
💼 **Project Title:**

HRMS Portal – A Product of Amazon Pvt. Ltd.

🚀 **1. Introduction**

The **Human Resource Management System (HRMS)** Portal is a full-stack web application designed to automate employee management, attendance tracking, payroll processing, and departmental coordination within an organization.

The goal is to **replace manual HR operations** with a centralized online system that allows HR administrators to manage employees, calculate payroll, record attendance, and generate reports — all in real-time.

🧩 **2. Technologies Used**

Layer	Technologies
Frontend	HTML, CSS, JavaScript, JSP
Backend	Java Servlets, JDBC
Database	MySQL
Build Tool	Maven
Server	Apache Tomcat
Version Control	Git
Testing	Postman (for CRUD API testing)
Optional Integration	DynamoDB for leave requests
Batch Processing	Java Queue (for payroll batch runs)

🧱 **3. System Architecture (MVC Pattern)**

The project follows the **Model-View-Controller (MVC)** architecture.

1. **Model (Data Layer)** – Java classes like Employee, Department, Payroll, and Attendance represent the data models.
   These connect to the database through the DAO (Data Access Object) layer.
2. **View (Presentation Layer)** – JSP pages (like addEmployee.jsp, listEmployee.jsp, attendance.jsp, payroll.jsp) display data to users.
3. **Controller (Logic Layer)** – Servlets like EmployeeServlet, PayrollServlet, AttendanceServlet handle user requests, process logic, and call DAO methods.

Example Flow:

User → JSP (form) → Servlet → DAO → MySQL → DAO → Servlet → JSP (response)

👨‍💼 4. Major Modules

A. Employee Management (CRUD)

This is the core feature for HR staff.

Features:

- Add new employee
- View employee list
- Update employee details
- Delete employee records

Example API Operations:

- POST /employees → Add Employee
- GET /employees → View All Employees
- PUT /employees/{id} → Update Employee
- DELETE /employees/{id} → Remove Employee

Code Example (Insert Employee):

```
public void addEmployee(Employee emp) throws SQLException {
    String query = "INSERT INTO employees (name, email, department_id, salary, join_date) VALUES (?, ?, ?, ?, ?)";
    PreparedStatement ps = conn.prepareStatement(query);
    ps.setString(1, emp.getName());
```

```
    ps.setString(2, emp.getEmail());
    ps.setInt(3, emp.getDepartmentId());
    ps.setDouble(4, emp.getSalary());
    ps.setDate(5, new java.sql.Date(emp.getJoinDate().getTime()));
    ps.executeUpdate();
}
```

B. Department Management

Departments are managed in a master table.

Schema:

```
CREATE TABLE departments (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    allowance_rate DECIMAL(5,4),
    deduction_rate DECIMAL(5,4)
);
```

Example:
If the IT department has a 10% allowance and 5% deduction, those values are applied during payroll calculations.

C. Attendance Management

Features:

- Record daily check-in and check-out times
- Calculate working hours
- Prevent duplicate attendance entries per day

Schema:

```
CREATE TABLE attendance (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    employee_id BIGINT,
```

```
    date DATE,
    check_in TIME,
    check_out TIME,
    hours_worked DECIMAL(4,2)
);
```

Example Logic:

```
hoursWorked = Duration.between(checkIn, checkOut).toHours();
```

Use Case:
When an employee checks in, an entry is created. When they check out, the system calculates hours worked and updates the record.

D. Payroll Management

Objective: Automatically compute total monthly salary for each employee.

Formula:

```
Total = Basic Salary + Allowances + Overtime - Deductions
```

Example:
If an employee's base salary is ₹50,000, allowance 10%, and deduction 5%,
then →
Total = 50,000 + 5,000 - 2,500 = ₹52,500

SQL Query for Payroll Computation:

```
SELECT e.id, e.name, e.salary,
    (e.salary * d.allowance_rate) AS allowances,
    (e.salary * d.deduction_rate) AS deductions,
    ((e.salary + (e.salary * d.allowance_rate)) - (e.salary * d.deduction_rate)) AS total
FROM employees e
JOIN departments d ON e.department_id = d.id;
```

E. Leave Management (Day 6 – DynamoDB Integration)

To demonstrate NoSQL integration, leave requests are stored in **AWS DynamoDB** instead of MySQL.

Example Fields:

LeaveID, EmployeeID, FromDate, ToDate, Reason, Status

Using AWS SDK for Java, the app can insert or retrieve leave requests dynamically.

F. Payroll Batch Processing (Day 7)

The payroll generation process for all employees at the end of the month uses a **Java Queue**.

Logic:

- Each employee record is added to a queue.
- A payroll worker dequeues each record and computes salary sequentially.

Example:

```
Queue<Employee> payrollQueue = new LinkedList<>();
for(Employee e : employeeList){
   payrollQueue.add(e);
}
while(!payrollQueue.isEmpty()){
   Employee emp = payrollQueue.poll();
   generatePayroll(emp);
}
```

G. Jenkins Automation (Day 9)

A **Jenkins pipeline job** automates:

- Code build using Maven
- Database migration scripts
- Deployment to Tomcat
- Automated payroll batch run

This ensures CI/CD (Continuous Integration & Delivery).

🗄️ 5. Database Design (Updated Schema)

Tables:

- departments
- employees
- payroll
- attendance

Relationships:

- **One Department → Many Employees**
- **One Employee → Many Attendance Records & Payroll Entries**

🧱 6. Example Queries

Insert Employee:

INSERT INTO employees (name, email, department_id, salary, join_date)

VALUES ('Ravi Kumar', 'ravi@example.com', 1, 60000, '2024-01-15');

Fetch Payroll Details:

SELECT * FROM payroll WHERE month = '2025-10';

🧠 7. Data Flow Summary

Step 1: HR logs into the portal.
Step 2: Views or adds employee details.
Step 3: Attendance automatically recorded daily.
Step 4: Payroll generation job processes salaries.
Step 5: Reports are generated and exported.

🔐 8. Security Measures

- PreparedStatements prevent SQL Injection.
- Validation on JSP forms to avoid invalid data.
- Role-based access control (Admin vs Employee).
- Passwords hashed before storage (if login is added).

🧪 9. Testing

- **Unit Tests:** DAO and Servlet methods tested independently.
- **Integration Tests:** Tested using Postman for REST APIs.
- **End-to-End Tests:** HRMS flows tested manually on localhost.

☁️ 10. Deployment

- Application built with Maven (mvn clean install)
- WAR file deployed on **Apache Tomcat**
- Database hosted in **MySQL Workbench**
- Optional AWS integration for DynamoDB leave management

📊 11. Future Enhancements

- Role-based dashboards (HR, Admin, Employee)
- Email notifications for salary slips
- AI-based attendance prediction
- Cloud-native microservice conversion

🏁 12. Conclusion

The **HRMS Portal** provides a complete, modular, and scalable solution to automate HR operations.
It demonstrates your ability to work on **end-to-end full-stack Java development**, including **database design, servlet logic, API handling, and deployment automation**.
