Employee Attendance Tracker with Basic Analytics

A Project Report

ON

Employee Attendance Tracker with Basic Analytics

BY

M Arsal Kamran : BCPE243020

Shahroz Shahid Tarar : BCPE343030

Employee Attendance Tracker with Basic Analytics

Table of Contents

1. Introduction 3
2. Objective 3
3. System Design and Architecture 3
    3.1. Input Data 3
    3.2. Core Components 3
    3.3. Data Structures 4
    3.4. Flow of the Program 4
4. Implementation Details 4
    4.1. Data Structures 4
    4.2. Code Explanation 4
5. Features 7
6. Testing and Results 7
7. Challenges and Limitations 8
8. Conclusion 8

1. Introduction

This report outlines the design, development, and implementation of an **Employee Attendance Tracker with Basic Analytics**, which is built using **C++** programming language. The primary purpose of the system is to efficiently record, track, and manage employee attendance, while also providing basic analytics, such as calculating attendance percentage, number of present days, and absenteeism.

With businesses needing a reliable solution to track employee work hours and attendance, this project aims to fulfill that requirement with a system that is simple, user-friendly, and easy to maintain.

---

2. Objective

The main objective of this project is to create a C++ application that helps businesses or organizations to track the attendance of their employees. The functionalities include:

- Recording employee attendance as "Present" or "Absent" for a specified period.
- Storing attendance data for multiple employees.
- Generating basic analytics such as:
    - Total attendance
    - Total absenteeism
    - Attendance percentage
- Ensuring data persistence, using files for storing records.

---

3. System Design and Architecture

3.1. Input Data

To operate the system, the user provides input data that includes:

- **Employee ID:** A unique identifier for each employee.
- **Employee Name:** The name of the employee.
- **Attendance Status:** For each working day, the system allows the user to mark whether the employee is Present or Absent.

3.2. Core Components

The system is divided into three main components:

- **Employee Class:** Manages individual employee data, such as name, ID, and attendance records for each day.
- **Attendance Tracker:** Keeps track of attendance and generates reports.

- **Analytics Module:** Calculates and displays summary statistics based on attendance data.

3.3. Data Structures

The application uses:

- **Arrays/Vectors** for storing attendance data for each employee.
- **Classes/Structures** to manage employee records and attendance.

3.4. Flow of the Program

The flow of the system involves the following steps:

1. **Input Employee Data:** The program accepts the ID, name, and attendance for each employee.
2. **Track Attendance:** The system records attendance as "Present" or "Absent" for each employee.
3. **Display Attendance:** The user can view attendance records for any given employee.
4. **Generate Analytics:** The system computes the total present and absent days, as well as the attendance percentage.

---

4. Implementation Details

4.1. Data Structures

In the program, the **Employee** class stores the following data:

- **Employee ID**
- **Employee Name**
- **Attendance Records** (as a vector of bools, with true representing present and false representing absent).

4.2. Code Explanation

The key part of the program is the **Employee** class and its associated methods:

```
#include <iostream>

#include <vector>

#include <string>

#include <iomanip> // for setting precision

using namespace std;


// Structure to store employee data
```

```
struct Employee {
    int id;
    string name;
    int totalDays;   // Total working days
    int daysPresent; // Days the employee was present
};

// Function prototypes
void markAttendance(vector<Employee>& employees);
void viewAnalytics(const vector<Employee>& employees);

int main() {
    vector<Employee> employees = {
        {1, "Alice", 30, 0},
        {2, "Bob", 30, 0},
        {3, "Charlie", 30, 0},
    };

    int choice;

    do {
        // Main menu
        cout << "\n--- Employee Attendance Tracker ---\n";
        cout << "1. Mark Attendance\n";
        cout << "2. View Analytics\n";
        cout << "3. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            markAttendance(employees);
            break;
        case 2:
            viewAnalytics(employees);
            break;
        case 3:
            cout << "Exiting...\n";
            break;
        default:
            cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 3);

    return 0;
}

// Function to mark attendance for employees
void markAttendance(vector<Employee>& employees) {
    cout << "\nMark Attendance: \n";
    for (auto& employee : employees) {
        char present;
        cout << "Is " << employee.name << " (ID: " << employee.id << ") present? (y/n): ";
        cin >> present;
        if (tolower(present) == 'y') {
            employee.daysPresent++;
        }
    }
    cout << "Attendance marked successfully.\n";
}

// Function to display analytics
```

```
void viewAnalytics(const vector<Employee>& employees) {
    cout << "\n--- Attendance Analytics ---\n";
    cout << left << setw(10) << "ID" << setw(20) << "Name" << setw(15) << "Days Present" <<
"Attendance %\n";
    cout << string(50, '-') << "\n";

    for (const auto& employee : employees) {
        double attendancePercentage = (static_cast<double>(employee.daysPresent) /
employee.totalDays) * 100;
        cout << left << setw(10) << employee.id << setw(20) << employee.name << setw(15) <<
employee.daysPresent << fixed << setprecision(2) << attendancePercentage << "%\n";
    }
}

 }
```

5. Features

The key features of the **Employee Attendance Tracker** include:

- **Marking Attendance:** Allows the system administrator to mark the employee as present or absent.
- **Analytics Generation:** Computes attendance statistics such as total attendance, absenteeism, and attendance percentage.
- **Persistence:** Attendance records are saved to a file and can be reloaded for future use.

6. Testing and Results

The program was tested with multiple employees, and the results were verified manually. The output of the program correctly calculated the attendance days and generated accurate analytics.

Example of a test output:

```
Attendance for John Doe (ID: 101):

Present Absent Present Present Absent

Total Present Days: 3
```

Total Absent Days: 2

Attendance Percentage: 60%

---

7. Challenges and Limitations

- **File Handling Complexity:** Ensuring that attendance data was written and read correctly from files was a challenge, especially when managing multiple employees.
- **Scalability:** The program is designed for small-scale usage. Handling a large number of employees and generating reports for hundreds of individuals may require additional optimizations.
- **Lack of Advanced Analytics:** While the system includes basic analytics, it could be expanded to include more advanced features, such as overtime calculations or leave management.

---

8. Conclusion

The **Employee Attendance Tracker** built using **C++** successfully addresses the problem of tracking employee attendance in an efficient and simple manner. The system not only records attendance but also provides basic analytics such as attendance percentage and absenteeism. While the system is effective for smaller organizations, future improvements could involve database integration for scalability and more advanced reporting features.

1	Project Title	Employee Attendance Tracker with Basic Analytics		
2	Lab	CYG1611- Applications of Information and Communication Technologies Lab	Semester	Fall 2024
3	Student Name & Registration No.	Student 1 M Arsal Kamran BCPE243030	Student 2 Shahroz Shahid Tarar BCPE243030	
4	Instructor Name & Signature	Mr. SM Waqas Ayub Shah		

Project Demonstration

Assessment Criteria	Very Poor 0-1	Poor 2-3	Satisfactory 4-5	Good 6-8	Excellent 9-10	Score Student 1	Score Student 2	Score Student3
Design Evaluation	No or very poor design prototype and	Design prototype is not working	Design prototype is partially functional	Design prototype is functional and some	Design prototype is fully functional			

and Testing	**demonstration.**	**and no testing of design has been done**	**and little testing of design has been done.**	**testing of design has been done.**	**and design has been exhaustively tested.**			
Usage of software tools (Visual Studio, MS Office Applications) in design and evaluation	**No or very poor software tool (Visual Studio, MS Office Applications) usage in project design and results evaluation**	**Insignificant evidence of software tool (Visual Studio, MS Office Applications) usage in project design and results evaluation**	**Little evidence of ability to select appropriate software tools (Visual Studio, MS Office Applications), in project design and results evaluation**	**Some evidence of skills to use software tools (Visual Studio, MS Office Applications) in project design and results evaluation**	**Clear evidence of skills to use software tools (Visual Studio, MS Office Applications) in project design and results evaluation**			

Project Report

Assessment Criteria	**Very Poor** **0-1**	**Poor** **2**	**Satisfactory** **3**	**Good** **4**	**Excellent** **5**	**Score**	**Score**	**Score**

						Student 1	Student 2	Student3
Literature Survey, Problem Analysis and Design Procedure	No or very poor literature survey done. No problem analysis performed. No worthwhile design procedure exists.	Insufficient literature survey Problem analysis part is skipped or does not contribute to creating an effective design. Does not follow any design procedure.	Partial literature survey. Problem Analyses performed is haphazard and design parameter selection is spontaneous. Little use of design procedure.	Adequate literature survey. Problem analysis performed correctly. Project demonstrates some use of design process.	Clear and complete literature survey, effective problem analyses is performed to choose design parameters. Project demonstrates effective use of design process.			
Language, Grammar and References	A lot of spelling and grammatical mistakes with poor English. The list of references is clearly inadequate. Table of content missing.	Frequent spellings and grammatical errors. The list of references should be expanded.	Occasional spellings and grammatical errors. The list of references appears reasonable but citation does not follow standard format.	Very few spellings and grammatical errors. Organization is good. The list of references appears reasonable and citation follow standard format.	Almost no spelling or grammatical mistake. Excellent organization. A comprehensive list of references is cited using the standard format.			

<u>**Viva Voce**</u>

Assessment Criteria	Very Poor 0-1	Poor 2	Satisfactory 3	Good 4	Excellent 5	Score	Score	Score

						Student 1	Student 2	Student 3
Knowledge of Project Implementation details (Q/A)	**No or very poor knowledge of implementation and design process.**	**Poor knowledge of implementation and design with wrong/no answers**	**Satisfactory knowledge of implementation, vague answers**	**Adequate knowledge of project implementation with majority of correct answers**	**Exceptional knowledge of implementation and overall design with clear and spontaneous answers.**			