**MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING**

*Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution*

*(A Unit of Rajalaxmi Education Trust®, Mangalore - 575001)*

*Affiliated to V.T.U., Belagavi, Approved by AICTE, New Delhi*

**TECHNICAL TRAINING PROJECT**

**Group-CP047**

**TOPIC : STUDENT ATTENDANCE MANAGEMENT SYSTEM**

**GROUP MEMBERS:**

|  |  |
| --- | --- |
| 4MT22CS417 | UMARUL KATHAD |
| 4MT22CS406 | KAVYANJALI |
| 4MT22CS416 | SURAKSHA |
| 4MT20CS046 | SHAMNEER |
| 4MT22CS172 | THAMEEM |

**TABLE OF CONTENTS**

**Problem Statement**

**1.Abstract**

**2.Introduction**

**2.1 Background**

**2.2 Objectives**

**3.Technologies Used**

**4.System Architecture**

**4.1 Front-End**

**4.2 Back-End**

**4.3 Database**

**5.Project Modules**

**5.1 Module 1: User Authentication**

**5.2 Module 2: Menu Handling**

**5.3 Module 3: Element Manipulation**

**6.Design and Implementation**

**6.1 Front-End Design**

**6.2 Back-End Design**

**6.3 Database Design**

**7.Features and Functionality**

**7.1 Feature 1: User Login**

**7.2 Feature 2: Menu Navigation**

**7.3 Feature 3: Element Management**

**8.Testing**

**8.1 Unit Testing**

**8.2 Integration Testing**

**8.3 User Acceptance Testing**

**9.Challenges Faced**

**10.Future Enhancements**

**11.Conclusion**

**12.References**

**13.Appendices**

**13.1 Screenshots**

**ABSTRACT:**

Today's educational landscape, efficient attendance tracking is essential for institutions to monitor student participation and engagement. This project aims to develop a robust and user-friendly system for storing student attendance across various classes and courses. By leveraging modern technologies and an intuitive user interface, this system streamlines the process of recording and managing attendance data, providing educators with valuable insights into student attendance patterns.

The system's objectives include creating a centralized platform accessible to teachers and administrators, facilitating the recording of attendance using digital methods, and generating comprehensive reports for better decision-making. This project will utilize a combination of front-end and back-end technologies to ensure a seamless user experience, along with a robust database system to store attendance records securely.

Through a modular approach, this project will break down the development process into manageable units, allowing for systematic design and implementation. Each module will address specific aspects of the system, from user interface design to database structure, ensuring a well-structured and scalable solution.

Key features and functionality will encompass real-time attendance tracking, the ability to associate attendance data with specific classes and courses, and customizable reporting options. Comprehensive testing, including unit testing, integration testing, and user acceptance testing, will be conducted to ensure the system's reliability and accuracy.

Throughout the development journey, challenges encountered and valuable lessons learned will be documented, providing insights into the implementation of attendance tracking systems in educational institutions. Future enhancements will be explored, such as integration with existing student information systems and the incorporation of advanced analytics for predictive attendance modeling.

**Introduction:**

In today's educational landscape, efficiently managing student attendance across various classes and courses is a crucial task for educational institutions. Traditional paper-based methods are not only time-consuming but also prone to errors. To address these challenges, we propose the development of a modern and robust "Student Attendance Management System."

**Project Overview:**

The Student Attendance Management System (SAMS) is a comprehensive software solution designed to simplify and streamline the process of recording and tracking student attendance. It will be a web-based application accessible by administrators, teachers, and students, ensuring seamless attendance management.

**Key Features:** User-Friendly Interface: SAMS will feature an intuitive user interface that makes it easy for teachers and administrators to record and access attendance data.

**Multi-Class Support:**  The system will support multiple classes and courses simultaneously, accommodating the diverse needs of educational institutions.

**Real-Time Attendance Tracking:** Teachers can mark attendance for each class in real-time, allowing for immediate updates and accurate records.

**Student and Parent Access:** Students and their parents or guardians will have limited access to view their attendance records, fostering transparency and accountability.

**Reporting and Analytics:** SAMS will generate attendance reports and analytics, providing valuable insights for academic planning and intervention.

**Automated Notifications:** Automated alerts and notifications will be sent to students and parents for unexcused absences or low attendance, promoting engagement.

**Security and Data Privacy:** The system will prioritize the security of student data, ensuring compliance with data protection regulations.

**Benefits:**

**Efficiency:** SAMS will reduce the administrative burden on educators and improve overall efficiency in attendance management.

**Accuracy:** The system will minimize errors associated with manual attendance recording, leading to more reliable data.

**Engagement:** By involving students and parents in the attendance tracking process, SAMS encourages active participation and accountability.

Data-Driven Decision Making: The availability of attendance analytics will empower educational institutions to make informed decisions about curriculum and student support.

**Project Background:**

Educational institutions, from schools to universities, often need an efficient way to track student attendance.

Manual attendance tracking can be time-consuming and prone to errors.

An automated attendance system can streamline this process, providing accurate records and saving time.

**Project Objectives:**

Automated Attendance Tracking: Develop a system that can automatically record student attendance for various classes and courses.User-Friendly Interface: Create an easy-to-use interface for teachers and administrators to input and access attendance data.

**Course and Class Management:** Implement features to manage different courses, classes, and academic terms within the system.

**Real-time Updates**: Ensure that attendance data is updated in real-time, allowing for immediate access to records.

**Reporting and Analytics**: Generate comprehensive reports on attendance patterns, trends, and individual student records.

**Security and Privacy**: Implement security measures to protect sensitive student information and ensure compliance with data privacy regulations.

**Integration:** Allow integration with existing student information systems if applicable.

**Scalability:** Design the system to accommodate future growth and additional features.

**User Training and Support:** Provide training resources and support for users to effectively utilize the attendance system.

**Feedback and Improvement:** Continuously gather feedback from users to make improvements and enhancements to the system.

**Documentation:** Maintain detailed documentation for system administrators and users.

**Cost-Efficiency:** Ensure that the system offers cost savings compared to manual attendance tracking methods.

Remember that this is a high-level outline, and the specifics of your project may vary based on the educational institution's requirements and available resources. It's essential to engage stakeholders, conduct a thorough needs analysis, and define a detailed project plan to successfully develop and implement the attendance system.

**Background:** This section can explain why the program was developed. For example, it could discuss the need for a student attendance management system.

**Objectives:** The objectives of the program can be mentioned here, such as managing student data and attendance efficiently.

**Technologies Used:** This section can list the technologies used in the program, which include C, standard libraries, and the use of time functions from the time.h library.

**System Architecture:** The code doesn't explicitly discuss system architecture, but you can mention that it's a console-based application, and the architecture mainly consists of a front-end (console interface), back-end (functions handling data), and a simple data storage structure (the struct student array).

**Front-End:** The console interface for user interaction.

**Back-End:** The functions handling student data and attendance.

**Database:** While not a traditional database, the student records are stored in memory using the struct student array.

**Project Modules:** This code represents the core functionality of managing student data and attendance. You can add a section that describes how the code is organized into functions and modules.

**Module 1:** Description of the student management module, including adding, removing, and marking attendance.

**Design and Implementation:**

**Front-End Design:** This can describe the console menu and how users interact with the program.

**Back-End Design:** Explain how the data is stored in the struct student array and manipulated by functions.

**Database Design**: Discuss the structure of the struct student used to store student data.

Features and Functionality:

**Feature 1:** Description of adding students.

**Feature 2:** Description of removing students.

**Feature 3:** Description of marking attendance.

**Feature 4:** Description of printing a report.

**Testing:**

**Unit Testing:** Explain how each function can be tested in isolation to ensure it works correctly.

**Integration Testing:** Discuss how different functions interact and are tested together.

**User Acceptance Testing:** Mention that users can interact with the program to verify its functionality.

**Challenges Faced:** Describe any challenges encountered during development, such as handling user input or managing student records.

**Future Enhancements:** Discuss potential improvements or additional features that could be added to the program, such as saving data to a file or providing a graphical user interface (GUI).

**Conclusion:** Summarize the project, its objectives, and key achievements. Mention any lessons learned during development.

**References:** If you used external resources or libraries, list them here.

**Appendices:** Include any additional materials, such as screenshots (mentioned in "Screenshots") or relevant code snippets (mentioned in "Code Snippets").

Incorporate the code and its explanations into the appropriate sections of your document following this table of contents structure to create a comprehensive report on your student attendance management program.

**Student Attendance Management System**

This C program is designed to manage a list of students and their attendance records. It provides the following functionalities:

**1.Add Student**: Allows you to add a new student to the list with their name, roll number, and course.

**2.Remove Student**: Removes a student from the list based on their roll number.

**3.Mark Attendance**: Marks attendance for a student based on their roll number (1 for present, 0 for absent).

**4.Print Report**: Displays a report of all students including their name, roll number, course, and attendance status.

**5.Exit**: Exits the program

**How to Use**

1.Compile the program using a C compiler (e.g., GCC).

2.Run the compiled executable.

3.Choose options from the menu to perform actions on the student list.

4.Follow the on-screen prompts to input data or make selections

**Sample Usage**

1.Add student

2.Remove student

3.Mark attendance

4.Print report

5.Exit Enter your choice: 1

Enter the student name: John Doe Enter the student roll number: 101 Enter the student course: Computer Science

1.Add student

2.Remove student

3.Mark attendance

4.Print report

5.Exit Enter your choice: 3

Enter the student roll number to mark attendance: 101 Is the student present (1 for yes, 0 for no)? 1

1.Add student

2.Remove student

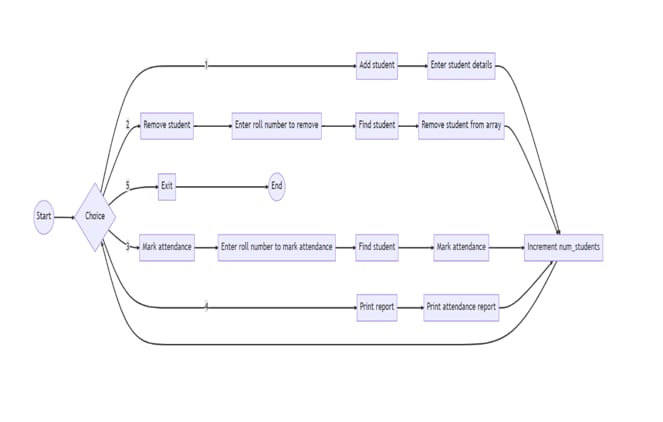
3.Mark attendance

4.Print report

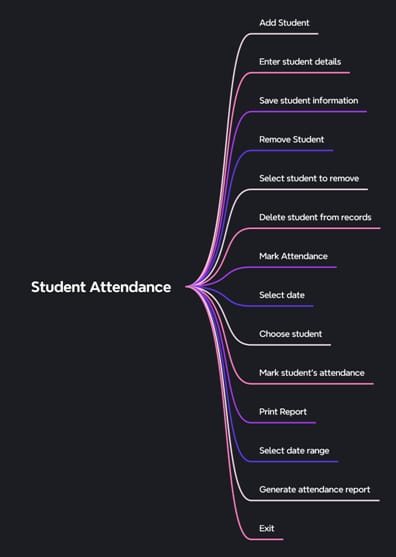
5.Exit Enter your choice: 4

Student name: John Doe Student roll number: 101 Student course: Computer Science Student attendance: 1

**Flowchart:**



**MIND MAP :**

****

**CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h> // Include the time header

struct student {

char name[50];

int roll\_number;

char course[50];

int attendance; // 1 for present, 0 for absent

char attendance\_date[20]; // Store attendance date and time

};

void add\_student(struct student \*students, int \*num\_students) {

// Get the student details from the user.

printf("Enter the student name: ");

scanf("%s", students[\*num\_students].name);

printf("Enter the student roll number: ");

scanf("%d", &students[\*num\_students].roll\_number);

printf("Enter the student course: ");

scanf("%s", students[\*num\_students].course);

students[\*num\_students].attendance = 0; // 0 for absent by default

strcpy(students[\*num\_students].attendance\_date, ""); // Initialize attendance\_date

// Increment the number of students.

(\*num\_students)++;

}

void remove\_student(struct student \*students, int \*num\_students, int roll\_number) {

// Find the student with the given roll number.

int i;

for (i = 0; i < \*num\_students; i++) {

if (students[i].roll\_number == roll\_number) {

break;

}

}

// If the student is found, remove it from the array.

if (i < \*num\_students) {

for (; i < \*num\_students - 1; i++) {

students[i] = students[i + 1];

}

(\*num\_students)--;

} else {

printf("Student with roll number %d not found.\n", roll\_number);

}

}

void mark\_attendance(struct student \*students, int \*num\_students, int roll\_number, int is\_present) {

// Find the student with the given roll number.

int i;

for (i = 0; i < \*num\_students; i++) {

if (students[i].roll\_number == roll\_number) {

break;

}

}

// If the student is found, mark their attendance and set the date.

if (i < \*num\_students) {

students[i].attendance = is\_present;

// Get the current date and time.

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

// Store the date in the student's record.

sprintf(students[i].attendance\_date, "%d-%02d-%02d %02d:%02d:%02d",

tm.tm\_year + 1900, tm.tm\_mon + 1, tm.tm\_mday, tm.tm\_hour, tm.tm\_min, tm.tm\_sec);

} else {

printf("Student with roll number %d not found.\n", roll\_number);

}

}

void print\_report(struct student \*students, int num\_students) {

// Print the attendance report in a table format.

printf("| %-20s | %-12s | %-20s | %-10s | %-20s |\n", "Name", "Roll Number", "Course", "Attendance", "Attendance Date");

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

for (int i = 0; i < num\_students; i++) {

printf("| %-20s | %-12d | %-20s | %-10d | %-20s |\n", students[i].name, students[i].roll\_number, students[i].course, students[i].attendance, students[i].attendance\_date);

}

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

}

int main() {

// Initialize the student array and the number of students.

struct student students[100];

int num\_students = 0;

// The main menu.

int choice;

while (1) {

printf("1. Add student\n");

printf("2. Remove student\n");

printf("3. Mark attendance\n");

printf("4. Print report\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

add\_student(students, &num\_students);

break;

case 2:

printf("Enter the student roll number to remove: ");

int roll\_number;

scanf("%d", &roll\_number);

remove\_student(students, &num\_students, roll\_number);

break;

case 3:

printf("Enter the student roll number to mark attendance: ");

scanf("%d", &roll\_number);

printf("Is the student present (1 for yes, 0 for no)? ");

int is\_present;

scanf("%d", &is\_present);

mark\_attendance(students, &num\_students, roll\_number, is\_present);

break;

case 4:

print\_report(students, num\_students);

break;

case 5:

// Exit the program.

exit(0);

default:

printf("Invalid choice. Please try again.\n");

break;

}

}

return 0;

}