A Mini Project Report

On

“EMPLOYEE MANAGEMENT SYSTEM”C++

Table Of Content

1. Introduction

1.1 Problem Statement

As businesses grow, so does the complexity of managing employee data. Manual record-keeping and calculations can lead to **data inconsistency**, **delays in payroll processing**, and **human errors**. The lack of a unified system for employee data often results in inefficiencies, especially when different types of employees (such as visiting or permanent employees) have different compensation rules. Additionally, employees' information may not be easily accessible for analysis, making it difficult for HR teams to make informed decisions.

This Employee Management System addresses these challenges by providing a **centralized, automated**, and **error-free** solution for managing employee data. It helps streamline key functions such as **employee creation**, **salary calculation**, and **data display**, while being scalable enough to adapt to various organizational needs.

1.2 Key Features

1.2.1 Employee Type Management:

· The system supports multiple employee types, such as **visiting** and **permanent** employees. Each type has its own salary calculation method, ensuring flexibility for different payroll structures.

1.2.2 Employee Information Input and Display:

· Employees' basic details such as **ID** and **name** can be entered into the system.

· The system allows for the **input of additional information** based on employee type (e.g., daily wages for visiting employees and monthly salary for permanent employees).

· The system displays **employee information** in a clear, organized manner for easy access and review.

1.2.3 **Salary Calculation:**

· The system calculates the **salary** of visiting employees based on their **daily wages** and the **number of days worked**.

· For permanent employees, the system calculates the **monthly salary** based on the fixed salary provided.

1.2.4 **Error Handling:**

· The system includes **input validation** and **error handling** mechanisms to ensure that invalid inputs are caught and appropriate messages are displayed, such as when an invalid employee type or incorrect data is entered.

2. SYSTEM ANALYSIS

2.1 FUNCTIONAL REQUIREMENTS

2.1.1 User Account Management

- Sign Up: Users provide name, email, age, and password. The system validates inputs (valid email, age > 5, name without numbers, password ≥ 8 chars with numbers).
- Login: Users log in with email and password. If incorrect, show error and deny access.

2.1.2 Task Management

- Add Task: Users can add tasks with descriptions and completion status. ●

View Tasks: Display all tasks with their completion status.

● Mark Task as Completed: Users can mark tasks as completed by entering the task number.

2.1.3 Task Persistence

● Tasks and passwords are saved to files linked to the user's email.

● On login, tasks are loaded from the respective file.

2.1.4 Menu and Options

● Options to add tasks, view tasks, mark tasks as completed, or exit.

● Validate input for menu options.

2.1.5 Error Handling

● Show error messages for invalid inputs (email, password, task number, menu choice).

2.1.6 Data Integrity

● Ensure task data is consistent and saved properly across sessions.

2.2 SOFTWARE REQUIREMENTS

2.2.1 Operating System:

● Windows 11

2.2.2 Compiler:

● MinGW (recommended for compatibility with Visual Studio Code).

● Ensure it supports at least C++11 standard.

2.2.3 IDE/Text Editor:

● Visual Studio Code with the C++ extension for syntax highlighting, debugging, and IntelliSense.

2.2.4 C++ Standard Library:

● <iostream>:Used for input/output operations, such as cin and cout.

● <fstream>:Used for file handling, such as reading from and writing to text files. ● <iomanip>:Provides features for manipulating output formatting, such as controlling spaces and alignment.

● <string>:Used to work with string objects for storing and manipulating text.

2.2.5 File System Access:

● Access permissions to save user data and tasks on your laptop.

2.3 HARDWARE REQUIREMENTS

2.3.1 Processor:

● Minimum: 1 GHz

2.3.2 Memory (RAM):

● Minimum: 2 GB

3. METHODOLOGY

3.1 Overview of C++

3.1.1 C++

C++ is a high-level, general-purpose programming language designed for system and application programming. It was developed by Bjarne Stroustrup at Bell Labs in 1983 as an extension of the C programming language. C++ is an object-oriented, multi-paradigm language that supports procedural, functional, and generic programming styles.[1]

3.1.2 ADVANTAGES OF C++

● **OOPS (Object-OrientedProgramming System):**

One major advantage of C++ is its Object-Oriented Programming (OOP) system, which helps developers solve real-world problems easily. Unlike C, which can be time consuming, C++ simplifies tasks with features like classes, inheritance, polymorphism, abstraction, and encapsulation, making it easier to manage data as objects.

● **Portability:**

C++ is highly portable, allowing code written on one system to be used on another. For example, a program written on Windows can be run and modified on a Linux system, offering flexibility across different operating systems.

· **Low-Level Language:**

C++ allows low-level data manipulation, similar to C, and is used for creating embedded systems, compilers, and operating systems due to its close relationship with machine-level languages.

● **Great Memory Management:**

C++ gives developers full control over memory management through dynamic memory allocation (DMA) using pointers, such as malloc and calloc , allowing for efficient memory handling, though it increases the programmer's responsibility.

● **Multi-Paradigm Language:**

C++ is a multi-paradigm language that supports generic, imperative, and

object-oriented programming styles, offering flexibility in how code is written and structured.

● **Compatibility with C:**

C++ is compatible with C, allowing virtually every valid C program to be a valid C++ program. Programs with a .cpp extension are recognized by C++ compilers. ●

Scalability:

C++ programs can scale from small to large data sets, enabling developers to build resource-intensive applications with ease.

- **Standardization:**

  C++ is an ISO standard language, with a well-defined set of rules that ensures consistency and ease of use, regardless of where or by whom it's being used.[3]

3.1.3 ADVANTAGES OF C++ OVER OTHER LANGUAGES

C++ offers several advantages over other programming languages, making it a popular choice for various applications:

- **Performance and Efficiency**: C++ is known for its high performance and efficient memory management. It allows developers to write code that closely interacts with hardware, leading to optimized and fast-executing applications.

- **Object-Oriented Programming (OOP)**: C++ supports OOP principles such as classes, inheritance, polymorphism, data abstraction, and encapsulation. These features promote modularity and code reusability, facilitating the development of complex software systems.[6]

- **Rich Function Library**: The language provides a comprehensive standard library, including the Standard Template Library (STL), which offers a variety of functions, algorithms, and data structures. This extensive library support accelerates development and reduces the need to write code from scratch.

- **Portability**: C++ is highly portable, enabling developers to create applications that can run on multiple platforms with minimal modifications. This cross-platform capability is particularly beneficial for developing multi-device and multi-platform applications.

- **Control Over System Resources**: C++ offers fine-grained control over system resources and memory management. This control is essential for developing resource-constrained applications, such as system software, game engines, and real-time simulations.

- **Performance in Gaming and Real-Time Systems**: In industries like gaming, C++ enables developers to create highly realistic visuals and immersive experiences while maintaining excellent performance. Its ability to handle complex operations efficiently makes it a preferred choice for game development.

- **Foundation for Learning Other Languages**: Learning C++ provides a solid foundation for understanding other programming languages. Its syntax and concepts are foundational to many modern languages, making it easier for developers to transition to or learn additional languages in the future.[2]

These advantages contribute to C++'s enduring relevance and widespread use in software development across various domains.

3.1.4 DISADVANTAGES OF C++

While C++ offers numerous advantages, it also presents certain challenges and disadvantages:

- **Complexity**: C++ is a complex language with a vast feature set, including multiple programming paradigms and intricate syntax. This complexity can make it difficult for beginners to learn and for experienced developers to use effectively.

- **Manual Memory Management**: Unlike some modern languages that provide automatic garbage collection, C++ requires developers to manage memory allocation and deallocation manually. This responsibility increases the risk of memory leaks and segmentation faults if not handled carefully.

- **Lack of Built-in Thread Support**: C++ does not include built-in support for multithreading, necessitating the use of external libraries or platform-specific APIs to implement concurrent programming. This can complicate the development process for applications requiring parallelism.[4]

- **Security Concerns**: The ability to perform low-level memory manipulation in C++ can lead to security vulnerabilities, such as buffer overflows, if the code is not written carefully. Developers must exercise caution to ensure that their programs are secure.

- **Difficulty in Debugging and Maintenance**: Due to its complexity and the potential for low-level errors, debugging and maintaining C++ code can be challenging. Errors may be hard to detect and fix, especially when dealing with issues like pointer arithmetic and manual memory management.

- **Slower Compilation Times**: C++ programs can have longer compilation times, particularly in large projects, due to features like templates and the extensive use of header files. This can slow down the development cycle.

- **Limited Standard Library Support for GUI Development**: C++ lacks a standardized library for graphical user interface (GUI) development, requiring developers to rely on third-party libraries or platform-specific tools, which can lead to portability issues.[5]

4. IMPLEMENTATION

4.1 SAMPLE CODE

```
#include<iostream>

#include<string>

using namespace std;

//abstract Class

class Employee{

     protected:

          int empId;

          string name;

     public:

          //concrete method

          void InputBasicInfo(){

               cout<<"Enter Employee ID :";

               cin>>empId;

               cout<<"Enter Employee Name :";

               cin.ignore();

               getline(cin,name);

          }

          void DisplayBasicInfo()const{

              cout<<"---------- Basic Information -----------
--"<<endl;
```

```
              cout<<"Employee ID :"<<empId<<endl;

              cout<<"Employee Name :"<<name<<endl;
```

```
          }
          //Pure Virtual Function
          virtual void inputInfo() = 0;
          virtual void calculateSalary() const= 0;
};
//dirived Class Visiting Employee
class visitingEmp : public Employee{
     private:
          int numberOfDays;
          float dailyWages;
     public:
          //@Override
          void inputInfo(){
               cout<<"Enter Daily Wages :";
               cin>>dailyWages;
               cout<<"Enter Number of Days :";
               cin>>numberOfDays;
          }
          //@Override
          void calculateSalary() const{
```

```
               float salary = dailyWages*numberOfDays;
```

```
                cout<<"Salary :"<<salary<<endl;
            }
};
//darived Class Parmanet Employee
class ParmanentEmp : public Employee{
     private:
          int monthlySalary;
     public:
          void inputInfo(){
               cout<<"Enter Monthly Salary :";
               cin>>monthlySalary;
               }
          void calculateSalary() const{
                cout<<"Salary :"<<monthlySalary<<" Per
Month"<<endl;
          }

};
main(){
     Employee *Emp;
     int choice;
     do{
          cout<<"-------- Main Menu ----------\n";
          cout<<"1. Create Visiting Employee \n";
          cout<<"2. Create Parmanent Employee \n";
          cout<<"0. Exit \n";
          cout<<"Enter Your Choice :";
          cin>>choice;
          switch(choice){
 case 1:
                    Emp = new visitingEmp();
                    break;
               case 2:
                    Emp = new ParmanentEmp();
                    break;
               case 0:
                    cout<<"Exiting The Program \n";
                    break;
               default:
                     cout<<"Invalid Number : Plz Enter Valid
Number \n";
                    continue;
          }
          if(choice != 0){
               Emp->InputBasicInfo();
```

```
                    Emp->inputInfo();

                   cout<<"\nEmployee Information\n";

                    Emp->DisplayBasicInfo();

                    Emp->calculateSalary();


                    delete Emp;

              }

        }while(choice!=0);


  }
```

5. RESULTS AND TESTING

5.1 RESULTS / OUTPUT

```
Welcome to Employee Management System :

-------- Main Menu ----------

1. Create Visiting Employee

2. Create Parmanent Employee

0. Exit

Enter Your Choice :Exiting The Program
```

4.2 TEST CASES

4.2.1 Test Case 1: Create a Visiting Employee

Description:

Test the functionality of creating a visiting employee, entering their daily wages and number of days, and calculating their salary.

Steps:

1. **Select option 1 to create a visiting employee.**
2. **Enter employee ID as 101.**
3. **Enter employee name as Alice.**
4. **Enter daily wages as 100.**
5. **Enter number of days as 15.**

```
-------- Main Menu ----------
 1. Create Visiting Employee
2. Create Parmanent Employee
0. Exit Enter Your Choice :1 Enter Employee ID :101 Enter Employee Name :Alice Enter
Daily Wages :100 Enter Number of Days :15 Employee Information ---------- Basic
Information ------------- Employee ID :101 Employee Name :Alice Salary :1500
```

4.2.2 Test Case 2: Create a Permanent Employee

Description:

Test the functionality of creating a permanent employee, entering their monthly salary, and displaying the salary.

Steps:

1. Select option 2 to create a permanent employee.

2. Enter employee ID as 102.

3. Enter employee name as Bob.

4. Enter monthly salary as 5000.

```
-------- Main Menu ----------
1. Create Visiting Employee
2. Create Parmanent Employee
0. Exit Enter Your Choice :2 Enter Employee ID :102 Enter Employee
Name :Bob Enter Monthly Salary :5000 Employee Information ---------- Basic Information ----
--------- Employee ID :102 Employee Name :Bob Salary :5000 Per Month
```

4.2.3 Test Case 3: Invalid Option Input

Description:

Test the system's behavior when an invalid option is selected from the menu.

Steps:

1. Select an invalid option, e.g., 5.

```
-------- Main Menu ---------- 1. Create Visiting Employee 2. Create Parmanent Employee 0.
Exit Enter Your Choice :5 Invalid Number : Plz Enter Valid Number
```

4.2. 4: Exit the Program

Description:

Test the system's behavior when the user selects the exit option.

Steps:

1. **Select option 0 to exit the program.**

```
-------- Main Menu ----------
1. Create Visiting Employee
2. Create Parmanent Employee 0. Exit Enter Your Choice :0 Exiting The Program
```

4.2.5 Test Case 5: Test for Multiple Employees

Description:

Test the system's behavior when creating multiple employees with different options.

Steps:

1. **Select option 1 to create a visiting employee and enter the following details:**
    - **ID: 201**
    - **Name: Charlie**
    - **Daily Wages: 80**
    - **Number of Days: 10**
2. **Select option 2 to create a permanent employee and enter the following details:** ○

ID: 202

- **Name: David**
- **Monthly Salary: 4000**

3. Select option 0 to exit.

```
-------- Main Menu ---------- 1. Create Visiting Employee 2. Create Parmanent Employee 0.
Exit Enter Your Choice :1 Enter Employee ID :201 Enter Employee Name :Charlie Enter  Daily
Wages :80 Enter Number of Days :10 Employee Information ---------- Basic Information
------------- Employee ID :201 Employee Name :Charlie Salary :800 -------- Main Menu --------
-- 1. Create Visiting Employee 2. Create Parmanent Employee 0. Exit Enter Your Choice :2
```

```
Enter Employee ID :202 Enter Employee Name :David Enter Monthly Salary :4000 Employee
Information ---------- Basic Information ------------- Employee ID :202 Employee Name :David
Salary :4000 Per Month -------- Main Menu ---------- 1. Create Visiting Employee 2. Create
Parmanent Employee 0. Exit Enter Your Choice :0 Exiting The Program
```

6. CONCLUSION & FUTURE SCOPE

6.1 Conclusion:

The Employee Management System implemented using C++ is a functional and scalable program designed to handle different types of employees—visiting and permanent—with features such as:

- **Basic information management (employee ID, name).**
- **Salary calculation based on different employee types.**

- **Use of polymorphism through abstract classes and virtual functions to manage employee-specific data.**

Through the use of derived classes, like visitingEmp and ParmanentEmp, the system demonstrates the power of inheritance and dynamic polymorphism, allowing different behavior for different types of employees while maintaining a common interface for common operations.

The system has been tested with various test cases, including edge cases, to ensure its correctness and robustness. The program works as expected for all tested scenarios, handling invalid input and performing necessary calculations.

6.2 Future Scope:

1. **Database Integration:**

   - Future improvements could involve integrating the system with a database (e.g., MySQL or SQLite) to store employee records persistently. This would allow for large-scale usage and retrieval of employee data.

2. **More Employee Types:**

   - Additional types of employees (e.g., Contractual, Intern, etc.) could be added, each with its own unique salary calculation and attributes.

3. GUI Development:

- **A graphical user interface (GUI) could be developed using libraries like Qt or SFML to make the system more user-friendly, moving away from a command-line interface (CLI).**

4. Automated Reports and Analytics:

- **The system could generate automatic reports and analytics, such as monthly salary slips, performance evaluations, and attendance reports, and allow for exporting to formats like PDF or Excel.**

5. Security Features:

- **Adding user authentication and authorization levels (admin vs. employee) would help protect sensitive employee data and provide role-based access control.**

6. Web-based or Cloud Integration:

- **The system could be converted into a web-based application, making it accessible across different devices and enhancing collaboration. Technologies like PHP, Node.js, and React could be employed for a modern web interface.**

7. Mobile Application:

- **Future expansion could involve the creation of a mobile application for employees and HR personnel to manage their information and monitor reports from anywhere, utilizing platforms like React Native or Flutter.**

8. Advanced Salary Calculation:

- **More complex salary models, such as performance-based bonuses, tax deductions, or overtime payments, could be implemented to accommodate different payroll policies.**

9. Leave Management System:

- **Integrating a leave management system where employees can apply for leave, and the system can track leave balances and approval processes, would make the program more comprehensive.**

10.Employee Feedback and Performance Evaluation:

- **A module for employee feedback, surveys, and performance evaluation could be added to help HR teams track employee satisfaction and growth.**

7. Bibliography

Here is a list of references that can be used for the concepts and technologies related to the Employee Management System:

1. C++ Programming:

- **Stroustrup, B. (2013). *The C++ Programming Language* (4th ed.). Addison Wesley.**

  This book is a definitive guide to C++ and is highly recommended for

understanding advanced features of the language such as polymorphism, inheritance, and virtual functions.

2. Object-Oriented Programming (OOP):

- **Lafore, R. (2002). *Object-Oriented Programming in C++* (4th ed.). Pearson Education.**
  This book provides an in-depth explanation of OOP concepts, which are crucial for understanding the design of the Employee Management System, especially the use of inheritance and abstract classes.

3. Database Integration:

- **Groff, J. R., & Weinberg, P. N. (2009). *SQL: The Complete Reference* (3rd ed.). McGraw-Hill.**
  This resource covers SQL, which is essential for integrating an employee management system with databases for data persistence.

4. Graphical User Interface (GUI) Development in C++:

- **Wirth, N., & Sutter, S. (2007). *Programming Windows with Qt 4*. Prentice Hall.**
  This book introduces the Qt framework, which can be used to develop GUI based applications in C++.

5. Software Design and Patterns:

- **Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.**
  This classic work discusses design patterns that can be applied to improve the maintainability and scalability of software systems like the Employee Management System.

6. Web Application Development:

- **Robbins, J. N., & Sutherland, J. (2014). *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics* (5th ed.). O'Reilly Media. For future web-based enhancements to the system, this book is an excellent guide to developing web applications with HTML, CSS, and JavaScript.**