Employee Management System (C++ – OOP Based)

1. Project Title

Employee Management System using C++ with Object Oriented Programming

---

2. Objective

The objective of this project is to develop a menu-driven Employee Management System using C++ that demonstrates the implementation of Object Oriented Programming concepts such as abstraction, inheritance, runtime polymorphism, dynamic memory management, and the use of STL vector for dynamic storage.

---

3. Problem Statement

Managing employee records manually is difficult and inefficient. This system provides an automated solution to:

- Add employee details
- Display all employees
- Search employee by ID or name
- Delete employee records

Each employee type has a different salary structure.

---

4. Technologies Used

- **Language:** C++
- **Concepts:** OOP, STL (vector), Runtime Polymorphism
- **Compiler:** Any standard C++ compiler

---

5. OOP Concepts Implemented

1. Abstraction

An abstract base class Employee is created using a pure virtual function:

virtual void display() = 0;

This ensures that every derived class provides its own implementation.

---

2. Inheritance

Two derived classes are created:

- Developer
- Manager

Both inherit from the base class Employee.

---

3. Runtime Polymorphism

A vector of base class pointers is used:

vector<Employee*> empList;

This allows storing different types of employee objects and calling the correct display() function at runtime.

---

4. Encapsulation

Employee data such as:

- id
- name

is stored in the protected section of the base class.

---

5. Dynamic Memory Allocation

Objects are created dynamically using:

new Developer(...)
new Manager(...)

---

6. Static Data Member

A static variable is used for automatic ID generation:

static int nextId;

Each new employee gets a unique ID.

---

6. Classes and Their Responsibilities

1. Employee (Abstract Base Class)

Data Members:

- id
- name
- static nextId

Functions:

- getId()
- getName()
- virtual display()

---

2. Developer Class

Additional Data Members:

- basic salary
- allowance

Salary Calculation:

Total Salary = Basic + Allowance

---

3. Manager Class

Additional Data Members:

- basic salary
- bonus

Salary Calculation:

Total Salary = Basic + Bonus

---

7. Functional Modules

1. Add Employee

- User selects employee type

- Enters name and salary details
- System auto-generates unique ID

---

2. Display All Employees

Displays complete details of all stored employees.

---

3. Search Employee

Search can be performed using:

- Employee ID
- Employee Name (case-insensitive)

---

4. Delete Employee

Employee record can be deleted using:

- Employee ID
- Employee Name

---

8. Special Features

- Automatic ID generation using static member
- Full name input using getline()
- Case-insensitive name search using transform()
- Dynamic storage using vector
- Different salary structure for different roles
- Menu-driven user interface

---

9. Sample Output (Flow)

1. Add Employee
2. Display Employees
3. Search Employee

4. Delete Employee
5. Exit

---

10. Advantages of the System

- Easy to use
- Scalable design
- Demonstrates real-world OOP implementation
- Reduces manual record handling

---

11. Limitations

- Data is not stored permanently (no file handling)
- Single user console application

---

12. Future Enhancements

- File handling for permanent storage
- Update employee details
- Role-based login system
- GUI implementation

---

13. Conclusion

This project successfully demonstrates the use of Object Oriented Programming concepts in C++. It provides a structured and efficient way to manage employee records using dynamic memory, runtime polymorphism, and STL vector, making it suitable for real-world applications and academic learning.