EMPLOYEE MANAGEMENT SYSTEM

Project Report

---

1. Introduction

The **Employee Management System (EMS)** is a console-based application developed using **Core Java** and **JDBC** with **PostgreSQL** as the backend database.
The purpose of this project is to manage employee records efficiently by performing basic **CRUD operations** (Create, Read, Update, Delete).

This project helps beginners understand how Java applications interact with relational databases using JDBC.

---

2. Objectives of the Project

The main objectives of this project are:

- To understand **Core Java concepts**
- To learn **JDBC database connectivity**
- To perform **CRUD operations**
- To integrate Java with **PostgreSQL**
- To create a **menu-driven application**
- To gain practical experience in backend development

---

3. Scope of the Project

The scope of the Employee Management System includes:

- Adding new employees
- Viewing all employee records
- Updating employee salary
- Deleting employee records
- Managing data securely using PreparedStatement

This system is suitable for **small organizations** and **educational purposes**.

---

4. Technologies Used

Technology	Description
Java	Core Java (JDK 17)
JDBC	Java Database Connectivity
Database	PostgreSQL
IDE	IntelliJ IDEA
OS	Windows

---

5. System Requirements

Hardware Requirements

- Processor: Intel i3 or higher
- RAM: 4 GB minimum
- Hard Disk: 20 GB free space

Software Requirements

- JDK 17 or above
- PostgreSQL Database
- PostgreSQL JDBC Driver
- IntelliJ IDEA / Eclipse

---

6. Database Design

Table Name: employees

Column Name	Data Type	Description
emp_id	INTEGER (PK)	Employee ID
name	VARCHAR(100)	Employee Name
email	VARCHAR(100)	Employee Email
department	VARCHAR(50)	Employee Department
salary	NUMERIC(12,2)	Employee Salary

---

7. Project Architecture

The project follows a **layered architecture**:

1. **Presentation Layer**
    - EmployeeManagementApp.java
    - Handles user input and menu display
2. **Business Logic Layer**
    - EmployeeDAO.java
    - Contains CRUD operations
3. **Data Layer**
    - PostgreSQL Database
    - JDBC Connection (DBConnection.java)
4. **Model Layer**
    - Employee.java
    - Represents employee entity

---

8. Core Java Concepts Used

- Classes and Objects
- Encapsulation
- Exception Handling
- Packages
- Scanner Class
- Switch Case
- Loops
- Static Methods

---

9. JDBC Concepts Used

- DriverManager
- Connection
- Statement
- PreparedStatement
- ResultSet
- SQL Queries
- Exception Handling

---

10. Functional Modules

1. Add Employee

- Accepts employee details

- Stores data into database

2. View All Employees

- Fetches all employee records
- Displays them in console

3. Update Employee Salary

- Updates salary using employee ID

4. Delete Employee

- Deletes employee record using ID

---

11. Advantages of the System

- Easy to use
- Secure database operations
- Prevents SQL injection
- Reduces manual work
- Fast and reliable
- Beginner-friendly

---

12. Limitations of the System

- Console-based UI
- No authentication system
- No data validation
- Single-user access
- Limited features

---

13. Future Enhancements

- GUI using JavaFX or Swing
- Login and authentication system
- Search employee by department
- Pagination and sorting
- Web-based version using Spring Boot
- Role-based access control

---

14. Sample Output

```
==== Employee Management System ====
1. Add Employee
2. View All Employees
3. Update Employee Salary
4. Delete Employee
5. Exit
Enter choice: 2

--- Employee List ---
1 | Alice | alice@example.com | HR | 50000
2 | Bob | bob@example.com | IT | 60000
```

15. Conclusion

The Employee Management System successfully demonstrates how **Core Java** and **JDBC** can be used to build a real-world application.
This project improves understanding of database operations, Java programming, and backend logic.
It is suitable as a **college mini project** and serves as a foundation for advanced Java applications.

16. References

- Java Documentation – Oracle
- PostgreSQL Official Documentation
- JDBC API Documentation
- IntelliJ IDEA Documentation