# 

# Attendance Tracking System

|  |  |  |
| --- | --- | --- |
| **TEAM MEMBERS(Group-13)** | | |
| **SL NO** | **NAME** | **REGD NO** |
| 1. | Ashirwad Shekhar | CL2025010601919829 |
| 2. | Apoorb Raj | CL20250106019422126 |
| 3. | Ashish Kumar | CL20250106018922104 |
| 4. | Aryabrat Sahoo | CL2025010601919728 |
|  | Paritosh Kumar | CL2025010601884426 |

# Introduction

The Attendance Tracking System is a console-based application written in C that simulates a basic attendance management experience. The system utilizes a CSV file named Attendance.csv—which stores student names along with their attendance status—to dynamically load and update records. Users can mark attendance, view detailed attendance reports, and update or correct entries as needed. This project served as a practical learning exercise to deepen understanding of fundamental C programming concepts and to explore the integration of file handling in procedural programming.

# Problems with Existing Attendance Tracking Systems

Before developing our solution, several issues were identified with traditional attendance tracking systems:

* Limited Data Management: Many existing systems lacked dynamic methods for managing and updating attendance records. Attendance data was often hardcoded or manually maintained, leading to data redundancy and inconsistency.
* Poor User Interaction: Conventional systems sometimes suffered from a non-intuitive user interface, making it difficult for users to mark or review attendance records efficiently.
* Inadequate Error Handling: Early implementations offered minimal validation for user input, which could result in corrupted data entries or unexpected system behavior.
* Scalability Challenges: Without proper file handling or data management strategies, scaling the system to handle larger groups (e.g., multiple classes or sessions) was often impractical.

# Solution with Our Attendance Tracking System

Our system addresses these challenges through the following features:

* Dynamic Data Loading: Attendance records are stored in a CSV file (Attendance.csv), allowing for simple and flexible updates. The system reads and writes to this file at runtime to keep records current.
* Modular Object-Oriented Design: The system leverages a structured procedural approach, organizing functionalities like marking attendance, viewing reports, and updating records into distinct, reusable functions.
* Efficient Data Structures: Arrays and structures are used to manage attendance data in memory, enabling fast lookups and easy updates for individual students.
* Robust Input and Error Handling: The application includes input validation to ensure proper menu selections, checks for file availability before accessing data, and manages invalid or duplicate entries gracefully.
* Interactive Menu System: A user-friendly, menu-driven interface guides users through core actions such as marking attendance, viewing daily reports, and editing previous entries.

These features combine to offer a reliable, intuitive, and scalable solution that improves upon traditional attendance tracking methods.

# Libraries Used and Their Descriptions

The following C++ libraries are integral to the functionality of the system:

* iostream: Used for standard input and output operations, enabling communication with the user through the console.
* string: Provides support for string manipulation, which is essential for handling product names and file input.
* limits: Facilitates handling of numeric limits, especially useful for clearing input errors and validating user data.
* vector: Offers dynamic array functionality, used here to manage the list of products in the shopping cart efficiently.
* fstream: Enables file input/output operations, crucial for reading the product data from the Products.csv file.
* sstream: Allows for string stream operations, which are used to parse individual lines of the CSV file into product attributes.

Each library plays a critical role in ensuring the system operates smoothly while reinforcing core C++ programming concepts.

# Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX\_STUDENTS 100

#define MAX\_NAME\_LENGTH 50

#define MAX\_REG\_LENGTH 20

struct Student {

    char name[MAX\_NAME\_LENGTH];

    char reg\_no[MAX\_REG\_LENGTH];

    char status;

};

struct Student students[MAX\_STUDENTS];

int student\_count = 0;

void load\_students() {

    FILE \*file = fopen("students.csv", "r");

    if (file == NULL) {

        perror("\nError opening students.csv");

        exit(EXIT\_FAILURE);

}

    char line[100];

    while (fgets(line, sizeof(line), file)) {

        char \*name = strtok(line, ",");

        char \*reg\_no = strtok(NULL, ",\n");

        if (name && reg\_no) {

            strncpy(students[student\_count].name, name, MAX\_NAME\_LENGTH);

            strncpy(students[student\_count].reg\_no, reg\_no, MAX\_REG\_LENGTH);

            students[student\_count].status = 'A'; // Default to absent

            student\_count++;

            if (student\_count >= MAX\_STUDENTS) {

                fprintf(stderr, "\nMaximum student capacity reached.");

                break;

            }

        }

    }

    fclose(file);

}

void load\_attendance() {

    FILE \*file = fopen("attendance.csv", "r");

if (file == NULL) return;

    char line[100];

    while (fgets(line, sizeof(line), file)) {

        char \*reg\_no = strtok(line, ",");

        char \*status = strtok(NULL, ",\n");

        if (reg\_no && status) {

            for (int i = 0; i < student\_count; i++) {

                if (strcmp(students[i].reg\_no, reg\_no) == 0) {

                    students[i].status = toupper(status[0]);

                    break;

                }

            }

        }

    }

    fclose(file);

}

void save\_attendance() {

    FILE \*file = fopen("attendance.csv", "w");

    if (file == NULL) {

        perror("\nError saving attendance data");

        return;

}

    for (int i = 0; i < student\_count; i++) {

        fprintf(file, "%s,%c\n", students[i].reg\_no, students[i].status);

    }

    fclose(file);

}

void show\_students() {

    printf("\n%-20s %-15s\n", "Name", "Registration No");

    printf("----------------------------------------\n");

    for (int i = 0; i < student\_count; i++) {

        printf("%-20s %-15s\n", students[i].name, students[i].reg\_no);

    }

}

void mark\_attendance() {

    char reg\_no[MAX\_REG\_LENGTH];

    printf("\nEnter registration number: ");

scanf("%s", reg\_no);

    int found = -1;

    for (int i = 0; i < student\_count; i++) {

        if (strcmp(students[i].reg\_no, reg\_no) == 0) {

            found = i;

            break;

        }

}

    if (found == -1) {

        printf("Student not found!\n");

        return;

}

    char status;

    printf("Enter status (P/A): ");

    scanf(" %c", &status);

status = toupper(status);

    if (status != 'P' && status != 'A') {

        printf("Invalid status! Use P or A.\n");

        return;

}

    students[found].status = status;

    printf("Attendance updated for %s.\n", students[found].name);

}

void show\_summary() {

    int present = 0, absent = 0;

    for (int i = 0; i < student\_count; i++) {

        students[i].status == 'P' ? present++ : absent++;

    }

    printf("\nAttendance Summary:\n");

    printf("Present: %d\nAbsent: %d\n", present, absent);

}

void show\_student\_attendance() {

    char reg\_no[MAX\_REG\_LENGTH];

    printf("\nEnter registration number: ");

scanf("%s", reg\_no);

    for (int i = 0; i < student\_count; i++) {

        if (strcmp(students[i].reg\_no, reg\_no) == 0) {

            printf("\nStudent Details:\n");

            printf("Name: %s\nRegistration No: %s\nStatus: %s\n",

                   students[i].name,

                   students[i].reg\_no,

                   students[i].status == 'P' ? "Present" : "Absent");

            return;

        }

    }

    printf("Student not found!\n");

}

int main() {

    load\_students();

load\_attendance();

    int choice;

    do {

        printf("\nAttendance Tracking System\n");

        printf("1. Show Students List\n");

        printf("2. Mark Attendance\n");

        printf("3. Show Summary\n");

        printf("4. Show Student Attendance\n");

        printf("5. Exit\n");

        printf("Enter your choice: ");

        if (scanf("%d", &choice) != 1) {

            while (getchar() != '\n');

            printf("Invalid input. Please enter a number.\n");

            continue;

        }

        switch (choice) {

            case 1: show\_students(); break;

            case 2: mark\_attendance(); break;

            case 3: show\_summary(); break;

            case 4: show\_student\_attendance(); break;

            case 5: save\_attendance(); break;

            default: printf("Invalid choice! Try again.\n");

        }

} while (choice != 5);

    printf("\nExiting system...\n");

    return 0;

}

# Future Scope

While the current system effectively showcases the core functionalities of an attendance tracking tool, several enhancements can be explored to increase its robustness, scalability, and usability:

* Graphical User Interface (GUI): Shifting from a console-based interface to a GUI-based system would significantly improve the user experience and make the system more accessible for educational institutions and organizations.
* Enhanced Exception Handling: Incorporating more sophisticated try-catch mechanisms and custom exceptions could further improve robustness, especially in handling file I/O errors.
* Extended Student and Session Management: Future versions can include more student attributes (such as ID, class, section, and attendance percentage) and support session-wise attendance tracking with options for date-wise filtering.
* Database Integration: Replacing the CSV-based approach with a database (e.g., SQLite or MySQL) would enhance data integrity, provide real-time access, and support multi-user environments.
* Security Features: Adding user authentication for teachers or administrators would enhance system security, ensuring that only authorized personnel can modify attendance records.
* Report Generation: The ability to generate and export comprehensive attendance reports in PDF or Excel format would be beneficial for academic use and record-keeping.

# Conclusion

The development of the Attendance Tracking System in C has served as a valuable learning experience by applying fundamental concepts of procedural programming, file handling, and structured design. By overcoming the limitations of traditional attendance systems—such as static data management and minimal user feedback—the project demonstrated how a simple yet efficient system can be built using basic C programming.

Through the use of structured code, modular functions, file operations, and user interaction techniques, this project has laid a strong foundation for more advanced system design in the future. It not only solidified core programming concepts but also opened up opportunities for further innovation and expansion into more robust attendance management solutions.