**UNIVERSITY OF GONDAR**

**COLLAGE OF INFORMATICS**

**DEPARTMENT OF COMPUTER SCIENCE**

**DATA STRACTURE AND ALGORITHM**

**Workforce Attendance & Payroll system.**

**GROUP PROJECT WORK**

**GROUP MEMBER ID**

**1, Adane dessie GUR/01669/15**

**2, Eyerusalem Tsegaye GUR/01575/15**

**3, Kalkidan Zemedkun GUR/01753/15**

**4, Ketsebaot Tariku GUR/01623/15**

**5, Kidest Chombe GUR/02368/15**

**6, Markos Ayiten GUR/02536/15**

**7, Mulatie Guadie GUR/03191/15**

**8, Tadese Belay GUR/02570/15**

**9, Melkamu Gebru GUR/02103/15**

**10. Shumet Yeserah GUR/02103/15**

**Submitted date:**

**Submitted to:**

**Table of Contents**

1. **Introduction**
2. **System Overview & Objectives**
3. **Employee Management Module**
4. **Attendance Management Module**
5. **Payroll Calculation Module**
6. **Code Structure & Implementation Details**
7. **User Interface & Menu Flow**
8. **Testing & Validation**
9. **Conclusion & Future Enhancements**
10. **References & Appendices**

**Introduction**

This document provides a detailed description of a C++-based Workforce Attendance & Payroll system. The system is designed to perform the following critical functions:

* **Employee Data Management:**  
  Add, search, update, delete, and display employee records using a fixed-size array.
* **Attendance Tracking:**  
  Record, update, delete, and display attendance data using a linked list structure.
* **Payroll Calculation:**  
  Compute employees’ salaries based on recorded attendance and their specified daily wages.

The system aims to offer a lightweight, console-based application suitable for small to medium-sized organizations. The design emphasizes simplicity, clarity, and efficient memory management.

**System Overview & Objectives**

**System Overview**

The Workforce Attendance & Payroll system is built using fundamental C++ programming concepts. It leverages:

* **Arrays** for managing and storing a limited number of employee records.
* **Linked Lists** for dynamic management of attendance records.
* **Control Structures** to navigate a multi-level menu that guides the user through various operations.

**Key Objectives**

1. **Data Integrity:**  
   Ensure that employee and attendance data are properly stored and can be updated or deleted as needed.
2. **User-Friendliness:**  
   Provide clear prompts and feedback for all operations within the system.
3. **Modularity:**  
   Separate concerns by distinguishing between employee management, attendance tracking, and payroll calculation.
4. **Efficiency:**  
   Utilize proper dynamic memory management for linked lists and ensure that operations (search, update, delete) are implemented efficiently.

**Employee Management Module**

**Description**

The Employee Management module handles all operations related to employee records. It uses an array (with a defined maximum of 100 employees) to store information such as:

* Employee ID
* Name
* Position
* Daily Wage

**Key Operations**

* **Add Employee:**  
  Collects user input for a new employee and appends it to the employee array. A counter tracks the number of active records.
* **Search Employee:**  
  Performs a linear search in the employee array to locate an employee record based on the employee ID.
* **Update Employee:**  
  Allows modifications to employee records. The system provides the original values as defaults during updates.
* **Delete Employee:**  
  Removes an employee’s record and shifts the remaining array elements to fill the gap, maintaining data continuity.
* **Display Employees:**  
  Prints a formatted list of all current employee records.

This module is responsible for ensuring that all aspects of employee data are maintained consistently throughout the system.

**Attendance Management Module**

**Description**

Attendance data is dynamic in nature. To accommodate variable numbers of records, a linked list is used. Each node in the list represents a single attendance record with:

* Employee ID
* Day (as an integer, representing the date)
* Attendance status (Present or Absent)

**Key Operations**

* **Add Attendance Record:**  
  Inserts a record at the head of the linked list to quickly register new attendance data.
* **Display Attendance for an Employee:**  
  Traverses the linked list to retrieve and display attendance records matching a specific employee ID.
* **Update Attendance Record:**  
  Modifies a record’s attendance status, based on a combination of employee ID and day.
* **Delete Attendance Record:**  
  Searches for and removes the desired attendance record. The system takes extra care to handle deletion at the start of the list (the head node).
* **Display All Attendance Records:**  
  Prints every attendance record present in the linked list for a detailed overview.

This module ensures that attendance tracking is dynamic and reflects real-time data modifications.

**Payroll Calculation Module**

**Description**

The payroll calculation component integrates data from both the employee array and the attendance linked list. It calculates the total salary for each employee by:

1. Counting the number of days the employee was marked as “present.”
2. Multiplying that count by the employee’s daily wage.

**Process Flow**

* **Step 1:** Iterate through each employee stored in the employee array.
* **Step 2:** For each employee, scan the entire attendance list.
* **Step 3:** For every attendance record where the employee was marked present, increment a counter.
* **Step 4:** Compute the salary by multiplying the number of days present by the employee’s daily wage.
* **Step 5:** Display the computed salary along with employee details.

This module automates the tedious calculation process inherent in payroll scenarios, enabling quick and accurate computation.

**Code Structure & Implementation Details**

**Overall Code Structure**

The program is divided into multiple logical sections:

1. **Data Structures:**  
   Defines the Employee structure for the array and the AttendanceNode for the linked list.
2. **Employee Operations:**  
   Contains functions for adding, searching, updating, deleting, and displaying employee records.
3. **Attendance Operations:**  
   Manages the linked list through functions for managing attendance records.
4. **Payroll Calculation:**  
   Integrates employee information and attendance records to calculate salaries.
5. **Main Menu:**  
   Provides a user interface loop that allows navigation between different modules.

**Implementation Highlights**

* **Memory Management:**  
  Attendance records are dynamically allocated with new and properly deleted after use to avoid memory leaks.
* **Input Validation:**  
  While basic error checking is incorporated, future iterations should refine user input handling.
* **Modular Design:**  
  The separation of concerns helps in maintaining and updating individual modules without affecting others.

**Development Considerations**

The programmer chose a procedural design with C++ constructs to keep the system simple and educational, focusing on fundamental data structures (arrays and linked lists) rather than advanced object-oriented techniques.

**Testing & Validation**

**Testing Strategy**

The system was tested through a series of manual tests on the following fronts:

* **Employee Module Tests:**  
  Verified that adding, searching, updating, deleting, and displaying employee records works as expected. Edge cases, such as attempting to update or delete a non-existent employee, were validated.
* **Attendance Module Tests:**  
  Confirmed that attendance records can be added, updated, or deleted correctly. Special focus was given to the handling of linked list nodes, including cases where the head node is modified or deleted.
* **Payroll Calculation Tests:**  
  Calculated payroll for various employee scenarios, ensuring that days present multiplied by daily wages yield accurate salary computations.

**Validation Methods**

* **Unit Testing:**  
  Individual functions were checked with sample data.
* **Integration Testing:**  
  The complete workflow—from employee registration to attendance recording and payroll computation—was tested.
* **Manual Testing:**  
  Due to the console-based nature of the application, manual testing was pivotal for verifying interactive menu options and system prompts.

**Issues & Resolutions**

* **Memory Leak Concerns:**  
  Dynamic memory allocation in the attendance module was audited, and proper deletion of nodes was implemented at program termination.
* **User Input Challenges:**  
  Although basic input extraction works, further enhancements are recommended for handling invalid input gracefully.

**Conclusion & Future Enhancements**

**Conclusion**

The Workforce Attendance & Payroll system serves as a robust example of applying core C++ programming concepts in a real-world scenario. It illustrates the integration of arrays and linked lists, dynamic memory management, and the development of an interactive console application to manage employee data and payroll processing.

**Future Enhancements**

1. **Transition to Object-Oriented Programming:**  
   Refactor the codebase to use classes and encapsulate data and functions for improved modularity and maintainability.
2. **Enhanced User Interface:**  
   Introduce graphical user interface elements or incorporate a web-based front end for a more modern user experience.
3. **Database Integration:**  
   Replace in-memory storage (arrays and linked lists) with a lightweight database (such as SQLite) to handle larger datasets.
4. **Advanced Error Handling:**  
   Implement more robust input validation and error handling strategies to ensure system stability.
5. **Additional Features:**  
   Consider incorporating features such as user authentication, detailed reporting, and data export capabilities for comprehensive payroll management.

**References:**

* + “The C++ Programming Language” by Bjarne Stroustrup
  + Various online C++ resources and documentation