# EMPLOYEE PAYROLL MANAGEMENT SYSTEM

Last Name, First Name M.I.
Last Name, First Name M.I.
Last Name, First Name M.I.
Last Name, First Name M.I.

Technological Institute of the Philippines
Quezon City

November 2025

## Table of Contents

## Introduction

<General Problem>
Running payroll for a business is a critical but highly complicated process. As a company grows, the amount and complexity of employee data—like hours worked, vacation days, bonuses, and deductions—also increases. This makes managing payroll difficult, and if it's mishandled, it can lead to serious legal issues and financial penalties. The general problems fall into categories like operational inefficiency (excessive time consumption) , financial risks (high potential for human error) , compliance issues (risk of non-compliance with changing laws) , and data security risks.

<Specific Problem>
This project specifically addresses the problems caused by using outdated manual systems for payroll and attendance tracking. These manual methods are extremely time-consuming and highly prone to human error, which often results in costly mistakes in paychecks and tax calculations. Furthermore, these systems don't provide real-time data, leading to inaccurate attendance records and a lack of transparency. These ongoing issues cause frustration for employees, hurt their morale, and reduce their trust in the organization.

<introduce your Solution>
To solve these issues, our solution is to design and develop a secure, centralized Employee Payroll Management System. This system will feature a secure login module to protect sensitive information and ensure only authorized users can perform payroll tasks. It will provide full CRUD (Create, Read, Update, and Delete) functions, allowing administrators to easily manage employee profiles, salary structures, attendance, and tax information. This solution will automate complex tasks, significantly reducing manual effort and errors, and ultimately leading to a transparent, accurate, and efficient payroll process.

## The Project

<Discuss how the project works THOROUGHLY>
Our Employee Payroll Management System is designed to replace outdated manual processes with an efficient, secure, and automated solution. The system is built around a secure login module, which is the first layer of defense. This ensures that sensitive employee data (like bank details, government IDs, and salary information) is protected and that only authorized personnel, such as HR staff or administrators, can access and modify payroll records.

Once logged in, administrators will have access to a centralized dashboard with full CRUD (Create, Read, Update, and Delete) capabilities. This means they can:

Create: Add new employees to the system, inputting their personal details, pay rate, and tax information.

Read: View and search for any employee profile, check attendance logs, or generate payroll reports for a specific pay period.

Update: Easily modify employee information, such as giving a raise (updating salary structure), recording approved vacation days, or adjusting tax withholding information.

Delete: Remove employees who are no longer with the company.

The core function of the system is the automation of complex calculations. Instead of manually calculating hours, overtime, tax withholdings, and deductions for every single employee, our system will do this automatically. It will process attendance data against the employee's saved salary structure and tax information to generate accurate pay slips. This automation directly eliminates the common human errors, like typos or miscalculations, that plague manual systems.

By automating these tasks, the system generates a payroll process that is transparent, accurate, and efficient, freeing up HR staff from hours of manual labor and ensuring employees are paid correctly and on time.

**Objectives**

<General Objectives

The objective of this project is to design and develop a secure, centralized Employee Payroll Management System. This system will minimize the errors in manual payroll methods by automating calculations, ensuring compliance with tax and labor laws , and protecting sensitive employee data. Specifically the system will  be able to:

1. Compute salary with tax deductions correctly.
2. Maintain data security and compliance with regulations.
3. Prepare the detailed salary record of all the employees in the company.
4. Maintain allowances, deductions, and arrears details for the employees.
5. Improve transparency and compliance with labor regulations.
6. To monitor employees attendance working hours, overtime to ensure compliance with company policies and for precise salary calculation.

The system also seeks to **enhance organizational efficiency** by providing management with detailed payroll analytics, generating comprehensive reports for auditing, budgeting, and strategic decision-making. By maintaining secure and organized employee records, it supports both short-term operational needs and long-term workforce management, ensuring **sustainable payroll processes, employee satisfaction, and regulatory compliance**.Employee Payroll Management System is to **ensure the accurate and efficient computation of employee salaries** by taking into account all relevant components such as basic pay, overtime, deductions for SSS, PhilHealth, and Pag-IBIG contributions, holiday pay, and the 13th-month bonus. This objective is designed to **eliminate the risk of human errors that often occur in manual payroll processing**, thereby ensuring that employees are compensated correctly and on time. By automating calculations, the system not only reduces the workload of HR and accounting personnel but also enhances employee satisfaction and trust in the organization. Moreover, maintaining precise salary records facilitates compliance with government regulations and allows for easy auditing, reporting, and historical tracking of salary data, ultimately supporting both the operational and strategic goals of the company.

1. <Specific Objective 2>

**Flowchart of the System**

<Flow Chart Subsection Introduction>

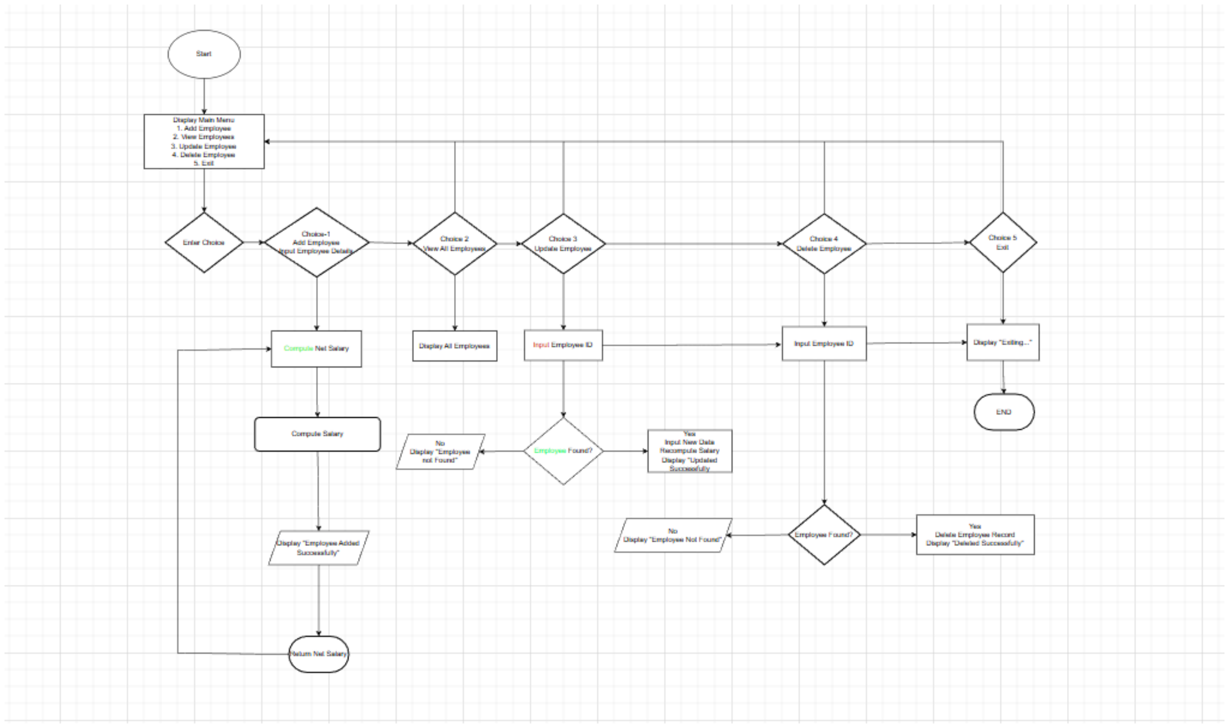


Figure 1: <flow chart name>

<Flow Chart Description> Example: Figure 1 illustrates…

## Pseudocode

<pseudo code Subsection Introduction>

Figure 2: Pseudo code of <system>

<Pseudo code Description> Figure 2 presents…

## Data Dictionary

Paragraph 1: <Data Dictionary sub section introduction >

Paragraph 2: <describe the table>

Table 1: Data Dictionary

| Data Name | Size | Data Type | Description |
|---|---|---|---|
| 1. id | | int | ID number of the employee |
| 2. First_Name | | string | Stores the first name of the employee |
| 3. Last_Name | | | Stores the last name of the employee |
| 4. basicSalary | | float | The employee’s base monthly salary |
| 5. OverTime | | float | Overtime worked by the employee |
| 6. Absent | | int | Number of days the employee was absent |
| 7. minutesLate | | int | Minutes of employee was late |
| 8. holidaysWorked | | int | Number of holidays the employee worked |
| 9. monthsWorked | | int | Number of months the employee has worked |
| 10. deductions | | float | Salary deductions |

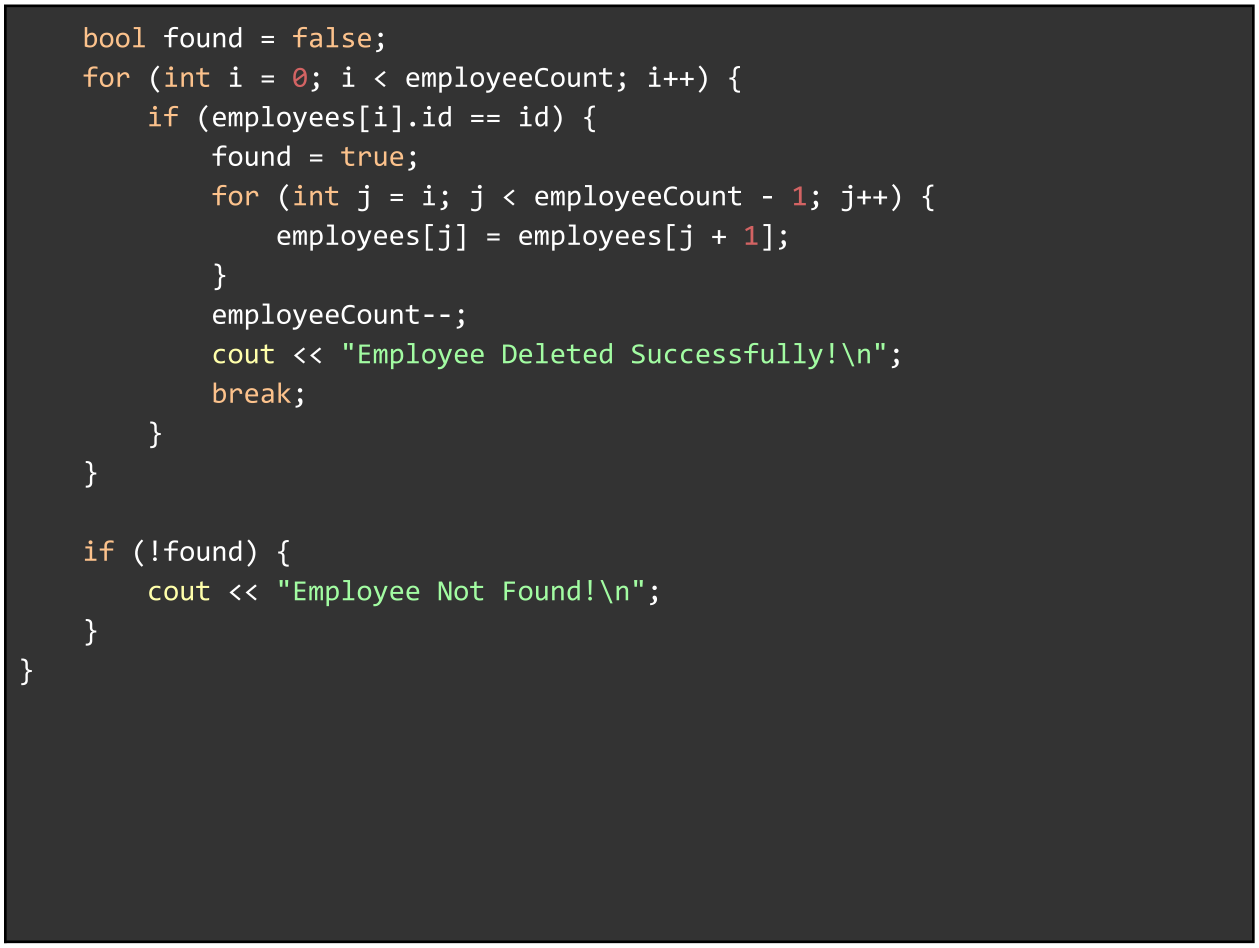
| | | | |
|---|---|---|---|
| | | | (SSS, PhilHealth, late, absences) |
| 11. holidayPay | | float | Extra pay for working on holidays |
| 12. thirteenthMonthPay | | float | Computed 13th month pay |
| 13. netSalary | | float | Final computed salary after all the deductions |
| 14. employees | Employee[500] | Array | An array to store 500 employee records |
| 15. isLegit13th | | string | Used to indicate if the employee is eligible for 13th month pay |
| 16. employeeCount | | int | Keeps track of how many employee currently exist in the system |
| 17. OVERTIME_RATE | | const float | Fixed amount paid per overtime |
| 18. ABSENCE_DEDUCTION | | const float | Fixed amount of deduction per day of absence |
| 19. LATE_DEDUCTION | | const float | |
| 20. SSS_RATE | | const float | |
| 21. PHILHEALTH_RATE | | const float | |
| 22. PAGIBIG_CONTRIBUTION | | const float | |
| 23. choice | | int | |

**Code**
<Subsection introduction >

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

//Employee infomation structure
struct Employee {
      int id;
      string First_Name;
      string Last_Name;
      float basicSalary;
      int OverTime;
      int Absent;
      int minutesLate;
      int holidaysWorked;
      string isLegit13th;
      float deductions;
      float thirteenthMonthpay;
      float netSalary;
      float holidayPay;
      float monthsWorked;
};

//Variables

Employee employees[500];
int employeeCount;


//Taxes and Deduction
const float OVERTIME_RATE = 100.0;
const float ABSENCE_DEDUCTION = 500.0;
const float LATE_DEDUCTION = 2.0;
```

```cpp
const float SSS_RATE = 0.045;
const float PHILHEALTH_RATE = 0.035;
const float PAGIBIG_CONTRIBUTION = 100.0;

//FUNCTION PROTOTYPES

float computeNetSalary(Employee &emp);
void addEmployee();
void viewEmployees();
void updateEmployee();
void deleteEmployee();

// MAIN MENU/FUNCTIONS

int main(){
      int choice;


      do{
            cout << "\n========== PAYROLL MANAGEMENT SYSTEM ===========\n";
            cout << "1. Add Employee\n";
            cout << "2. View Employees\n";
            cout << "3. Update Employee\n";
            cout << "4. Delete Employee\n";
            cout << "5. Exit\n";
            cout << "Enter your choice: ";
            cin >> choice;

            switch (choice){
            case 1: addEmployee(); break;
            case 2: viewEmployees(); break;
            case 3: updateEmployee(); break;
            case 4: deleteEmployee(); break;
            case 5: cout << "Exiting program...\n"; break;
            default: cout << "Invalid choice!\n";
            }
      } while (choice != 5);

      return 0;
}

// SALARY COMPUTATION
```

```cpp
float computeNetSalary(Employee &emp){
      float overtimePay = emp.OverTime * OVERTIME_RATE;
    float sss = emp.basicSalary * SSS_RATE;
    float philhealth = emp.basicSalary * PHILHEALTH_RATE;
    float pagibig = PAGIBIG_CONTRIBUTION;
    float lateDeduction = emp.minutesLate * LATE_DEDUCTION;
    float absenceDeduction = emp.Absent * ABSENCE_DEDUCTION;

    float dailyRate = emp.basicSalary / 22;
    emp.holidayPay = emp.holidaysWorked * dailyRate * 2;

    if (emp.monthsWorked >= 1) {
        emp.thirteenthMonthpay = emp.basicSalary / 12;
    } else {
        emp.thirteenthMonthpay = 0;
    }

    emp.deductions = sss + philhealth + pagibig + lateDeduction +
absenceDeduction;
    emp.netSalary = emp.basicSalary + overtimePay + emp.holidayPay +
emp.thirteenthMonthpay - emp.deductions;

    return emp.netSalary;
}



// ADD EMPLOYEEE FUNCTION

void addEmployee(){
      if (employeeCount >= 500){
            cout << "Employee list if currently full\n";
            return;
      }

      Employee emp;
      cout << "\n--- Add Employee ---\n";
      cout << "Enter ID: ";
      cin >> emp.id;
      cin.ignore();
      cout << "Enter First Name: ";
      getline(cin, emp.First_Name);

      cout << "Enter Last Name: ";
      getline(cin, emp.Last_Name);


      cout << "Enter Basic Salary: ";
    cin >> emp.basicSalary;
    cout << "Enter Overtime Hours: ";
    cin >> emp.OverTime;
    cout << "Enter Days Absent: ";
    cin >> emp.Absent;
    cout << "Enter Minutes Late: ";
    cin >> emp.minutesLate;
    cout << "Enter Paid Holidays: ";
    cin >> emp.holidaysWorked;
    cout << "Enter Months Worked: ";
    cin >> emp.monthsWorked;

    computeNetSalary(emp);
    employees[employeeCount] = emp;
    employeeCount++;

      cout << "Employee Successfully Added";
}


//View Employee List

void viewEmployees() {
      if (employeeCount == 0){
            cout << "\nNO Employee(s) Found please add employee(s)\n";
            return;
      }

      cout <<
"\n=======================================================================
===================================\n";
      cout << left
           << setw(6)  << "ID"
         << setw(20) << "First Name"
         << setw(20) << "Last Name"
         << setw(12) << "Basic"
         << setw(12) << "Overtime"
         << setw(10) << "Abesent"
         << setw(10) << "Late"
         << setw(14) << "Holiday Pay"
         << setw(16) << "13thMonth"
         << setw(14) << "Deductions"
         << setw(12) << "Net Salary" << endl;

      cout <<
"\n========================================================================
===================================\n";

      cout << fixed << setprecision(3);
      for (int i = 0; i < employeeCount; i++) {
            cout << left
           << setw(6)  << employees[i].id << " | "
         << setw(20) << employees[i].First_Name << " | "
         << setw(20) << employees[i].Last_Name << " | "
         << setw(12) << employees[i].basicSalary << " | "
         << setw(12) << employees[i].OverTime << " | "
         << setw(10) << employees[i].Absent << " | "
         << setw(10) << employees[i].minutesLate << " | "
         << setw(14) << employees[i].holidayPay << " | "
         << setw(16) << employees[i].thirteenthMonthpay << " | "
         << setw(14) << employees[i].deductions
         << setw(12) << employees[i].netSalary << endl << " | ";

      }

      cout <<
"\n========================================================================
===================================\n";

}


//Update Employee(s)
void updateEmployee(){
      int id;
      cout <<" Enter Employee ID to Update Information: ";
      cin >> id;

      bool found = false;
      for (int i = 0; i < employeeCount; i++){
            if(employees[i].id == id){
            found = true;
            cout << "Updating " << employees[i].First_Name << " " <<
employees[i].Last_Name << endl;
                  cout << "Enter Basic Salary: ";
                  cin >> employees[i].basicSalary;
                  cout << "Enter Overtime Hours: ";
                  cin >> employees[i].OverTime;
                  cout << "Enter Days Absent: ";
                  cin >> employees[i].Absent;
                  cout << "Enter Minutes Late: ";
                  cin >> employees[i].minutesLate;
                  cout << "Enter Paid Holidays: ";
                  cin >> employees[i].holidaysWorked;
                  cout << "Enter Months Worked: ";
                cin >> employees[i].monthsWorked;


                computeNetSalary(employees[i]);
                cout << "Employee Updated Sucessfully";
                break;
      }

  }

  if (!found){
      cout << "EMPLOYEE CANNOT BE FOUND"<< endl;
  }

}

//DELETE EMPLOYEE ACCOUNT

 void deleteEmployee() {
    int id;
    cout << "Enter Employee ID to Delete: ";
    cin >> id;
```



```
    bool found = false;
    for (int i = 0; i < employeeCount; i++) {
        if (employees[i].id == id) {
            found = true;
            for (int j = i; j < employeeCount - 1; j++) {
                employees[j] = employees[j + 1];
            }
            employeeCount--;
            cout << "Employee Deleted Successfully!\n";
            break;
        }
    }

    if (!found) {
        cout << "Employee Not Found!\n";
    }
}
```

Figure 3. <name of figure>

Code discussion

## Results and Discussion

<Subsection introduction >

<Results Discussion> ( output of code + discussion)

## Conclusion

<conclude results and discussions, add recommendations>

## References

<APA Format Alphabetical Order>