De La Salle University – Manila Campus

Term 1, A.Y. 2023 – 2024

In Partial Fulfillment of the Requirements of the

Course Computer Fundamentals and Programming 2 (LBYEC2B)

“Student Attendance System”

Submitted by:

[illegible]

Reyes, Edward Sam M.

Section

EB4

Submitted to:

Mr. Ramon Stephen Ruiz

I. Introduction

The necessity for an effective and dependable Student Attendance System has grown more important in today's educational scene. This system attempts to simplify the time-consuming chore of tracking student attendance by providing teachers with a powerful tool for easily managing student attendance data.

Traditional manual procedures are frequently time-consuming, error-prone, and incapable of producing insightful data. The Student Attendance System remedies these flaws by providing a computerized solution. It is designed to help teachers collect, analyze, and manage attendance data.

The implementation of the Student Attendance System offers not just effective attendance tracking but also actionable data to improve teaching approaches and increase student involvement.

II. Objectives

The objective of the project is to create a reliable Student Attendance System, focusing on achieving three primary aims:

1. Facilitating instructors in effortlessly inputting attendance records.

2. Gathering valuable insights regarding attendance trends, classroom participation, and average attendance.

3. Simplifying the processes involved in adding, modifying, and removing attendance data.

III. Related Work

Various solutions in educational technology strive to simplify student attendance management. These systems include automatic attendance recording, user-friendly interfaces, and data analytic tools for analyzing attendance patterns and student involvement.

Existing solutions are focused on streamlining teacher attendance processes, giving insights through data analysis, and delivering detailed reports. By automating attendance tasks and providing useful information into student participation, these technologies attempt to improve teaching efficiency.

By incorporating user-friendly features, automated analysis, and effective attendance management, the Student Attendance System matches with these trends. It adds to this subject by resolving the constraints of manual methods and adhering to current educational technology practices.

IV. Methodology

The development of the Student Attendance System follows a structured process:

1. Requirement Analysis: Understanding the system's needs, including features like data entry, attendance pattern generation, and ease of record management for instructors.

2. Design: Creating an easy-to-use system by developing simple functionalities based on the established functionality.

3. Development: Translating the design into functional code for the system's features.

4. Testing: Ensuring system functionality through comprehensive tests such as unit, integration, and system testing.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#define MAX_STUDENTS 100
#define MAX_DAYS 7 // Maximum number of days for attendance

struct Date {
  int day;
  int month;
  int year;
};

struct AttendanceRecord {
  char studentName[50];
  struct Date dates[MAX_DAYS];
  int isPresent[MAX_DAYS];
  int totalDays;
};

struct StudentAttendanceSystem {
  struct AttendanceRecord attendanceRecords[MAX_STUDENTS];
  int recordCount;
};

void addAttendanceRecord(struct StudentAttendanceSystem *attendanceSystem, const
char *studentName) {
  if (attendanceSystem->recordCount >= MAX_STUDENTS) {
    printf("Maximum student limit reached!\n");
    return;
  }

  struct AttendanceRecord *record = &(attendanceSystem-
>attendanceRecords[attendanceSystem->recordCount]);
```

```
    strcpy(record->studentName, studentName);
    memset(record->isPresent, -1, sizeof(record->isPresent));
    record->totalDays = 0;
    attendanceSystem->recordCount++;
}

void markAttendance(struct StudentAttendanceSystem *attendanceSystem, const char
*studentName) {
    if (attendanceSystem->recordCount == 0) {
        printf("No students added yet!\n");
        return;
    }

    // Print all stored student names for debugging purposes
    printf("List of Students:\n");
    for (int i = 0; i < attendanceSystem->recordCount; ++i) {
        printf("%d. %s\n", i + 1, attendanceSystem->attendanceRecords[i].studentName);
    }

    printf("Enter date (DD MM YYYY): ");
    struct Date currentDate;
    scanf("%d %d %d", &currentDate.day, &currentDate.month, &currentDate.year);

    int studentIndex = -1;
    for (int i = 0; i < attendanceSystem->recordCount; ++i) {
        if (strcmp(attendanceSystem->attendanceRecords[i].studentName, studentName) ==
0) {
            studentIndex = i;
            break;
        }
    }
    if (studentIndex == -1) {
        printf("Student not found!\n");
        return;
    }

    struct AttendanceRecord *record = &(attendanceSystem-
>attendanceRecords[studentIndex]);
    record->dates[record->totalDays] = currentDate;
    printf("Mark attendance for %s on %d/%d/%d (1 for present, 0 for absent): ",
studentName, currentDate.day, currentDate.month, currentDate.year);
    scanf("%d", &record->isPresent[record->totalDays]);
    record->totalDays++;
}

void displayStudentAttendance(const struct StudentAttendanceSystem
*attendanceSystem) {
    printf("List of Students with Attendance:\n");
    printf("--------------------------------\n");
    for (int i = 0; i < attendanceSystem->recordCount; ++i) {
        printf("Student: %s\n", attendanceSystem->attendanceRecords[i].studentName);
        printf("Attendance:\n");
        for (int j = 0; j < attendanceSystem->attendanceRecords[i].totalDays; ++j) {
            printf("%d/%d/%d - %s\n", attendanceSystem->attendanceRecords[i].dates[j].day,
                   attendanceSystem->attendanceRecords[i].dates[j].month,
                   attendanceSystem->attendanceRecords[i].dates[j].year,
                   attendanceSystem->attendanceRecords[i].isPresent[j] == 1 ? "Present" :
"Absent");
        }
        printf("--------------------------------\n");
    }
```

```
}

void editAttendanceRecord(struct StudentAttendanceSystem *attendanceSystem, const
char *studentName) {
    if (attendanceSystem->recordCount == 0) {
        printf("No students added yet!\n");
        return;
    }

    printf("Enter date (DD MM YYYY) to edit attendance: ");
    struct Date editDate;
    scanf("%d %d %d", &editDate.day, &editDate.month, &editDate.year);

    int studentIndex = -1;
    for (int i = 0; i < attendanceSystem->recordCount; ++i) {
        if (strcmp(attendanceSystem->attendanceRecords[i].studentName, studentName) ==
0) {
            studentIndex = i;
            break;
        }
    }

    if (studentIndex == -1) {
        printf("Student not found!\n");
        return;
    }

    struct AttendanceRecord *record = &(attendanceSystem-
>attendanceRecords[studentIndex]);

    int dayIndex = -1;
    for (int j = 0; j < record->totalDays; ++j) {
        if (record->dates[j].day == editDate.day && record->dates[j].month ==
editDate.month && record->dates[j].year == editDate.year) {
            dayIndex = j;
            break;
        }
    }

    if (dayIndex == -1) {
        printf("Attendance for specified date not found!\n");
        return;
    }

    printf("Mark attendance for %s on %d/%d/%d (1 for present, 0 for absent): ",
studentName, editDate.day, editDate.month, editDate.year);
    scanf("%d", &record->isPresent[dayIndex]);
}

void deleteAttendanceRecord(struct StudentAttendanceSystem *attendanceSystem, const
char *studentName) {
    for (int i = 0; i < attendanceSystem->recordCount; ++i) {
        if (strcmp(attendanceSystem->attendanceRecords[i].studentName, studentName) ==
0) {
            for (int j = i; j < attendanceSystem->recordCount - 1; ++j) {
                attendanceSystem->attendanceRecords[j] = attendanceSystem-
>attendanceRecords[j + 1];
            }
            attendanceSystem->recordCount--;
            printf("Attendance record deleted for: %s\n", studentName);
            return;
        }
    }
    printf("Student not found!\n");
}

void deleteAttendanceForDate(struct StudentAttendanceSystem *attendanceSystem,
const char *studentName) {
    if (attendanceSystem->recordCount == 0) {
        printf("No students added yet!\n");
        return;
    }

    printf("Enter date (DD MM YYYY) to delete attendance: ");
    struct Date deleteDate;
    scanf("%d %d %d", &deleteDate.day, &deleteDate.month, &deleteDate.year);

    int studentIndex = -1;
    for (int i = 0; i < attendanceSystem->recordCount; ++i) {
        if (strcmp(attendanceSystem->attendanceRecords[i].studentName, studentName) ==
0) {
            studentIndex = i;
            break;
        }
    }

    if (studentIndex == -1) {
        printf("Student not found!\n");
        return;
    }

    struct AttendanceRecord *record = &(attendanceSystem-
>attendanceRecords[studentIndex]);
```

```
    int dayIndex = -1;
    for (int j = 0; j < record->totalDays; ++j) {
        if (record->dates[j].day == deleteDate.day && record->dates[j].month ==
deleteDate.month && record->dates[j].year == deleteDate.year) {
            dayIndex = j;
            break;
        }
    }

    if (dayIndex == -1) {
        printf("Attendance for specified date not found!\n");
        return;
    }

    // Shift array elements to remove the entry for the specified date
    for (int k = dayIndex; k < record->totalDays - 1; ++k) {
        record->dates[k] = record->dates[k + 1];
        record->isPresent[k] = record->isPresent[k + 1];
    }
    record->totalDays--;
    printf("Attendance for %d/%d/%d deleted for %s\n", deleteDate.day,
deleteDate.month, deleteDate.year, studentName);
}

void generateReport(const struct StudentAttendanceSystem *attendanceSystem) {
    printf("Attendance Report:\n");
    printf("------------------\n");
    for (int i = 0; i < attendanceSystem->recordCount; ++i) {
        int daysPresent = 0;
        int daysAbsent = 0;
        for (int j = 0; j < attendanceSystem->attendanceRecords[i].totalDays; ++j) {
            if (attendanceSystem->attendanceRecords[i].isPresent[j] == 1) {
                daysPresent++;
            } else {
                daysAbsent++;
            }
        }

        double percentageAttendance = (double)daysPresent / attendanceSystem-
>attendanceRecords[i].totalDays * 100;

        printf("Student: %s\n\n", attendanceSystem->attendanceRecords[i].studentName);
        printf("Days Present: %d\n", daysPresent);
        printf("Days Absent: %d\n\n", daysAbsent);
        printf("Percentage of Attendance: %.2f%%\n", percentageAttendance);
        printf("--------------------------------\n");
    }

    int totalDaysPresent = 0;
    int totalDaysAbsent = 0;
    int totalStudents = attendanceSystem->recordCount;

    for (int i = 0; i < attendanceSystem->recordCount; ++i) {
        for (int j = 0; j < attendanceSystem->attendanceRecords[i].totalDays; ++j) {
            if (attendanceSystem->attendanceRecords[i].isPresent[j] == 1) {
                totalDaysPresent++;
            } else {
                totalDaysAbsent++;
            }
        }
    }

    double totalAttendance = (double)totalDaysPresent / (totalDaysPresent +
totalDaysAbsent) * 100;

    printf("Total Days Present: %d\n", totalDaysPresent);
    printf("Total Days Absent: %d\n\n", totalDaysAbsent);
    printf("Percentage of All Students' Attendance: %.2f%%\n", totalAttendance);
    printf("-----------------\n");
}

int main() {
    struct StudentAttendanceSystem attendanceSystem;
    attendanceSystem.recordCount = 0;

    printf("Welcome to the Student Attendance System\n");

    while (1) {
        printf("\n1. Add Student\n2. Display Attendance Patterns\n3. Mark Attendance\n4.
Edit Attendance\n5. Delete Attendance Record\n6. Delete Attendance for a Date\n7.
Generate Report\n8. Exit\n");
        printf("\nEnter your choice: ");

        int choice;
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                if (attendanceSystem.recordCount >= MAX_STUDENTS) {
                    printf("Maximum student limit reached!\n");
                    break;
    }

    printf("Enter student name: ");
    char name[50];
    scanf(" %[^\n]", name);
    addAttendanceRecord(&attendanceSystem, name);
    printf("Student added successfully!\n");
    break;

case 2:
    displayStudentAttendance(&attendanceSystem);
    break;

case 3:
    printf("Enter student name to mark attendance: ");
    char markName[50];
    scanf(" %[^\n]", markName);
    markAttendance(&attendanceSystem, markName);
    break;

case 4:
    printf("Enter student name to edit attendance: ");
    char editName[50];
    scanf(" %[^\n]", editName);
    editAttendanceRecord(&attendanceSystem, editName);
    break;

case 5:
    printf("Enter student name to delete: ");
    char deleteName[50];
    scanf(" %[^\n]", deleteName);
```

```
                deleteAttendanceRecord(&attendanceSystem, deleteName);
                break;

            case 6:
                printf("Enter student name to delete attendance for a date: ");
                char deleteDateName[50];
                scanf(" %[^\n]", deleteDateName);
                deleteAttendanceForDate(&attendanceSystem, deleteDateName);
                break;

            case 7:
                generateReport(&attendanceSystem);
                break;

            case 8:
                printf("Exiting program. Goodbye!\n");
                exit(0);

            default:
                printf("Invalid choice! Please enter a valid option.\n");
                break;
        }
    }

    return 0;
}
```

V. Results and Discussion

The implementation of the Student Attendance System considerably simplifies attendance monitoring within the educational setting by offering effective recording capabilities and basic analytical functions, leading to simplified attendance management.

The use of this technology dramatically lowers the time and effort teachers spend on attendance management by replacing human tracking with automated operations. The automated data analysis tools create attendance patterns and give important insight into class engagement, allowing instructors to make educated teaching decisions.

Despite its text-based interface, the system is user-friendly, allowing educators simple navigation and accessibility. Instructors may create detailed reports for a complete examination of attendance patterns and student involvement thanks to the reporting features.

VI. Conclusion and Future Work

The Student Attendance System simplifies attendance management by providing an easy-to-use interface for data entry and automatic attendance analysis of patterns. It makes attendance record management easier for teachers, allowing them to focus more on teaching while also maintaining accurate data capture.

Future additions might include including a more user-friendly graphical interface to improve accessibility, particularly for those who are unfamiliar with text-based systems.

In conclusion, the Student Attendance System streamlines attendance monitoring while giving critical insights into student participation. Its reduced functionality and automated analysis make it a potential tool for refining teaching methods and improving academic performance.

VII. Contributions

Reyes, Edward Sam

- Planning
- Project Proposal
- Coding
- Project Documentation

VIII. References

GeeksforGeeks. (2023, September 11). Examination Management System in C. GeeksforGeeks. https://www.geeksforgeeks.org/examination-management-system-in-c/

Student Management System Project in C programming language. insideTheDiv. (n.d.). https://www.insidethediv.com/student-management-system-in-c

C++ project on attendance management with source code. CppBuzz. (n.d.). https://www.cppbuzz.com/projects/c++/c++-project-on-attendance-management