AMITY UNIVERSITY
JHARKHAND

Home Assignment + Viva

Topic: College Management System

Course code: ES203

Course Title: Object Oriented Programming Using C++

Student Name: Ahmad Faraz

Enrollment No: A35705223006

Program: BTech CSE

Section: A

Semester: 2

Batch: 2023-27

Signature of the student with date

INTRODUCTION

The **College Management System** project is designed to streamline administrative tasks and enhance communication within a college or educational institution. It provides a comprehensive platform for managing various aspects of college operations, including student information, teacher details, course management, attendance tracking, fee management, and more.

Key features of the College Management System project include:

1. **User Authentication**: Users such as administrators, teachers, and students can log in securely using their credentials.

2. **Role-based Access Control**: Different user roles have access to specific functionalities based on their roles, ensuring data security and privacy.

3. **Student Management**: Enables the management of student profiles, including personal information, academic records, attendance, and fee details.

4. **Teacher Management**: Facilitates the management of teacher profiles, including personal information, qualifications, contact details, and salary details.

5. **Course Management**: Allows administrators to create, update, and manage course schedules, syllabi, and timetables.

6. **Attendance Tracking**: Provides functionality for teachers to take and record student attendance for various classes and sessions.

7. **Grade Management**: Allows teachers to upload and manage student grades and marks for assessments, examinations, and assignments.

8. **Fee Management**: Enables administrators to manage student fees, including fee collection, overdue reminders, and fee waivers.

9. **Communication**: Facilitates communication between administrators, teachers, and students through notices, announcements, and messages.

10. **Data Persistence**: Utilizes CSV files to store and manage data, ensuring easy access, retrieval, and manipulation of information.

Overall, the College Management System project aims to automate and streamline administrative tasks, enhance communication, and improve efficiency within educational institutions.

SYSTEM REQUIREMENTS

1. **Operating System**:

- The project should be compatible with most major operating systems, including Windows, macOS, and various Linux distributions.

2. **Compiler**:

- A C++ compiler is required to build and run the project. Common options include GCC (GNU Compiler Collection), Clang, and Microsoft Visual C++.

3. **Memory**:

- Adequate RAM is needed to run the application smoothly. The memory requirements depend on factors such as the size of the dataset and the complexity of operations.

4. **Storage**:

- Sufficient disk space is required to store the CSV data files used by the application. The amount of storage needed depends on the size of the dataset and expected growth.

5. **Processor**:

- A standard modern processor should be sufficient for running the application. The processing power required depends on the complexity of operations and the number of concurrent users.

6. **Concurrency**:

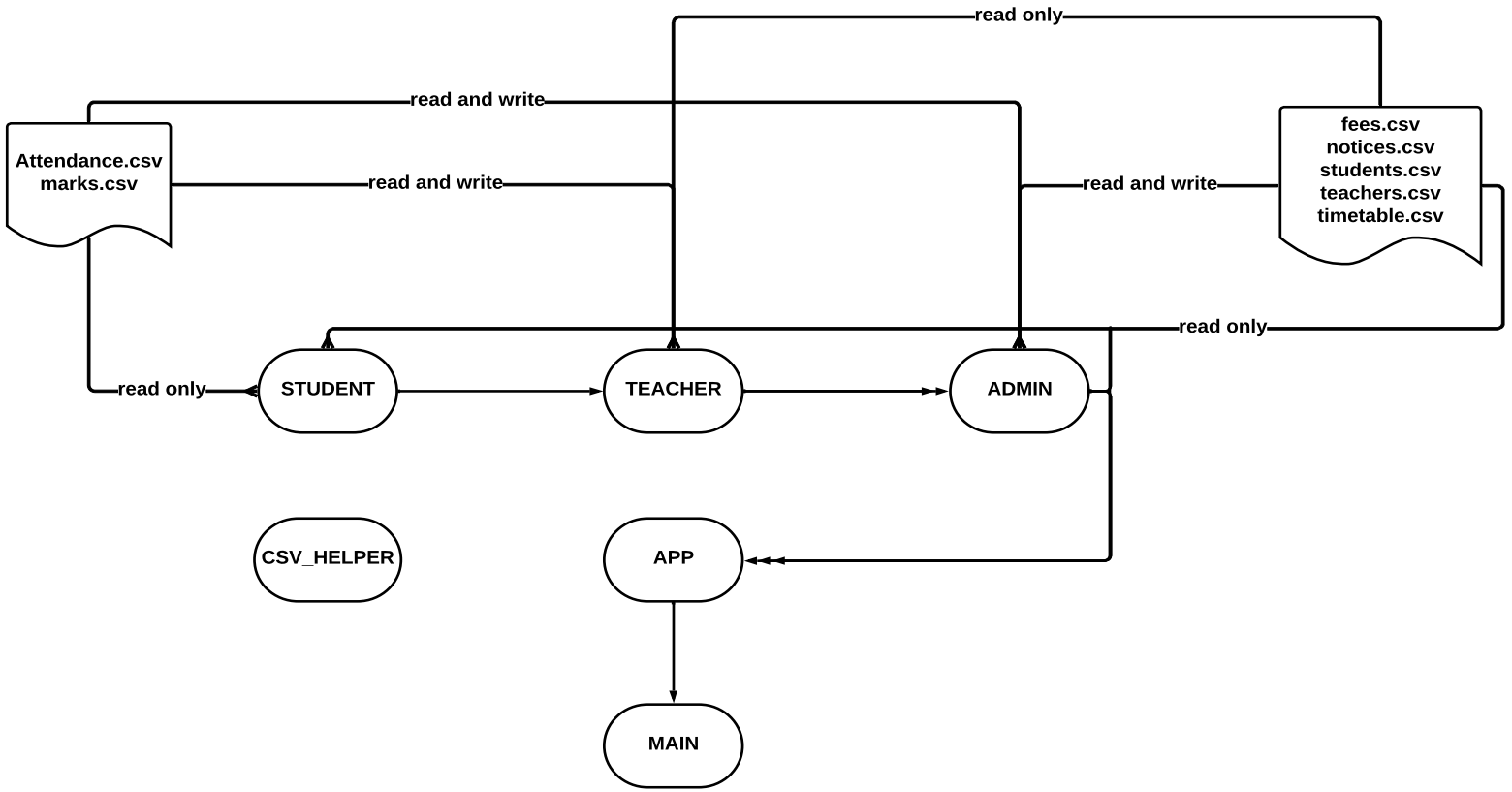
- If the system is expected to handle multiple users concurrently, mechanisms for concurrency control may need to be implemented to ensure data consistency and avoid conflicts.

7. **Dependencies**:

- Ensure that the necessary dependencies, including standard C++ libraries and any third-party libraries used in the project, are available on the system.

These requirements provide a basic framework for setting up and running the project. Depending on specific needs and usage scenarios, additional resources and optimizations may be necessary.

DATA FLOW DIAGRAM



SOURCE CODE

Csv_helper.h

```
#ifndef CSV_HELPER_H
#define CSV_HELPER_H

#include <string>

using namespace std;

class CSV
{
public:
    void write(string filename, string *data, int size, const char delimiter);
    string *split_line(string filename, int line_no, const char delimiter);
    int total_lines(string filename);
    void delete_line(string filename, int line_no);
    void update_line(string filename, int line_no, string *data, int size, const char
delimiter);
    int find_line(string filename, string to_find, const char delimiter);
};

#endif
```

csv_helper.cpp

```
#include <iostream>
#include <fstream>
#include <string>

#include "Csv_helper.h"

using namespace std;

void CSV::write(string filename, string *data, int size, const char delimiter)
{
    ofstream file;
    file.open(filename, ios::app);
    for (int i = 0; i < size; i++)
    {
        try
        {
            if (data[i].find(delimiter) != string::npos)
```

```
            {
                throw "Invalid argument: Data contains delimiter.\nFalied to update
file.";
            }
        }
        catch (const char *msg)
        {
            cerr << msg << endl;
            throw msg;
        }
    }
    for (int i = 0; i < size; i++)
    {
        file << data[i];
        if (i < size - 1)
        {
            file << delimiter;
        }
    }
    file << endl;
    file.close();
}

int CSV::total_lines(string filename)
{
    ifstream file;
    file.open(filename, ios::in);
    string line;
    int count = 0;
    while (getline(file, line))
    {
        count++;
    }
    return count;
}

string *CSV::split_line(string filename, int line_no, const char delimiter)
{
    ifstream file(filename);
    if (!file.is_open())
    {
        cerr << "Error: Unable to open file." << endl;
        return nullptr;
    }
```

```
    string line;
    int i = 0;
    while (getline(file, line))
    {
        if (i == line_no)
        {
            string *arr = new string[10];
            int j = 0;
            string token;
            for (char c : line)
            {
                if (c == delimiter)
                {
                    arr[j++] = token;
                    token.clear();
                }
                else
                {
                    token += c;
                }
            }
            arr[j] = token;
            return arr;
        }
        i++;
    }
    cerr << "Error: Line number out of bounds." << endl;
    return nullptr;
}

void CSV::delete_line(string filename, int line_no)
{
    ifstream file(filename);
    if (!file.is_open())
    {
        cerr << "Error: Unable to open file." << endl;
        return;
    }
    int i = 0;
    string line, temp_line;
    ofstream temp("temp.csv");
    while (getline(file, line))
    {
        if (i != line_no)
        {
```

```
            temp << line << endl;
        }
        else
        {
            temp_line = line;
        }
        i++;
    }
    file.close();
    temp.close();
    remove(filename.c_str());
    rename("temp.csv", filename.c_str());
}

void CSV::update_line(string filename, int line_no, string *data, int size, const char
delimiter)
{
    string *temp_line = split_line(filename, line_no, delimiter);
    if (temp_line == nullptr)
    {
        cerr << "Error: Failed to update line. Reverting changes." << endl;
        return;
    }

    delete_line(filename, line_no);

    try
    {
        write(filename, data, size, delimiter);
    }
    catch (const char *msg)
    {
        write(filename, temp_line, size, delimiter);
    }
}

int CSV::find_line(string filename, string to_find, const char delimiter)
{
    ifstream file(filename);
    if (!file.is_open())
    {
        cerr << "Error: Unable to open file." << endl;
        return -1;
    }
```

```
    string line;
    int i = 0;
    while (getline(file, line))
    {
        string token;
        for (char c : line)
        {
            if (c == delimiter)
            {
                if (token == to_find)
                {
                    return i;
                }
                token.clear();
            }
            else
            {
                token += c;
            }
        }
        i++;
    }
    return -1;
}
```

Admin.h

```
#ifndef ADMIN_H
#define ADMIN_H

#include <string>

#include "Teacher.h"

using namespace std;

class Admin : public Teacher
{
public:
    string notice, notice_desc;

    int admin_login();
    void add_student();
    void add_teacher();
```

```
    void update_fee();
    void add_fee();
    void add_timetable();
    void update_attendance();
    void update_marks();
    void add_notice();
    void remove_student();
    void remove_teacher();
    void admin_logout();
};

#endif
```

admin.cpp

```
#include <iostream>
#include <string>
#include <fstream>

#include "Admin.h"
#include "Csv_helper.h"

#define DELIMETER ','

int Admin::admin_login()
{
    string username, password;
    cin.ignore();
    cout << "Enter username: ";
    getline(cin, username);
    cout << "Enter password: ";
    getline(cin, password);
    if (username == "admin" && password == "password")
    {
        cout << "Login successful" << endl;
        return 1;
    }
    else
    {
        cout << "Invalid username or password" << endl;
        return 0;
    }
}
```

```
void Admin::add_student()
{
    CSV csv;
    char passwd[20];
    cin.ignore();
    cout << "Enter student roll no: ";
    getline(cin, roll_no);
    for (int i = 0; i < csv.total_lines("students.csv"); i++)
    {
        string *data = csv.split_line("students.csv", i, DELIMETER);
        if (data[0] == roll_no)
        {
            cout << "Student with this roll no already exists" << endl;
            return;
        }
    }
    cout << "Enter student name: ";
    getline(cin, name);
    cout << "Enter student branch: ";
    getline(cin, branch);
    cout << "Enter student semester: ";
    getline(cin, semester);
    cout << "Enter student email: ";
    getline(cin, email);
    cout << "Enter student phone: ";
    getline(cin, phone);
    cout << "Enter student address: ";
    getline(cin, address);
    cout << "Enter room number: ";
    getline(cin, room);
    strcpy(passwd, roll_no.c_str());
    strcat(passwd, name.c_str());
    string data[] = {roll_no, name, branch, semester, email, phone, address, passwd,
room};
    csv.write("students.csv", data, 9, DELIMETER);
}

void Admin::add_teacher()
{
    CSV csv;
    char passwd[20];
    cin.ignore();
    cout << "Enter teacher unique id: ";
    getline(cin, unique_id);
    for (int i = 0; i < csv.total_lines("teachers.csv"); i++)
```

```
    {
        string *data = csv.split_line("teachers.csv", i, DELIMETER);
        if (data[0] == unique_id)
        {
            cout << "Teacher with this unique id already exists" << endl;
            return;
        }
    }
    cout << "Enter teacher name: ";
    getline(cin, tname);
    cout << "Enter teacher degree: ";
    getline(cin, tdegree);
    cout << "Enter teacher email: ";
    getline(cin, temail);
    cout << "Enter teacher phone: ";
    getline(cin, tphone);
    cout << "Enter teacher address: ";
    getline(cin, taddress);
    strcpy(passwd, unique_id.c_str());
    strcat(passwd, tname.c_str());
    string data[] = {unique_id, tname, tdegree, temail, tphone, taddress, passwd};
    csv.write("teachers.csv", data, 7, DELIMETER);
}
void Admin::update_fee()
{
    CSV csv;
    string id, fees;
    cin.ignore();
    cout << "Enter roll_no or unique_id: ";
    getline(cin, id);
    cout << "Enter new fees: ";
    getline(cin, fees);
    for (int i = 0; i < csv.total_lines("fees.csv"); i++)
    {
        string *data = csv.split_line("fees.csv", i, DELIMETER);
        if (data[0] == id)
        {
            char ch;
            cout << "Are you sure you want to update this fee? (y/n): " << endl;
            cout << "Old fee: " << data[1] << endl;
            cin >> ch;
            if (ch != 'y')
            {
                cout << "Fee not updated" << endl;
                return;
```

```
            }
            string new_data[] = {id, fees};
            csv.update_line("fees.csv", i, new_data, 2, DELIMETER);
            cout << "Fee updated successfully" << endl;
            return;
        }
    }
    cout << "Id not found" << endl;
}
void Admin::add_fee()
{
    CSV csv;
    string id, fees;
    cin.ignore();
    cout << "Enter roll_no or unique_id: ";
    getline(cin, id);
    cout << "Enter fees: ";
    getline(cin, fees);
    string data[] = {id, fees};
    if (csv.find_line("fees.csv", id, DELIMETER) != -1)
    {
        cout << "Id already added" << endl;
        return;
    }
    csv.write("fees.csv", data, 2, DELIMETER);
}
void Admin::add_timetable()
{
    CSV csv;
    string room, unique_id, date, day, time, course;
    cin.ignore();
    cout << "Enter teacher unique id: ";
    getline(cin, unique_id);
    if (csv.split_line("teachers.csv", 0, DELIMETER)[0] != unique_id)
    {
        cout << "Teacher with this unique id does not exist" << endl;
        return;
    }
    cout << "Enter date: ";
    getline(cin, date);
    cout << "Enter day: ";
    getline(cin, day);
    cout << "Enter time: ";
    getline(cin, time);
    cout << "Enter room number: ";
```

```
    getline(cin, room);
    cout << "Enter course: ";
    getline(cin, course);
    string data[] = {room, unique_id, date, day, time, course};
    csv.write("timetable.csv", data, 6, DELIMETER);
}

void Admin::update_attendance()
{
    CSV csv;
    string roll_no, date, status, time;
    cin.ignore();
    cout << "Enter student roll no: ";
    getline(cin, roll_no);
    cout << "Enter date: ";
    getline(cin, date);
    cout << "Enter time: ";
    getline(cin, time);
    for (int i = 0; i < csv.total_lines("attendance.csv"); i++)
    {
        string *data = csv.split_line("attendance.csv", i, DELIMETER);
        if (data[3] == roll_no && data[0] == date && data[1] == time)
        {
            char ch;
            cout << "Are you sure you want to update this attendance? (y/n): " << endl;
            cout << "Old status: " << data[4] << endl;
            cin >> ch;
            if (ch != 'y')
            {
                cout << "Attendance not updated" << endl;
                return;
            }
            cin.ignore();
            cout << "Enter new status: ";
            getline(cin, status);
            string new_data[] = {date, time, data[2], roll_no, status, data[5],
data[6]};
            csv.update_line("attendance.csv", i, new_data, 7, DELIMETER);
            cout << "Attendance updated successfully" << endl;
            return;
        }
    }
    cout << "Attendance not found" << endl;
}
void Admin::update_marks()
```

```
{
    CSV csv;
    string roll_no, date, marks, time;
    cin.ignore();
    cout << "Enter student roll no: ";
    getline(cin, roll_no);
    cout << "Enter date: ";
    getline(cin, date);
    cout << "Enter time: ";
    getline(cin, time);
    cout << "Enter marks: ";

    getline(cin, marks);
    for (int i = 0; i < csv.total_lines("marks.csv"); i++)
    {
        string *data = csv.split_line("marks.csv", i, DELIMETER);
        if (data[3] == roll_no && data[0] == date && data[1] == time)
        {
            char ch;
            cout << "Are you sure you want to update this marks? (y/n): " << endl;
            cout << "Old marks: " << data[4] << endl;
            cin >> ch;
            if (ch != 'y')
            {
                cout << "Marks not updated" << endl;
                return;
            }
            string new_data[] = {date, time, data[2], roll_no, marks, data[5],
data[6]};
            csv.update_line("marks.csv", i, new_data, 7, DELIMETER);
            cout << "Marks updated successfully" << endl;
            return;
        }
    }
    cout << "Marks not found" << endl;
}

void Admin::add_notice()
{
    CSV csv;
    cin.ignore();
    cout << "Enter date: ";
    getline(cin, notice);
    cout << "Enter notice description: ";
    getline(cin, notice_desc);
```

```
    string data[] = {notice, notice_desc};
    csv.write("notices.csv", data, 2, DELIMETER);
}

void Admin::remove_student()
{
    CSV csv;
    string roll_no;
    cin.ignore();
    cout << "Enter student roll no: ";
    getline(cin, roll_no);
    for (int i = 0; i < csv.total_lines("students.csv"); i++)
    {
        string *data = csv.split_line("students.csv", i, DELIMETER);
        if (data[0] == roll_no)
        {
            char ch;
            cout << "Are you sure you want to remove this student? (y/n): " << endl;
            csv_line = i;
            student_display();
            cin >> ch;
            if (ch != 'y')
            {
                cout << "Student not removed" << endl;
                return;
            }
            csv.delete_line("students.csv", i);
            cout << "Student removed successfully" << endl;
            return;
        }
    }
    cout << "Student not found" << endl;
}

void Admin::remove_teacher()
{
    CSV csv;
    string unique_id;
    cin.ignore();
    cout << "Enter teacher unique_id: ";
    getline(cin, unique_id);
    for (int i = 0; i < csv.total_lines("teachers.csv"); i++)
    {
        string *data = csv.split_line("teachers.csv", i, DELIMETER);
        if (data[0] == unique_id)
```

```
        {
            char ch;
            cout << "Are you sure you want to remove this teacher? (y/n): " << endl;
            tcsv_line = i;
            teacher_display();
            cin >> ch;
            if (ch != 'y')
            {
                cout << "Teacher not removed" << endl;
                return;
            }
            csv.delete_line("teachers.csv", i);
            cout << "Teacher removed successfully" << endl;
            return;
        }
    }
    cout << "Teacher not found" << endl;
}

void Admin::admin_logout()
{
    cout << "Admin logged out successfully" << endl;
}
```

Student.h

```
#ifndef STUDENT_H
#define STUDENT_H

#include <iostream>

using namespace std;
class Student
{
private:
    string password;

public:
    string name;
    string roll_no;
    string branch;
    string semester;
    string email;
    string phone;
```

```
    string address;
    string room;
    int csv_line;

    int student_login();
    void student_display();
    void student_update();
    void student_marks();
    void student_attendance();
    void student_timetable();
    void student_fee();
    void student_logout();
};

#endif
```

student.cpp

```
#include <iostream>
#include <string>
#include <fstream>

#include "Student.h"
#include "Csv_helper.h"

#define DELIMETER ','

using namespace std;

int Student::student_login()
{
    CSV csv;
    string roll, passwd;
    cin.ignore();
    cout << "Enter your roll no: ";
    getline(cin, roll);
    cout << "Enter your password: ";
    getline(cin, passwd);
    for (int i = 0; i < csv.total_lines("students.csv"); i++)
    {
        string *data = csv.split_line("students.csv", i, DELIMETER);
        if (data[0] == roll && data[7] == passwd)
        {
            cout << "Login successful" << endl;
```

```
            csv_line = i;
            roll_no = data[0];
            name = data[1];
            branch = data[2];
            semester = data[3];
            email = data[4];
            phone = data[5];
            address = data[6];
            room = data[8];
            return 1;
        }
    }
    cout << "Invalid roll no or password" << endl;
    return 0;
}

void Student::student_display()
{
    CSV csv;
    string *data = csv.split_line("students.csv", csv_line, DELIMETER);
    cout << "Roll No: " << data[0] << endl;
    cout << "Name: " << data[1] << endl;
    cout << "Branch: " << data[2] << endl;
    cout << "Semester: " << data[3] << endl;
    cout << "Email: " << data[4] << endl;
    cout << "Phone: " << data[5] << endl;
    cout << "Address: " << data[6] << endl;
    cout << "Room: " << data[8] << endl;
}
void Student::student_update()
{
    CSV csv;
    string passwd;
    cin.ignore();
    cout << "Enter old password: ";
    getline(cin, passwd);
    string *data = csv.split_line("students.csv", csv_line, DELIMETER);
    if (data[7] != passwd)
    {
        cout << "Invalid old password" << endl;
        return;
    }
    cout << "Enter new password: ";
    getline(cin, passwd);
    cout << "Enter new password again: ";
```

```
    string passwd2;
    getline(cin, passwd2);
    if (passwd != passwd2)
    {
        cout << "Passwords do not match" << endl;
        return;
    }
    string new_data[] = {data[0], name, branch, semester, email, phone, address,
passwd2, room};
    csv.update_line("students.csv", csv_line, new_data, 9, DELIMETER);
}
void Student::student_marks() {
    CSV csv;
    int c = 0;
    for (int i = 0; i < csv.total_lines("marks.csv"); i++)
    {
        string *data = csv.split_line("marks.csv", i, DELIMETER);
        if (data[3] == roll_no)
        {
            cout << "Course: " << data[6];
            cout << " | Marks: " << data[4] << endl;
            c++;
        }
    }
    if (c == 0)
    {
        cout << "No marks found" << endl;
    }
}
void Student::student_attendance() {
    CSV csv;
    string date;
    int c = 0;
    cin.ignore();
    cout << "Enter date: ";
    getline(cin, date);
    for (int i = 0; i < csv.total_lines("attendance.csv"); i++)
    {
        string *data = csv.split_line("attendance.csv", i, DELIMETER);
        if (data[3] == roll_no && data[0] == date)
        {
            string teacher_name = csv.split_line("teachers.csv",
csv.find_line("teachers.csv", data[5], DELIMETER), DELIMETER)[1];
            cout << "Date: " << data[0];
            cout << " | Time: " << data[1];
```

```
            cout << " | Room: " << data[2];
            cout << " | Course: " << data[6];
            cout << " | Teacher: " << teacher_name;
            cout << " | Status: " << data[4] << endl;
            c++;
        }
    }

    if (c == 0)
    {
        cout << "No attendance on this day" << endl;
    }
}

void Student::student_timetable()
{
    CSV csv;
    string date;
    int c = 0;
    cout << "Enter date: ";
    cin.ignore();
    getline(cin, date);
    for (int i = 0; i < csv.total_lines("timetable.csv"); i++)
    {
        string *data = csv.split_line("timetable.csv", i, DELIMETER);
        if (data[2] == date && data[0] == room)
        {
            cout << "Time: " << data[4];
            cout << " | Day: " << data[3];
            cout << " | Course: " << data[5];
            cout << " | Room: " << data[0];
            int line_no = csv.find_line("teachers.csv", data[1], DELIMETER);
            string teacher_name = csv.split_line("teachers.csv", line_no,
DELIMETER)[1];
            cout << " | Teacher: " << teacher_name << endl;
            c++;
        }
    }
    if (c == 0)
    {
        cout << "No classes on this day" << endl;
    }
}
void Student::student_fee() {
    CSV csv;
```

```
    for (int i = 0; i < csv.total_lines("fees.csv"); i++)
    {
        string *data = csv.split_line("fees.csv", i, DELIMETER);
        if (data[0] == roll_no)
        {
            cout << "Fees: " << data[1] << endl;
            return;
        }
    }
    cout << "No fees found" << endl;
}

void Student::student_logout()
{
    cout << name << " logged out" << endl;
}
```

Teacher.h

```
#ifndef TEACHER_H
#define TEACHER_H

#include <string>

#include "Student.h"

using namespace std;

class Teacher : public Student
{
private:
    string tpassword;

public:
    string unique_id;
    string tname;
    string tdegree;
    string temail;
    string tphone;
    string taddress;
    int tcsv_line;

    int teacher_login();
```

```
    void teacher_display();
    void teacher_update();
    void upload_marks();
    void take_attendance();
    void teacher_timetable();
    void teacher_salary();
    void teacher_logout();
};

#endif
```

teacher.cpp

```
#include <iostream>
#include <string>
#include <fstream>

#include "Csv_helper.h"
#include "Teacher.h"

#define DELIMETER ','

using namespace std;

int Teacher::teacher_login()
{
    CSV csv;
    string _id, passwd;
    cin.ignore();
    cout << "Enter your unique id: ";
    getline(cin, _id);
    cout << "Enter your password: ";
    getline(cin, passwd);
    for (int i = 0; i < csv.total_lines("teachers.csv"); i++)
    {
        string *data = csv.split_line("teachers.csv", i, DELIMETER);
        if (data[0] == _id && data[6] == passwd)
        {
            cout << "Login successful" << endl;
            tcsv_line = i;
            unique_id = data[0];
            tname = data[1];
            tdegree = data[2];
            temail = data[3];
```

```
            tphone = data[4];
            taddress = data[5];
            return 1;
        }
    }
    cout << "Invalid unique_id or password" << endl;
    return 0;
}

void Teacher::teacher_display()
{
    CSV csv;
    string *data = csv.split_line("teachers.csv", tcsv_line, DELIMETER);
    cout << "Unique ID: " << data[0] << endl;
    cout << "Name: " << data[1] << endl;
    cout << "Degree: " << data[2] << endl;
    cout << "Email: " << data[3] << endl;
    cout << "Phone: " << data[4] << endl;
    cout << "Address: " << data[5] << endl;
}

void Teacher::teacher_update()
{
    CSV csv;
    string passwd;
    cin.ignore();
    cout << "Enter old password: ";
    getline(cin, passwd);
    string *data = csv.split_line("teachers.csv", csv_line, DELIMETER);
    if (data[6] != passwd)
    {
        cout << "Invalid old password" << endl;
        return;
    }
    cout << "Enter new password: ";
    getline(cin, passwd);
    cout << "Enter new password again: ";
    string passwd2;
    getline(cin, passwd2);
    if (passwd != passwd2)
    {
        cout << "Passwords do not match" << endl;
        return;
    }
```

```
    string new_data[7] = {data[0], data[1], data[2], data[3], data[4], data[5],
passwd};
    csv.update_line("teachers.csv", csv_line, new_data, 7, DELIMETER);
}

void Teacher::upload_marks() {
    CSV csv;
    string date, room, time, course;
    int c = 0;
    cin.ignore();
    cout << "Enter date: ";
    getline(cin, date);
    cout << "Enter time: ";
    getline(cin, time);
    string unique_id = csv.split_line("teachers.csv", tcsv_line, DELIMETER)[0];
    for (int i = 0; i < csv.total_lines("timetable.csv"); i++)
    {
        string *data = csv.split_line("timetable.csv", i, DELIMETER);
        if (data[1] == unique_id && data[2] == date && data[4] == time)
        {
            cout << "Room: " << data[0];
            cout << " | Date: " << data[2];
            cout << " | Day: " << data[3];
            cout << " | Time: " << data[4];
            cout << " | Course: " << data[5] << endl;
            room = data[0];
            course = data[5];
            c++;
            break;
        }
    }
    if (c == 0)
    {
        cout << "No classes on this date or time" << endl;
        return;
    }
    for (int i = 0; i < csv.total_lines("marks.csv"); i++)
    {
        string *data = csv.split_line("marks.csv", i, DELIMETER);
        if (data[2] == room && data[0] == date && data[1] == time)
        {
            cout << "Marks already uploaded for this class" << endl;
            return;
        }
    }
```

```
    char ch;
    cout << "Do you want to upload marks for all students in this class? (y/n): ";
    cin >> ch;
    if (ch == 'y'){
    for (int i = 0; i < csv.total_lines("students.csv"); i++)
    {
        string *st_data = csv.split_line("students.csv", i, DELIMETER);
        if (st_data[8] == room)
        {
            cout << "Roll No: " << st_data[0];
            cout << " | Name: " << st_data[1];
            cout << " | Room: " << st_data[8] << endl;
            cout << "Enter marks: ";
            string marks;
            cin >> marks;
            string final_data[] = {date, time, room, st_data[0], marks, unique_id,
course};
            csv.write("marks.csv", final_data, 7, DELIMETER);
        }
    }}
}

void Teacher::take_attendance()
{
    CSV csv;
    string date, room, time, course;
    int c = 0;
    cin.ignore();
    cout << "Enter date: ";
    getline(cin, date);
    cout << "Enter time: ";
    getline(cin, time);
    string unique_id = csv.split_line("teachers.csv", tcsv_line, DELIMETER)[0];
    for (int i = 0; i < csv.total_lines("timetable.csv"); i++)
    {
        string *data = csv.split_line("timetable.csv", i, DELIMETER);
        if (data[1] == unique_id && data[2] == date && data[4] == time)
        {
            cout << "Room: " << data[0];
            cout << " | Date: " << data[2];
            cout << " | Day: " << data[3];
            cout << " | Time: " << data[4];
            cout << " | Course: " << data[5] << endl;
            room = data[0];
            course = data[5];
```

```
            c++;
            break;
        }
    }
    if (c == 0)
    {
        cout << "No classes on this date or time" << endl;
        return;
    }
    for (int i = 0; i < csv.total_lines("attendance.csv"); i++)
    {
        string *data = csv.split_line("attendance.csv", i, DELIMETER);
        if (data[2] == room && data[0] == date && data[1] == time)
        {
            cout << "Attendance already taken for this class" << endl;
            return;
        }
    }

    for (int i = 0; i < csv.total_lines("students.csv"); i++)
    {
        string *st_data = csv.split_line("students.csv", i, DELIMETER);
        if (st_data[8] == room)
        {
            cout << "Roll No: " << st_data[0];
            cout << " | Name: " << st_data[1];
            cout << " | Room: " << st_data[8] << endl;
            cout << "Enter attendance (P/A): ";
            string attendance;
            cin >> attendance;
            if (attendance != "P" && attendance != "A")
            {
                cout << "Invalid attendance" << endl;
                return;
            }
            string final_data[] = {date, time, room, st_data[0], attendance, unique_id,
course};
            csv.write("attendance.csv", final_data, 7, DELIMETER);
        }
    }
}

void Teacher::teacher_timetable()
{
    CSV csv;
```

```
    string date;
    int c = 0;
    cin.ignore();
    cout << "Enter date: ";
    getline(cin, date);
    string unique_id = csv.split_line("teachers.csv", tcsv_line, DELIMETER)[0];
    for (int i = 0; i < csv.total_lines("timetable.csv"); i++)
    {
        string *data = csv.split_line("timetable.csv", i, DELIMETER);
        if (data[1] == unique_id && data[2] == date)
        {
            cout << "Room: " << data[0];
            cout << " | Date: " << data[2];
            cout << " | Day: " << data[3];
            cout << " | Time: " << data[4];
            cout << " | Course: " << data[5] << endl;
            c++;
        }
    }
    if (c == 0)
    {
        cout << "No classes on this date" << endl;
    }
}

void Teacher::teacher_salary() {
    CSV csv;
    cin.ignore();
    for (int i = 0; i < csv.total_lines("fees.csv"); i++)
    {
        string *data = csv.split_line("fees.csv", i, DELIMETER);
        if (data[0] == unique_id)
        {
            cout << "Salary: " << data[1] << endl;
            return;
        }
    }
    cout << "No fees record found" << endl;
}

void Teacher::teacher_logout()
{
    cout << tname << " logged out" << endl;
}
```

App.h

```h
#ifndef APP_H
#define APP_H

#include "Admin.h"

class App : public Admin
{
public:
    void start();
    void admin_menu();
    void teacher_menu();
    void student_menu();
};

#endif
```

app.cpp

```cpp
#include <iostream>
#include <string>
#include <fstream>

#include "App.h"
#include "Csv_helper.h"

using namespace std;

void App::start()
{
    CSV csv;
    int ch;
    while (1)
    {
        cout << "Welcome to College Management System" << endl;
        if (csv.total_lines("notices.csv") > 0)
        {
            cout << "Notice: " << csv.split_line("notices.csv",
csv.total_lines("notices.csv") - 1, ',')[0] << ": " << csv.split_line("notices.csv",
csv.total_lines("notices.csv") - 1, ',')[1] << endl;
        }
```

```
        cout << "Please select your role" << endl;
        cout << "1. Admin" << endl;
        cout << "2. Teacher" << endl;
        cout << "3. Student" << endl;
        cout << "4. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> ch;
        switch (ch)
        {
        case 1:
            if (admin_login())
            {
                admin_menu();
            }
            break;
        case 2:
            if (teacher_login())
            {
                teacher_menu();
            }
            break;
        case 3:
            if (student_login())
            {
                student_menu();
            }
            break;
        case 4:
            cout << "Exiting..." << endl;
            exit(0);
            break;
        default:
            cout << "Invalid choice" << endl;
            break;
        }
    }
}

void App::admin_menu()
{
    int ch;
    while (1)
    {
        cout << "Welcome Admin" << endl;
        cout << "1. Add Student" << endl;
```

```
        cout << "2. Add Teacher" << endl;
        cout << "3. Update Fee" << endl;
        cout << "4. Add Fee" << endl;
        cout << "5. Add Timetable" << endl;
        cout << "6. Update Attendance" << endl;
        cout << "7. Update Marks" << endl;
        cout << "8. Add Notice" << endl;
        cout << "9. Remove Student" << endl;
        cout << "10. Remove Teacher" << endl;
        cout << "11. Logout" << endl;
        cout << "Enter your choice: ";
        cin >> ch;
        switch (ch)
        {
        case 1:
            add_student();
            break;
        case 2:
            add_teacher();
            break;
        case 3:
            update_fee();
            break;
        case 4:
            add_fee();
            break;
        case 5:
            add_timetable();
            break;
        case 6:
            update_attendance();
            break;
        case 7:
            update_marks();
            break;
        case 8:
            add_notice();
            break;
        case 9:
            remove_student();
            break;
        case 10:
            remove_teacher();
            break;
        case 11:
```

```
            admin_logout();
            return;
            break;
        default:
            cout << "Invalid choice" << endl;
            break;
        }
    }
}

void App::student_menu()
{
    int ch;
    while (1)
    {
        cout << "Welcome Student" << endl;
        cout << "1. Display Profile" << endl;
        cout << "2. Update Password" << endl;
        cout << "3. View Marks" << endl;
        cout << "4. View Attendance" << endl;
        cout << "5. View Timetable" << endl;
        cout << "6. View Fee" << endl;
        cout << "7. Logout" << endl;
        cout << "Enter your choice: ";
        cin >> ch;
        switch (ch)
        {
        case 1:
            student_display();
            break;
        case 2:
            student_update();
            break;
        case 3:
            student_marks();
            break;
        case 4:
            student_attendance();
            break;
        case 5:
            student_timetable();
            break;
        case 6:
            student_fee();
            break;
```

```
        case 7:
            student_logout();
            return;
            break;
        default:
            cout << "Invalid choice" << endl;
            break;
        }
    }
}

void App::teacher_menu()
{
    int ch;
    while (1)
    {
        cout << "Welcome Teacher" << endl;
        cout << "1. Display Profile" << endl;
        cout << "2. Update Password" << endl;
        cout << "3. Upload Marks" << endl;
        cout << "4. Upload Attendance" << endl;
        cout << "5. View Timetable" << endl;
        cout << "6. View Salary" << endl;
        cout << "7. Logout" << endl;
        cout << "Enter your choice: ";
        cin >> ch;
        switch (ch)
        {
        case 1:
            teacher_display();
            break;
        case 2:
            teacher_update();
            break;
        case 3:
            upload_marks();
            break;
        case 4:
            take_attendance();
            break;
        case 5:
            teacher_timetable();
            break;
        case 6:
            teacher_salary();
```

```
            break;
        case 7:
            teacher_logout();
            return;
            break;
        default:
            cout << "Invalid choice" << endl;
            break;
        }
    }
}
```

main.cpp

```
#include <iostream>

#include "App.h"

// Default Password: unique_id/roll_no + name
// Admin username_password: admin_password

// g++ -std=c++11 ./*.cpp -o main
// chmod +x main
// ./main

using namespace std;

int main(void)
{
    App app;
    app.start();
    return 0;
}
```

*Go to: https://github.com/Shevilll/CollegeManagementSystem for source code.

APPLICATION OUTPUT

Commands

```
Ahmads-MacBook-Air:CollegeManagementSystem ahmadfaraz$ g++ -std=c++11 ./*.cpp -o main
Ahmads-MacBook-Air:CollegeManagementSystem ahmadfaraz$ chmod +x main
Ahmads-MacBook-Air:CollegeManagementSystem ahmadfaraz$ ./main
```

Main Menu

```
Welcome to College Management System
Please select your role
1. Admin
2. Teacher
3. Student
4. Exit
Enter your choice:
```

Admin Portal

```
Enter your choice: 1
Enter username: admin
Enter password: password
Login successful
Welcome Admin
1. Add Student
2. Add Teacher
3. Update Fee
4. Add Fee
5. Add Timetable
6. Update Attendance
7. Update Marks
8. Add Notice
9. Remove Student
10. Remove Teacher
11. Logout
Enter your choice:
```

Adding Student

```
Enter your choice: 1
Enter student roll no: 101
Enter student name: Nick
Enter student branch: BTech
Enter student semester: 5
Enter student email: nick@univ.com
Enter student phone: 931-341-3425
Enter student address: Block-C-Street
Enter room number: 121
```

Adding Teacher

```
Enter your choice: 2
Enter teacher unique id: 901
Enter teacher name: Samuel
Enter teacher degree: MTech
Enter teacher email: samuel@tuniv.com
Enter teacher phone: 324-243-5443
Enter teacher address: xyz-street
```

Adding Timetable

```
Enter your choice: 5
Enter teacher unique id: 901
Enter date: 20/04/2024
Enter day: Saturday
Enter time: 08:00:00-09:00:00
Enter room number: 121
Enter course: Python
```

Adding Fees / Salary

```
Enter your choice: 4
Enter roll_no or unique_id: 901
Enter fees: 120000
```

```
Enter your choice: 4
Enter roll_no or unique_id: 101
Enter fees: 48000
```

Teacher Portal

```
Enter your choice: 11
Admin logged out successfully
Welcome to College Management System
Please select your role
1. Admin
2. Teacher
3. Student
4. Exit
Enter your choice: 2
Enter your unique id: 901
Enter your password: 901Samuel
Login successful
Welcome Teacher
1. Display Profile
2. Update Password
3. Upload Marks
4. Upload Attendance
5. View Timetable
6. View Salary
7. Logout
Enter your choice:
```

Displaying Profile and Salary

```
Enter your choice: 1
Unique ID: 901
Name: Samuel
Degree: MTech
Email: samuel@tuniv.com
Phone: 324-243-5443
Address: xyz-street
```

```
Enter your choice: 6
Salary: 120000
```

Displaying Timetable

```
Enter your choice: 5
Enter date: 20/04/2024
Room: 121 | Date: 20/04/2024 | Day: Saturday | Time: 08:00:00-09:00:00 | Course: Python
```

Updating Password

```
Enter your choice: 2
Enter old password: 901Samuel
Enter new password: samuelxyz
Enter new password again: samuelxyz
```

Uploading Attendance

```
Enter your choice: 4
Enter date: 20/04/2024
Enter time: 08:00:00-09:00:00
Room: 121 | Date: 20/04/2024 | Day: Saturday | Time: 08:00:00-09:00:00 | Course: Python
Roll No: 101 | Name: Nick | Room: 121
Enter attendance (P/A): P
```

Uploading Marks

```
Enter your choice: 3
Enter date: 20/04/2024
Enter time: 08:00:00-09:00:00
Room: 121 | Date: 20/04/2024 | Day: Saturday | Time: 08:00:00-09:00:00 | Course: Python
Do you want to upload marks for all students in this class? (y/n): y
Roll No: 101 | Name: Nick | Room: 121
Enter marks: 98
```

Student Portal

```
Enter your choice: 7
Samuel logged out
Welcome to College Management System
Please select your role
1. Admin
2. Teacher
3. Student
4. Exit
Enter your choice: 3
Enter your roll no: 101
Enter your password: 101Nick
Login successful
Welcome Student
1. Display Profile
2. Update Password
3. View Marks
4. View Attendance
5. View Timetable
6. View Fee
7. Logout
Enter your choice:
```

Displaying Profile

```
Enter your choice: 1
Roll No: 101
Name: Nick
Branch: BTech
Semester: 5
Email: nick@univ.com
Phone: 931-341-3425
Address: Block-C-Street
Room: 121
```

Displaying Timetable

```
Enter your choice: 5
Enter date: 20/04/2024
Time: 08:00:00-09:00:00 | Day: Saturday | Course: Python | Room: 121 | Teacher: Samuel
```

Displaying Fees

```
Enter your choice: 6
Fees: 48000
```

Displaying Attendance

```
Enter your choice: 4
Enter date: 20/04/2024
Date: 20/04/2024 | Time: 08:00:00-09:00:00 | Room: 121 | Course: Python | Teacher: Samuel | Status: P
```

Displaying Marks

```
Enter your choice: 3
Course: Python | Marks: 98
```

Updating Password

```
Enter your choice: 2
Enter old password: 101Nick
Enter new password: nickbtech
Enter new password again: nickbtech
```

Updating Attendance

```
Enter your choice: 6
Enter student roll no: 101
Enter date: 20/04/2024
Enter time: 08:00:00-09:00:00
Are you sure you want to update this attendance? (y/n):
Old status: P
y
Enter new status: A
Attendance updated successfully
```

```
Enter your choice: 4
Enter date: 20/04/2024
Date: 20/04/2024 | Time: 08:00:00-09:00:00 | Room: 121 | Course: Python | Teacher: Samuel | Status: A
```

Updating Marks

```
Enter your choice: 7
Enter student roll no: 101
Enter date: 20/04/2024
Enter time: 08:00:00-09:00:00
Enter marks: 0
Are you sure you want to update this marks? (y/n):
Old marks: 98
y
Marks updated successfully
```

```
Enter your choice: 3
Course: Python | Marks: 0
```

Adding Notice

```
Enter your choice: 8
Enter date: 25/05/2024
Enter notice description: roll_no 101 and unique_id 901 - removed
```

```
Welcome to College Management System
Notice: 25/05/2024: roll_no 101 and unique_id 901 - removed
Please select your role
1. Admin
2. Teacher
3. Student
4. Exit
Enter your choice:
```

Removing Student

```
Enter your choice: 9
Enter student roll no: 101
Are you sure you want to remove this student? (y/n):
Roll No: 101
Name: Nick
Branch: BTech
Semester: 5
Email: nick@univ.com
Phone: 931-341-3425
Address: Block-C-Street
Room: 121
y
Student removed successfully
```

Removing Teacher

```
Enter your choice: 10
Enter teacher unique_id: 901
Are you sure you want to remove this teacher? (y/n):
Unique ID: 901
Name: Samuel
Degree: MTech
Email: samuel@tuniv.com
Phone: 324-243-5443
Address: xyz-street
y
Teacher removed successfully
```

Updating Fees / Salary

```
Enter your choice: 3
Enter roll_no or unique_id: 101
Enter new fees: 0
Are you sure you want to update this fee? (y/n):
Old fee: 48000
y
Fee updated successfully
```

```
Enter your choice: 3
Enter roll_no or unique_id: 901
Enter new fees: 0
Are you sure you want to update this fee? (y/n):
Old fee: 120000
y
Fee updated successfully
```

Exiting

```
Enter your choice: 11
Admin logged out successfully
Welcome to College Management System
Notice: 25/05/2024: roll_no 101 and unique_id 901 - removed
Please select your role
1. Admin
2. Teacher
3. Student
4. Exit
Enter your choice: 4
Exiting...
```