UE22CS352B-Object Oriented Analysis & Design

Mini Project Report

"Employee management system"

Submitted by:

PRAKASH	**PES1UG22CS425**
SRIHARI	**PES1UG22CS610**

Semester 6th

Under the guidance of

BHARGAVI MOKASHI

Assistant Professor

January - May2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100ft Ring Road, Bengaluru–560085, Karnataka, India.

Problem Statement

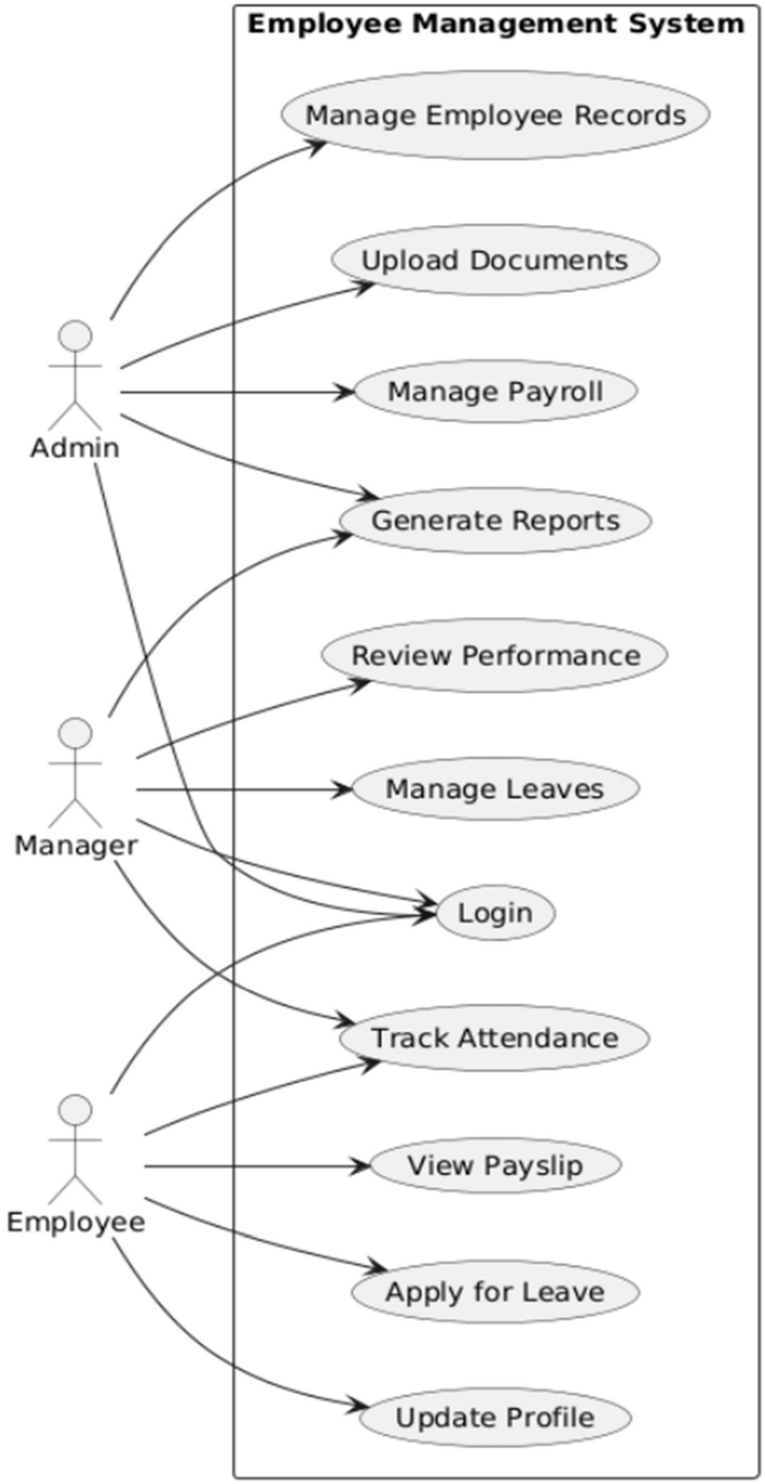
In many organizations, managing employee data, tracking attendance, handling leave requests, and ensuring effective communication between departments remains a time-consuming and error-prone process when done manually or using outdated systems. This often leads to inefficiencies in HR operations, poor employee engagement, and difficulty in accessing accurate information for decision-making. The lack of a centralized, automated system can result in data redundancy, delayed processes, and reduced productivity. Therefore, there is a clear need for a robust Employee Management System that streamlines core HR functions, enhances data accuracy, and provides managers with real-time insights to support better workforce planning and performance monitoring.

Key Features

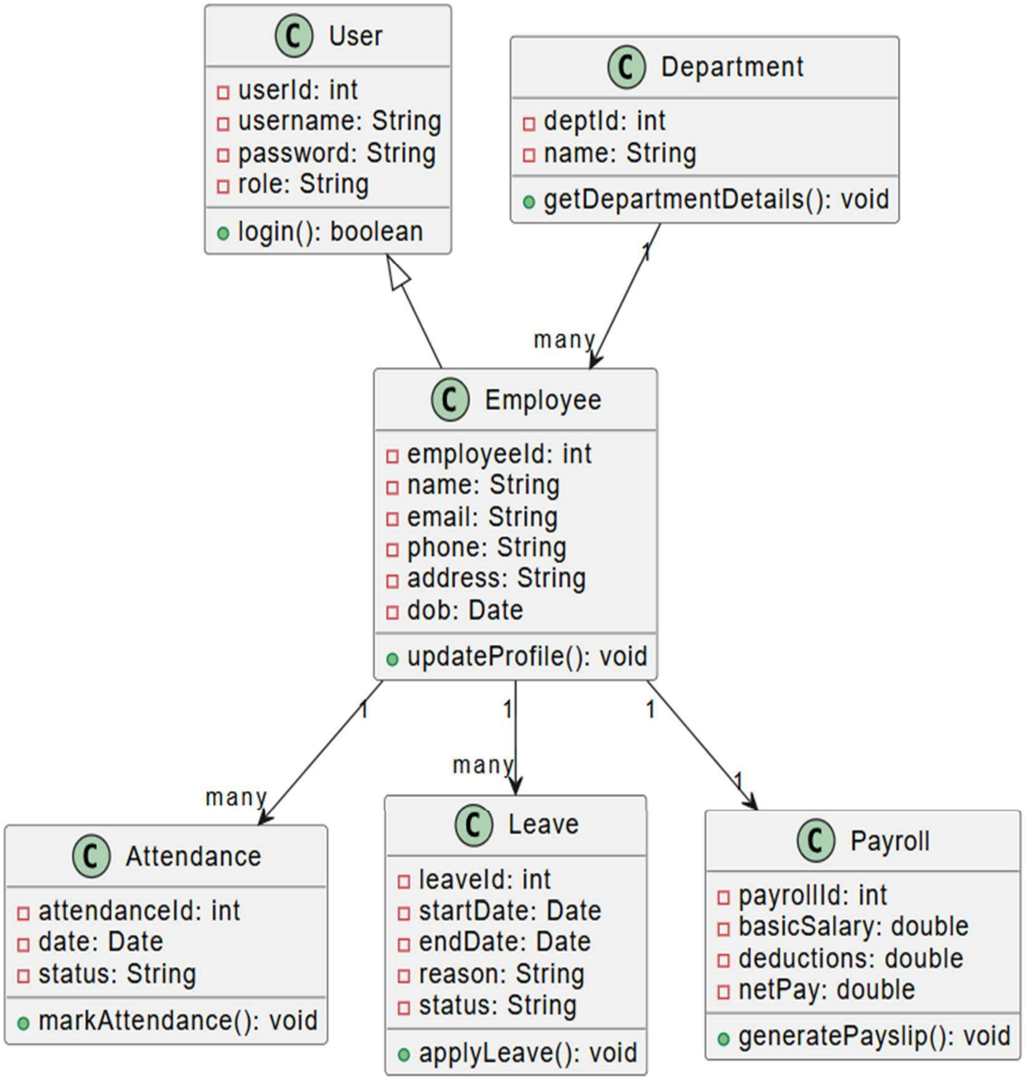
- **Employee Profile Management :** Stores and manages complete employee records including personal details, job role, contact info, and employment history.
- **Attendance Tracking :** Tracks employee working hours and attendance using manual input or automated systems like biometric scanners or time clocks.
- **Leave Management :** Allows employees to apply for leaves and enables HR or managers to review, approve, and maintain leave balances efficiently.
- **Payroll Integration:** Calculates and processes employee salaries automatically based on attendance, leave data, tax rules, and benefits.
- **Performance Management:** Facilitates goal setting, performance evaluations, and tracking of employee achievements for appraisals and development.
- **Department & Role Management:** Organizes the workforce into structured departments and assigns roles to define access levels and responsibilities.
- **Document Management:** Provides a secure repository for storing employee-related documents such as resumes, contracts, and identification proofs.
- **User Authentication and Access Control:** Ensures secure access to the system through login credentials and restricts access based on user roles and permissions.

Models

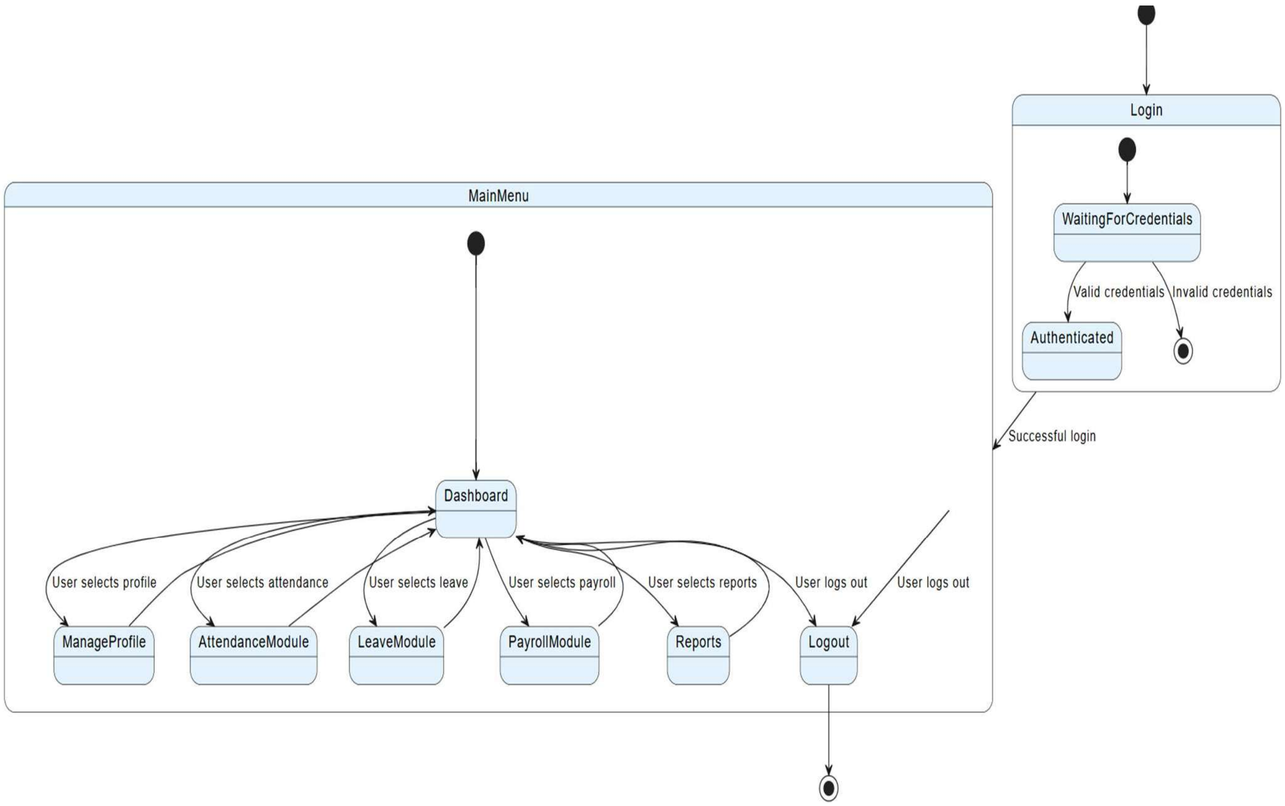
- **Use Case Diagram**



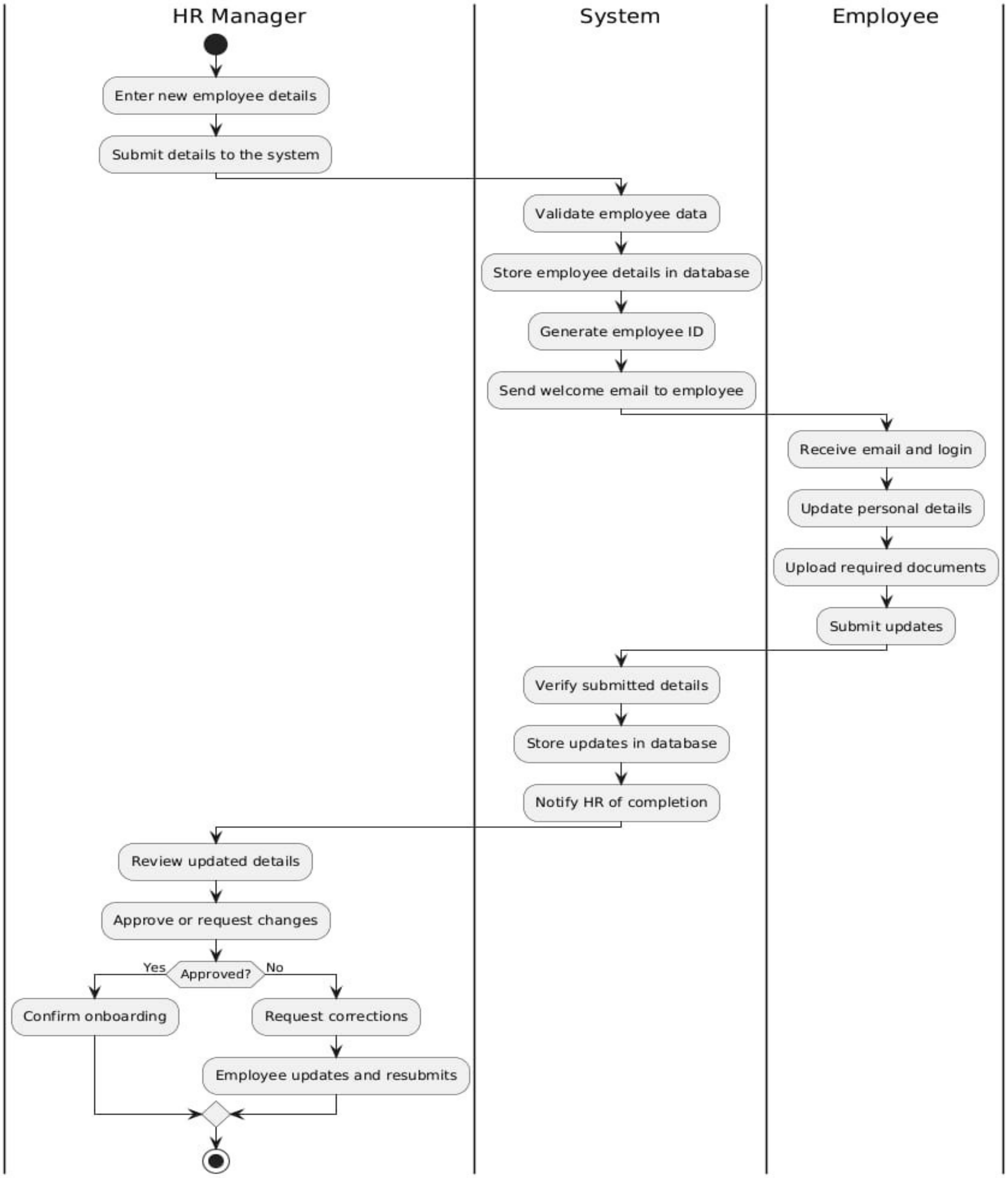
- **Class diagram**

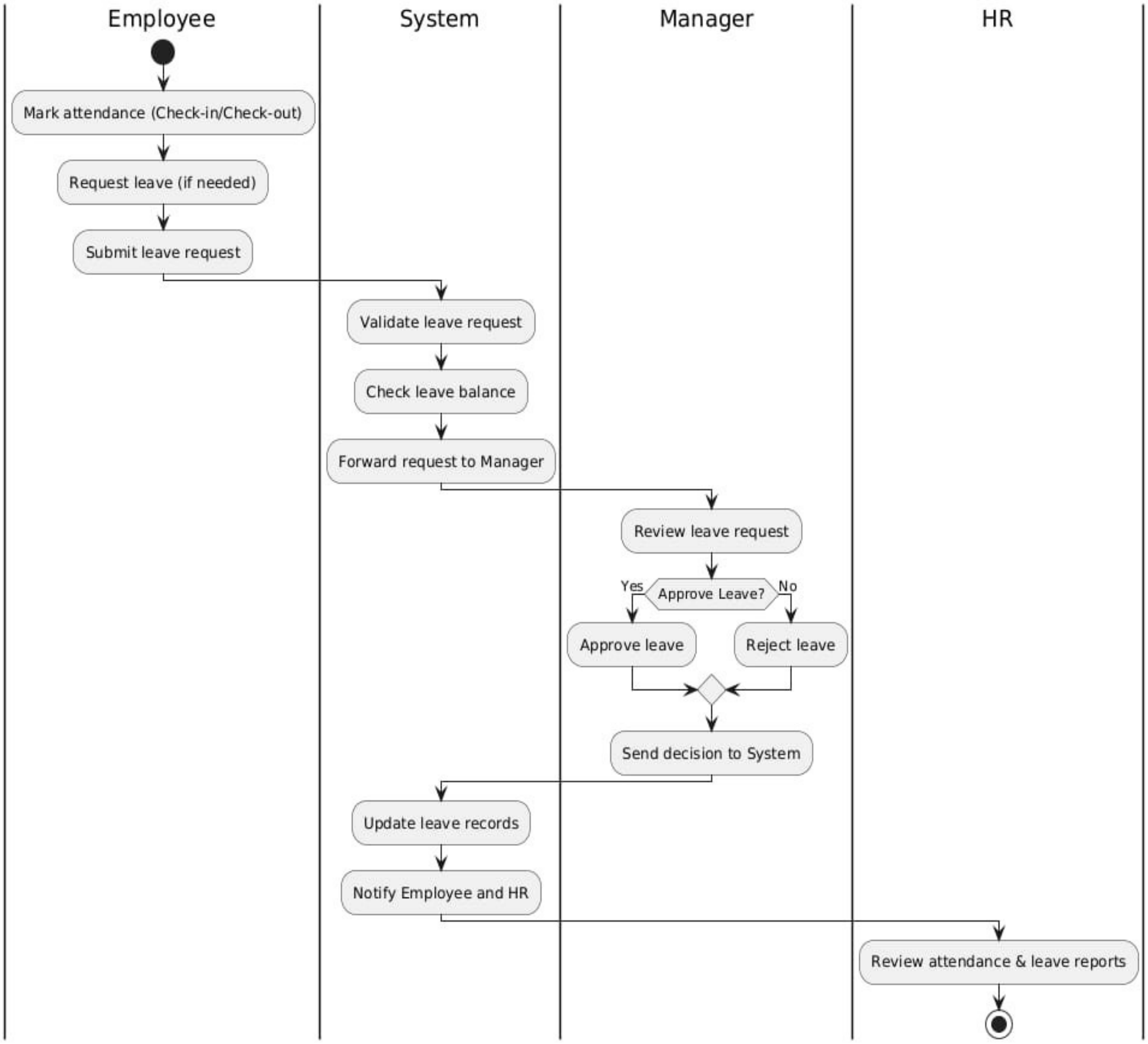


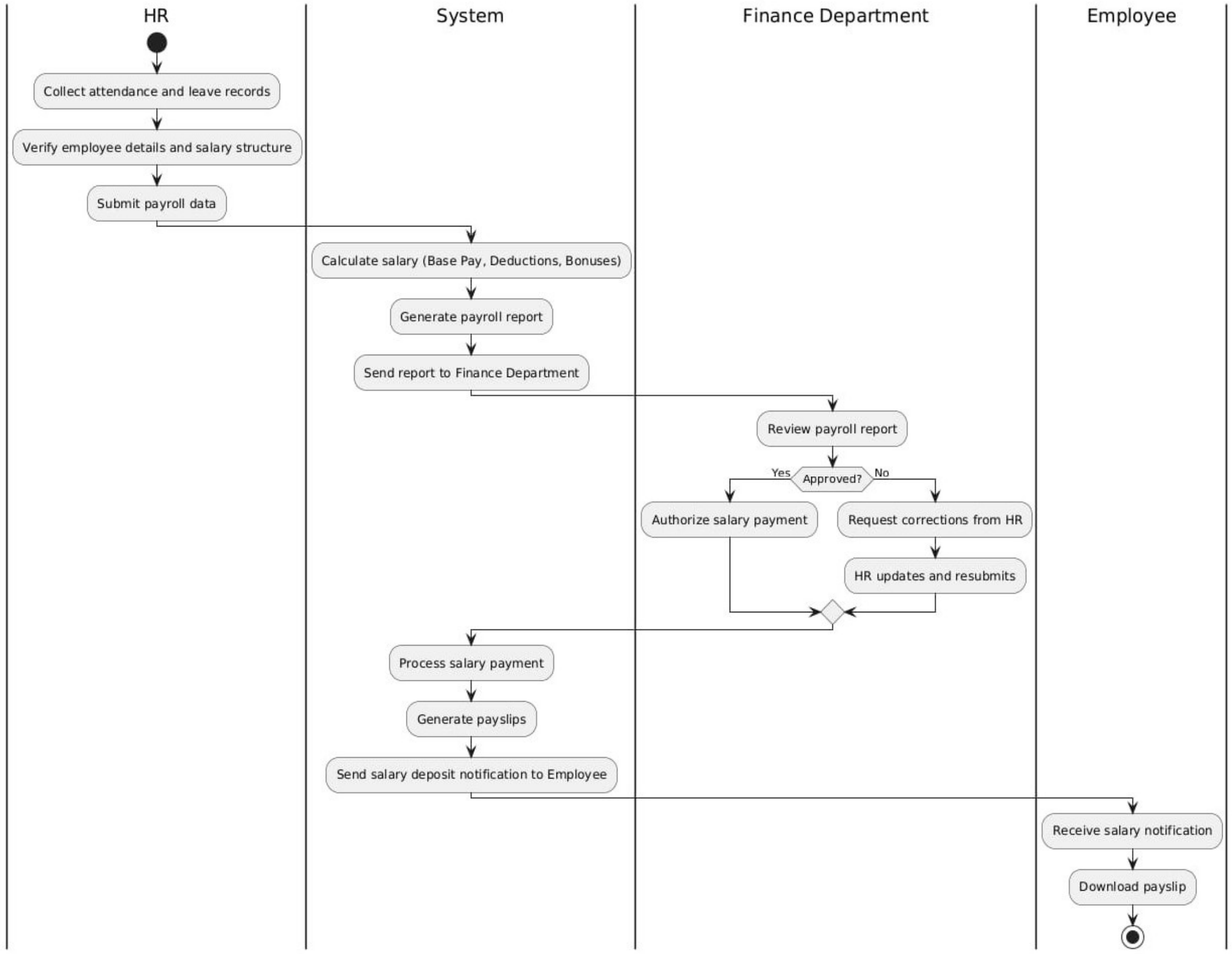
- **State diagram**

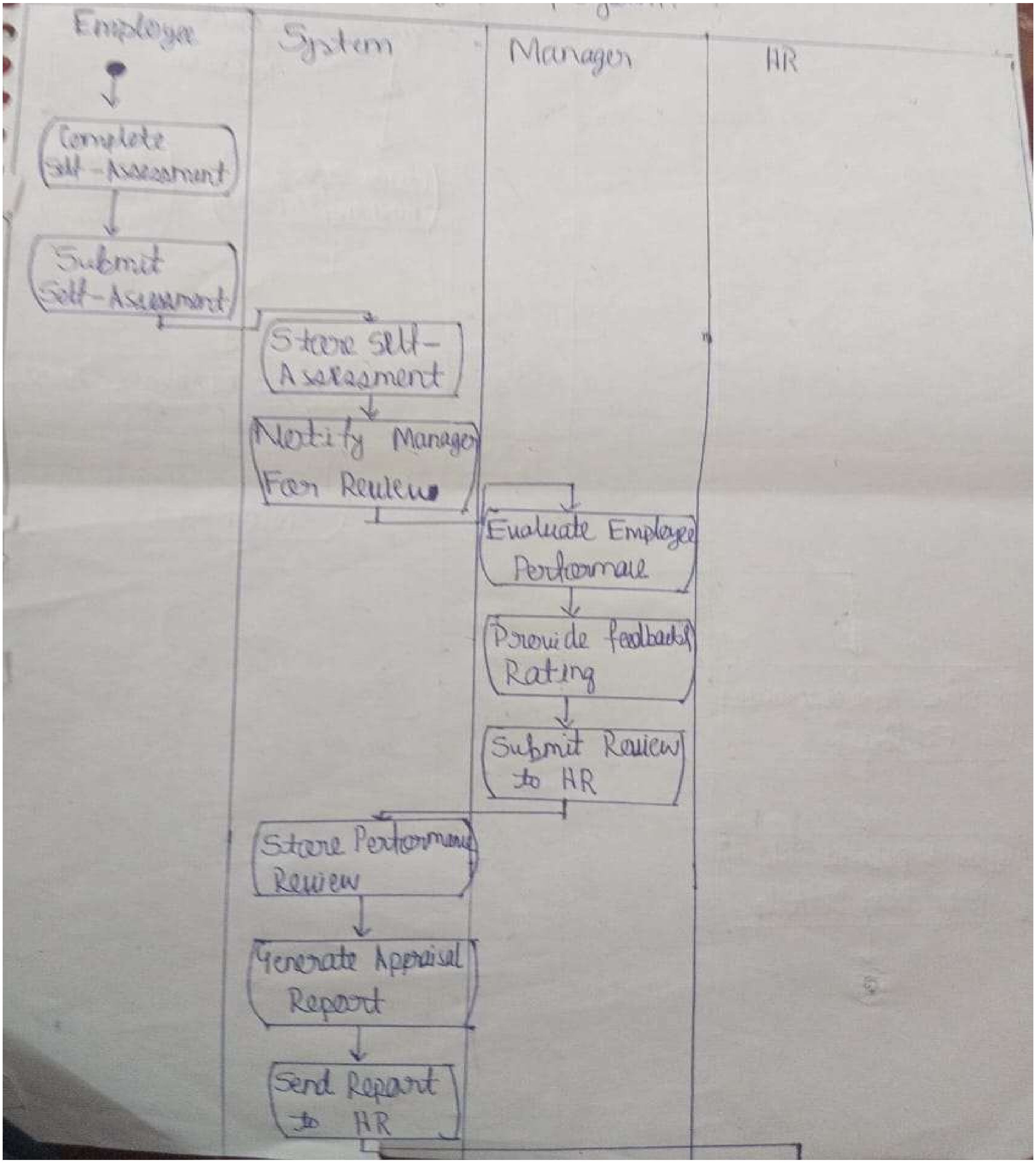


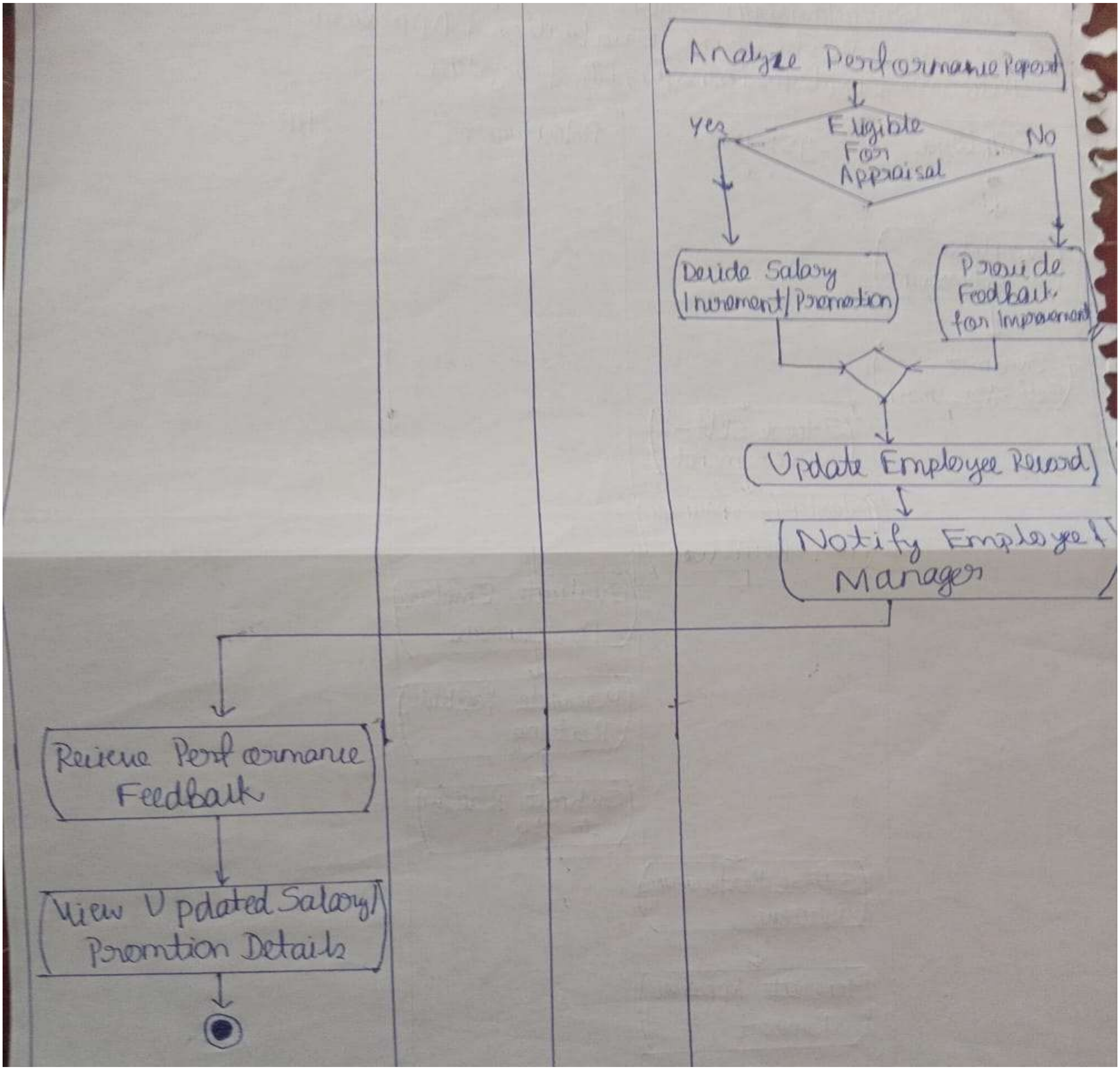
- **Swim lane diagram**




Employee
System
Manager
HR
Mark attendance (Check-in/Check-out)
Request leave (if needed)
Submit leave request
Validate leave request
Check leave balance
Forward request to Manager
Review leave request
Approve Leave?
Yes
No
Approve leave
Reject leave
Send decision to System
Update leave records
Notify Employee and HR
Review attendance & leave reports


HR
System
Finance Department
Employee
Collect attendance and leave records
Verify employee details and salary structure
Submit payroll data
Calculate salary (Base Pay, Deductions, Bonuses)
Generate payroll report
Send report to Finance Department
Review payroll report
Approved?
Yes
No
Authorize salary payment
Request corrections from HR
HR updates and resubmits
Process salary payment
Generate payslips
Send salary deposit notification to Employee
Receive salary notification
Download payslip


Employee
System
Manager
HR
Complete Self-Assessment
Submit Self-Assessment
Store Self-Assessment
Notify Manager For Review
Evaluate Employee Performance
Provide feedback Rating
Submit Review to HR
Store Performance Review
Generate Appraisal Report
Send Report to HR


Analyze Performance Report
Yes
Eligible For Appraisal
No
Decide Salary Increment/Promotion
Provide Feedback for Improvement
Update Employee Record
Notify Employee & Manager
Recieve Performance Feedback
View Updated Salary/ Promtion Details

Architecture Patterns

The architecture of the Employee Management System is primarily built on the Model-View-Controller (MVC) pattern, which is widely adopted in Spring Boot applications. This pattern divides the system into three interconnected layers that separate responsibilities effectively. The Model layer represents the business entities such as Employee, Department, Payroll, and Leave, along with the underlying business logic. The View layer is responsible for rendering the user interface, typically using Thymeleaf or HTML templates in a Spring Boot context. The Controller layer acts as a mediator that processes incoming HTTP requests, invokes business logic via services, and returns appropriate views or responses. This separation improves modularity and makes the application easier to test, maintain, and extend.

Beneath the MVC structure, the system adheres to a layered (N-tier) architecture, which organizes the codebase into logical layers such as the Controller layer, Service layer, Repository layer, and Data (Persistence) layer. The Controller handles user input and API endpoints, delegating business logic to the Service layer, which contains reusable methods for handling core operations like leave approval, salary generation, and attendance tracking. The Repository layer, implemented using Spring Data JPA, abstracts all database interactions, enabling developers to use simple method declarations to perform CRUD operations without writing SQL queries. This layered approach promotes separation of concerns, allowing developers to focus on specific aspects of the system independently.

One of the key architectural patterns used in the system is the Repository pattern. This pattern abstracts the complexity of data access, allowing data retrieval and persistence logic to be encapsulated within repository interfaces. By doing this, the code becomes cleaner and more maintainable. Additionally, the Service Layer pattern is employed to centralize business logic, which helps keep controllers light and focused solely on handling HTTP requests. This not only improves code reuse but also makes it easier to apply cross-cutting concerns like logging and transaction management. These service classes interact with repositories to fetch or update data, and with controllers to serve processed results to the user interface.

The system also heavily utilizes Dependency Injection (DI), a core feature of the Spring Framework, which follows the Inversion of Control (IoC) principle. Instead of creating instances manually, Spring automatically injects dependencies into beans at runtime, promoting loose coupling and making the application more modular and testable. Another important pattern used is the Front Controller pattern, implemented via Spring’s DispatcherServlet, which acts as a centralized entry point for all HTTP requests and delegates them to the appropriate controller. Lastly, most Spring-managed components (such as services and controllers) are instantiated as singletons by default, reflecting the Singleton pattern to ensure a single shared instance throughout the application lifecycle. Together, these architectural patterns form a robust, maintainable, and scalable backbone for the Employee Management System.

Design Principles

Separation of Concerns (SoC)

This principle encourages the division of the system into distinct sections, each responsible for a specific functionality. In the context of the system, it is achieved through the use of the MVC pattern, where the Model, View, and Controller are kept separate. This separation makes it easier to modify or extend one part of the application without affecting others.

Don't Repeat Yourself (DRY)

The DRY principle is followed by centralizing business logic in the service layer, where it can be reused by multiple controllers. Additionally, Spring Data JPA repositories abstract database operations, preventing the need to write repetitive SQL queries. This reduces redundancy, ensuring that logic is written only once and reused efficiently throughout the system.

Single Responsibility Principle (SRP)

Each class in the system is designed to have a single responsibility, making the codebase more modular and easier to maintain. For example, the `EmployeeService` is responsible solely for employee-related business logic, while the `PayrollService` focuses exclusively on payroll calculations. This reduces complexity and ensures that each class can be modified without affecting others.

Open/Closed Principle (OCP)

The system is designed to be open for extension but closed for modification. This is achieved by using interfaces and abstract classes, which allow new functionality to be added through subclassing or implementing new classes without altering existing code. For instance, new modules or features such as a performance review system can be integrated without modifying existing services or controllers.

Liskov Substitution Principle (LSP)

The LSP principle is followed by ensuring that objects of a subclass can be substituted for objects of a superclass without affecting the behavior of the system. This is particularly useful in the service and repository layers, where interfaces are used to define standard operations, allowing flexibility in changing or extending functionality without breaking the system.

Interface Segregation Principle (ISP)

The system adheres to this principle by defining small, specific interfaces for different modules, ensuring that clients only depend on the methods they actually use. For example, a `PayrollRepository` might be separate from an `AttendanceRepository`, ensuring that services only depend on relevant repository interfaces, reducing unnecessary coupling.

Dependency Inversion Principle (DIP)

By leveraging Spring's Dependency Injection (DI) framework, the system follows the DIP, where high-level modules (such as service classes) do not depend on low-level modules (like repository classes). Instead, both depend on abstractions (interfaces), which makes the system more flexible, maintainable, and decoupled.

Design Patterns

1. SingletonPattern

The Singleton pattern ensures that a class has only one instance and provides a global point of access to it. In a Spring Boot application, the Singleton pattern is commonly used for service classes and other Spring beans, where the Spring IoC (Inversion of Control) container ensures that only one instance of a class is created and managed throughout the application lifecycle. For instance, the `EmployeeService` or `PayrollService` could be a singleton, ensuring that a single, consistent instance handles all requests during the application runtime..

2. Factory Pattern

The Factory pattern is used to create objects without specifying the exact class of object that will be created. In the context of an Employee Management System, it can be used to instantiate various objects based on certain parameters or conditions. For instance, a Report Factory could generate different types of reports (e.g., payroll report, attendance report, performance review report) based on user input or other business logic, without exposing the specific implementation details to the client.

3. Repository Pattern

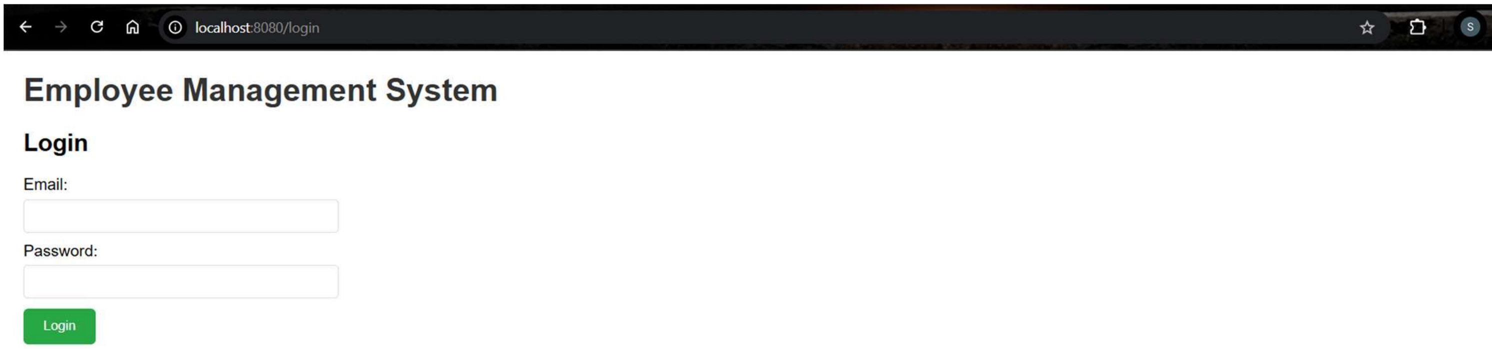
The Repository pattern is used to encapsulate the logic needed to access data sources. In Spring Boot, the repository layer can be implemented using Spring Data JPA, where repository interfaces extend `JpaRepository`or `CrudRepository` to automatically provide CRUD operations. This pattern abstracts database access, allowing the system to interact with data in a more object-oriented way, reducing the need to write raw SQL queries and making the code more maintainable and testable. Other than that some of the repositories were not only extended but some custom methods were also written related to finding an employee by Department etc. The reason we chose this is because it has more security as it prevents SQL injection, CSRF attacks better than the traditional prepared statement SQL queries.

4. Façade Pattern

The Facade pattern provides a simplified interface to a complex subsystem. In an Employee Management System, a Facade could be used to provide a simple interface for common functionalities such as employee management, leave management, and payroll processing. By using the facade pattern, clients can interact with the system using a single point of entry (e.g.,`PaymentFacade`) without needing to be aware of the underlying complexity or the interactions between various subsystems.

Screenshots

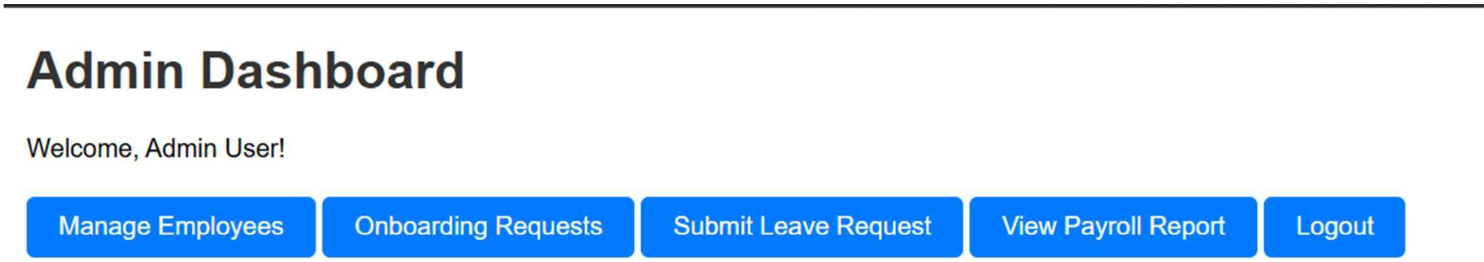
SS1



Employee Management System

Login

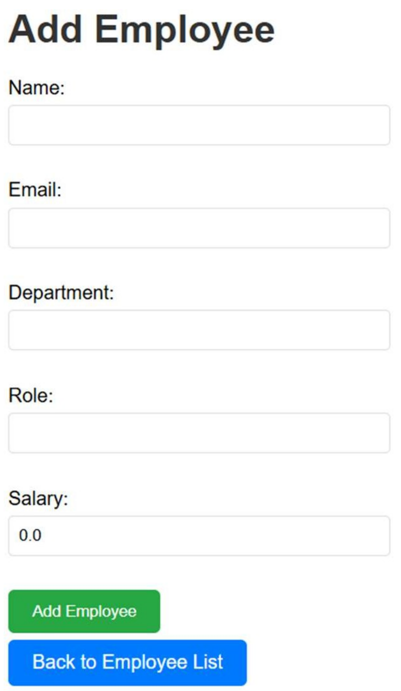
SS2



Admin Dashboard

Welcome, Admin User!

SS3



Add Employee

Name:

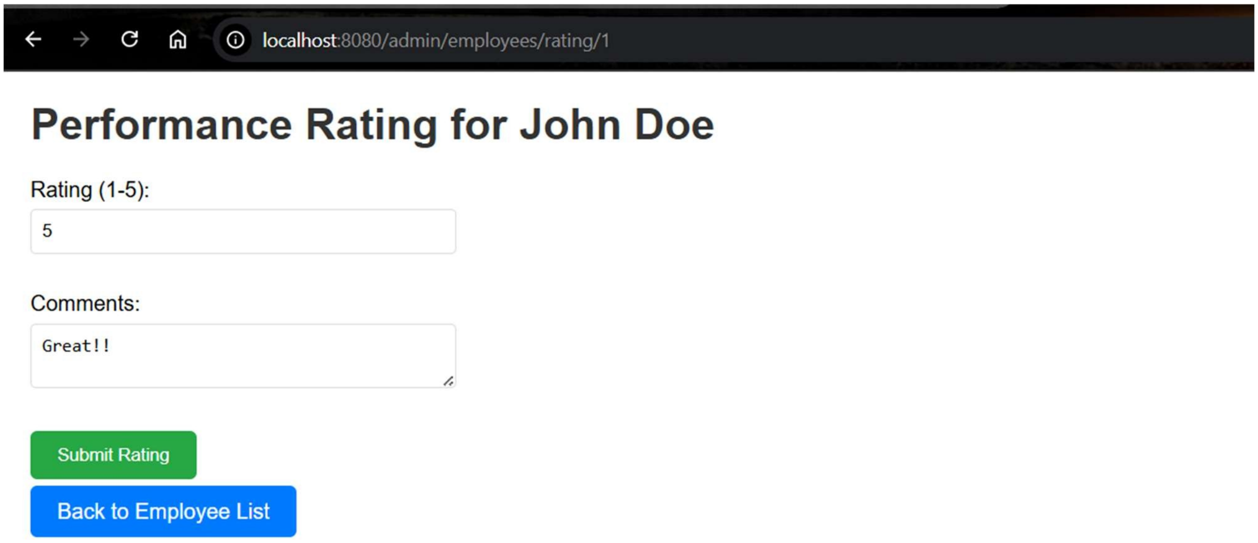
Email:

Department:

Role:

Salary: 0.0

SS4



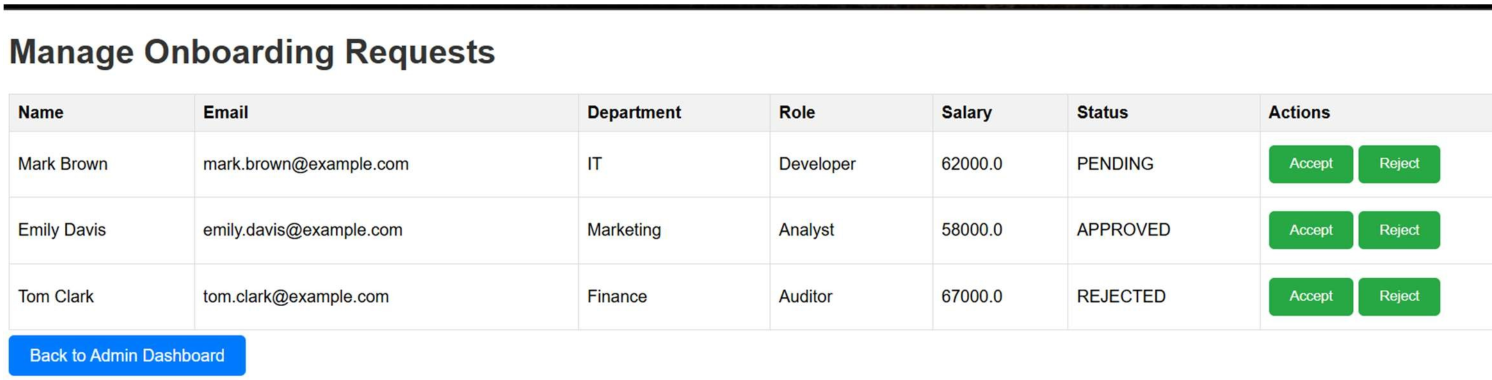
localhost:8080/admin/employees/rating/1

Performance Rating for John Doe

Rating (1-5): 5

Comments: Great!!

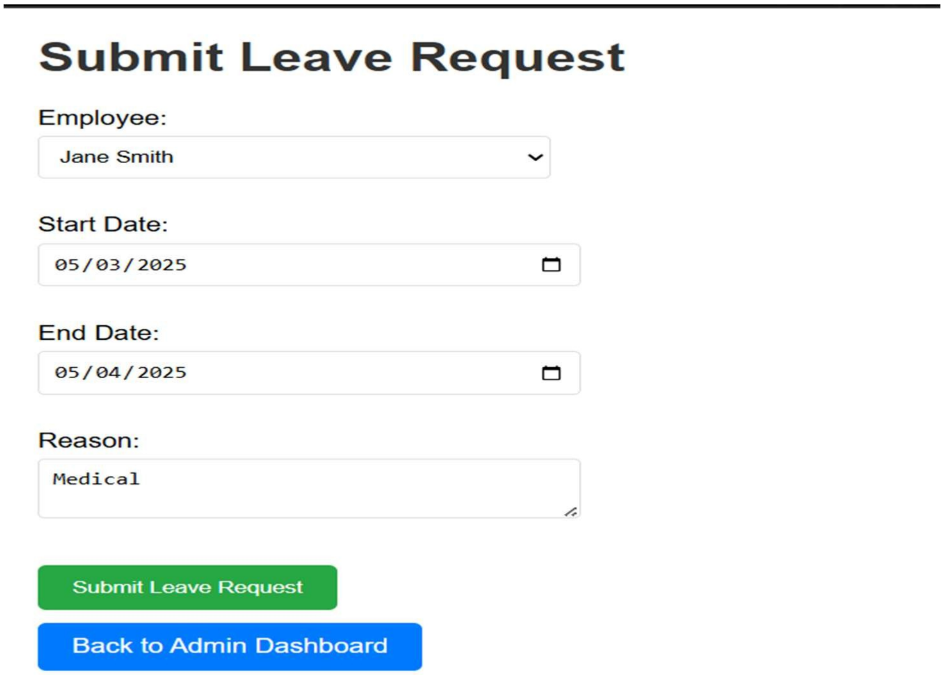
SS5



Manage Onboarding Requests

Name	Email	Department	Role	Salary	Status	Actions
Mark Brown	mark.brown@example.com	IT	Developer	62000.0	PENDING	Accept Reject
Emily Davis	emily.davis@example.com	Marketing	Analyst	58000.0	APPROVED	Accept Reject
Tom Clark	tom.clark@example.com	Finance	Auditor	67000.0	REJECTED	Accept Reject

SS6



Submit Leave Request

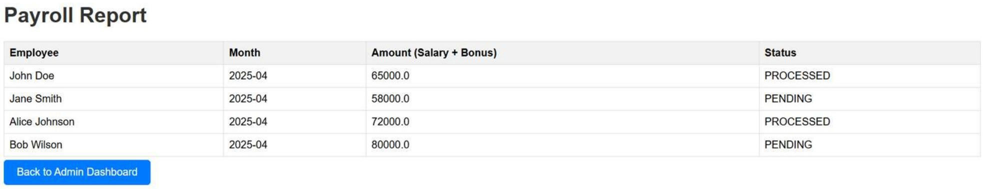
Employee:
Jane Smith

Start Date:
05/03/2025

End Date:
05/04/2025

Reason:
Medical

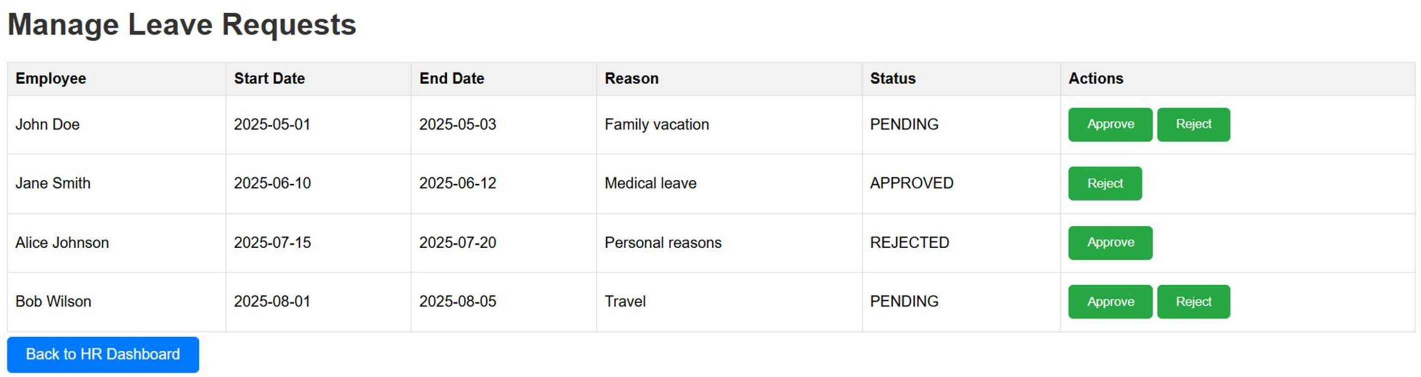
SS7



Payroll Report

Employee	Month	Amount (Salary + Bonus)	Status
John Doe	2025-04	65000.0	PROCESSED
Jane Smith	2025-04	58000.0	PENDING
Alice Johnson	2025-04	72000.0	PROCESSED
Bob Wilson	2025-04	80000.0	PENDING

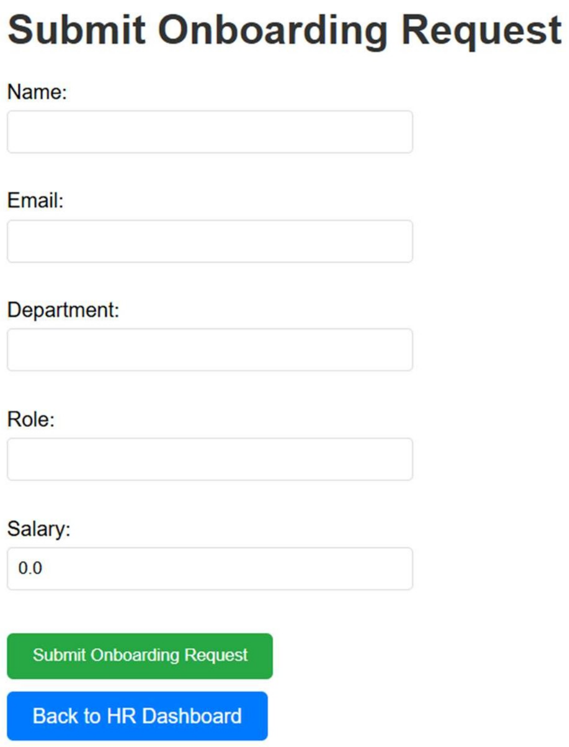
SS8



Manage Leave Requests

Employee	Start Date	End Date	Reason	Status	Actions
John Doe	2025-05-01	2025-05-03	Family vacation	PENDING	Approve Reject
Jane Smith	2025-06-10	2025-06-12	Medical leave	APPROVED	Reject
Alice Johnson	2025-07-15	2025-07-20	Personal reasons	REJECTED	Approve
Bob Wilson	2025-08-01	2025-08-05	Travel	PENDING	Approve Reject

SS9



Submit Onboarding Request

Name:

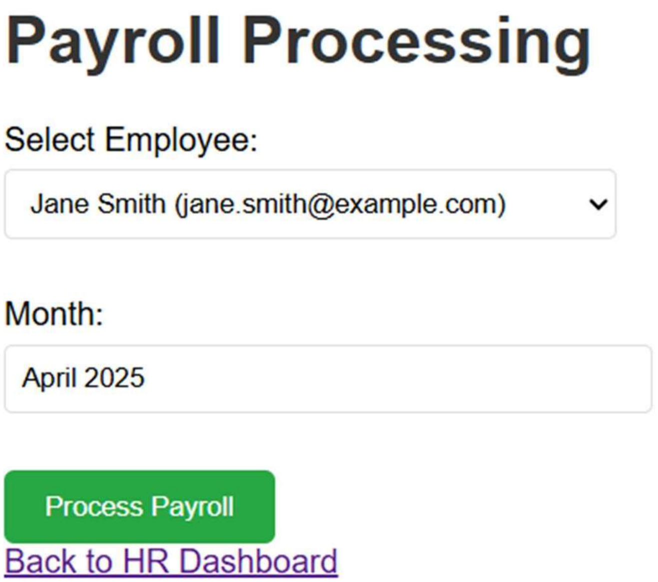
Email:

Department:

Role:

Salary:
0.0

SS10

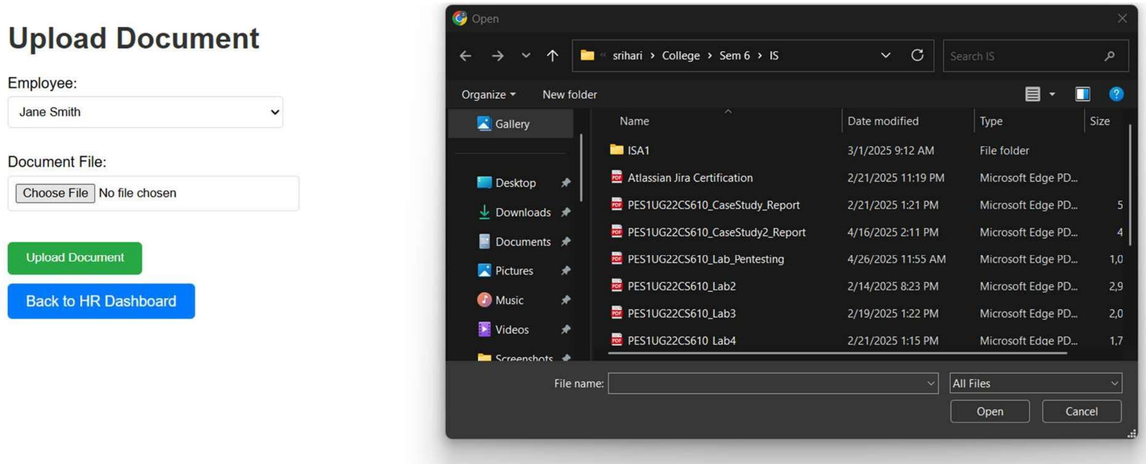


Payroll Processing

Select Employee: Jane Smith (jane.smith@example.com)

Month: April 2025

SS11



Upload Document

Employee: Jane Smith

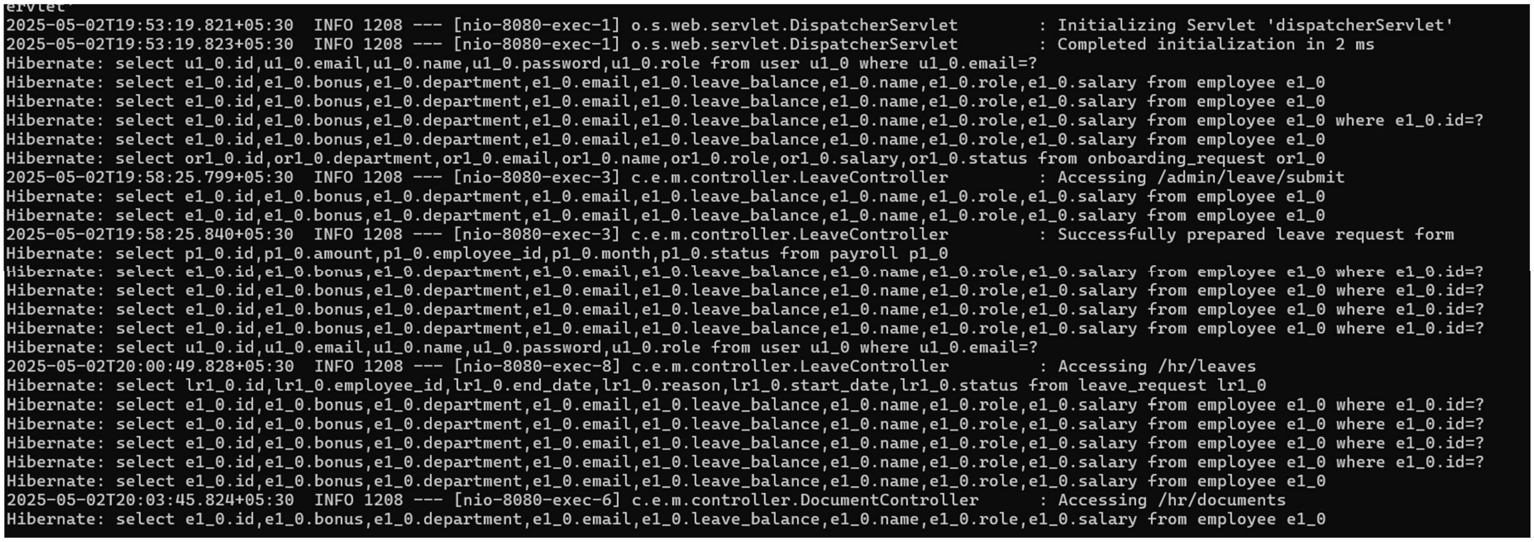
Document File: Choose File No file chosen

Name	Date modified	Type	Size
ISA1	3/1/2025 9:12 AM	File folder	
Atlassian Jira Certification	2/21/2025 11:19 PM	Microsoft Edge PD...	
PES1UG22CS610_CaseStudy_Report	2/21/2025 1:21 PM	Microsoft Edge PD...	5
PES1UG22CS610_CaseStudy2_Report	4/16/2025 2:11 PM	Microsoft Edge PD...	4
PES1UG22CS610_Lab_Pentesting	4/26/2025 11:55 AM	Microsoft Edge PD...	1,0
PES1UG22CS610_Lab2	2/14/2025 8:23 PM	Microsoft Edge PD...	2,9
PES1UG22CS610_Lab3	2/19/2025 1:22 PM	Microsoft Edge PD...	2,0
PES1UG22CS610_Lab4	2/21/2025 1:15 PM	Microsoft Edge PD...	1,7

File name:

All Files

SS12(Additionally)



```
ervlet'
2025-05-02T19:53:19.821+05:30  INFO 1208 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet       : Initializing Servlet 'dispatcherServlet'
2025-05-02T19:53:19.823+05:30  INFO 1208 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet       : Completed initialization in 2 ms
Hibernate: select u1_0.id,u1_0.email,u1_0.name,u1_0.password,u1_0.role from user u1_0 where u1_0.email=?
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0 where e1_0.id=?
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0
Hibernate: select or1_0.id,or1_0.department,or1_0.email,or1_0.name,or1_0.role,or1_0.salary,or1_0.status from onboarding_request or1_0
2025-05-02T19:58:25.799+05:30  INFO 1208 --- [nio-8080-exec-3] c.e.m.controller.LeaveController        : Accessing /admin/leave/submit
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0
2025-05-02T19:58:25.840+05:30  INFO 1208 --- [nio-8080-exec-3] c.e.m.controller.LeaveController        : Successfully prepared leave request form
Hibernate: select p1_0.id,p1_0.amount,p1_0.employee_id,p1_0.month,p1_0.status from payroll p1_0
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0 where e1_0.id=?
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0 where e1_0.id=?
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0 where e1_0.id=?
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0 where e1_0.id=?
Hibernate: select u1_0.id,u1_0.email,u1_0.name,u1_0.password,u1_0.role from user u1_0 where u1_0.email=?
2025-05-02T20:00:49.828+05:30  INFO 1208 --- [nio-8080-exec-8] c.e.m.controller.LeaveController        : Accessing /hr/leaves
Hibernate: select lr1_0.id,lr1_0.employee_id,lr1_0.end_date,lr1_0.reason,lr1_0.start_date,lr1_0.status from leave_request lr1_0
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0 where e1_0.id=?
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0 where e1_0.id=?
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0 where e1_0.id=?
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0 where e1_0.id=?
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0
2025-05-02T20:03:45.824+05:30  INFO 1208 --- [nio-8080-exec-6] c.e.m.controller.DocumentController      : Accessing /hr/documents
Hibernate: select e1_0.id,e1_0.bonus,e1_0.department,e1_0.email,e1_0.leave_balance,e1_0.name,e1_0.role,e1_0.salary from employee e1_0
```

As depicted in the screenshot of Command prompt we have added detailed logging of transactions related to database to maintain a record.

Name	**Module Worked On**
Prakash	**Performance Rating and Document Upload**
Prakash	**Payroll Processing and Bonus Management**
Srihari	**Leave and Attendance Management**
Srihari	**On boarding Management and Logging**

Note : The basic CRUD operations, SQL scripts and The UI design were handled by both of us collectively.

Github link : https://github.com/prakash0491/Employee_Management_System/