Programing and Logic Design

# Attendance Report System

GROUP 1 | CPE 11S1

# GROUP 1

**Cenar**,
Marqui Joshua

**Formalejo**,
Francis Elijah

**Ramirez**,
Angel Mae

**Silao**,
Johrel Louie

**Tobias**,
Lawrence

**Engr. Jimlord Quejado**
Adviser

# Table of Content

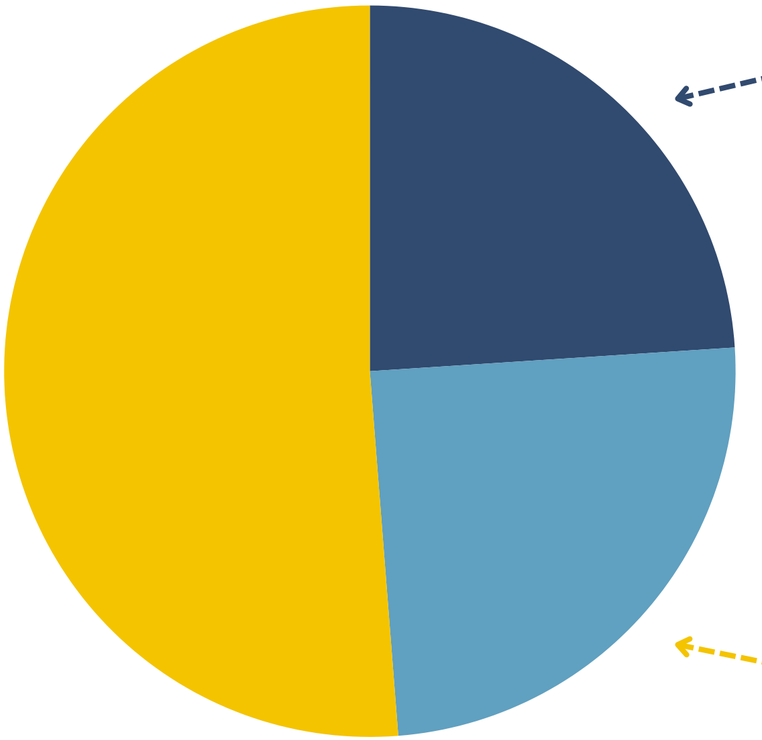
01

# The Problem

In educational institutions today, attendance is important for assessing student involvement and behavior. Numerous educational organizations continue to depend on manual methods for taking attendance. The majority of Philippine schools use laborious and time-consuming attendance tracking. Some employ paper-based methods, requiring students to turn them in along with their section and name. (Santos et al.,2021) This conventional approach can consume a significant amount of time during class sessions and may easily result in errors, including overlooked names, incorrect attendance, or misplaced records.(Lenuf, 2025). In a generation where technology continues to evolve rapidly, it seems outdated and inefficient for schools to still depend on manual systems when there are faster and more accurate alternatives available. This has become a general problem in many academic institutions, especially in large classes where monitoring attendance manually becomes more difficult and disorganized.

The specific problem arises when attendance data is not properly recorded or updated. Teachers may face challenges in tracking who is present, late, or absent, especially if there are multiple classes in a day. According to (Nwabuwe, et al.) “The most common act of misconduct was to sign a proxy for an absent friend.” This directly states the problem of false attendance being common in student settings. Over time, these small inaccuracies can affect grading and performance evaluation. Furthermore, the process of computing attendance percentages manually can take too much time leaving teachers less time to focus on their lessons and student engagement. In this sense, the problem is not just about recording attendance, but also about maintaining accuracy.

# 1.1 Academic Dishonesty among Students



Many students requested proxy attendance.
24%

Students rejected the proxy attendance statement.
51.2%

Students covered attendance for peers.
24.8

**Gazi, S., & Jamal, A. (2020). Academic Honesty among Students of Selected Dental Colleges of Bangladesh. Bangladesh Journal of Medical Education, 10(1), 6–13. https://doi.org/10.3329/bjme.v10i1.44588**

# 02 The Project

**The project**
aims to improve the monitoring and management of student attendance by providing a digital system to record, track, and view attendance reports. The program allows users to register by providing a username and password. Once registered, users can log in using their credentials, which are securely stored in the system's internal data structure, and are then presented with a menu of options.

**Register and Login:** Users can create an account by registering with a username and password. Once registered, they can log in using their credentials, which are stored securely in the system. This allows the program to identify each user and maintain individual attendance records.

**Mark Attendance:** After logging in, the user can input the student's name, date, and time in. The system automatically calculates the attendance status as Present, Late, or Absent based on the time entered, and the entry is saved under the logged-in user's records.

**View Full Attendance Report:** Users can view all attendance entries in a formatted table that includes the student's name, date, time in, and attendance status. This gives a clear overview of all recorded attendance data.

**View Attendance by Date:** Users can filter attendance records for a specific date, allowing them to quickly check attendance for a particular day.

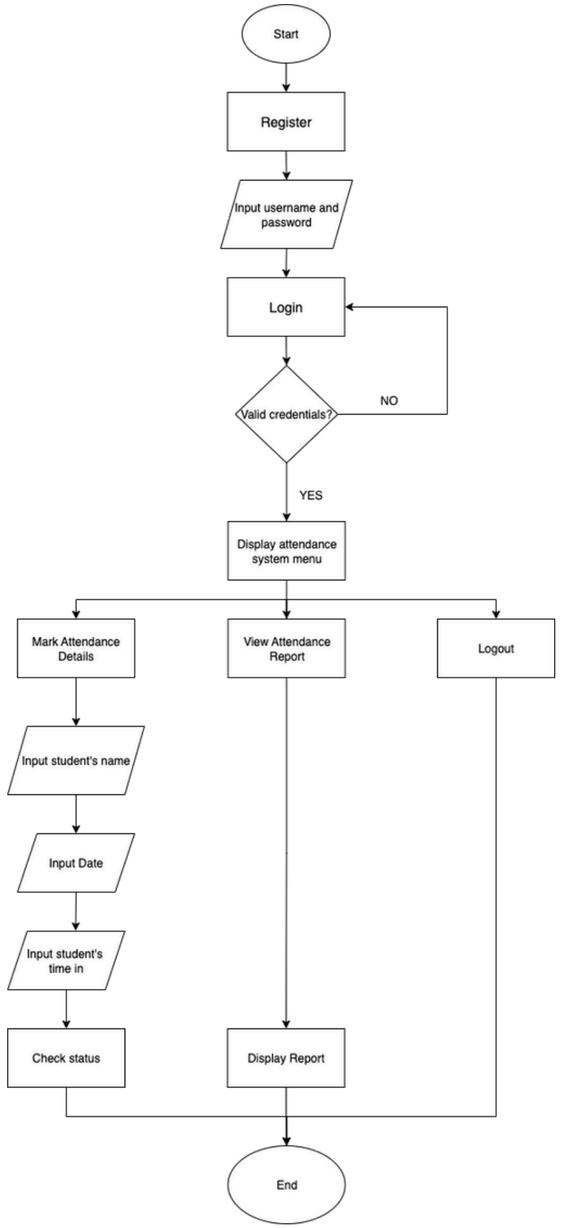
**Logout:** Users can securely log out from the system, ending their session and protecting access to their attendance records.

# 2.1 Objectives

**The project aims to develop an Attendance Report System that automates the process of recording and managing student attendance. Specifically:**

- Enable users to register and log in securely using a unique username and password.
- Allow users to mark attendance by inputting the student's name, date, and time in.
- Automatically determine and record the attendance status as Present, Late, or Absent based on the time entered.
- Display attendance reports in a clear and organized format, including options to view all records or filter by a specific date.

# 03 Flowchart of the System



# 03 Pseudocode

***main.cpp***

```
START

users = empty list
currentUser = NULL

LOOP forever

  IF currentUser is NULL THEN
    DISPLAY "1. Login 2. Register 3. Exit"
    READ choice

    IF choice == 1 THEN
      IF users is empty THEN
        PRINT "Register first"
      ELSE
        currentUser = login(users)
        IF currentUser is NULL THEN
          PRINT "Invalid login"
        ENDIF
      ENDIF

    ELSE IF choice == 2 THEN
      registerUser(users)

    ELSE IF choice == 3 THEN
      PRINT "Goodbye"
      BREAK
    ELSE
      PRINT "Invalid choice"
    ENDIF

  ELSE
    displayMenu()
    READ choice

    SWITCH choice
      CASE 1: markAttendance(currentUser)
      CASE 2: viewReport(currentUser)
      CASE 3: viewReportByDate(currentUser)
      CASE 4: currentUser = NULL // logout
      DEFAULT: PRINT "Invalid choice"
    END SWITCH

  ENDIF

END LOOP

END
```

***Attendance_system.h***

```
STRUCT Attendance:
 student_name
 date
 time_in
 status
```

# 3.1 Pseudocode

```
STRUCT User:
  username
  password
  attendances (list of Attendance)

FUNCTION getCurrentDate()
FUNCTION trim(text)
FUNCTION lowerCopy(text)
FUNCTION parseTimeToMinutes(time_in)
FUNCTION calculateStatus(time_in)

FUNCTION displayMenu()
FUNCTION login(users)
FUNCTION registerUser(users)
FUNCTION markAttendance(user)
FUNCTION viewReport(user)
FUNCTION viewReportByDate(user)
```

***Attendance_system.cpp***

```
STRUCT Attendance:
 student_name, date, time_in, status

STRUCT User:
 username, password, attendances

FUNCTION getCurrentDate()
 RETURN system date in YYYY-MM-DD

FUNCTION trim(text)
  RETURN text without leading/trailing spaces

FUNCTION lowerCopy(text)
  RETURN text in lowercase

FUNCTION parseTimeToMinutes(time_in)
  CLEAN time_in
  CONVERT to hours and minutes
  HANDLE am/pm if present
  IF invalid -> RETURN -1
  RETURN total_minutes

FUNCTION calculateStatus(time_in)
  minutes = parseTimeToMinutes(time_in)
  IF minutes == -1 -> RETURN "Absent"
  IF minutes <= 7:30 AM -> "Present"
  ELSE IF minutes <= 4:00 PM -> "Late"
  ELSE -> "Absent"

FUNCTION displayMenu()
   PRINT "1. Mark Attendance 2. View Full Report 3. View
by Date 4. Logout"

FUNCTION login(users)
  INPUT username, password
```

## 3.2 Pseudocode

```
FOR each user:
IF username AND password match -> RETURN user
RETURN NULL

FUNCTION registerUser(users)
  INPUT username
     IF username already exists -> PRINT "Taken" and
RETURN
  INPUT password
  CREATE new User and add to users

FUNCTION markAttendance(user)
  INPUT student_name, date (default today), time_in
  status = calculateStatus(time_in)
  ADD Attendance to user.attendances
  PRINT summary

FUNCTION viewReport(user)
  IF no attendances -> PRINT "No records"
   ELSE PRINT all attendance records (name, date, time,
status)

FUNCTION viewReportByDate(user)
  INPUT date (default today)
  FILTER attendances by date
  IF none -> PRINT "No records"
  ELSE PRINT filtered records (name, time, status)
```

## 04 Data Dictionary

| Data Name | Size | Data Type | Description |
|---|---|---|---|
| student_name | 32 bytes | string | Stores the full name of the student whose attendance |
| date | 16 bytes | string | Represents the date of attendance in the format YYYY-MM-DD. |
| time_in | 15 bytes | string | Holds the time the student arrived |
| status | 15 bytes | string | Indicates attendance status: “Present”, “Late”, or “Absent.” |
| attendances | depends on number of records | vector<Attendance> | A list (vector) that holds multiple attendance records for each user. |
| users | depends on number of users | vector<User> | A list containing all registered users in the system. |
| choice | 4 bytes | int | Stores the menu selection input from the user. |
| currentUser | 8 bytes | User* | A pointer that refers to the currently logged-in user. |
| buffer | 11 bytes | char[11] | Stores formatted date (e.g., "2025-11-11"). |
| username | 30 bytes | string | Stores the username used during registration and login. |

# 4.1 code

main.cpp | Attendance_system.h | Attendance_system.cpp

```cpp
#include "attendance_system.h"

int main() {
    vector<User> users; // Start empty, no admin
    User* currentUser = NULL;
    int choice = 0;

    while (true) {
        if (currentUser == NULL) {
            cout << "\n--- Welcome to Attendance System ---\n";
            cout << "1. Login\n";
            cout << "2. Register\n";
            cout << "3. Exit\n";
            cout << "Enter your choice: ";

            if (!(cin >> choice)) {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "Please enter a valid number.\n";
                continue;
            }

            if (choice == 1) {
                if (users.empty()) {
                    cout << "No users registered yet. Please register first.\n";
                    continue;
                }
                currentUser = login(users);
                if (currentUser == NULL) {
                    cout << "Invalid credentials. Try again.\n";
                    continue;
                }
                cout << "Login successful! Welcome, " << currentUser->username << "\n";
            }
            else if (choice == 2) {
                registerUser(users);
                continue;
            }
            else if (choice == 3) {
                cout << "Goodbye!\n";
                break;
            }
            else {
                cout << "Invalid choice. Try again.\n";
                continue;
            }
        }
        displayMenu();
        if (!(cin >> choice)) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Please enter a valid number.\n";
            continue;
        }

        switch (choice) {
            case 1:
                markAttendance(*currentUser);
                break;
            case 2:
                viewReport(*currentUser);
                break;
            case 3:
                viewReportByDate(*currentUser);
                break;
            case 4:
                cout << "Logged out.\n";
                currentUser = NULL;
                break;
            default:
                cout << "Invalid choice. Try again.\n";
        }
    }

    return 0;
}
```

# 4.2 code

main.cpp | Attendance_system.h | Attendance_system.cpp

```cpp
#ifndef ATTENDANCE_SYSTEM_H
#define ATTENDANCE_SYSTEM_H

#include <iostream>
#include <string>
#include <vector>
#include <iomanip>
#include <ctime>
#include <limits>
#include <sstream>
#include <cctype>
#include <cstdlib>

using namespace std;

struct Attendance {
    string student_name;
    string date;       // YYYY-MM-DD
    string time_in;
    string status;     // Present, Late, Absent
};

struct User {
    string username;
    string password;
    vector<Attendance> attendances;
};

// Utility functions
string getCurrentDate();
string trim(const string &s);
string lowerCopy(const string &s);
int parseTimeToMinutes(const string &raw);
string calculateStatus(const string &time_in);

// Menu & user functions
void displayMenu();
User* login(vector<User> &users);
void registerUser(vector<User> &users);
void markAttendance(User &user);
void viewReport(const User &user);
void viewReportByDate(const User &user);

#endif
```

main.cpp | Attendance_system.h | Attendance_system.cpp

```cpp
#include "attendance_system.h"
#include <cstdlib>
#include <cctype>
#include <sstream>
#include <limits>
#include <iomanip>
#include <ctime>

using namespace std;

// --- Utility Functions ---

string getCurrentDate() {
    time_t now = time(NULL);
    tm* tm_info = localtime(&now);
    char buffer[11];
    strftime(buffer, sizeof(buffer), "%Y-%m-%d", tm_info);
    return string(buffer);
}

string trim(const string& s) {
    size_t a = s.find_first_not_of(" \t\r\n");
    if (a == string::npos) return "";
    size_t b = s.find_last_not_of(" \t\r\n");
    return s.substr(a, b - a + 1);
}

string lowerCopy(const string& s) {
    string out;
    out.reserve(s.size());
    for (size_t i = 0; i < s.size(); i++) {
        unsigned char c = static_cast<unsigned char>(s[i]);
        out.push_back((char)tolower(c));
    }
    return out;
}

int parseTimeToMinutes(const string& raw) {
    string s = trim(raw);
    if (s.empty()) return -1;

    string low = lowerCopy(s);
    bool hasAM = (low.find("am") != string::npos);
    bool hasPM = (low.find("pm") != string::npos);
```

# 4.3 code

main.cpp | Attendance_system.h | Attendance_system.cpp

```
45
46      string digits = "";
47      for (size_t i = 0; i < low.size(); i++) {
48          char c = low[i];
49          if ((c >= '0' && c <= '9') || c == ':') digits.push_back(c);
50          else if (isspace((unsigned char)c)) digits.push_back(' ');
51      }
52      digits = trim(digits);
53
54      size_t colon = digits.find(':');
55      if (colon == string::npos) return -1;
56
57      string hs = trim(digits.substr(0, colon));
58      string ms = trim(digits.substr(colon + 1));
59
60      if (hs.empty() || ms.empty()) return -1;
61
62      for (size_t i = 0; i < hs.size(); i++) {
63          if (!isdigit((unsigned char)hs[i])) return -1;
64      }
65      for (size_t i = 0; i < ms.size(); i++) {
66          if (!isdigit((unsigned char)ms[i])) return -1;
67      }
68
69      int hours = atoi(hs.c_str());
70      int minutes = atoi(ms.c_str());
71      if (minutes < 0 || minutes > 59) return -1;
72
73      if (hasAM || hasPM) {
74          if (hours < 1 || hours > 12) return -1;
75          if (hasPM && hours != 12) hours += 12;
76          else if (!hasPM && hours == 12) hours = 0;
77      } else {
78          if (hours < 0 || hours > 23) return -1;
79      }
80
81      return hours * 60 + minutes;
82  }
83
84  string calculateStatus(const string& time_in) {
85      int total = parseTimeToMinutes(time_in);
86      if (total == -1) return "Absent";
87
```

main.cpp | Attendance_system.h | Attendance_system.cpp

```
88      int present_cutoff = 7 * 60 + 30; // 7:30 AM
89      int late_cutoff = 16 * 60;        // 4:00 PM
90
91      if (total <= present_cutoff) return "Present";
92      if (total <= late_cutoff) return "Late";
93      return "Absent";
94  }
95
96  // --- Menu Functions ---
97
98  void displayMenu() {
99      cout << "\n--- Attendance System Menu ---\n";
100     cout << "1. Mark Attendance\n";
101     cout << "2. View Full Attendance Report\n";
102     cout << "3. View Attendance by Date\n";
103     cout << "4. Logout\n";
104     cout << "Enter your choice: ";
105 }
106
107 User* login(vector<User>& users) {
108     string username, password;
109     cout << "\n--- Login ---\n";
110     cout << "Enter username: ";
111     cin >> username;
112     cout << "Enter password: ";
113     cin >> password;
114
115     for (size_t i = 0; i < users.size(); i++) {
116         if (users[i].username == username && users[i].password == password) {
117             return &users[i];
118         }
119     }
120     return NULL;
121 }
122
123 void registerUser(vector<User>& users) {
124     string username, password;
125     cout << "\n--- Register New User ---\n";
126     cout << "Enter new username: ";
127     cin >> username;
128
129     for (size_t i = 0; i < users.size(); i++) {
130         if (users[i].username == username) {
131             cout << "Username already taken. Please choose another.\n";
```

# 4.4 code

main.cpp | Attendance_system.h | Attendance_system.cpp

```
131             cout << "Username already taken. Please choose another.\n";
132             return;
133         }
134     }
135
136     cout << "Enter new password: ";
137     cin >> password;
138
139     User newUser;
140     newUser.username = username;
141     newUser.password = password;
142     users.push_back(newUser);
143
144     cout << "Registration successful! You can now log in.\n";
145 }
146
147 void markAttendance(User& user) {
148     string student_name, date, time_in;
149     cin.ignore(); // Clear newline
150
151     cout << "\nEnter student name: ";
152     getline(cin, student_name);
153
154     cout << "Enter date (YYYY-MM-DD) or press Enter for today: ";
155     getline(cin, date);
156     if (date.empty()) date = getCurrentDate();
157
158     cout << "Enter time in: ";
159     getline(cin, time_in);
160
161     string status = calculateStatus(time_in);
162
163     Attendance att;
164     att.student_name = student_name;
165     att.date = date;
166     att.time_in = time_in;
167     att.status = status;
168
169     user.attendances.push_back(att);
170
171     cout << "\nAttendance marked successfully!\n";
172     cout << "Student: " << att.student_name << "\n";
173     cout << "Date:    " << att.date << "\n";
174     cout << "Time In: " << att.time_in << "\n";
```

main.cpp | Attendance_system.h | Attendance_system.cpp

```
175     cout << "Status:  " << att.status << "\n";
176 }
177
178 void viewReport(const User& user) {
179     cout << "\n--- Attendance Report for " << user.username << " ---\n";
180     if (user.attendances.empty()) {
181         cout << "No attendance records found.\n";
182         return;
183     }
184
185     cout << left << setw(20) << "Student Name"
186          << setw(15) << "Date"
187          << setw(18) << "Time In"
188          << setw(10) << "Status" << "\n";
189     cout << "-----------------------------------------------------------------\n";
190
191     for (size_t i = 0; i < user.attendances.size(); i++) {
192         const Attendance& a = user.attendances[i];
193         cout << left << setw(20) << a.student_name
194              << setw(15) << a.date
195              << setw(18) << a.time_in
196              << setw(10) << a.status << "\n";
197     }
198 }
199
200 void viewReportByDate(const User& user) {
201     string date;
202     cin.ignore(); // Clear newline
203     cout << "\nEnter date to view attendance (YYYY-MM-DD) or press Enter for today: ";
204     getline(cin, date);
205     if (date.empty()) date = getCurrentDate();
206
207     bool found = false;
208     cout << "\n--- Attendance Report for " << user.username << " on " << date << " ---\n";
209     cout << left << setw(20) << "Student Name"
210          << setw(18) << "Time In"
211          << setw(10) << "Status" << "\n";
212     cout << "-----------------------------------------------------\n";
213
214     for (size_t i = 0; i < user.attendances.size(); i++) {
215         const Attendance& a = user.attendances[i];
216         if (a.date == date) {
217             cout << left << setw(20) << a.student_name
218                  << setw(18) << a.time_in
219                  << setw(10) << a.status << "\n";
220             found = true;
221         }
222     }
223
224     if (!found) cout << "No attendance records found for this date.\n";
225 }
```

# 05 Results & Discussion

```
--- Welcome to Attendance System ---
1. Login
2. Register
3. Exit
Enter your choice: 2

--- Register New User ---
Enter new username: Marqui
Enter new password: 123456
Registration successful! You can now log in.

--- Welcome to Attendance System ---
1. Login
2. Register
3. Exit
Enter your choice: 1

--- Login ---
Enter username: 123456
Enter password: Maqrui
Invalid credentials. Try again.
```

Running the programs first displays a Welcome Menu where the user will choose to Login, Register, or Exit. To register a new user, the system requires an input of a new username and password, after which the system saves the new account and displays a message telling the user that they can now log in. When logging in the username and password must be the same as the newly registered one, in this output I switched them to check if the system detects the error and prevents any wrong input to continue.

```
--- Welcome to Attendance System ---
1. Login
2. Register
3. Exit
Enter your choice: 1

--- Login ---
Enter username: Marqui
Enter password: 123456
Login successful! Welcome, Marqui

--- Attendance System Menu ---
1. Mark Attendance
2. View Full Attendance Report
3. View Attendance by Date
4. Logout
Enter your choice: 1

Enter student name: Joshua Garcia
Enter date (YYYY-MM-DD) or press Enter for today:
Enter time in: 7:30 AM

Attendance marked successfully!
Student: Joshua Garcia
Date:    2025-11-11
Time In: 7:30 AM
Status:  Present
```

In this output the username and password are correct so the user continues to the Attendance System Menu which shows the user actions that they can possibly do. The user marks attendance for a student using the current date with the time in of 7:30 AM. After that the system displays that the attendance for the student is marked successfully, with full details, and their status is “Present” since the time fits the cutoff.

# 5.1 Results & Discussion

```
--- Attendance System Menu ---
1. Mark Attendance
2. View Full Attendance Report
3. View Attendance by Date
4. Logout
Enter your choice: 1

Enter student name: Daniel Caesar
Enter date (YYYY-MM-DD) or press Enter for today:
Enter time in: 12:30 PM

Attendance marked successfully!
Student: Daniel Caesar
Date:    2025-11-11
Time In: 12:30 PM
Status:  Late
```

**The user marks attendance for another student still using the current date but with a different time. The system shows it has been marked successfully, with full details, and the student's status is "Late" because his time fits the requirement of being marked as "Late".**

```
--- Attendance System Menu ---
1. Mark Attendance
2. View Full Attendance Report
3. View Attendance by Date
4. Logout
Enter your choice: 1

Enter student name: Frank Ocean
Enter date (YYYY-MM-DD) or press Enter for today:
Enter time in: 5:00 PM

Attendance marked successfully!
Student: Frank Ocean
Date:    2025-11-11
Time In: 5:00 PM
Status:  Absent
```

**Another student's attendance record is being marked with the same date being the current date and still a different time than the previous ones. This student is marked as "Absent" because their time is already beyond the cutoff.**

# 5.2 Results & Discussion

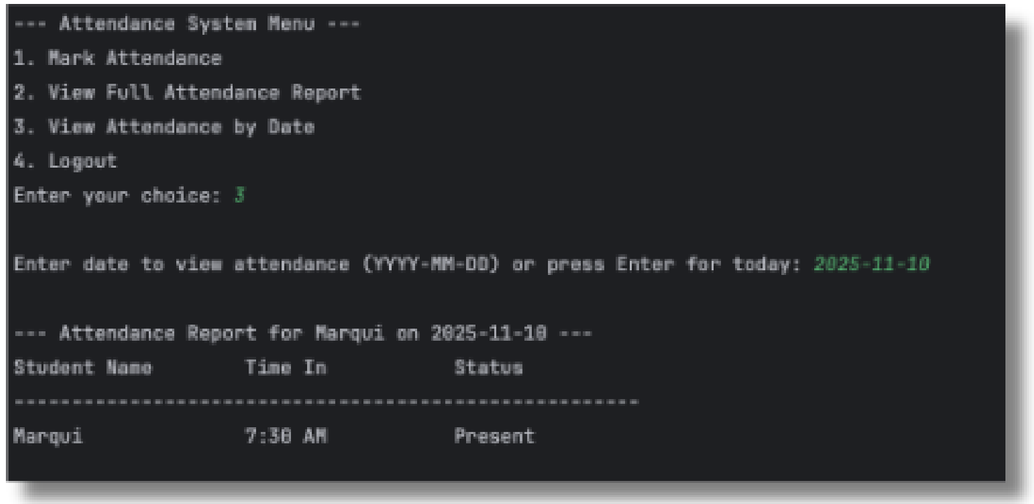
```
--- Attendance System Menu ---
1. Mark Attendance
2. View Full Attendance Report
3. View Attendance by Date
4. Logout
Enter your choice: 1

Enter student name: Marqui
Enter date (YYYY-MM-DD) or press Enter for today: 2025-11-10
Enter time in: 7:30 AM

Attendance marked successfully!
Student: Marqui
Date:    2025-11-10
Time In: 7:30 AM
Status:  Present
```

**In this output, the user marks attendance for a student but this time enters an exact date that is not the current date with a time of 7:30 AM. The student was marked as “Present” for that date and it showed the full details.**

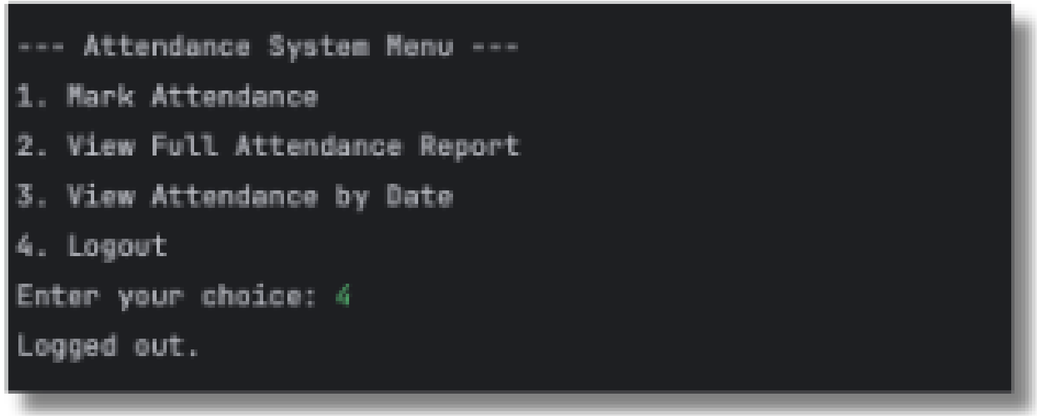
```
--- Attendance System Menu ---
1. Mark Attendance
2. View Full Attendance Report
3. View Attendance by Date
4. Logout
Enter your choice: 2

--- Attendance Report for Marqui ---
Student Name        Date           Time In          Status
-----------------------------------------------------------------
Joshua Garcia       2025-11-11     7:30 AM          Present
Daniel Caesar       2025-11-11     12:30 PM         Late
Frank Ocean         2025-11-11     5:00 PM          Absent
Marqui              2025-11-10     7:30 AM          Present
```

**This output is the Full Attendance Report including the details of each student that was marked by the user. This report shows the student’s name, date when they were marked, time in, and status. It clearly shows which students were Present, Late, or Absent on what day and what time did they get to class.**

# 5.3 Results & Discussion




This Attendance Report shows only the report for a specific date. Only one student was marked on a specific date so the report only shows one student, including their name, time in, and status which is “Present” for that specific day.




In this output shows the Attendance System Menu and the user’s choice to Logout of the system. The system receives its input and display a message indicating that the user has been logged out of the system.

# 6 Conclusion

Throughout the development of the Attendance Report System, the group went through a detailed and sometimes challenging process of designing, coding, and debugging the program. At first, several errors were encountered, especially in organizing the functions, linking the header file, and ensuring that each part of the system worked together properly. These issues required careful troubleshooting and revisions, which helped the team better understand how C++ handles structures, vectors, and function declarations. Despite the difficulties, these experiences allowed the group to strengthen their problem-solving and coding skills.

The completed program successfully allows users to register, log in, record attendance, and generate attendance reports efficiently. It can automatically determine whether a student is Present, Late, or Absent based on the time entered, providing an accurate and organized record of attendance. Additionally, the system includes features for viewing all attendance data or filtering reports by a specific date, enhancing its usability and accuracy.

Overall, the project demonstrates the effectiveness of using programming to automate and improve traditional attendance recording systems. Despite the technical difficulties encountered during development, the group was able to create a functioning system that meets its intended purpose and contributes to more efficient attendance monitoring and management.

# 7 References


Gazi, S., & Jamal, A. (2020). Academic Honesty among Students of Selected Dental Colleges of Bangladesh. Bangladesh Journal of Medical Education, 10(1), 6–13. https://doi.org/10.3329/bjme.v10i1.44588

Lenuf. (2025, January 16). Challenges of manual attendance marking and how to overcome them. SoftDunamis Business Meal Consulting. https://softdunamis.com/blog/manual-attendance-marking/

Nwabuwe, A., Sanghera, B., Alade, T., & Olajide, F. (2023). Fraud Mitigation in Attendance Monitoring Systems using Dynamic QR Code, Geofencing and IMEI Technologies. International Journal of Advanced Computer Science and Applications, 14(4). https://doi.org/10.14569/ijacsa.2023.01404104

Rahaman, M., Islam, M. M., & Nandi, D. (2025). SmartPresence: Wi-Fi-based online attendance management for smart academic assistance. Journal of Electrical Systems and Information Technology, 12(1). https://doi.org/10.1186/s43067-025-00215-y

Santos, A. B. G., Balba, N. P., & Rebong, C. B. (2021). Attendance Monitoring System of Schools in the Philippines with an Inclusion of Optimization Query Algorithm. International Journal of Innovative Technology and Exploring Engineering, 10(8), 142–146. https://doi.org/10.35940/ijitee.h9149.0610821