Employee Management System Project

Introduction

The Employee Management System project is a Java-based web application that allows companies to manage their employee data efficiently. The project is developed using Java programming language, MySQL database, and JDBC driver, and is built in Eclipse Integrated Development Environment (IDE). The objective of this project is to create a system that is user-friendly, reliable, and can handle a large number of employees.

Objective

The objective of the Employee Management System project is to provide a web-based solution for managing employee data. The project aims to create a system that is easy to use and can handle a large number of employees. The project also aims to ensure data security and reliability by using the MySQL database and JDBC driver. The web-based nature of the system makes it easily accessible and scalable.

Hardware Required

The hardware requirements for running the Employee Management System project are as follows:

- Processor: Intel Core i5 or higher
- RAM: 4 GB or higher
- Hard Disk Space: 1 GB or higher
- Display: 1024 x 768 resolution or higher

Software Required

The software requirements for running the Employee Management System project are as follows:

- Eclipse IDE
- Java Development Kit (JDK) version 8 or later
- MySQL database server
- JDBC driver

Introduction to technologies

1. Java

Java is a popular programming language that is widely used in software development. Developed by Sun Microsystems in the mid-1990s, Java was designed to be a platform-independent language that could run on any computer system. Java programs are compiled into byte code that can run on any system that has a Java Virtual Machine (JVM) installed. Java is used for developing web, mobile, and desktop applications, as well as embedded systems and gaming platforms. Java is known for its simplicity, readability, and ease of use, making it a popular choice for developers of all skill levels.

2. JDBC

JDBC (Java Database Connectivity) is a Java API that provides a standard interface for connecting Java applications to relational databases. JDBC enables Java applications to access data stored in a variety of relational database management systems (RDBMS), including MySQL, Oracle, and Microsoft SQL Server. JDBC provides a set of classes and methods that allow Java applications to send SQL commands to the database, retrieve data, and update data. JDBC is widely used in enterprise applications, web applications, and other Java-based systems.

3. MySQL

MySQL is a popular open-source relational database management system that is widely used in software development. Developed by MySQL AB (now owned by Oracle Corporation), MySQL is known for its reliability, scalability, and ease of use. MySQL supports a variety of platforms, including Windows, Linux, and macOS. MySQL is used in a wide range of applications, including web applications, content management systems, and e-commerce platforms. MySQL is compatible with many programming languages, including Java, PHP, and Python.

In the context of the Employee Management System project, Java is used to develop the web application, JDBC is used to connect the application to the MySQL database, and MySQL is used to store and retrieve employee data. Together, these technologies form a powerful and reliable stack for developing web applications that can handle a large number of employees and provide a secure and user-friendly experience for the end users.

Features

The Employee Management System project comes with the following features:

- Add new employee details to the database
- Update employee details such as employee id, name, salary, age, etc.
- Delete employee details from the database
- View employee details
- Search employee details based on ID.

Installation and Configuration

To install and configure the Employee Management System project, follow these steps:

- Download the project files from the repository. https://github.com/Viki3223/Employee-Management-Application
- Install the JDK and MySQL database server if not already installed.
- Create a new database in MySQL and execute the SQL script provided with the project files to create the required tables.
- Configure the JDBC driver in the project to connect to the MySQL database.
- Build and run the project.

Steps involves Create the Employee management Application

1. Define the Employee class: Define a class for the Employee that includes the employee's name, ID, salary, and other relevant attributes.
2. Create the Employee management class: Create a class to manage the Employee objects. The Employee management class should be able to add, remove, update, and retrieve Employee records from the console.
3. Implement the user interface: Implement a simple user interface that allows the user to interact with the Employee management class. The user interface can be implemented using the console I/O, where the user is prompted to enter options like adding, deleting, or updating an employee record.
4. Store data in a file: Implement a file system to store the Employee records. You can use a simple text file or a more structured format like JSON to store the data.
5. Implement CRUD operations: Implement create, read, update, and delete (CRUD) operations to manage the Employee records. The user should be able to create new records, read existing records, update records, and delete records from the console.
6. Validate user input: Validate user input to ensure that the data entered by the user is accurate and appropriate.
7. Test the application: Test the application to ensure that it is functioning correctly and that the Employee records are being stored and retrieved as expected.

These are the general steps to create a console-based employee management application in Java.

NOTE: Putting all the code and classes screenshot here all together. Here we have 5 main java files

Main.java

```java
package com.skilllync.empapp;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        EmployeeDaoImpl dao = new EmployeeDaoImpl();
        System.out.println("Welcome to employee management System");

        Scanner sc = new Scanner(System.in);
        do {
            System.out.println("\n\n1. Add Employee\n" +
                    "2. Show All Employee\n" +
                    "3. show Employee based on id\n" +
                    "4. Update the employee\n" +
                    "5. Delete the employee\n" +
                    "6. Delete All employees\n" +
                    "7. Exit");
            System.out.print("Enter choise: ");
            int ch = sc.nextInt();
            switch (ch) {
                case 1:
                    Employee emp = new Employee();
                    System.out.println("\n\nEnter Id :");
                    int id = sc.nextInt();
                    System.out.println("Enter name : ");
                    String name = sc.next();
                    System.out.println("Enter Salary : ");
                    double salary = sc.nextDouble();
                    System.out.println("Enter age : ");
                    int age = sc.nextInt();
                    emp.setId(id);
                    emp.setName(name);
                    emp.setSalary(salary);
                    emp.setAge(age);
                    dao.createEmployee(emp);

                    break;

                case 2:
                    dao.showAllEmployee();
                    break;

                case 3:
                    System.out.print("\n\nEnter the employe id to show details : ");
                    int empid = sc.nextInt();
                    dao.showEmployeeBasedOnId(empid);
                    break;
                case 4:
                    System.out.print("\n\nEnter id to update the details : ");
                    int empid_for_update = sc.nextInt();
                    System.out.print("Enter the updated name : ");
                    String updated_name = sc.next();
                    dao.updateEmployee(empid_for_update, updated_name);
                    break;
                case 5:
                    System.out.print("\n\nEnter the employe id to delete the employee : ");
                    int empid_for_Delete = sc.nextInt();
                    dao.deleteEmployee(empid_for_Delete);
                    break;
                case 6:
                    System.out.println("Clearing all details of the for all the employee");
                    dao.deleteAllEmployee();
                    break;
                case 7:
                    System.out.println("\n\nThanku for using our Application !!!");
                    System.exit(0);
                default:
                    System.out.println("\n\nEnter valid choise ");
                    break;
            }

        } while (true);
    }
}
```

Employee.java

```java
package com.skilllync.empapp;

public class Employee {
    private int id;
    private String name;
    private double salary;
    private int age;

    public Employee() {

    }

    public Employee(int id, String name, double salary, int age)
{
        this.id = id;
        this.name = name;
        this.salary = salary;
        this.age = age;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Employee [id=" +
                id + ", name=" +
                name + ", salary=" +
                salary + ", age=" +
                age + "]";
    }

}
```

EmployeeDaoIntrf.java

```java
package com.skilllync.empapp;

public interface EmployeeDaoIntrf {

    public void createEmployee(Employee emp);
    public void showAllEmployee();
    public void showEmployeeBasedOnId(int id);
    public void updateEmployee(int id, String n
    public void deleteEmployee(int id);
    public void deleteAllEmployee();
}
```

DBconnection.java

```java
package com.skilllync.empapp;

import java.sql.Connection;
import java.sql.DriverManager;


public class DBconnection {
    static Connection con;
    public static  Connection createDBConnection(){
        try {
            // first load the driver
            Class.forName("com.mysql.jdbc.Driver");
            // next get the connection
            String url = "jdbc:mysql://localhost:3306/employeeDB?useSSL=false";
            String user = "root";
            String pwd = "root";
            con = DriverManager.getConnection(url,user,pwd);
        } catch (Exception e) {

            e.printStackTrace();
        }
        return con;
    }
}
```

EmployeeDaoImpl.java

```
package com.skilllync.empapp;

import java.sql.Statement;
import java.sql.Connection;

import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class EmployeeDaoImpl implements EmployeeDaointrf {
    Connection con;

    @Override
    public void createEmployee(Employee emp) {

        con = DBconnection.createDBConnection();
        String query = "insert into employee values(?,?,?,?)";
        try {
            PreparedStatement pstmt = con.prepareStatement(query);
            pstmt.setInt(1, emp.getId());
            pstmt.setString(2, emp.getName());
            pstmt.setDouble(3, emp.getSalary());
            pstmt.setInt(4, emp.getAge());

            int x = pstmt.executeUpdate();
            if (x > 0) {
                System.out.println("Employee Inserted Successfully !!!");

            } else {
                System.out.println("Not Inserted successfully ");
            }

        } catch (Exception ex) {
            ex.printStackTrace();
        }

    }

    @Override
    public void showAllEmployee() {
        con = DBconnection.createDBConnection();
        String query = "Select * from employee";
        System.out.println("\n\nEmployee Details :");
        System.out.format("%s\t%s\t%s\t%s\n", "ID", "Name", "Salary", "age");
        System.out.println("--------------------");

        try {
            Statement stmt = con.createStatement();
            ResultSet result = stmt.executeQuery(query);
            while (result.next()) {
                System.out.format("%d   %s   %8.2f   %d \n", result.getInt(1),
                        result.getString(2),
                        result.getDouble(3),
                        result.getInt(4));
                System.out.println("--------------------");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    @Override
    public void showEmployeeBasedOnId(int id) {
        // TODO Auto-generated method stub
        con = DBconnection.createDBConnection();
        String querry = "Select * from employee where id = " + id;
        System.out.println("\n\nEmployee Details :");
        System.out.format("%s\t%s\t%s\t%s\n", "ID", "Name", "Salary", "age");
        System.out.println("--------------------");

        try {
            Statement stmt = con.createStatement();
            ResultSet result = stmt.executeQuery(querry);
            while (result.next()) {
                System.out.format("%d   %s   %8.2f   %d \n", result.getInt(1),
                        result.getString(2),
                        result.getDouble(3),
                        result.getInt(4));
                System.out.println("--------------------");
            }
        } catch (Exception e) {
            // TODO: handle exception
        }

    }

    @Override
    public void updateEmployee(int id, String name) {
        con = DBconnection.createDBConnection();
        String querry = "Update employee set name = ? where id = ?";
        try {
            PreparedStatement pstmt = con.prepareStatement(querry);
            pstmt.setString(1, name);
            pstmt.setInt(2, id);
            int x = pstmt.executeUpdate();
            if (x != 0) {
                System.out.println("Employee UPDATED Successfully !!!");

            } else {
                System.out.println("Not UPDATED successfully ");
            }
        } catch (Exception e) {
            e.printStackTrace();

        }

    }

    @Override
    public void deleteEmployee(int id) {
        con = DBconnection.createDBConnection();
        String querry = "Delete from employee where id = ?";
        try {
            PreparedStatement pstmt = con.prepareStatement(querry);
            pstmt.setInt(1, id);
            int x = pstmt.executeUpdate();
            if (x > 0) {
                System.out.println("Employee Deleted Successfully !!!");

            } else {
                System.out.println("Not Deleted successfully ");
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }

    }

    @Override
    public void deleteAllEmployee() {
        con = DBconnection.createDBConnection();
        String querry = "Delete from employee";
        try {
            Statement stmt = con.createStatement();
            int x = stmt.executeUpdate(querry);
            if (x > 0) {
                System.out.println("Deleted all the employees successfully!!!");
            } else {
                System.out.println("Not able to delete the employess, try again");
            }

        } catch (Exception ek) {
            ek.printStackTrace();
        }
    }
}
```

Conclusion

The Employee Management System project is a reliable and efficient system that can help companies manage their employee data effectively. It is easy to use and provides a wide range of features to handle a large number of employees. The use of Java, MySQL, and JDBC makes it a robust and scalable system that can be customized to meet specific requirements.