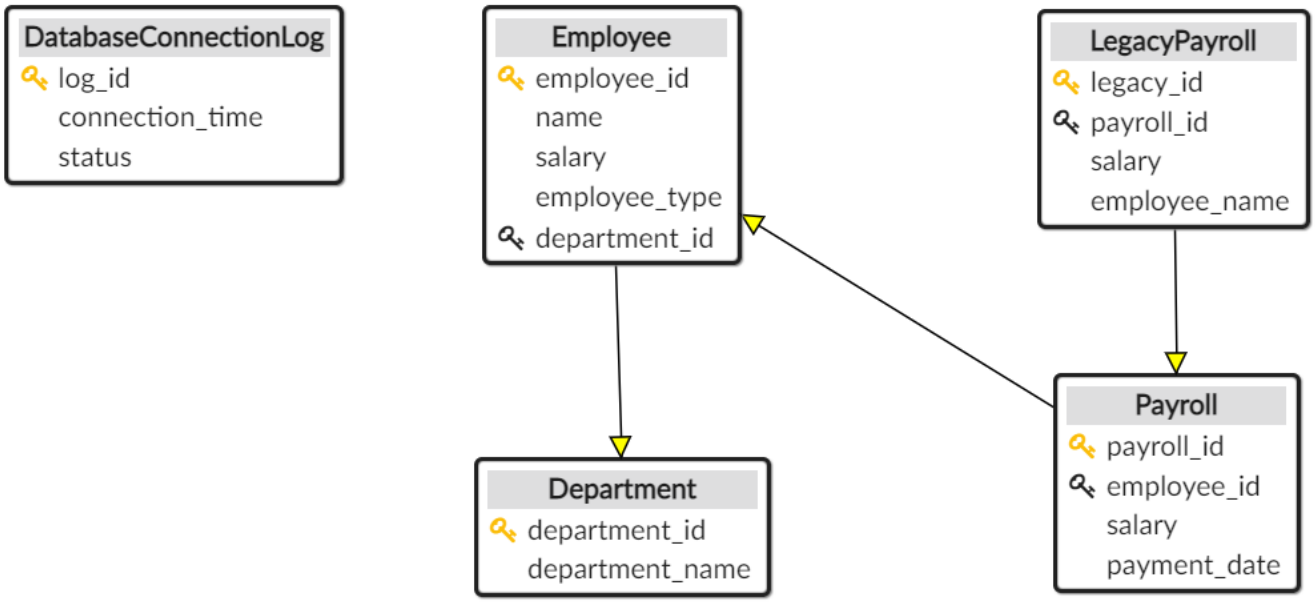
This project enables real-time data management, including employee creation, department assignments, and payroll processing, while also supporting legacy systems through an adapter. The secure and centralized database connection ensures smooth operation and reliable performance.

Overall, the system provides a solid foundation for managing company data and can be easily enhanced to include new features such as reporting tools, authentication, and automated payroll processing. It is scalable, user-friendly, and designed for real-world enterprise use.

ERD


department_id
Department
department_name
1
works
m
employee_id
name
Employee
salary
employee-type
1
department-id
has
log-id
DatabaseConnectionLog
connection-time
status
m
payroll-id
LegacyPayroll
1
has
1
payroll
employee-id
legacy-id
salary
payroll-id
employee-name
payment-date
salary

Schema



Thank you !