Real-Time Employee Leave Management Application

Project Overview

The objective of this mini-project is to design and develop a full-stack web application using Java technologies (Servlets, JSP with Expression Language (EL), and JSTL) integrated with a MySQL database. The application will manage employee details, leave requests, attendance, payroll, and system logs for an organization. This CRUD (Create, Read, Update, Delete) application will allow users with different roles (Admin, HR, Manager, Employee) to perform operations based on their access levels. The system aims to streamline employee leave management in real-time while ensuring data integrity and security.

Project Goals

1. Implement a robust backend using Java Servlets to handle business logic and database interactions.
2. Design a dynamic frontend using JSP with EL and JSTL for efficient rendering and minimal scripting.
3. Create a relational database schema in MySQL to store and manage employee-related data.
4. Ensure role-based access control (RBAC) for secure and appropriate user interactions.
5. Provide a user-friendly interface to perform CRUD operations on employee records, leave requests, and related entities.

Technology Stack

- **Frontend**: JSP (JavaServer Pages), Expression Language (EL), JSTL (JavaServer Pages Standard Tag Library), HTML, CSS (optional: Bootstrap for styling)
- **Backend**: Java Servlets
- **Database**: MySQL
- **Server**: Apache Tomcat (or any compatible servlet container)
- **IDE**: Eclipse, IntelliJ IDEA, or NetBeans (recommended)
- **Build Tool**: Maven (optional, for dependency management)

Database Schema

The MySQL database schema is provided below (as per your input). Students must implement it and ensure proper relationships and constraints are maintained.

1. **Database Creation**

```
CREATE DATABASE EmployeeLeaveManagement;
USE EmployeeLeaveManagement;
```

2. **Employee Table**

   Stores core employee information with leave balances computed automatically.

```
CREATE TABLE Employee (
    employee_id     INT PRIMARY KEY AUTO_INCREMENT,
    first_name      VARCHAR(50) NOT NULL,
    last_name       VARCHAR(50) NOT NULL,
    email           VARCHAR(100) UNIQUE NOT NULL,
    phone_number    VARCHAR(15) UNIQUE NOT NULL,
    dob             DATE NOT NULL,
    gender          ENUM('Male', 'Female', 'Other') NOT NULL,
    job_title       VARCHAR(50) NOT NULL,
    department      VARCHAR(50) NOT NULL,
    employee_type   ENUM('Full-time', 'Part-time', 'Contract', 'Intern') NOT NULL,
    date_of_joining DATE NOT NULL,
    manager_id      INT DEFAULT NULL,
    total_leave     INT DEFAULT 30,
    leave_taken     INT DEFAULT 0,
    leave_balance   INT GENERATED ALWAYS AS (total_leave - leave_taken) STORED,
    employee_status ENUM('Active', 'On Leave', 'Resigned', 'Terminated') DEFAULT 'Active',
    username        VARCHAR(50) UNIQUE NOT NULL,
    password_hash   VARCHAR(255) NOT NULL,
    role            ENUM('Admin', 'HR', 'Manager', 'Employee') DEFAULT 'Employee',
    last_login      TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (manager_id) REFERENCES Employee(employee_id) ON DELETE SET NULL
);
```


Archer Infotech Pune

3. **Leave Requests Table**

   Tracks leave applications and their statuses.

```
CREATE TABLE LeaveRequests (
  leave_id      INT PRIMARY KEY AUTO_INCREMENT,
  employee_id    INT NOT NULL,
  leave_type     ENUM('Sick Leave', 'Casual Leave', 'Annual Leave', 'Maternity Leave', 'Unpaid Leave') NOT NULL,
  start_date     DATE NOT NULL,
  end_date       DATE NOT NULL,
  leave_days     INT NOT NULL,
  leave_status   ENUM('Pending', 'Approved', 'Rejected', 'Cancelled') DEFAULT 'Pending',
  applied_on     TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  approved_by    INT DEFAULT NULL,
  approval_date  TIMESTAMP NULL DEFAULT NULL,
  FOREIGN KEY (employee_id) REFERENCES Employee(employee_id) ON DELETE CASCADE,
  FOREIGN KEY (approved_by) REFERENCES Employee(employee_id) ON DELETE SET NULL
);
```

4. **Attendance Table**

   Records daily attendance for employees.

```
CREATE TABLE Attendance (
  attendance_id  INT PRIMARY KEY AUTO_INCREMENT,
  employee_id    INT NOT NULL,
  attendance_date DATE NOT NULL,
  check_in_time  TIME NOT NULL,
  check_out_time  TIME NULL,
  status        ENUM('Present', 'Absent', 'On Leave', 'Half-day') NOT NULL DEFAULT 'Present',
  FOREIGN KEY (employee_id) REFERENCES Employee(employee_id) ON DELETE CASCADE
);
```

5. **Payroll Table**

   Manages salary and payment details.

```
CREATE TABLE Payroll (
  payroll_id     INT PRIMARY KEY AUTO_INCREMENT,
  employee_id    INT NOT NULL,
  salary         DECIMAL(10,2) NOT NULL,
  bank_account   VARCHAR(50) NOT NULL,
  tax_deductions  DECIMAL(10,2) DEFAULT 0,
  net_salary     DECIMAL(10,2) GENERATED ALWAYS AS (salary - tax_deductions) STORED,
  payment_date   DATE,
  FOREIGN KEY (employee_id) REFERENCES Employee(employee_id) ON DELETE CASCADE
);
```

System Logs Table

Logs user actions for auditing purposes.

```
CREATE TABLE SystemLogs (
  log_id         INT PRIMARY KEY AUTO_INCREMENT,
  employee_id    INT NOT NULL,
  action         VARCHAR(255) NOT NULL,
  timestamp      TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (employee_id) REFERENCES Employee(employee_id) ON DELETE CASCADE
);
```

Functional Requirements

The application must support the following features:

1. **Employee Management (CRUD Operations)**
   - **Create**: Add a new employee with all required details (e.g., personal info, job details, leave entitlement).
   - **Read**: Display a list of employees and allow viewing individual employee details.
   - **Update**: Modify employee information (e.g., update job title, leave balance, or status).

Archer Infotech Pune

- **Delete**: Remove an employee record (only for Admin/HR roles).

2. **Leave Management**
   - Employees can submit leave requests with start/end dates and leave type.
   - Managers/HR can approve or reject leave requests, updating the employee's leave balance.
   - View leave request history for each employee.
3. **Role-Based Access Control**
   - **Admin**: Full access (CRUD on all tables, manage users).
   - **HR**: Manage employee records, payroll, and approve leaves.
   - **Manager**: View team details, approve/reject leave requests.
   - **Employee**: View own details, apply for leaves, check leave balance.
4. **Authentication**
   - Login system using username and password (stored as hashed values in the database).
   - Redirect users to role-specific dashboards after login.
5. **Dashboard**
   - Display key metrics (e.g., total employees, pending leave requests, active employees).
   - Provide quick links to CRUD operations based on user role.

Non-Functional Requirements

- **Usability**: Simple and intuitive UI with minimal scripting in JSP (rely on EL and JSTL).
- **Performance**: Efficient database queries and servlet handling for real-time updates.
- **Security**: Passwords must be hashed (e.g., using BCrypt or SHA-256). Validate inputs to prevent SQL injection and XSS attacks.
- **Scalability**: Design the application to handle at least 100 employee records efficiently.

Detailed Expectations

1. **Frontend (JSP with EL and JSTL)**
   - Use JSP pages for all UI components (e.g., employee list, leave request form).
   - Leverage EL (${variable}) to display dynamic data from servlets.
   - Use JSTL tags (<c:forEach>, <c:if>, etc.) for looping and conditional rendering.
   - Avoid inline Java code in JSP files; keep logic in servlets.
2. **Backend (Servlets)**
   - Create servlets for each major operation (e.g., EmployeeServlet, LeaveServlet).
   - Use JDBC to connect to MySQL and execute CRUD operations.
   - Handle HTTP requests (GET/POST) and forward data to JSP pages.
3. **Database Integration**
   - Implement the provided schema in MySQL.
   - Write prepared statements in JDBC to ensure security and efficiency.
   - Maintain referential integrity (e.g., cascading deletes for leave records when an employee is removed).
4. **Deliverables**
   - Complete source code (Servlets, JSP files, SQL scripts).
   - A deployed application on Apache Tomcat (or equivalent).
   - A brief report (1-2 pages) explaining:
     - Project structure (folder hierarchy).
     - Key features implemented.
     - Challenges faced and solutions applied.
   - A demo presentation (5-10 minutes) showcasing the application's functionality.
5. **Bonus Features (Optional)**
   - Add pagination for employee and leave request lists.
   - Implement search/filter functionality (e.g., by department or status).
   - Generate PDF reports for payroll or attendance using a library like iText.

Evaluation Criteria

- **Functionality** (40%): All required features work as expected.
- **Code Quality** (30%): Clean, modular code with proper use of EL/JSTL and minimal scripting.
- **Database Design** (15%): Correct implementation of schema and efficient queries.
- **Presentation & Documentation** (15%): Clear explanation of work and demo.

Sample Workflow

1. **Login**: User enters username/password → Servlet validates → Redirects to dashboard.
2. **Add Employee**: HR fills a form → Servlet inserts into Employee table → Updates UI.
3. **Apply for Leave**: Employee submits request → Servlet adds to LeaveRequests → Manager approves → Updates leave_taken and leave_balance.

Archer Infotech Pune