FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

UNIVERSITI TUN HUSSEIN ONN MALAYSIA

SEMESTER 1 2023/2024

OBJECT- ORIENTED PROGRAMMING (BIC20904)

SECTION 04

GROUP PROJECT REPORT (GROUP 02)

TITLE: **EMPLOYEE PAYROLL SYSTEM**

LECTURER'S NAME: **PROF. TS. DR. ROSZIATI BINTI IBRAHIM**

STUDENT'S NAME	**MATRIC NUMBER**
AKIF BIN KHAIRUL NIZAM	AI220086
MOHAMMAD NASRULLAH BIN MANAP	AI220183
NURKHAIRINA BALQIS BINTI MOHAMMAD JOE	AI220206
NURUL JANNAH BINTI KAMARUL ZAMAN	AI220147
TUAN NUR RIFAQIAH BINTI TUAN HANIZI	AI220040
TUAN KHALIDAH SYAZWANA BINTI TUAN MOHD KASMAWI	AI220118

DATE SUBMISSION : 02nd JANUARY 2024

TABLE OF CONTENTS

1.0 Introduction

Salary is a monetary payment made to a worker in exchange for the task or work done for the company as a whole. Companies that provide pay have several systems in place, with the compensation offered to employees based on their class and position. This might make it difficult for a corporation to calculate the wages of its employees. This is owing to the huge number of employees and the short time spent calculating salaries, which is normally done at the end of the month [1].

In this dynamic landscape of modern businesses, the efficient management of employee payroll stands as a cornerstone for organizational success. As companies expand and workforce intricacies evolve, the need for a robust and streamlined Employee Payroll System becomes increasingly paramount. This report explains the comprehensive analysis and evaluation of our newly implemented Employee PayrollSystem, designed to not only automate payroll processes but also enhance accuracy, compliance, and overall efficiency within our organization.

A payroll system is a software programme that companies use to automate and manage employee payments. It tracks hours worked, computes taxes and employee compensation, and sends checks or direct deposits for payments [3]. The Employee Payroll System represents a strategic investment in optimizing the payroll workflow, ensuring timely and error-free disbursement of salaries and benefits. This report aims to provide a detailed overview of the system's functionalities, its impact on organizational operations and the benefits accrued since its implementation. In this report, we will dive into the key features of the Employee Payroll System, the technology underpinning its architecture, the integration process and its alignment with current industry standards and regulatory requirements. Furthermore, we will examine the system’s adaptability to the unique needs of our organization and its potential to scale alongside our future growth.

2.0 Problem Backgrounds

Historically, manual payroll management proved to be not only labor-intensive but also susceptible to errors leading to challenges in compliances, data accuracy, and timely disbursements. As our organization experienced growth in both size and complexity, it became evident that a modernized approach to payroll administration was essential to meet the evolving demands of our workforce. The project was conceived as a strategic initiative to harness technology, streamline processes and elevate the overall efficiency of our payroll operations.

First of all, manual systems lack the ability to automate complex payroll calculations. This absence of automation increases the likelihood of errors in salary computations and benefit deductions. In addition, manual processes cannot achieve real-time processing of payroll data. The lack of immediacy in calculations and updates hampers the organization's ability to respond promptly to changes and delays salary disbursements. Manual payroll processes are also typically not scalable. As an organization grows, manual systems face challenges in accommodating an increasing number of employees, adapting to changes in compensation structures, and incorporating additional functionalities. Moreover, manual systems often lack the capability to generate efficient and customizable reports. Extracting valuable insights from payroll data for strategic decision-making becomes challenging without the analytics capabilities offered by modern payroll systems [3].

Understanding these limitations provides a context for the imperative to transition to a modern, automated Employee Payroll System, which can address these challenges, enhance efficiency, and provide a more robust solution for managing payroll processes in today's dynamic business environment.

3.0 Objectives

- **To develop a system that will reduce manual errors, enhance efficiency and expedite the payroll processing cycle.**

  The system can employ validation checks and algorithms to ensure accurate data entry and calculations, minimizing the risk of errors. This not only enhances the precision of payroll processing but also reduces the need for subsequent corrections.

- **To implement a system that ensures accurate and error-free payroll calculations.**

  This system aims to leverage technology to eliminate these errors and enhance the overall accuracy of payroll calculations. The system would be designed to perform calculations systematically, following predefined rules and formulas, thus minimizing the risk of computational mistakes. Automation ensures that each employee's compensation is accurately calculated based on factors such as working hours, overtime, allowances, and deductions, without the inherent variability associated with manual processes.

- **To develop a system that is user-friendly and minimizes the learning curve for administrators.**

  Employee Payroll System is focused on creating an interface that is intuitive, accessible, and efficiently navigable. In the payroll domain, administrators are responsible for managing and overseeing intricate tasks, including configuring payroll parameters, entering employee data, and generating reports. A user-friendly system is crucial to enhance the overall experience for administrators, ensuring that they can efficiently and effectively carry out their responsibilities without encountering unnecessary complexities.

- **To develop a system that can save more time and effort to calculate the employee’s salary.**

  The system is engineered to automate intricate salary calculations, including base pay, overtime, bonuses, and deductions. By incorporating predefined rules and formulas, the system performs these calculations swiftly and accurately, eliminating the need for manual computations. This not only accelerates the process but also minimizes the risk of errors associated with manual calculations.

4.0 Methodology

The meaning of methodology is a body of methods, rules, and postulates employed by a discipline: a particular procedure or set of procedures. The methodology for implementing an employee payroll system can vary depending on the specific requirements and context of the organization. As for this project, the first step that we took was to define the requirements needed in the system. For instance, Identify the document and the specific requirements that relate to the employee payroll system. This includes determining the types of employee information to be captured, such as their name and ID, the calculation of salary and benefits, and overtime. After that, we started the second step, which is system design. Based on the requirements list, we designed the overall structure and components of the payroll system. This includes determining the database structure, user interface design, system architecture, and integration with other relevant systems.

Next, the programming is created based on the required data. One of the important algorithms in our coding that we list as the main algorithm in the program is salary calculation and deductions. So we design and develop algorithms or rules to calculate employee salaries based on factors such as hours worked, overtime, bonuses, and deductions. The next step is system implementation. After the main algorithms are done, we develop and deploy the payroll system based on the design and requirements. This involves coding the software components, configuring the database, setting up security measures, and integrating the system with other relevant systems.

Finally, after finishing the coding, we did some testing and validation, conducting comprehensive testing of the payroll system to ensure its accuracy, reliability, and compliance with the defined requirements. Test various scenarios, including different types of employees, salary structures, and deductions. Validate the system outputs against manual calculations or sample employee data. Monitor the system for any issues or errors, and provide ongoing maintenance and support, including system upgrades, bug fixes, and compliance updates. And last but not least, it comes to reporting and documenting the entire project as a report and record for this project for future reference. Before finishing up, we ensure the payroll system complies with relevant labor laws and is based on our requirement data that we collected at the start of the project. Overall, every project will have different methodologies that are being used to make the project successful and working well, and those methods that we used are considered the best for this project.

5.0 Analysis & Design

The development of an employee payroll system involves a comprehensive analysis and design process to ensure its effectiveness and efficiency. During the analysis phase, it is crucial to gather requirements through stakeholder interviews, encompassing HR, finance, and IT perspectives. An evaluation of the existing system, if applicable, helps identify strengths and weaknesses. Compliance with overtime and data security measures must be prioritized. The design phase includes the creation of a well-structured database for employee information, a user-friendly interface for payroll administrators, and the formulation of payroll processing logic. Integration with other systems, such as HR and accounting, should be seamless, and reporting and analytics features should be designed to facilitate decision-making. Security measures, such as role-based access control and data encryption, are paramount. The implementation phase involves development, training for users, and a systematic deployment strategy. Ongoing maintenance and support are essential to address evolving regulatory requirements and ensure the system's longevity. This holistic approach ensures the creation of a robust payroll system that aligns with organizational needs while upholding accuracy, compliance, and data security.

5.1 System Analysis

5.1.1 Requirement Gathering

To make a working and functional system that reaches our stakeholders goal , we must identify and document the requirements for the payroll system.An Interview with the stakeholders, including Employer, employee, and IT personnel,must be conducted to understand their needs.Gathering information on employee details, salary structures, and other relevant data are necessary to gain a better insight for the system.

5.1.2 Current System Evaluation

When analyzing the current payroll system, evaluate its strengths, weaknesses, and potential areas for improvement. Be mindful of the system's processing speed, processing accuracy, and data security measures. Find ways to maximize processing speed, increase accuracy, and strengthen security in order to develop the system as a whole.

5.1.3 Data Security and Privacy

The implementation of strong security measures to protect confidential employee data is essential when addressing data security and privacy concerns. To stop unwanted access, this requires setting up security procedures, access limits, and encryption techniques. To further guarantee that the processing and storage of personal data abides by legal requirements and industry regulations, give compliance with data protection legislation first priority. Maintaining a high degree of data security and upholding privacy requirements can be facilitated by regular audits and upgrades to security protocols.

5.2 System Design

5.2.1 UML Diagrams

A system's behavior and structure may be seen by developers and stakeholders through the use of Unified Modeling Language (UML) diagrams, which are graphical representations [2]. System design and analysis are often expressed using UML, a standardized modeling language used in software engineering. Different UML diagram types are used to represent different components of the design of an employee payroll system.

5.2.1.1 Class Diagram

A class diagram outlines the classes that enable up the system and their attributes, methods, and relationships, giving the user a general idea of the structure of the system. The Class Diagram for an Employee Payroll System displayed entities like "**MainSystem**," "**PayrollSystem**," and "**EmployeeDatabase**" It provides a visual representation of the relationships and interactions between these classes.

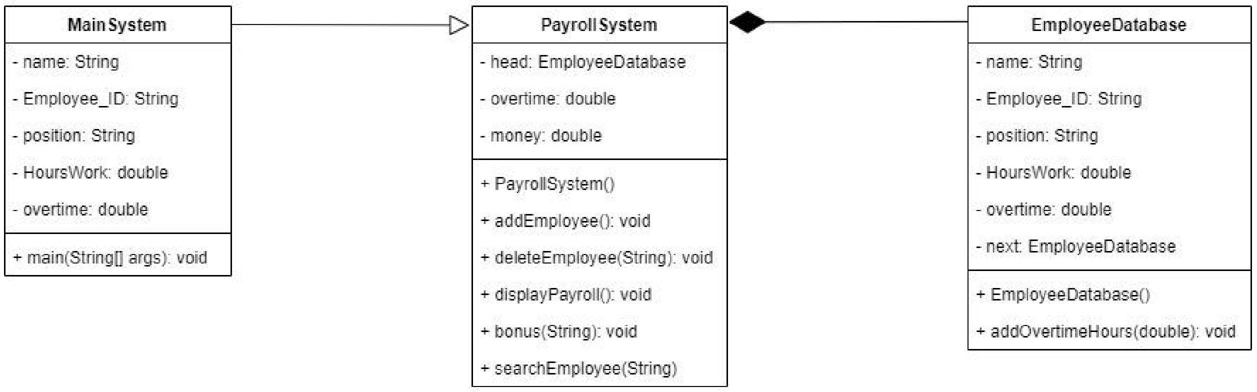


Figure 1 : Class Diagram for Employee Payroll System

5.2.1.2 Use Case Diagram

The different interactions between actors, and the system are shown in a use case diagram. Actor for an Employee Payroll System is "Admin." The representation of use cases such as "Add Employee" and "Search For Employee" provide a high-level comprehension of how the system operates from the perspective of a user.

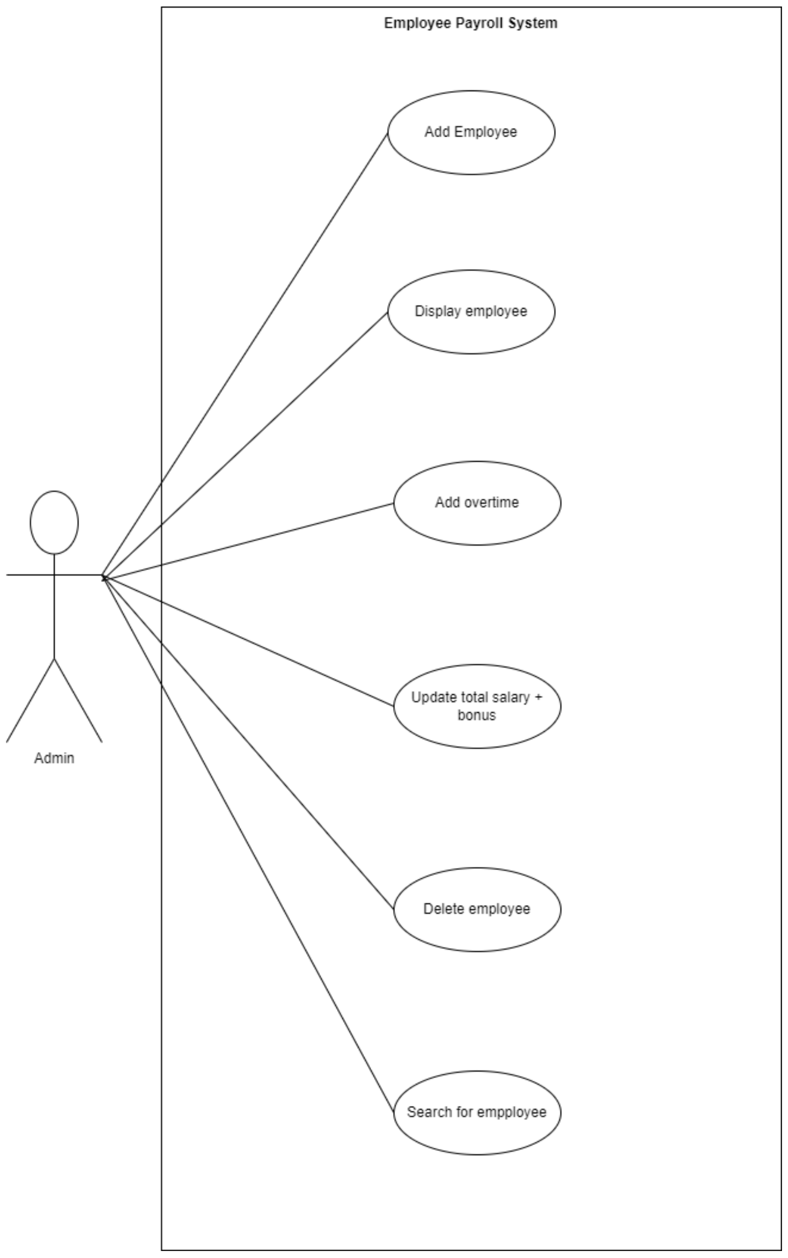


Figure 2 : Use Case Diagram for Employee Payroll System

5.2.1.3 Sequence Diagram

The chronological order of interactions between every aspect of the system is represented visually in a sequence diagram. A sequence diagram might be used in the context of the Employee Payroll System to show the steps involved in calculating payroll. It illustrates the links between the "**main system**", "**payroll system",** and the "**employee database,**" as well as the data flow between each of them.

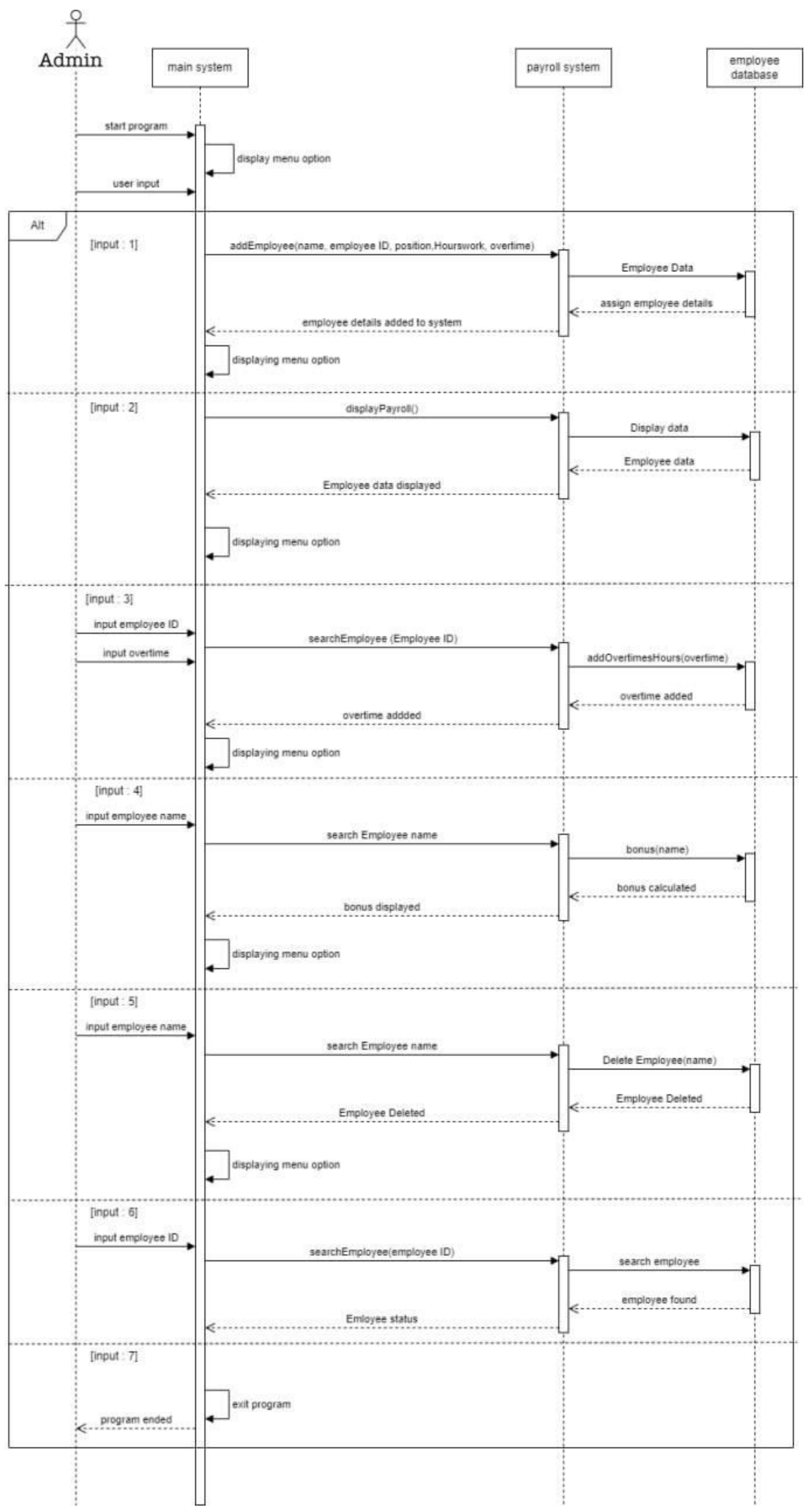

Admin
main system
payroll system
employee database
start program
display menu option
user input
Alt
[input : 1]
addEmployee(name, employee ID, position,Hourswork, overtime)
Employee Data
assign employee details
employee details added to system
displaying menu option
[input : 2]
displayPayroll()
Display data
Employee data
Employee data displayed
displaying menu option
[input : 3]
input employee ID
input overtime
searchEmployee (Employee ID)
addOvertimesHours(overtime)
overtime added
overtime addded
displaying menu option
[input : 4]
input employee name
search Employee name
bonus(name)
bonus calculated
bonus displayed
displaying menu option
[input : 5]
input employee name
search Employee name
Delete Employee(name)
Employee Deleted
Employee Deleted
displaying menu option
[input : 6]
input employee ID
searchEmployee(employee ID)
search employee
employee found
Emloyee status
[input : 7]
exit program
program ended

Figure 3 : Sequence Diagram for Employee Payroll System

5.2.1.4 Activity Diagram

An activity diagram displays parallel and sequential operations and shows how activities navigate a system. An activity diagram might be used to show each phase in an employee payroll system's "Update Employee Record" process. It describes the tasks carried out by the "Admin" and their interactions with all aspects of the system.

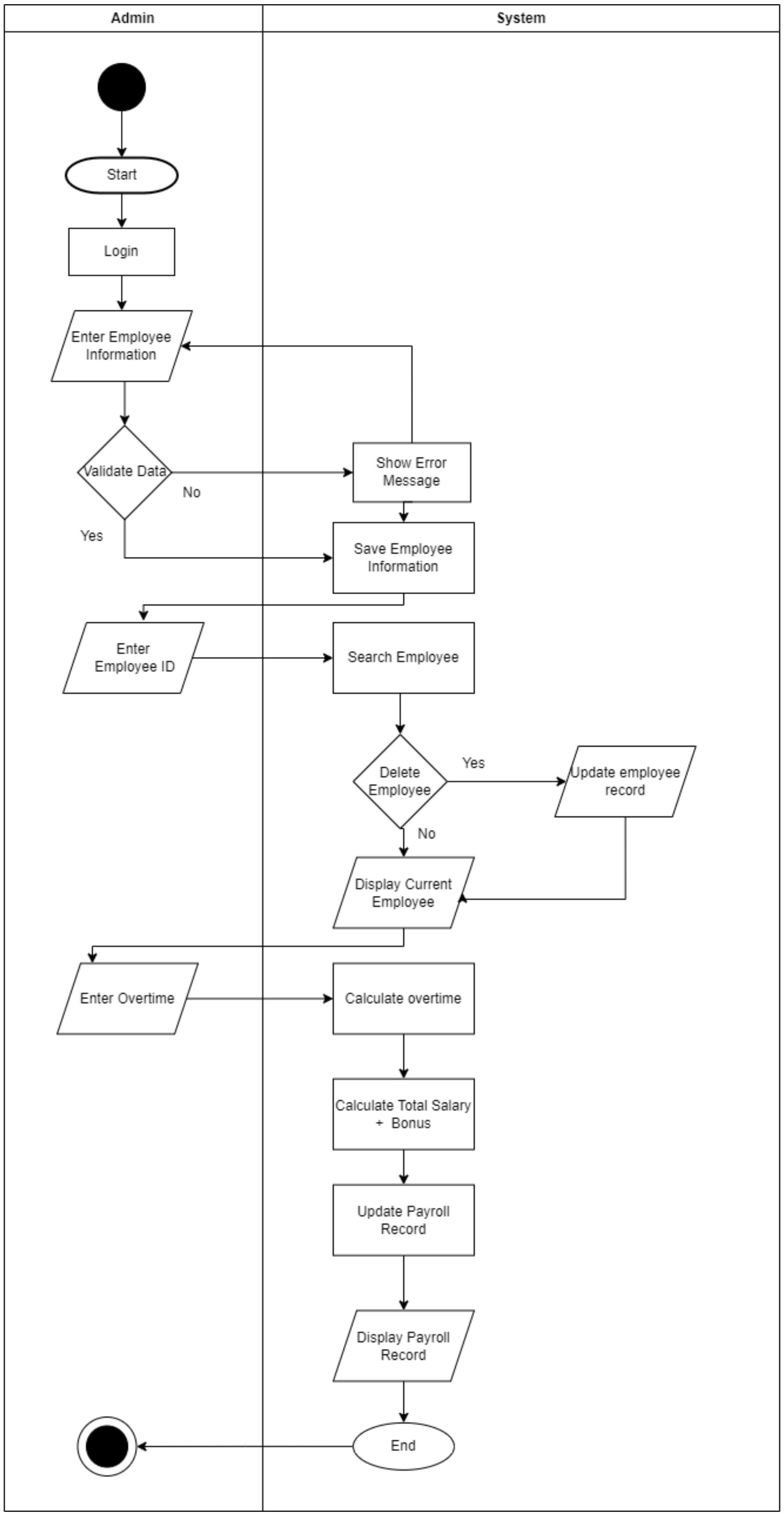
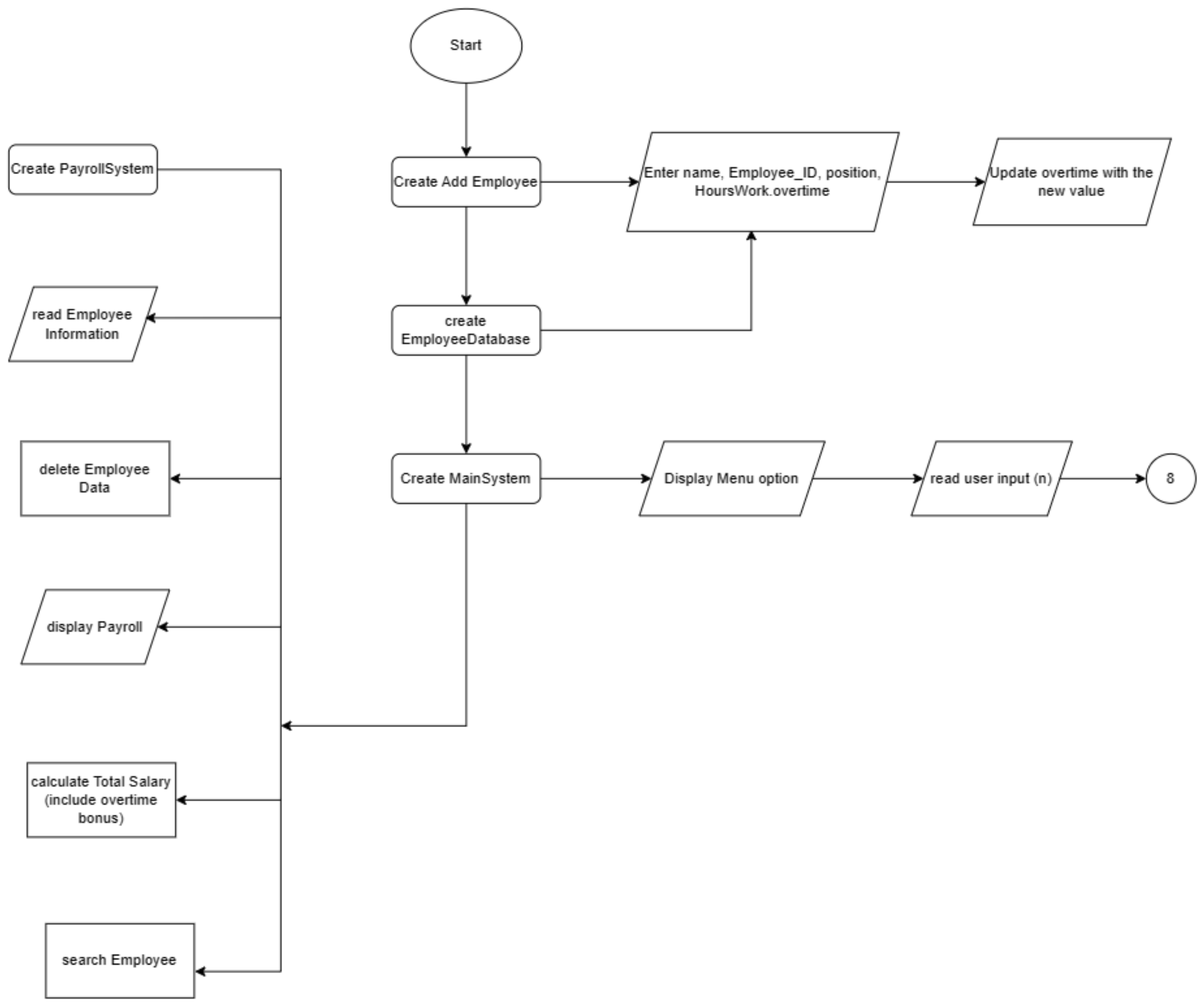
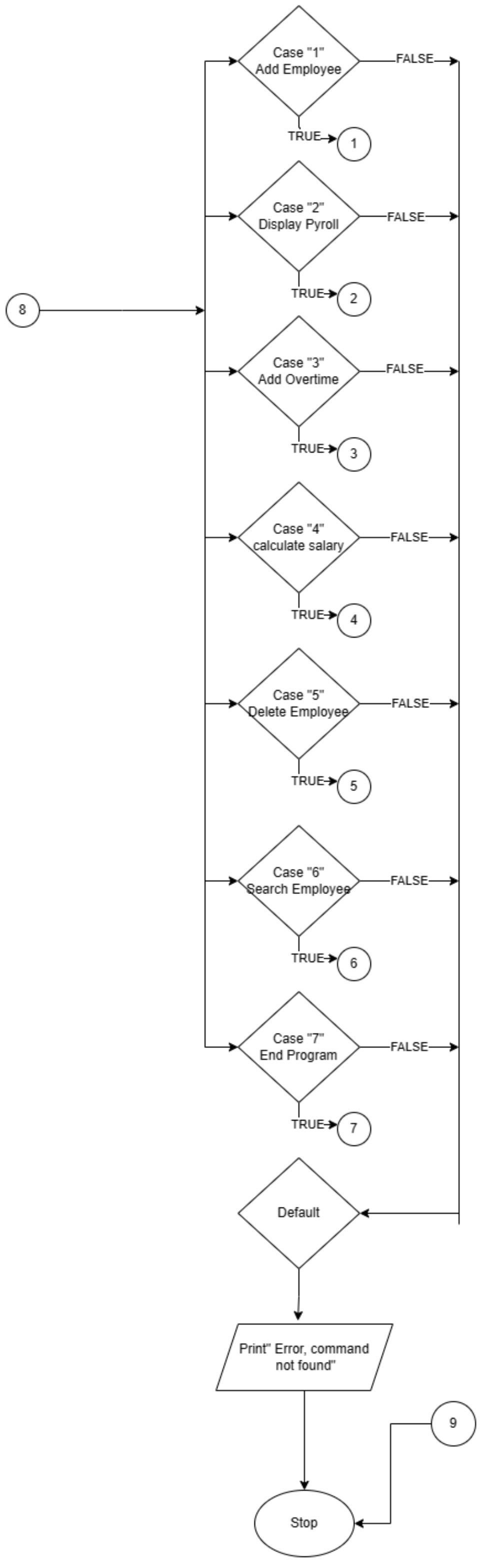


Figure 4 : Activity Diagram for Employee Payroll System

5.2.2 Flowchart

A flowchart is a visual representation that shows the steps and vital choice points in a process. An employee payroll system's total payroll processing system might be shown using a flowchart. Steps like "**read Employee Information**," "**display Menu option**," and "**display Payroll**" may be included, in addition to deciding indicators for coping with deductions or exceptions.


Start
Create PayrollSystem
Create Add Employee
Enter name, Employee_ID, position, HoursWork.overtime
Update overtime with the new value
read Employee Information
create EmployeeDatabase
delete Employee Data
Create MainSystem
Display Menu option
read user input (n)
8
display Payroll
calculate Total Salary (include overtime bonus)
search Employee



Case "1"
Add Employee
Case "2"
Display Pyroll
Case "3"
Add Overtime
Case "4"
calculate salary
Case "5"
Delete Employee
Case "6"
Search Employee
Case "7"
End Program
Default
Print" Error, command not found"
Stop
TRUE
FALSE
1
2
3
4
5
6
7
8
9

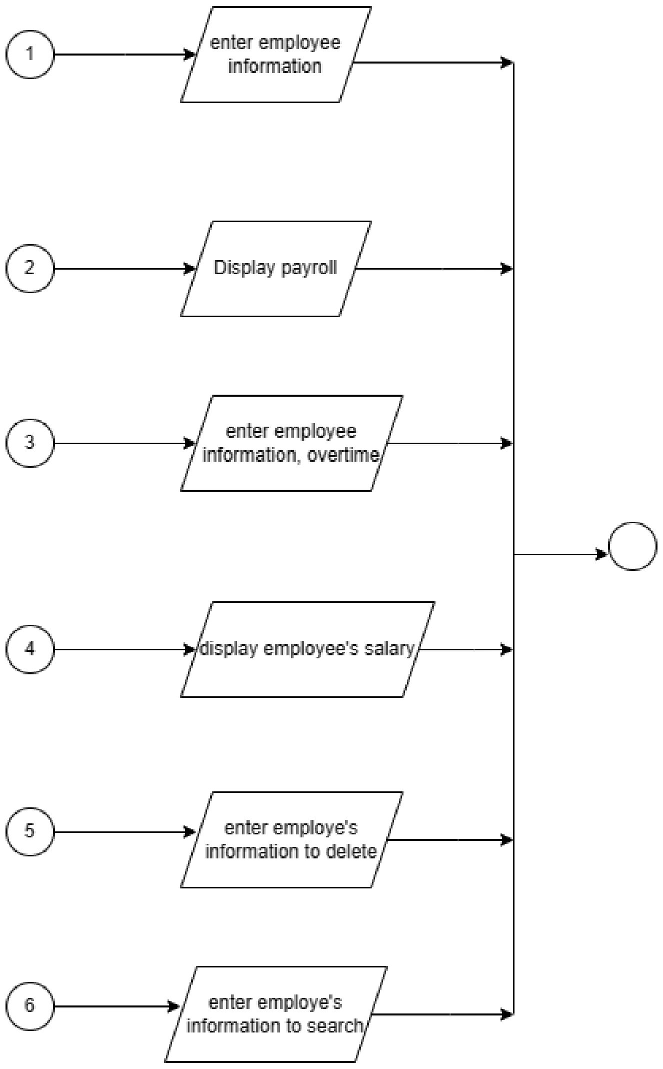

1
enter employee information
2
Display payroll
3
enter employee information, overtime
4
display employee's salary
5
enter employe's information to delete
6
enter employe's information to search

Figure 5 : Flowchart for Employee Payroll System

6.0 Implementation

6.1 Programming Language and Frameworks

The programming language used for this project was Java, which served as the foundation for creating basic functionality. Eclipse was the primary Integrated Development Environment (IDE) used, which was supplemented with a number of necessary tools to speed the development process. This project also uses classes, attributes, appropriate access modifiers, constructors, mutators, accessors, methods, and inheritance. There is also the linked list to store data that user input.

6.2 Class Implementations

6.2.1 Overview of Key Classes

6.2.1.1 PayrollSystem

The **PayrollSystem** method contains the primary characteristics and methods required to manage an employee database and payroll procedures. It has features like **head**, which refers to the beginning of the employee list, overtime, which tracks the accumulated overtime hours, and money, which holds the overtime bonus for computations.

The **addEmployee** method is critical in the class since it allows new employees to be added to the payroll system. To establish and incorporate an employee's information in the database, it requires factors such as **name**, **Employee_ID**, **position**, **HoursWork**, and **overtime**. The **deleteEmployee** function removes employees by removing a specific employee's data from the payroll based on the specified **name**.

The **displayPayroll** method displays detailed payroll information for all employees, including their **name**, **employee ID**, **position**, and computed salary, as well as any **overtime bonuses** received. The bonus technique is used to compute a specific employee's total compensation with overtime bonuses. It requires the employee's **name** to compute and show the entire compensation, which includes any collected **overtime bonuses.**

Finally, the **searchEmployee** function allows you to search for an employee within the database using their unique **Employee_ID**, providing information if the individual is discovered or showing the employee's absence if they are not found.

6.2.1.2 MainSystem

The Employee Payroll System includes various features and ways for properly managing employee data. The MainSystem method is the main function in the code.

The employee's name, unique Employee_ID, assigned position within the organization, total hours worked **(HoursWork)**, overtime hours finished **(overtime)**, and a reference to the next employee in the list **(next)** are all attributes inside the system. These characteristics jointly store and organize critical information about each employee in the system.

The **main(String[] args)** function acts as the program's entry point and orchestrates a menu-based interface. This interface allows users to engage with the employee payroll system by allowing them to conduct different operations. Users may add new employees, see current employee data, include extra hours, compute pay with bonuses, delete employees from the system, and search for individual employees using their unique identifier (Employee_ID). This technique encompasses the functionality and user interface, allowing for seamless payroll system navigation and management.

6.2.1.3 EmployeeDatabase

The **EmployeeDatabase** class contains critical characteristics and methods related to an employee's data in a payroll system. It includes data like **name**, **Employee_ID**, **position**, **HoursWork**, **overtime**, and **next**. These attributes store an employee's identifying information (name, ID, and position), the total hours worked (HoursWork), specifically for regular working hours, overtime hours (overtime), and a reference to the next employee in the linked list structure (next), allowing for the systematic organization and management of employee data.

EmployeeDatabase(String name, String Employee_ID, String position, double HoursWork, double overtime) is the constructor function responsible for initializing an employee's data upon object creation. This constructor allows to set an employee's **name**, **employee ID**, **position**, total regular **hours worked**, and **extra hours worked**, allowing to create employee objects with all of their data in one place.

The class also has the **addOvertimeHours(double overtime)** function, which simplifies the adding of extra hours for an employee. This approach allows to increase the recorded overtime hours for a single employee, allowing for the dynamic updating of overtime data across the payroll system as needed.

6.2.2 Implementation of Class Methods

The **PayrollSystem** method in this Java programme acts as the backbone for keeping the list of employees and performing other employee management procedures. It includes methods for adding new workers to the database, deleting old employee records, presenting the current employee list with their details, and calculating pay including bonuses or overtime compensation. This class effectively orchestrates the key processes necessary for the system to retain and handle employee information.

The **MainSystem** method, on the other hand, serves as the core programme that communicates with the PayrollSystem. It serves as the primary entry point for users, providing a menu-driven interface via which they may engage with the payroll system's features. This class is in charge of accepting user input, interpreting instructions, and activating the PayrollSystem class's appropriate methods to carry out particular activities requested by the user. It provides an easy and user-friendly interface for managing the payroll system's operations without gaining direct access to its core functions.

Finally, the **EmployeeDatabase** method serves as the foundation for individual employee data. It establishes the framework for storing and managing employee-specific data, such as their name, employee ID, position, hours worked, overtime, and other essential information. This class encompasses the properties and methods associated with individual employee records, allowing the system to consistently and organized produce, edit, and change employee data. It encapsulates the data connected with each employee and offers means for operating on this data within the context of the payroll system as a whole.

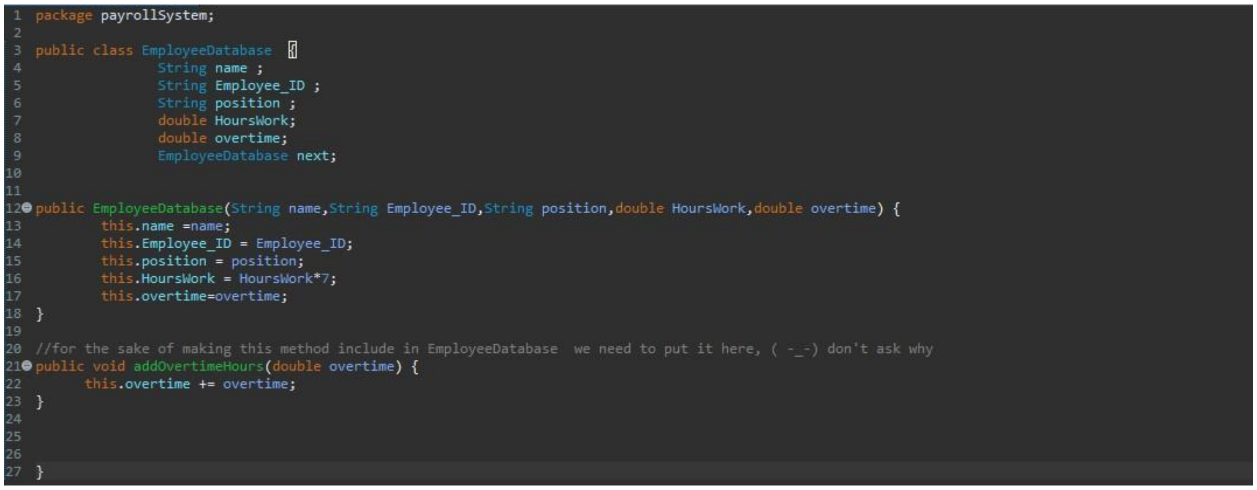
6.2.3 Handling Relationship

Using instances of the **EmployeeDatabase** class, the **PayrollSystem** class in the payroll administration system orchestrates a linked list structure to hold employee records. This approach uses the **next** reference within each **EmployeeDatabase** instance to connect individual employee data, resulting in a sequential chain of employees. The **PayrollSystem** class properly manages and manipulates the employee database using this linked list structure. It manages tasks like adding new workers, removing current ones, presenting employee information, calculating pay with bonuses, and searching for specific employees.

The relationship between these classes is centered on the hierarchical structure of the employee list, which allows the **PayrollSystem** to navigate and perform various operations on the interconnected **EmployeeDatabase** instances, resulting in a well-organized and accessible system for employee management.

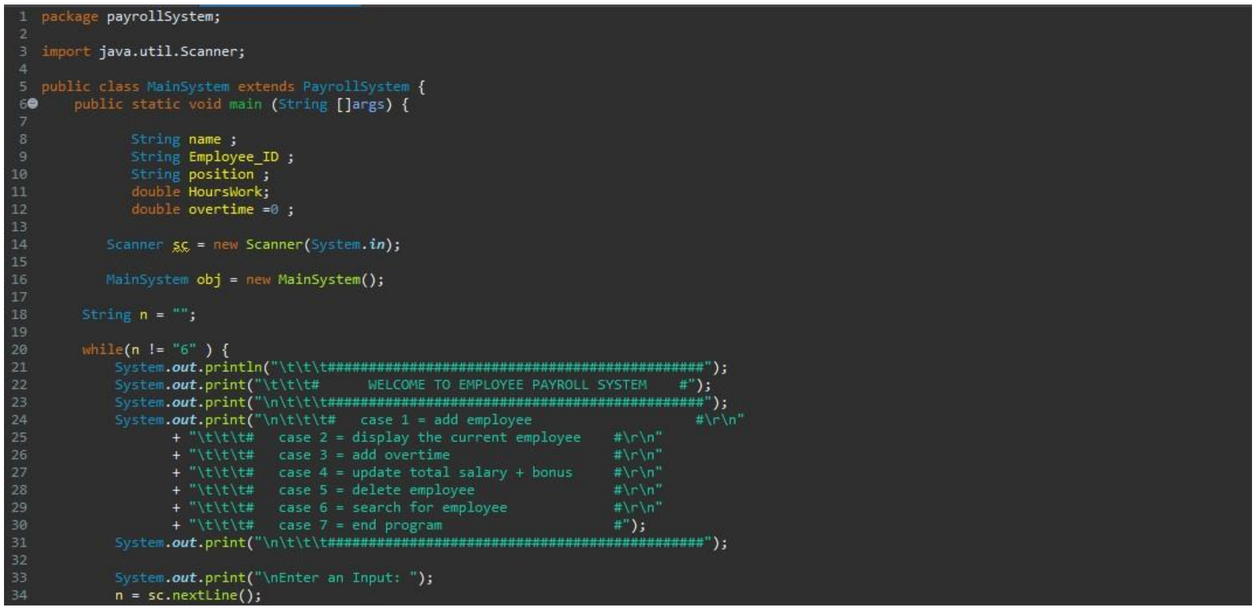
6.3 Code Documentation

6.3.1 Source Code



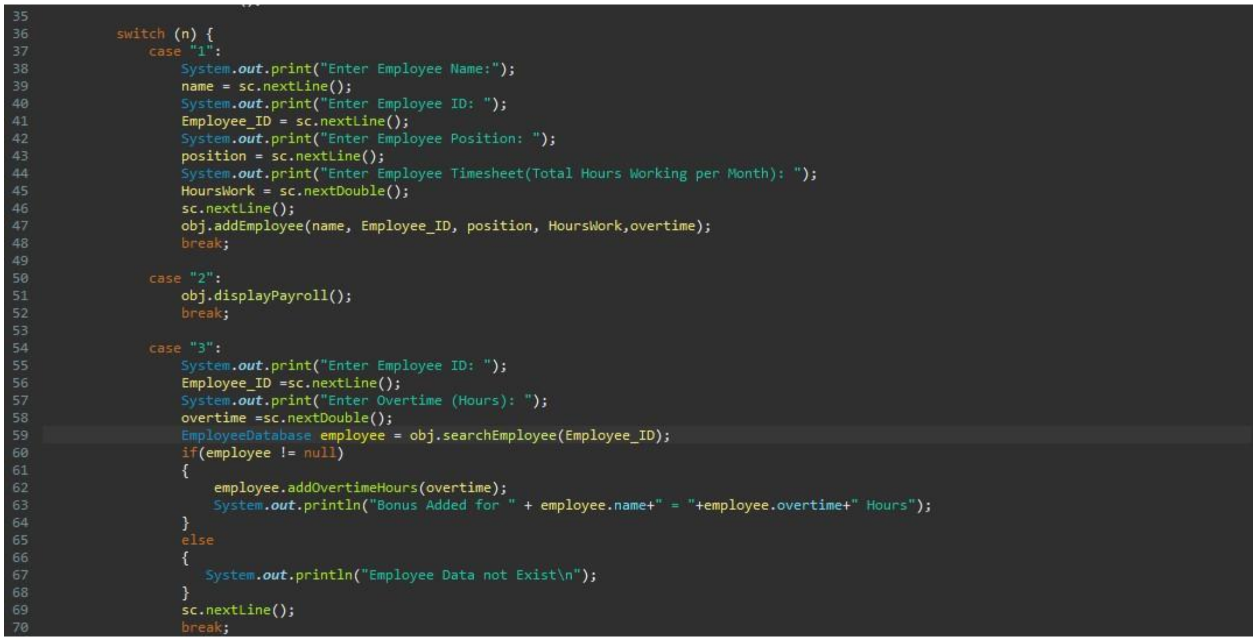
```java
package payrollSystem;

public class EmployeeDatabase  {
               String name ;
               String Employee_ID ;
               String position ;
               double HoursWork;
               double overtime;
               EmployeeDatabase next;


public EmployeeDatabase(String name,String Employee_ID,String position,double HoursWork,double overtime) {
        this.name =name;
        this.Employee_ID = Employee_ID;
        this.position = position;
        this.HoursWork = HoursWork*7;
        this.overtime=overtime;
}

//for the sake of making this method include in EmployeeDatabase  we need to put it here, ( -_-) don't ask why
public void addOvertimeHours(double overtime) {
      this.overtime += overtime;
}



}
```

Figure 6 : EmployeeDatabase Method



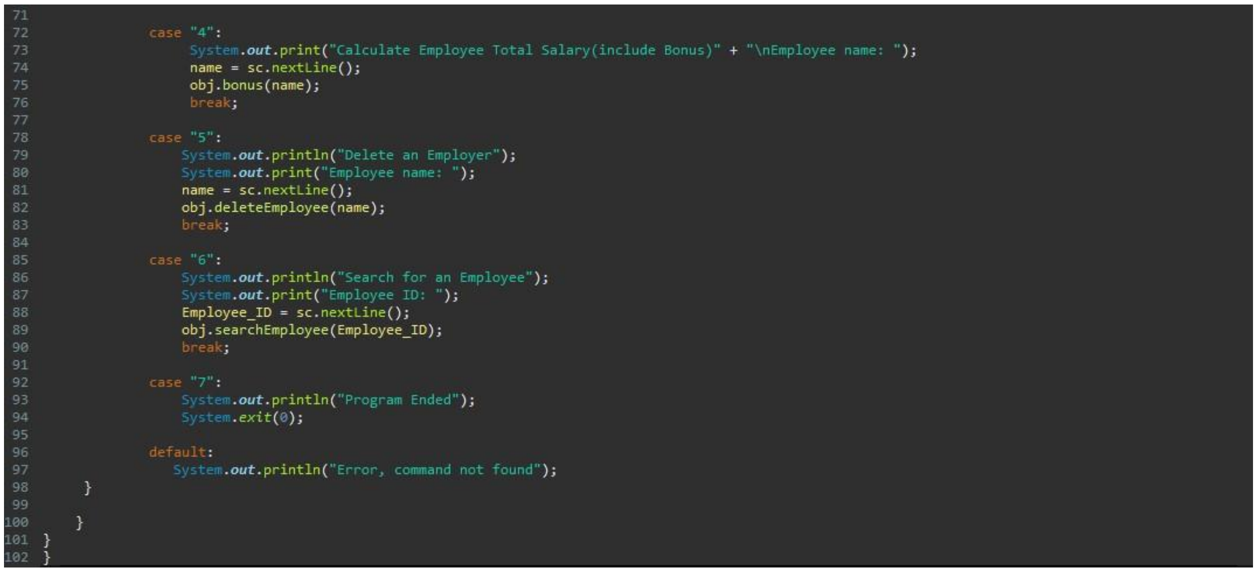
```java
package payrollSystem;

import java.util.Scanner;

public class MainSystem extends PayrollSystem {
    public static void main (String []args) {

           String name ;
           String Employee_ID ;
           String position ;
           double HoursWork;
           double overtime =0 ;

        Scanner sc = new Scanner(System.in);

        MainSystem obj = new MainSystem();

     String n = "";

     while(n != "6" ) {
         System.out.println("\t\t\t###############################################");
         System.out.print("\t\t\t#      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #");
         System.out.print("\n\t\t\t###############################################");
         System.out.print("\n\t\t\t#   case 1 = add employee                    #\r\n"
                + "\t\t\t#   case 2 = display the current employee    #\r\n"
                + "\t\t\t#   case 3 = add overtime                    #\r\n"
                + "\t\t\t#   case 4 = update total salary + bonus     #\r\n"
                + "\t\t\t#   case 5 = delete employee                 #\r\n"
                + "\t\t\t#   case 6 = search for employee             #\r\n"
                + "\t\t\t#   case 7 = end program                     #");
         System.out.print("\n\t\t\t###############################################");

         System.out.print("\nEnter an Input: ");
         n = sc.nextLine();
```

Figure 7 : MainSystem Method



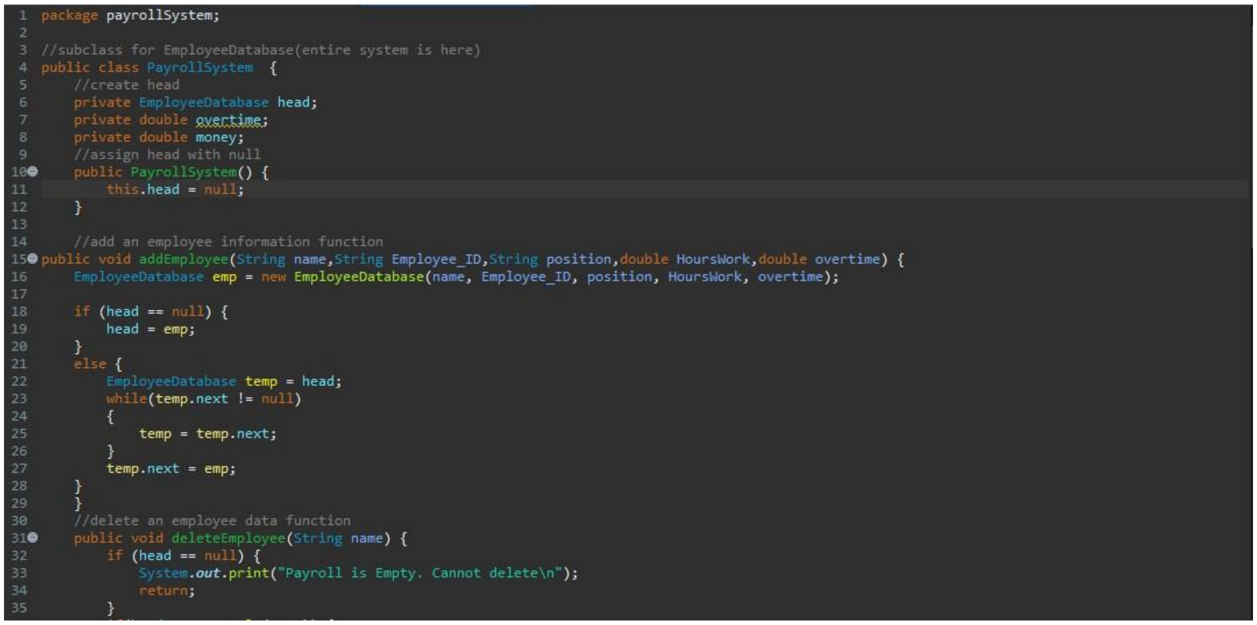
```java

        switch (n) {
            case "1":
                System.out.print("Enter Employee Name:");
                name = sc.nextLine();
                System.out.print("Enter Employee ID: ");
                Employee_ID = sc.nextLine();
                System.out.print("Enter Employee Position: ");
                position = sc.nextLine();
                System.out.print("Enter Employee Timesheet(Total Hours Working per Month): ");
                HoursWork = sc.nextDouble();
                sc.nextLine();
                obj.addEmployee(name, Employee_ID, position, HoursWork,overtime);
                break;

            case "2":
                obj.displayPayroll();
                break;

            case "3":
                System.out.print("Enter Employee ID: ");
                Employee_ID =sc.nextLine();
                System.out.print("Enter Overtime (Hours): ");
                overtime =sc.nextDouble();
                EmployeeDatabase employee = obj.searchEmployee(Employee_ID);
                if(employee != null)
                {
                    employee.addOvertimeHours(overtime);
                    System.out.println("Bonus Added for " + employee.name+" = "+employee.overtime+" Hours");
                }
                else
                {
                   System.out.println("Employee Data not Exist\n");
                }
                sc.nextLine();
                break;
```

Figure 8 : MainSystem Method cont…2



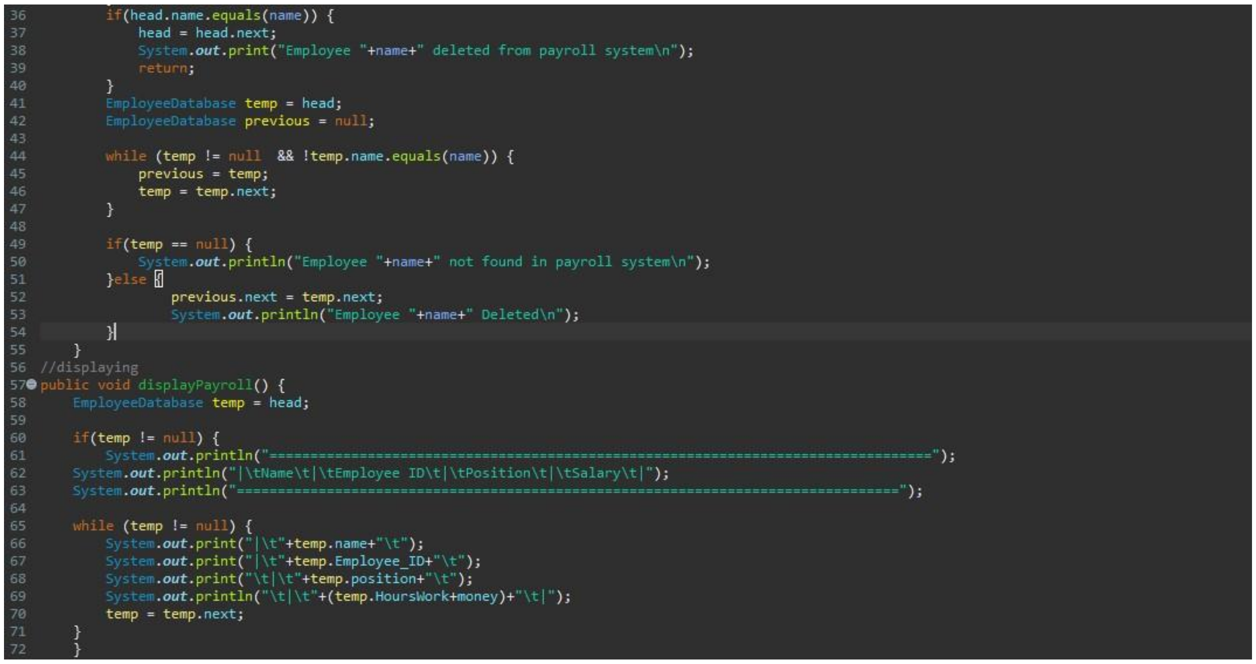
```java

            case "4":
                 System.out.print("Calculate Employee Total Salary(include Bonus)" + "\nEmployee name: ");
                 name = sc.nextLine();
                 obj.bonus(name);
                 break;

            case "5":
                System.out.println("Delete an Employer");
                System.out.print("Employee name: ");
                name = sc.nextLine();
                obj.deleteEmployee(name);
                break;

            case "6":
                System.out.println("Search for an Employee");
                System.out.print("Employee ID: ");
                Employee_ID = sc.nextLine();
                obj.searchEmployee(Employee_ID);
                break;

            case "7":
                System.out.println("Program Ended");
                System.exit(0);

            default:
               System.out.println("Error, command not found");
     }

    }
}
}
```

Figure 9 : MainSystem Method cont…3



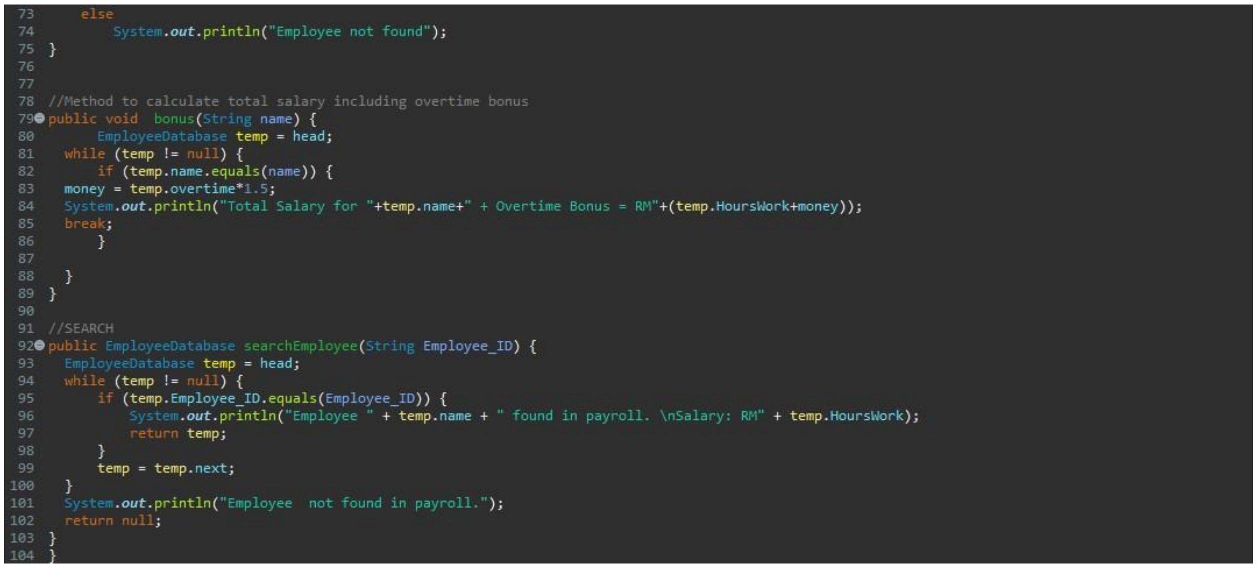
```java
package payrollSystem;

//subclass for EmployeeDatabase(entire system is here)
public class PayrollSystem  {
    //create head
    private EmployeeDatabase head;
    private double overtime;
    private double money;
    //assign head with null
    public PayrollSystem() {
        this.head = null;
    }

    //add an employee information function
public void addEmployee(String name,String Employee_ID,String position,double HoursWork,double overtime) {
    EmployeeDatabase emp = new EmployeeDatabase(name, Employee_ID, position, HoursWork, overtime);

    if (head == null) {
        head = emp;
    }
    else {
        EmployeeDatabase temp = head;
        while(temp.next != null)
        {
            temp = temp.next;
        }
        temp.next = emp;
    }
    }
    //delete an employee data function
    public void deleteEmployee(String name) {
        if (head == null) {
            System.out.print("Payroll is Empty. Cannot delete\n");
            return;
        }
```

Figure 10 :PayrollSystem Method



```java
        if(head.name.equals(name)) {
            head = head.next;
            System.out.print("Employee "+name+" deleted from payroll system\n");
            return;
        }
        EmployeeDatabase temp = head;
        EmployeeDatabase previous = null;

        while (temp != null  && !temp.name.equals(name)) {
            previous = temp;
            temp = temp.next;
        }

        if(temp == null) {
            System.out.println("Employee "+name+" not found in payroll system\n");
        }else {
                previous.next = temp.next;
                System.out.println("Employee "+name+" Deleted\n");
        }
    }
//displaying
public void displayPayroll() {
    EmployeeDatabase temp = head;

    if(temp != null) {
        System.out.println("=================================================================================");
    System.out.println("|\tName\t|\tEmployee ID\t|\tPosition\t|\tSalary\t|");
    System.out.println("=================================================================================");

    while (temp != null) {
        System.out.print("|\t"+temp.name+"\t");
        System.out.print("|\t"+temp.Employee_ID+"\t");
        System.out.print("\t|\t"+temp.position+"\t");
        System.out.println("\t|\t"+(temp.HoursWork+money)+"\t|");
        temp = temp.next;
    }
    }
```

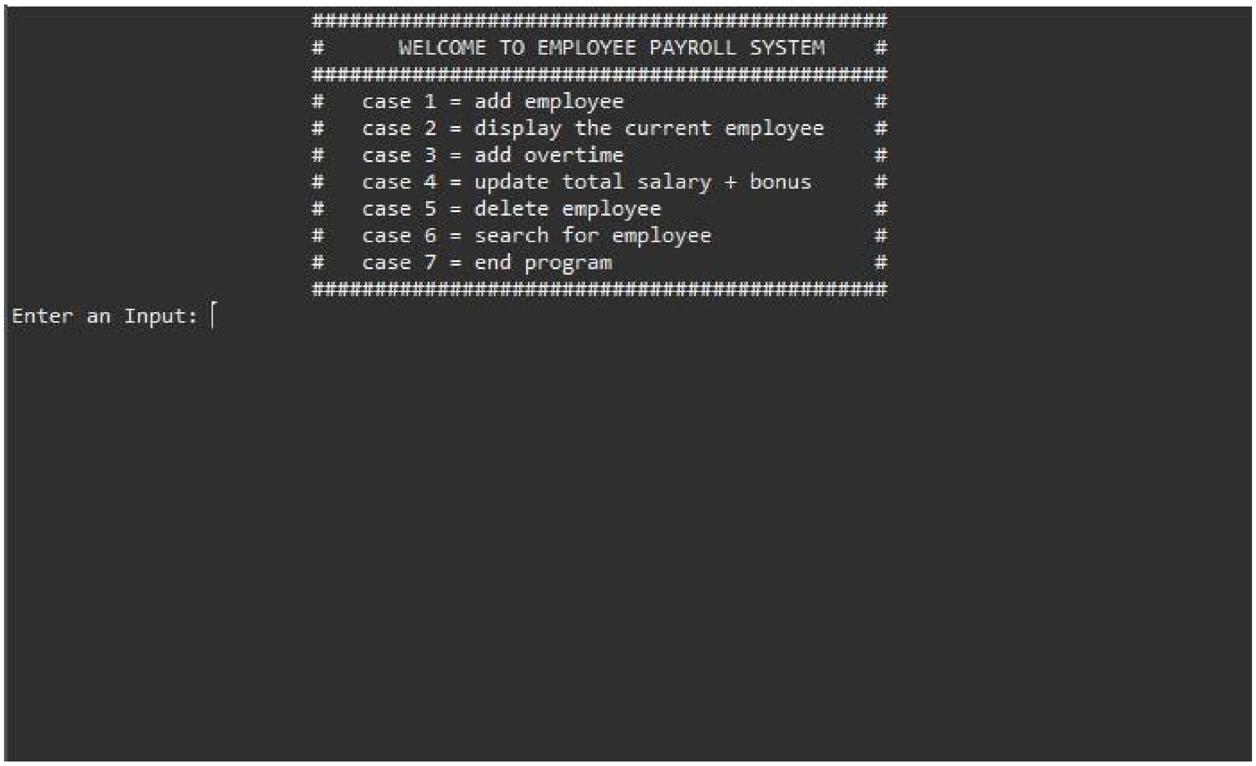
Figure 11 :PayrollSystem Method cont…2



```java
    else
        System.out.println("Employee not found");
}


//Method to calculate total salary including overtime bonus
public void  bonus(String name) {
      EmployeeDatabase temp = head;
  while (temp != null) {
      if (temp.name.equals(name)) {
  money = temp.overtime*1.5;
  System.out.println("Total Salary for "+temp.name+" + Overtime Bonus = RM"+(temp.HoursWork+money));
  break;
      }

  }
}

//SEARCH
public EmployeeDatabase searchEmployee(String Employee_ID) {
  EmployeeDatabase temp = head;
  while (temp != null) {
      if (temp.Employee_ID.equals(Employee_ID)) {
          System.out.println("Employee " + temp.name + " found in payroll. \nSalary: RM" + temp.HoursWork);
          return temp;
      }
      temp = temp.next;
  }
  System.out.println("Employee  not found in payroll.");
  return null;
}
}
```

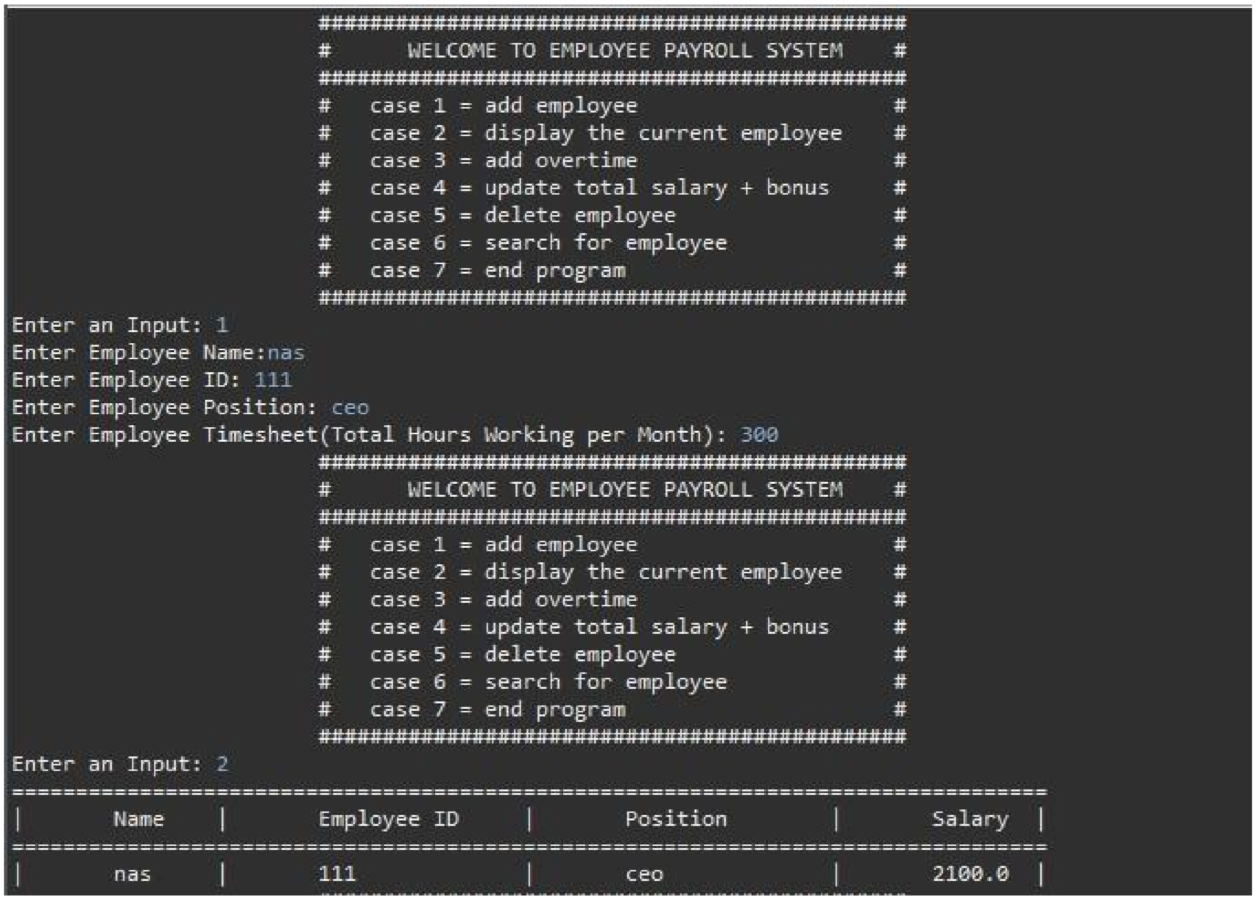
Figure 12 :PayrollSystem Method cont…3

6.3.2 Output



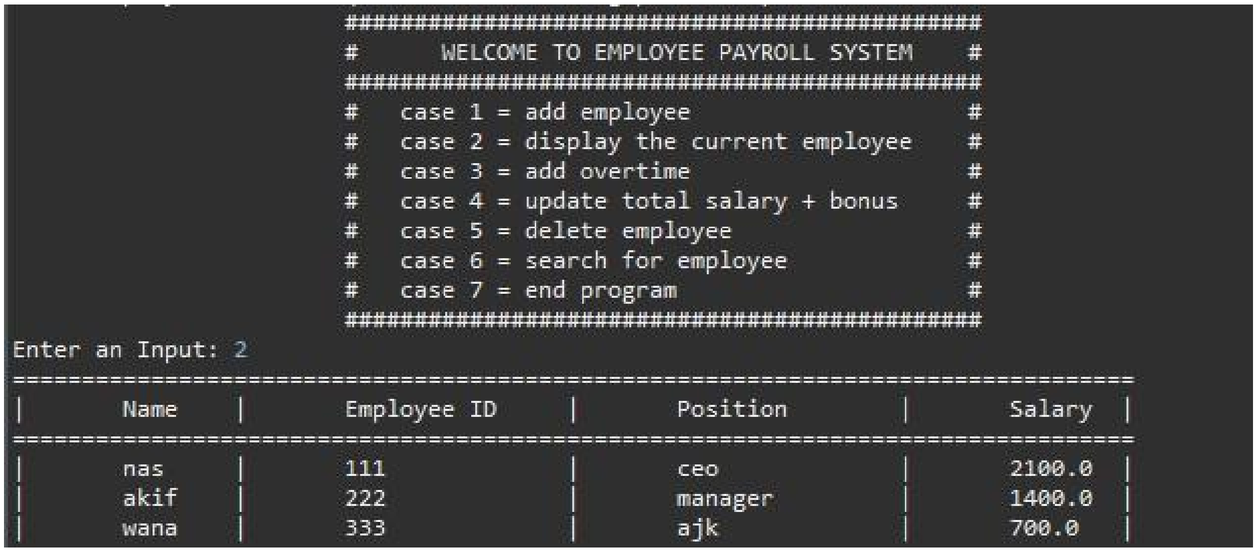
```
                        ##############################################
                        #      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #
                        ##############################################
                        #   case 1 = add employee                    #
                        #   case 2 = display the current employee    #
                        #   case 3 = add overtime                    #
                        #   case 4 = update total salary + bonus     #
                        #   case 5 = delete employee                 #
                        #   case 6 = search for employee             #
                        #   case 7 = end program                     #
                        ##############################################
Enter an Input: 
```

Figure 13 : User Interface



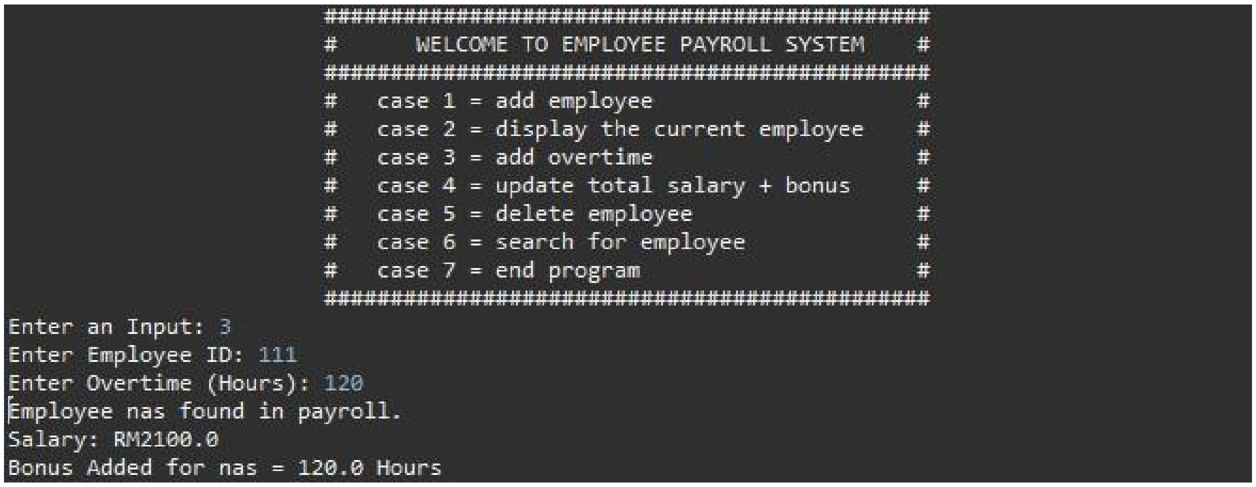
```
                        ##############################################
                        #      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #
                        ##############################################
                        #   case 1 = add employee                    #
                        #   case 2 = display the current employee    #
                        #   case 3 = add overtime                    #
                        #   case 4 = update total salary + bonus     #
                        #   case 5 = delete employee                 #
                        #   case 6 = search for employee             #
                        #   case 7 = end program                     #
                        ##############################################
Enter an Input: 1
Enter Employee Name:nas
Enter Employee ID: 111
Enter Employee Position: ceo
Enter Employee Timesheet(Total Hours Working per Month): 300
                        ##############################################
                        #      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #
                        ##############################################
                        #   case 1 = add employee                    #
                        #   case 2 = display the current employee    #
                        #   case 3 = add overtime                    #
                        #   case 4 = update total salary + bonus     #
                        #   case 5 = delete employee                 #
                        #   case 6 = search for employee             #
                        #   case 7 = end program                     #
                        ##############################################
Enter an Input: 2
=================================================================================
|       Name     |       Employee ID      |       Position        |       Salary  |
=================================================================================
|       nas      |       111              |       ceo             |       2100.0  |
```

Figure 14 : Add Employee



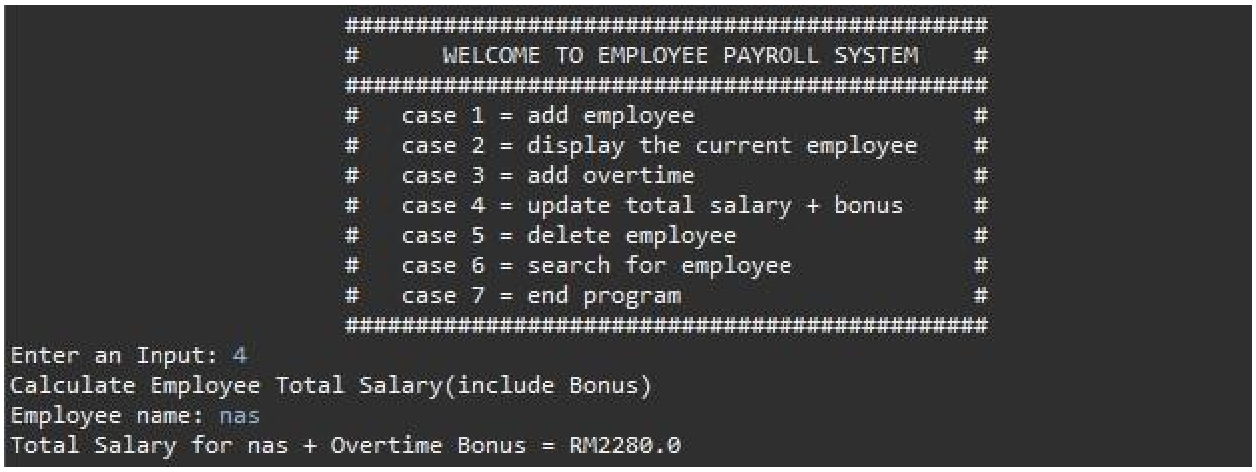
```
                        ##############################################
                        #      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #
                        ##############################################
                        #   case 1 = add employee                    #
                        #   case 2 = display the current employee    #
                        #   case 3 = add overtime                    #
                        #   case 4 = update total salary + bonus     #
                        #   case 5 = delete employee                 #
                        #   case 6 = search for employee             #
                        #   case 7 = end program                     #
                        ##############################################
Enter an Input: 2
=================================================================================
|       Name     |       Employee ID      |       Position        |       Salary  |
=================================================================================
|       nas      |       111              |       ceo             |       2100.0  |
|       akif     |       222              |       manager         |       1400.0  |
|       wana     |       333              |       ajk             |       700.0   |
```

Figure 15 : Display Employee Details



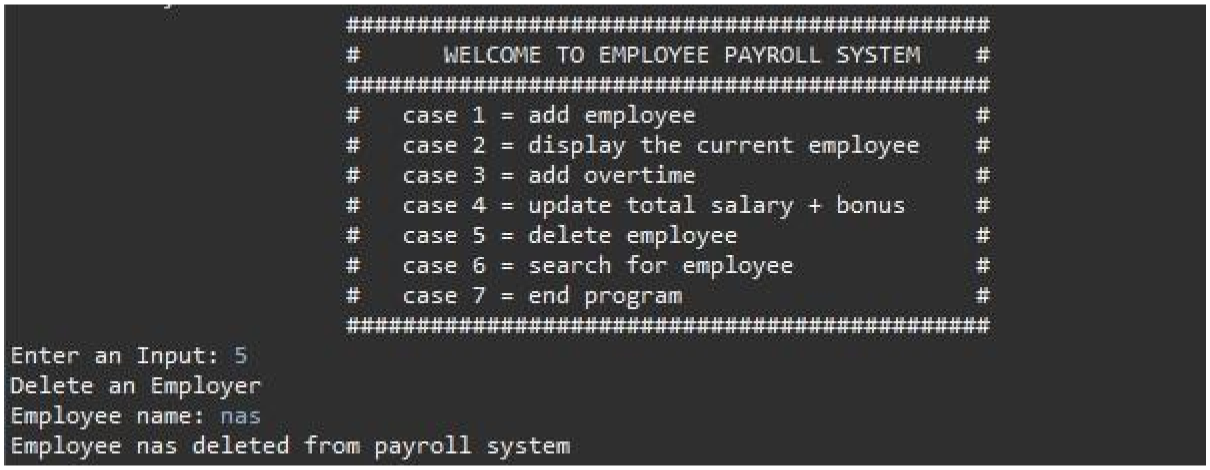
```
                              ##############################################
                              #      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #
                              ##############################################
                              #   case 1 = add employee                    #
                              #   case 2 = display the current employee    #
                              #   case 3 = add overtime                    #
                              #   case 4 = update total salary + bonus     #
                              #   case 5 = delete employee                 #
                              #   case 6 = search for employee             #
                              #   case 7 = end program                     #
                              ##############################################
Enter an Input: 3
Enter Employee ID: 111
Enter Overtime (Hours): 120
Employee nas found in payroll.
Salary: RM2100.0
Bonus Added for nas = 120.0 Hours
```

Figure 16 : Add Overtime For Employee



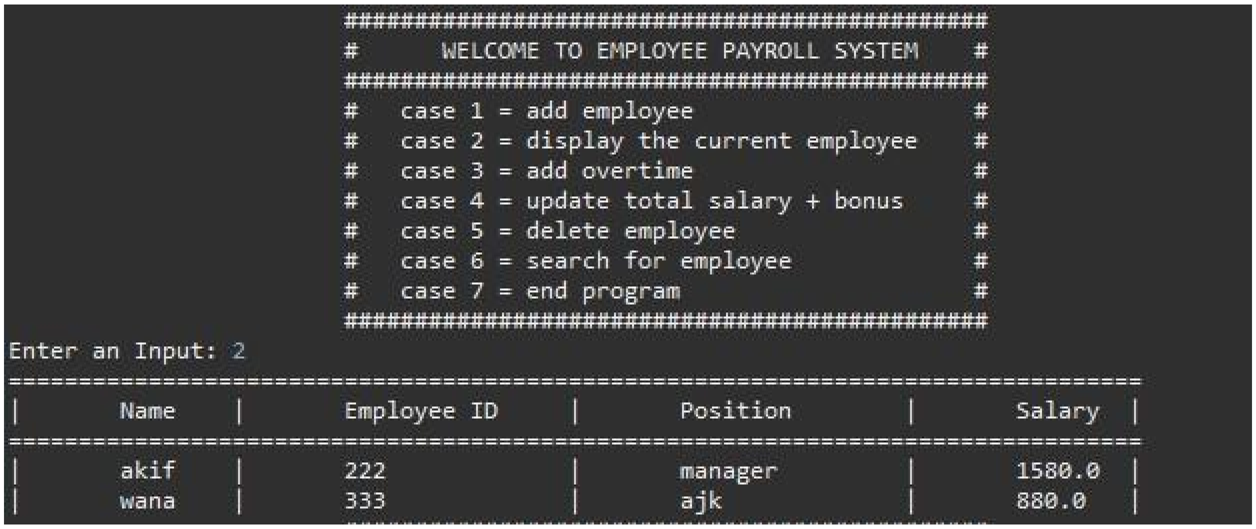
```
                              ##############################################
                              #      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #
                              ##############################################
                              #   case 1 = add employee                    #
                              #   case 2 = display the current employee    #
                              #   case 3 = add overtime                    #
                              #   case 4 = update total salary + bonus     #
                              #   case 5 = delete employee                 #
                              #   case 6 = search for employee             #
                              #   case 7 = end program                     #
                              ##############################################
Enter an Input: 4
Calculate Employee Total Salary(include Bonus)
Employee name: nas
Total Salary for nas + Overtime Bonus = RM2280.0
```

Figure 17: Update Total Salary + Bonus Employee



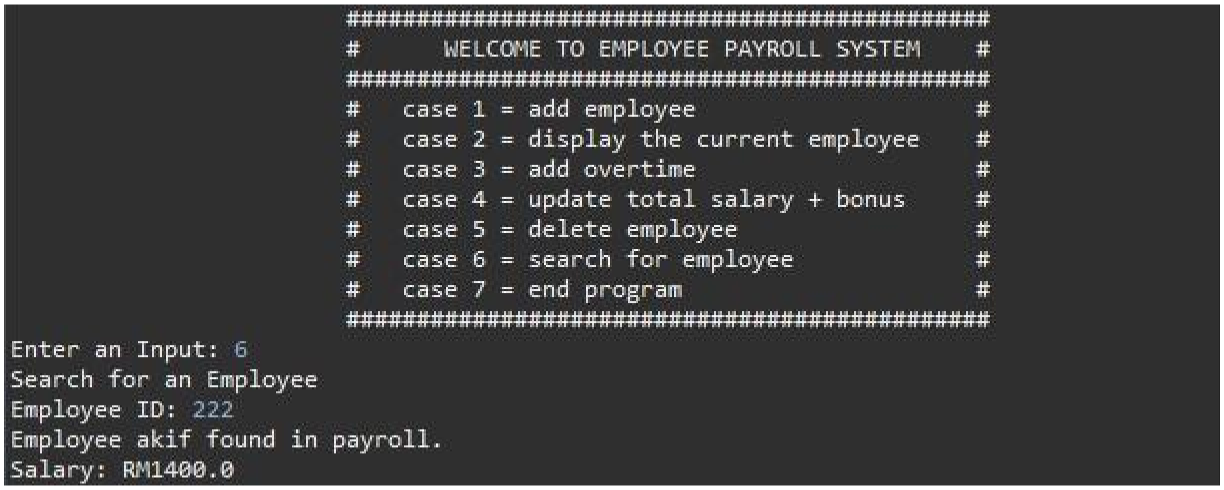
```
                              ##############################################
                              #      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #
                              ##############################################
                              #   case 1 = add employee                    #
                              #   case 2 = display the current employee    #
                              #   case 3 = add overtime                    #
                              #   case 4 = update total salary + bonus     #
                              #   case 5 = delete employee                 #
                              #   case 6 = search for employee             #
                              #   case 7 = end program                     #
                              ##############################################
Enter an Input: 5
Delete an Employer
Employee name: nas
Employee nas deleted from payroll system
```

Figure 18 : Delete Employee



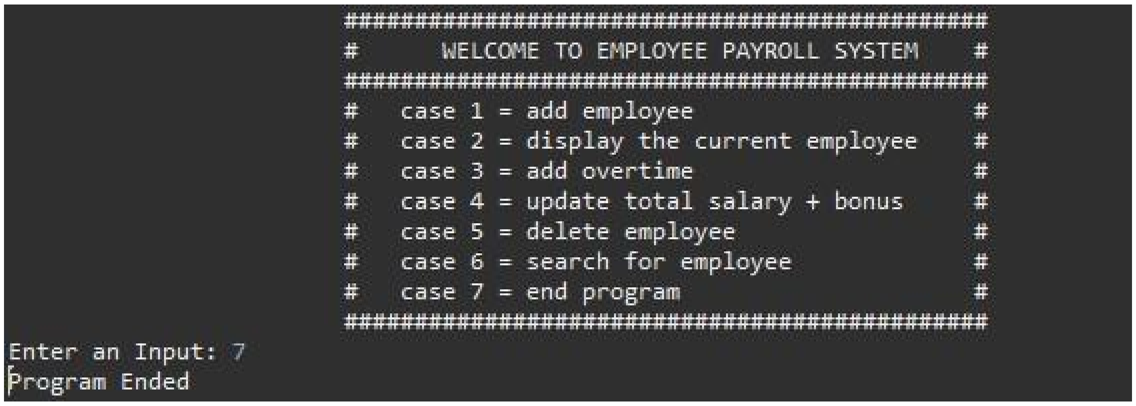
```
                        ##############################################
                        #      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #
                        ##############################################
                        #   case 1 = add employee                    #
                        #   case 2 = display the current employee    #
                        #   case 3 = add overtime                    #
                        #   case 4 = update total salary + bonus     #
                        #   case 5 = delete employee                 #
                        #   case 6 = search for employee             #
                        #   case 7 = end program                     #
                        ##############################################
Enter an Input: 2
=================================================================================
|       Name    |       Employee ID     |       Position        |       Salary  |
=================================================================================
|       akif    |       222             |       manager         |       1580.0  |
|       wana    |       333             |       ajk             |       880.0   |
```

Figure 19 : After Delete



```
                        ##############################################
                        #      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #
                        ##############################################
                        #   case 1 = add employee                    #
                        #   case 2 = display the current employee    #
                        #   case 3 = add overtime                    #
                        #   case 4 = update total salary + bonus     #
                        #   case 5 = delete employee                 #
                        #   case 6 = search for employee             #
                        #   case 7 = end program                     #
                        ##############################################
Enter an Input: 6
Search for an Employee
Employee ID: 222
Employee akif found in payroll.
Salary: RM1400.0
```

Figure 20 : Search



```
                        ##############################################
                        #      WELCOME TO EMPLOYEE PAYROLL SYSTEM    #
                        ##############################################
                        #   case 1 = add employee                    #
                        #   case 2 = display the current employee    #
                        #   case 3 = add overtime                    #
                        #   case 4 = update total salary + bonus     #
                        #   case 5 = delete employee                 #
                        #   case 6 = search for employee             #
                        #   case 7 = end program                     #
                        ##############################################
Enter an Input: 7
Program Ended
```

Figure 21 : End Program

7.0 Results

7.1 Discussion

For the result, through the process of creating and executing a payroll system for employees, a thorough grasp of several aspects of financial operations has been gained. The main goal has been to develop a system that will reduce manual errors, enhance efficiency and expedite the payroll processing cycle. A system has been created by addressing all the payroll processing, which includes add employee, delete employee, display employee, search employee, add overtime, update total salary and bonus.

The addition of a new employee guarantees thorough and accurate record-keeping in addition to streamlining the onboarding process. Maintaining an organized, current employee database is essential for admin procedures like employee payroll. This capability helps achieve these goals. The option to insert additional hours worked by employees is an example of how flexible the system is in adapting shifting work patterns. Using this function, administrators may precisely determine how many overtime hours each employee has worked, which expedites the payroll calculations process. This feature streamlines the whole payroll process inside the employee system and guarantees accuracy when calculating salary based on real hours worked.

Concurrently, the function that updates the total salary and bonus is essential in guaranteeing that workers receive just compensation for their contributions. This dynamic pay adjustment encourages higher productivity and fosters employee happiness, which all help to maintain a pleasant work environment and retain employees. Next, deleting employee functionality optimizes admin procedures. This feature makes sure that the process of removing an employee from the company is easy and updates the database to avoid having outdated information. Conversely, the employee search feature boosts productivity by providing instant access to certain employee data. When combined, these features help to keep an employee database that is both compliant and well-organized.

In addition to guaranteeing the system's seamless functioning, a successful implementation of these features offers the organization certain advantages. Reducing mistakes in payroll processing, lowering compliance risks, and improving overall organizational efficiency are all achieved via accurate and effective handling of employee data. To ensure that admin processes continue smoothly and to build user confidence in the system's dependability, the code should be built with strong error-handling capabilities.

To guarantee that users, who are administrators, can easily navigate the system, an intuitive interface is essential. This lowers the learning curve for new users while simultaneously increasing overall productivity. Clear feedback and direction in the event of problems are provided via efficient error handling, which enhances the user experience even more. Maintainability is improved by a well-documented codebase, which makes it possible for effective troubleshooting and future improvements, eventually saving the company time and money.

In summary, an effective payroll employee system has advantages that go beyond its practical aspects. The technology turns into a priceless tool that accelerates admin procedures, guarantees pay accuracy, and encourages organizational authenticity. The system's long-term effectiveness is influenced by best practices in software development, such as consistent upgrades based on user input and coding standards observance. Through acknowledgement of the important part these capabilities play and adoption of optimal practices, organizations can establish an enhanced and employee-focused work environment, resulting in better outcomes.

7.2 Software Test

Test Case 1 : Valid are input with 1

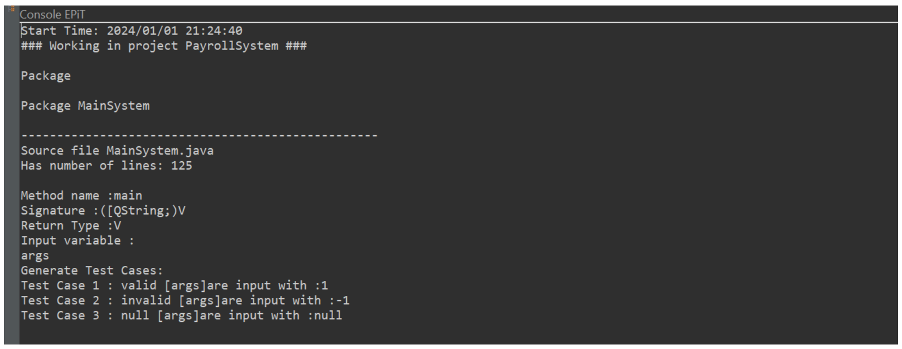
- This test case involves providing a valid input. In this case, the array has a length of 1. The purpose of this test might be to check the behavior of the method when it receives a valid and expected input.
- If the function is done correctly, there shouldn't be any issues with handling the valid input (array of length 1). Assuming that the test is successful in this instance, it can be concluded that the method is capable of processing valid input and responding accordingly.

Test Case 2 : Invalid are input with -1

- This test case tests the system's handling of invalid input. Specifically, it provides a negative value (-1) as the length of the array. The goal may be to assess how the system handles unexpected or invalid inputs and whether it gracefully manages such scenarios.
- The outcome of this test case may be expected to fail if the code is properly constructed to handle faulty inputs. By displaying error warnings or implementing the necessary correction steps, the system would be able to show that it detects and handles incorrect input instances in an acceptable manner.

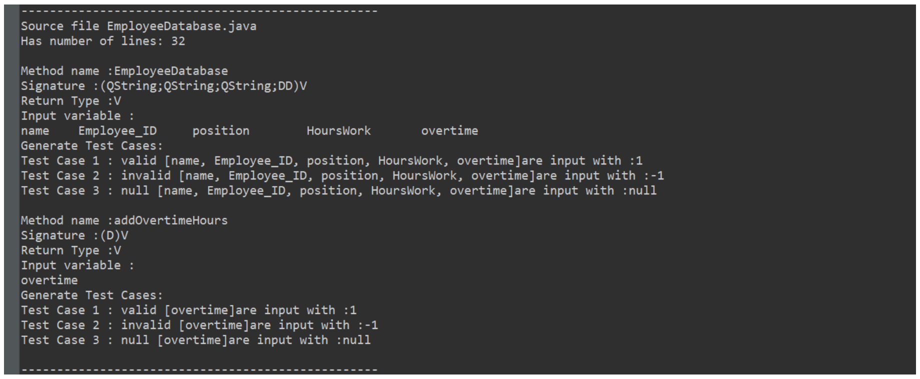
Test Case 3 : Null are input with null

- This test case explores the behavior of the method when it receives a null value. Testing with null inputs is crucial to ensure that the system can handle such cases without encountering runtime errors or unexpected behavior.
- When testing with null input, the outcome should be successful if the code is strong. The application's general stability should be ensured by the system's graceful handling of null input, which should not result in any unexpected behavior or crash.



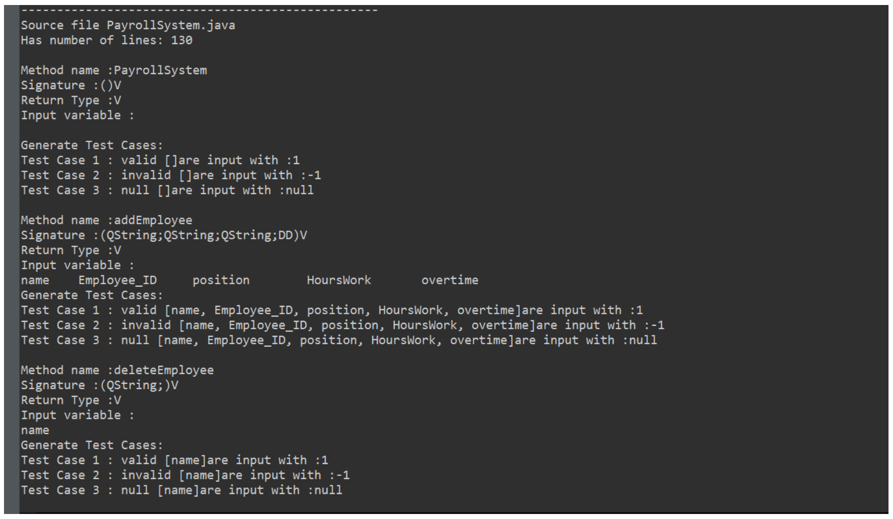
```
Start Time: 2024/01/01 21:24:40
### Working in project PayrollSystem ###

Package

Package MainSystem

--------------------------------------------------
Source file MainSystem.java
Has number of lines: 125

Method name :main
Signature :([QString;)V
Return Type :V
Input variable :
args
Generate Test Cases:
Test Case 1 : valid [args]are input with :1
Test Case 2 : invalid [args]are input with :-1
Test Case 3 : null [args]are input with :null
```

Figure 22 : Software test result for MainSystem

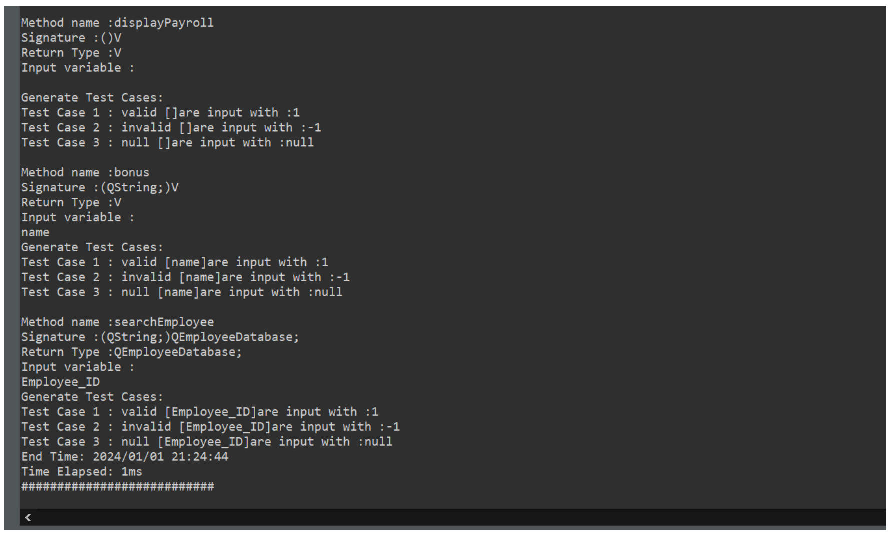


```
--------------------------------------------------
Source file EmployeeDatabase.java
Has number of lines: 32

Method name :EmployeeDatabase
Signature :(QString;QString;QString;DD)V
Return Type :V
Input variable :
name    Employee_ID     position        HoursWork       overtime
Generate Test Cases:
Test Case 1 : valid [name, Employee_ID, position, HoursWork, overtime]are input with :1
Test Case 2 : invalid [name, Employee_ID, position, HoursWork, overtime]are input with :-1
Test Case 3 : null [name, Employee_ID, position, HoursWork, overtime]are input with :null

Method name :addOvertimeHours
Signature :(D)V
Return Type :V
Input variable :
overtime
Generate Test Cases:
Test Case 1 : valid [overtime]are input with :1
Test Case 2 : invalid [overtime]are input with :-1
Test Case 3 : null [overtime]are input with :null

--------------------------------------------------
```

Figure 23 : Software test result for EmployeeDatabase



```
--------------------------------------------------
Source file PayrollSystem.java
Has number of lines: 130

Method name :PayrollSystem
Signature :()V
Return Type :V
Input variable :

Generate Test Cases:
Test Case 1 : valid []are input with :1
Test Case 2 : invalid []are input with :-1
Test Case 3 : null []are input with :null

Method name :addEmployee
Signature :(QString;QString;QString;DD)V
Return Type :V
Input variable :
name    Employee_ID     position        HoursWork       overtime
Generate Test Cases:
Test Case 1 : valid [name, Employee_ID, position, HoursWork, overtime]are input with :1
Test Case 2 : invalid [name, Employee_ID, position, HoursWork, overtime]are input with :-1
Test Case 3 : null [name, Employee_ID, position, HoursWork, overtime]are input with :null

Method name :deleteEmployee
Signature :(QString;)V
Return Type :V
Input variable :
name
Generate Test Cases:
Test Case 1 : valid [name]are input with :1
Test Case 2 : invalid [name]are input with :-1
Test Case 3 : null [name]are input with :null
```



```
Method name :displayPayroll
Signature :()V
Return Type :V
Input variable :

Generate Test Cases:
Test Case 1 : valid []are input with :1
Test Case 2 : invalid []are input with :-1
Test Case 3 : null []are input with :null

Method name :bonus
Signature :(QString;)V
Return Type :V
Input variable :
name
Generate Test Cases:
Test Case 1 : valid [name]are input with :1
Test Case 2 : invalid [name]are input with :-1
Test Case 3 : null [name]are input with :null

Method name :searchEmployee
Signature :(QString;)QEmployeeDatabase;
Return Type :QEmployeeDatabase;
Input variable :
Employee_ID
Generate Test Cases:
Test Case 1 : valid [Employee_ID]are input with :1
Test Case 2 : invalid [Employee_ID]are input with :-1
Test Case 3 : null [Employee_ID]are input with :null
End Time: 2024/01/01 21:24:44
Time Elapsed: 1ms
###########################
```

Figure 24 : Software test result for PayrollSystem

8.0 Conclusion

In conclusion, a well-designed payroll employee system is an important component of organizational administration since it ensures correct and timely salary for employees while conforming to legal and regulatory standards. It improves efficiency and reduces the possibility of mistakes by streamlining payroll operations. The capacity of the system to provide comprehensive reports and ease secure transactions, such as direct payments and payslip production, adds to efficient financial administration. Integration with other organizational systems expands its possibilities, resulting in a more smooth and comprehensive approach to worker pay. Finally, a payroll personnel system is critical to ensuring transparency, compliance, and general operational effectiveness inside a company.

9.0 References

[1] Resca, Y., & Munandar, A. (2022). Analysis of implementation: the payroll accounting system and employee wages. Fair Value: Jurnal Ilmiah Akuntansi dan Keuangan, 4(10), 4564-4570. https://www.journal.ikopin.ac.id/index.php/fairvalue/article/view/1756

[2] What is Unified Modeling Language (UML)? (2023). https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

[3] Myranda Mondry June 21, 2023, What is a payroll system? A beginner’s guide for 2023, QuickBooks Payroll
https://quickbooks.intuit.com/payroll/what-are-payroll-systems/