**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

#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

#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();

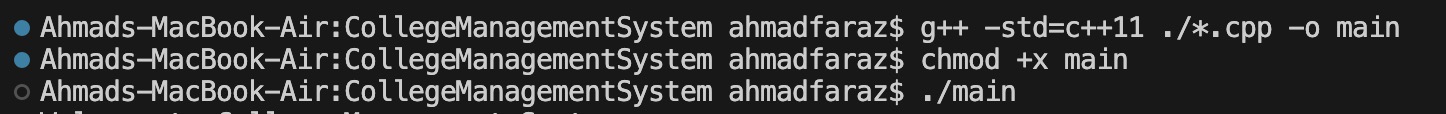
return 0;

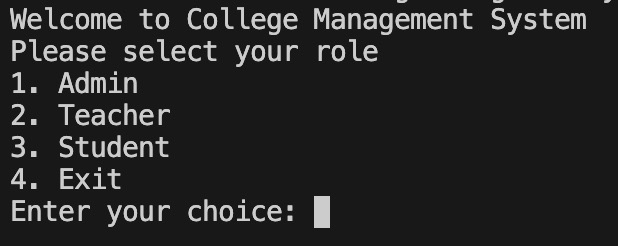
}

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

**APPLICATION OUTPUT**

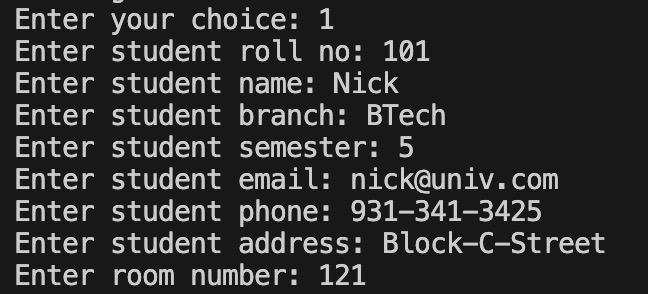
Commands

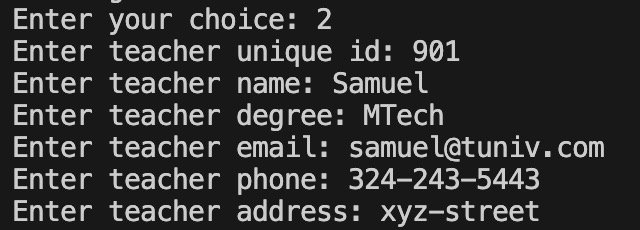
****

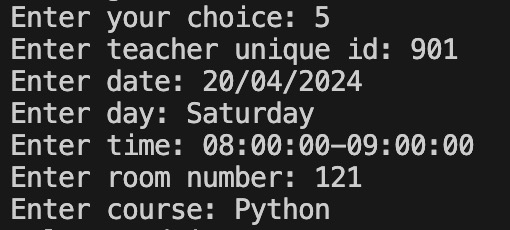
Main Menu

A screen shot of a computer

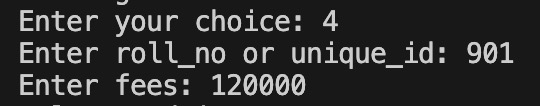
Description automatically generatedAdmin Portal

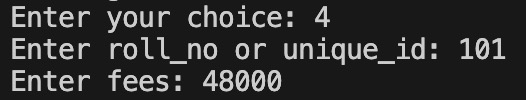
Adding Student

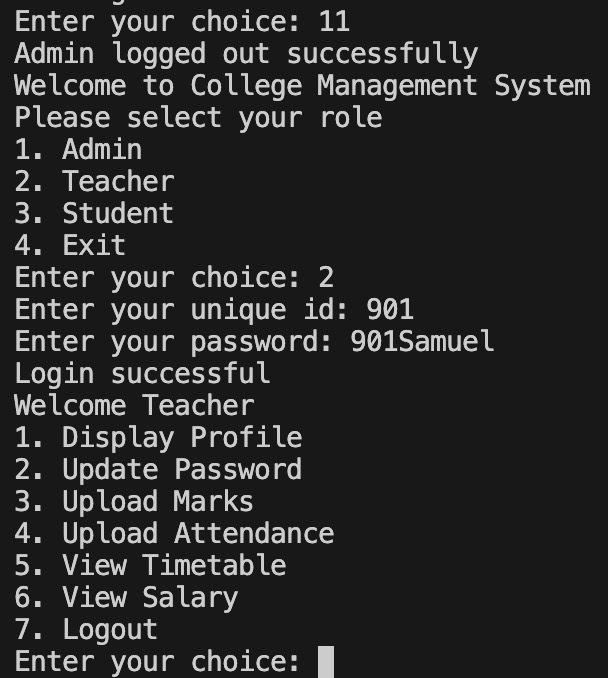
Adding Teacher

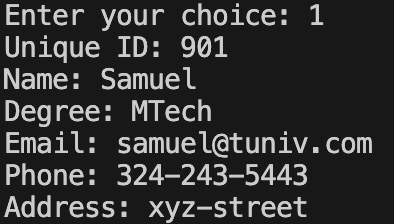
Adding Timetable

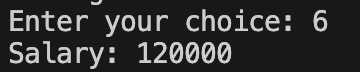
Adding Fees / Salary

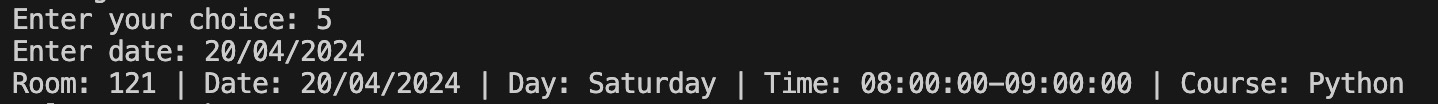


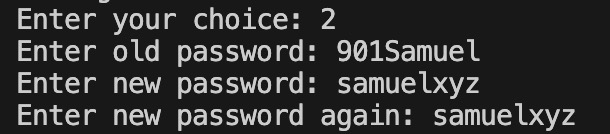


Teacher Portal

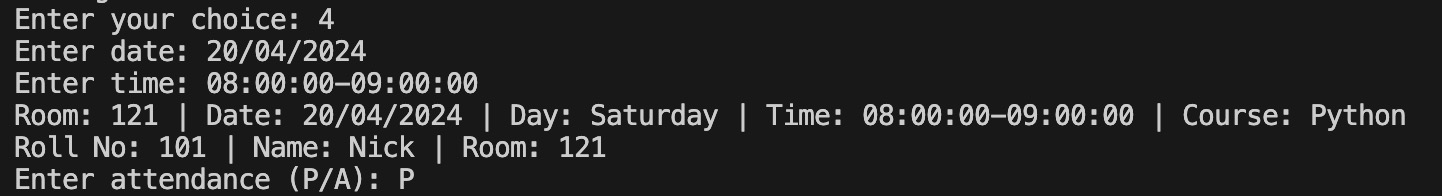
Displaying Profile and Salary



Displaying Timetable

Updating Password

Uploading Attendance



Uploading Marks

A black and white text

Description automatically generated

A screenshot of a computer screen

Description automatically generatedStudent Portal

A screen shot of a computer

Description automatically generatedDisplaying Profile

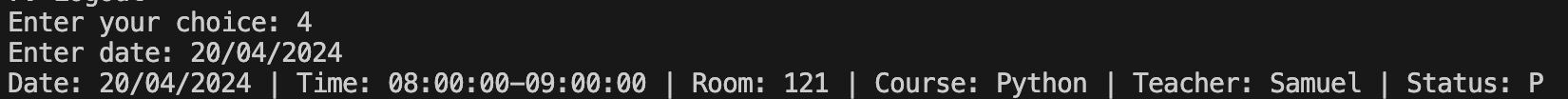
Displaying Timetable

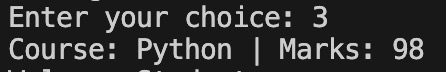


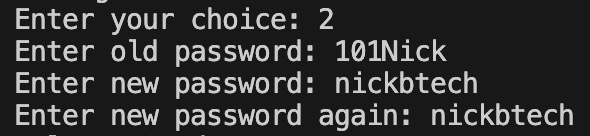
A black and white sign with white text

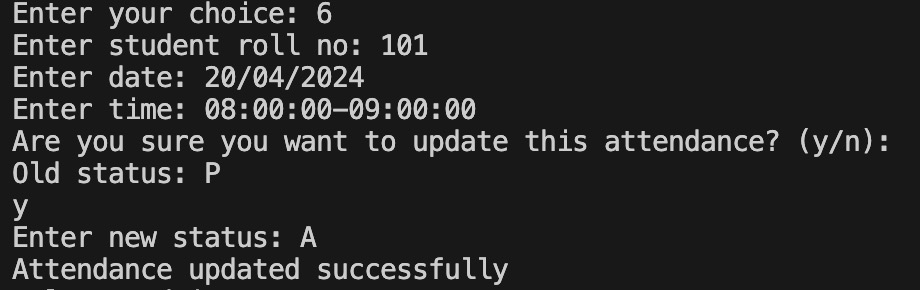
Description automatically generatedDisplaying Fees

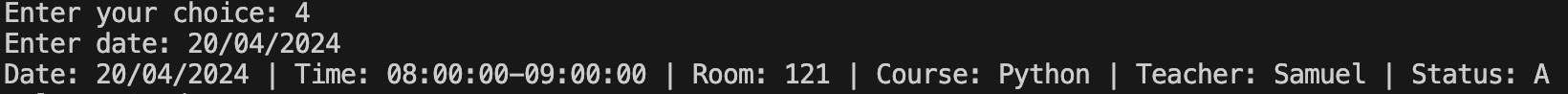
Displaying Attendance



Displaying Marks

Updating Password

Updating Attendance



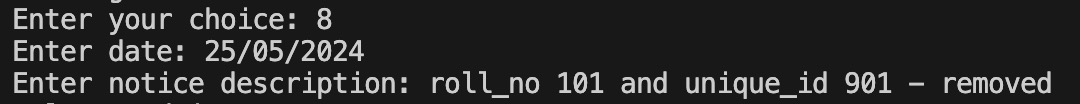
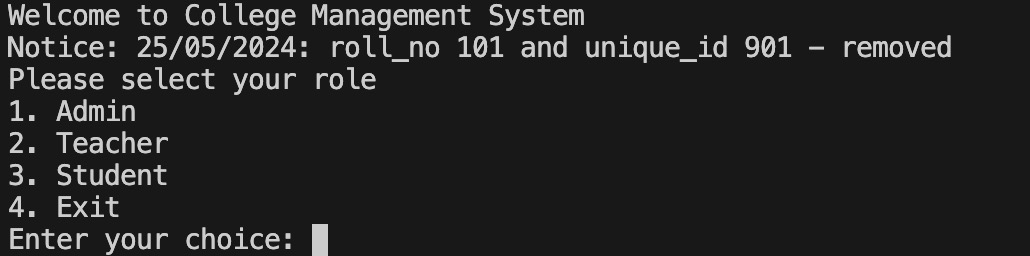
A screen shot of a computer

Description automatically generatedUpdating Marks

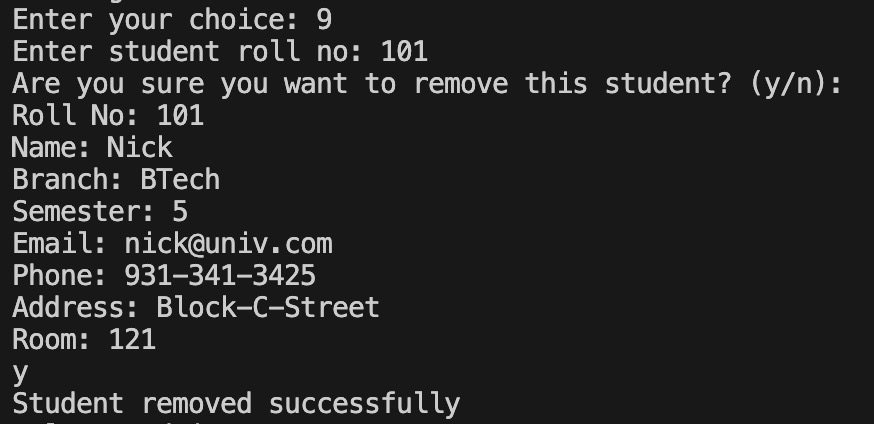
A black background with white text

Description automatically generated

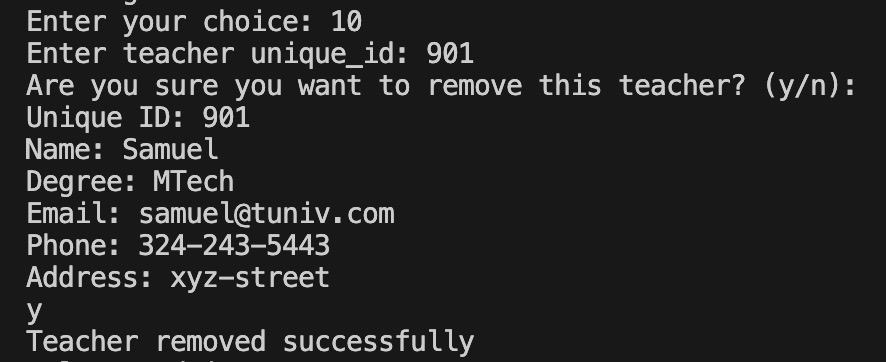
Adding Notice



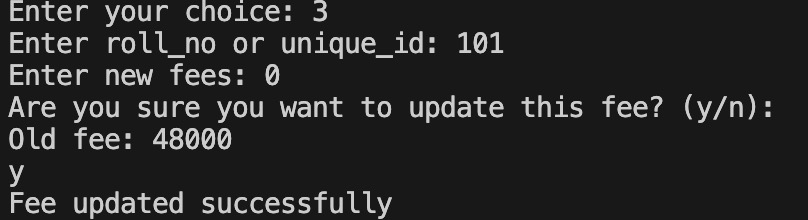
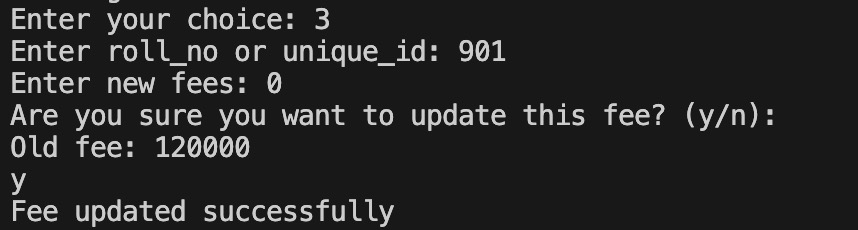
Removing Student



Removing Teacher



Updating Fees / Salary



Exiting

