**Logo, company name

Description automatically generated**

**Department of (Computer Science)**

**Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur, Pakistan**

**COMP-112L Data Structure** **& Algorithm Lab**

**FINAL PROJECT REPORT**

**\*CMS PORTAL\***

**Class: BS Computer Science**

**Name: Ahmed Raza**

**Registration No.: B20F0436CS031**

**Semester: 4th**

**Submission Date: 13 June 2022**

**Submitted to: Engr. Rafi-Ullah**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Instructor Signature**

Contents

[EXECUTIVE SUMMARY 3](#_Toc106204330)

[Chapter 1: Introduction 4](#_Toc106204331)

[Background 5](#_Toc106204332)

[1.2. Motivations and Challenges 6](#_Toc106204333)

[1.3. Goals/Scope/Objectives 7](#_Toc106204334)

[Objectives 8](#_Toc106204335)

[1.4. Project Plan 9](#_Toc106204336)

[1.4.1. Work Breakdown Structure 9](#_Toc106204337)

[User Classes and Characteristics 10](#_Toc106204338)

[1.5. System Features 11](#_Toc106204339)

[C++ CODE 12](#_Toc106204340)

[OUTPUT 49](#_Toc106204341)

# EXECUTIVE SUMMARY

I’m developing a C++ program for university with some outstanding highlights. Program is fully developing on OOP & Data Structure. The program name is College Management System. In this program I provide three dashboards 1st for Admin, 2nd for Teacher and 3rd for students. In this website we gave fully access to admin. Admin can add, delete, update and view data about teacher, student in detail from the database. Teacher can also view his record of attendance, his salary record and, he can view his personal data and he can change his personal information and teacher can also add student exam data of the students and they can change in the short time after some time they cannot change the exam data. Teacher can also add the attendance of the student can also view his record of attendance, his fee record, his exam record and, he can view his personal data and he can change his personal information. Admin, Teacher, and Student have different three dashboards where they can play their task. Admin can change the record modify and play everything with the data, but teacher and student can just view their record does not change anything in the record except their personal information. Personal information they can change.

## Chapter 1: Introduction

Management system is a very important and essential part of any educational institute. The Project defines all management of College System. College management system means it controls all the management tasks and functions. It performs all functions which is require to any Educational Institute. Admin can view and change all record of the college. Teacher can also view their information and they can change personal information. Students can search their result. They can see more information about the college on the college management system software. They can see the timetable of the lectures in the college. They can see also their exam data fee data and all personal information. Students can print out their roll no slips with just one click. Student can view rules and regulations for college in details. Students can view his / her Fee details for previous semester and current semester. Student can view his/her all fee package admission details the details of courses completed and pending courses and repeating courses. Students. can view academic calendar. Students can fill their course evaluation forms at the end of each semester and can give suggestions to make the program and educational environment better.

Teachers have been authorized at this site to update results (including assignments, quiz, midterm, and final term marks details), attendance on daily bases. Teacher can view all details of each student of any semester.

Management system is a very important and essential part of any educational institute. The Project defines all management of College System. College management system means it controls all the management tasks and functions. It performs all functions which is require to any Educational Institute. Admin can view and change all record of the college. Teacher can also view their information and they can change personal information. Students can search their result. They can see more information about the college on the college management system software.

## Background

As the name suggests "College Management System" suggests the application will manage the different functions about the students, teachers, and other staff of the college, searching the result of the student, admin can enter, save, search, and modify the record of teachers and his salary, students and their fee record, record about the exam. To keep the information, correct and fast, it will manage all the services correct and fast.

Students can search their result. They can see more information about the college on the college management system software. They can see the timetable of the lectures in the college. They can see also their exam data fee data and all personal information. Students can print out their roll no slips with just one click. Student can view rules and regulations for college in details. Students can view his/her Fee details for previous semester and current semester. Student can view his /her all fee package admission details the details of courses completed and pending courses and repeating courses. Students can view academic calendar. Students can fill their course evaluation forms at the end of each semester and can give suggestions to make the program and educational environment better.

Teachers have been authorized at this site to update results (including assignments, quiz, midterm, and final term marks details), attendance on daily bases. Teacher can view all details of each student of any semester.

We aim to manage all the work of college using the College Management System.

The College Management System is the educational management system.

At the time to run the Educational Institute the education is the basic and main responsibility of the College.

It helps the admin to keep the record, manage and control the entire data of the college.

It helps the admin to access the data easily.

Student can view rules and regulations for college in details. Students can view his/her Fee details for previous semester and current semester.

Student can view his/her all fee package admission details the details of courses completed and pending courses and repeating courses.

Students can view academic calendar. Students can fill their course evaluation forms at the end of each semester and can give suggestions to make the program and educational environment better.

## 1.2. Motivations and Challenges

We must face many problems int the old system that is painful for students, teacher and other staff if we want to find any record in old system that was take much more time old college system was a time wasted, facing problem in every field it was difficult to store to a huge amount of data about the college. Then we feel need a system who manage all management system of the college. The system gives a quick response on the request. Then we develop a College Management System website who manage all the management of the system. It contains all record about the teacher. It contains all record about courses. It contains all record about other staff of the college. It contains all record about the student and exam record. It works smoothly. The system gives us a quick and accurate and correct result which we want to search. It is less time-consuming software. It helps us in every field of the college. Now everything will be managing easily.

## 1.3. Goals/Scope/Objectives

The Goals are as follows:

Provide better services through Application software "College Management System" than the old system of college.

Students can search their result. They can see more information about the college on the college management system software. They can see the timetable of the lectures in the college. They can see also their exam data fee data and all personal information. Students can print out their roll no slips with just one click. Student can view rules and regulations for college in details. Students can view his / her Fee details for previous semester and current semester. Student can view his/her all fee package admission details the details of courses completed and pending courses and repeating courses. Students can view academic calendar. Students can fill their course evaluation forms at the end of each semester and can give suggestions to make the program and educational environment better.

Teachers have been authorized at this site to update results (including assignments, quiz, midterm, and final term marks details), attendance on daily bases. Teacher can view all details of each student of any semester.

Aim to manage all the work of college using the College Management System.

The College Management System is the educational management system.

At the time to run the Educational Institute the education is the basic and main responsibility of the College.

It helps the admin to keep the record, manage and control the entire data of the college.

It helps the admin to access the data easily.

Student can view rules and regulations for college in details. Students can view his / her Fee details for previous semester and current semester.

Student can view his / her all fee package admission details the details of courses completed and pending courses and repeating courses.

Students can view academic calendar. Students can fill their course evaluation forms at the end of each semester and can give suggestions to make the program and educational environment better.

## Objectives

The Objectives are as follows:

* We can add, modify, search, and delete data about teachers, student, and other staff of the college.
* It contains the teacher and other staff's personal information.
* It contains the teacher and other staff's attendance.
* It saves the data of teachers and other staff's salary.
* It contains the personal information.
* It contains attendance of the student.
* It manages the fee of students.
* It manages the exam management system.
* It contains more information about the college.
* It also contains the timetable of the lectures.
* Student can view rules and regulations for college in details.

## 1.4. Project Plan

College Management System provide us the facility of electronic media content. Employee and students now they can be checking their detail on their electronic devices like phone, tablet, and pcs. It saves the time of admin student and teachers. It is also very cost effective if we compare this system to current system of the college.

## 1.4.1. Work Breakdown Structure

Front Look: On the front page of the College Management System contain some important information about the college.

**Login System:** It manages the Login of admin, employee, and student.

**Admin System:** It perform all functions of the admin.

**Employee System:** It manages the employee detail.

**Student System:** It manages the student's details.

**Exam System:** It manages the exam system.

## User Classes and Characteristics

This website can be used by the following:

**Students:**

Registered students can use the website to get their required information.

**Teachers:**

Teachers can update the related data on the website.

**Management:**

Management can use the site to handle their related tasks.

**Overall population:**

Overall population can visit the website to know about college, can view their required information such as admission criteria etc.

## 1.5. System Features

**Account**: User can make his/her account on the website.

**Login**: After creating account user can log into the website.

**Update**: Specific user can update the related data.

**View detail**: User can view his/her required details on the website.

## C++ CODE

//=================================================  HEADER SECTION

#include <iostream>

#include <fstream>

#include <string>

#include <cstring>

#include <iomanip>

#include <conio.h>

#include <stdlib.h>

//================================================= DEFINING STANDARD NAMESPACE

using namespace std;

//================================================= CLASSES SECTION for menus and passwords

class Password

{

private:

    string ad\_name;             // ----> admin names

    string ad\_pw;               // ----> admin passwords

    string adm\_name;                                    // ----> strings to hold input credentials

    string adm\_pw;

    string try\_top;

    ifstream admin;

    ifstream stu,marks;

    ofstream mar;

    bool login;

    bool log;

public:

    bool isLogin()                                          // ----> Function for authenticating the admin login through saved credentials in a file.

    {

        system("cls");

        Password data[3];                                       // ----> Structure Array for storing the read data from the file

        admin.open("Admin\_logs.txt");                       // ----> Opening the file for reading

        for (int i = 0; i < 3; i++)

        {

            admin >> data[i].ad\_name;

            admin >> data[i].ad\_pw;

        }

        login = false;

        cin.ignore();

        cout << " Enter the name of admin: ";               // ----> taking input for credentials

        getline(cin, adm\_name);

        cout << " Enter the admin password: ";

        getline(cin, adm\_pw);

        for (int i = 0; i < 3; i++)

        {

            if (adm\_name == data[i].ad\_name && adm\_pw == data[i].ad\_pw)           // ----> checking for password and user name match

            {

                cout << "\n\n You are successfully logged in ADMIN mode \n\n";

                login = true;

            }

        }

        admin.close();

        return login;                                                           // ----> returning the login value to the function

    }

    bool isLoginTry()

    {

        cout << endl << " Wrong Ussername or Password  Entered!!!!!\n\n" << endl;

        log = false;

        cout << " Do you want to choose another choice ? (y/n)    ";

        cin >> try\_top;                 // ----> string input for decision

        if (try\_top == "y" || try\_top == "Y")

        {

            system("cls");

            if (isLogin() == 1)

            {

                log = true;

                system("pause");

                return log;

            }

            else

            {

                isLoginTry();

            }

        }

        else if (try\_top == "n" || try\_top == "N")

        {

            system("pause");

            exit(0);

        }

        else

        {

            cout << endl << endl << " Wrong choice entered!! The system is exiting due to the wrong format used in admin login. " << endl << endl;

            system("pause");

            exit(0);

        }

        return log;

    }

    bool isLogin1()                                         // ----> Function for authenticating the admin login through saved credentials in a file.

    {

        int i = 0;

        system("cls");

        stu.open("student.txt");                        // ----> Opening the file for reading

        marks.open("Marks.txt");

        mar.open("Marksx.txt");

        int realid,batch,cour;

        char realpass[30];

        char name[30],name1[10], name2[50];

        login = false;

        int idm;

        cout << " Enter id : ";             // ----> taking input for credentials

        cin >> idm;

        cin.ignore();

        cout << " Enter password: ";

        getline(cin, adm\_pw);

        while (!stu.eof())

        {

            stu >> realid;

            stu.ignore(10, ',');

            stu.getline(realpass, 30, '-');

            stu.getline(name, 30, ',');

            stu.getline(name1, 10, ',');

            stu >> batch;

            stu.ignore();

            marks.ignore(100, '-');

            marks.getline(name2, 60, '!');

            marks >> cour;

            marks.ignore(10, ',');

            marks.ignore(10, '\n');

            if (realpass == adm\_pw && idm == realid)

            {

                mar << realid << "," << name << "-"  << name2 << "!" << cour<<",";

                while (cour > 0)

                {

                    mar << "$";

                    cour--;

                }

                cout << "\n\n You are successfully logged in Student section \n\n";

                system("pause");

                login = true;

            }

        }

        marks.close();

        mar.close();

        stu.close();

        return login;                                                           // ----> returning the login value to the function

    }

    bool isLoginTry1()

    {

        cout << endl << " Wrong Ussername or Password  Entered!!!!!\n\n" << endl;

        log = false;

        cout << " Do you want to choose another choice ? (y/n)    ";

        cin >> try\_top;                 // ----> string input for decision

        if (try\_top == "y" || try\_top == "Y")

        {

            system("cls");

            if (isLogin1() == 1)

            {

                log = true;

                system("pause");

                return log;

            }

            else

            {

                isLoginTry1();

            }

        }

        else if (try\_top == "n" || try\_top == "N")

        {

            system("pause");

            exit(0);

        }

        else

        {

            cout << endl << endl << " Wrong choice entered!! The system is exiting due to the wrong format used. " << endl << endl;

            system("pause");

            exit(0);

        }

        return log;

    }

    bool isLogin2()                                         // ----> Function for authenticating the admin login through saved credentials in a file.

    {

        int i = 0;

        system("cls");

        stu.open("teacher.txt");                        // ----> Opening the file for reading

        mar.open("teacher\_open.txt");

        int realid, id2;

        char realpass[30];

        char name[30], name2[50];

        login = false;

        int idm;

        cout << " Enter id : ";             // ----> taking input for credentials

        cin >> idm;

        cin.ignore();

        cout << " Enter password: ";

        getline(cin, adm\_pw);

        while (!stu.eof())

        {

            stu >> realid;

            stu.ignore(10, ',');

            stu.getline(realpass, 30, '-');

            stu.getline(name, 30, '$');

            stu >> id2;

            stu.ignore(10, '.');

            stu.getline(name2, 50, '\n');

            if (realpass == adm\_pw && idm == realid)

            {

                mar  << realid << "," << realpass << "-" << name << "$" << id2 << "." << name2;

                cout << "\n\n You are successfully logged in Teacher section \n\n";

                system("pause");

                login = true;

            }

        }

        mar.close();

        stu.close();

        return login;                                                           // ----> returning the login value to the function

    }

    bool isLoginTry2()

    {

        cout << endl << " Wrong Ussername or Password  Entered!!!!!\n\n" << endl;

        log = false;

        cout << " Do you want to choose another choice ? (y/n)    ";

        cin >> try\_top;                 // ----> string input for decision

        if (try\_top == "y" || try\_top == "Y")

        {

            system("cls");

            if (isLogin2() == 1)

            {

                log = true;

                system("pause");

                return log;

            }

            else

            {

                isLoginTry2();

            }

        }

        else if (try\_top == "n" || try\_top == "N")

        {

            system("pause");

            exit(0);

        }

        else

        {

            cout << endl << endl << " Wrong choice entered!! The system is exiting due to the wrong format used. " << endl << endl;

            system("pause");

            exit(0);

        }

        return log;

    }

};

class Menu

{

private:

public:

    void dis()

    {

        system("COLOR 70");

        cout << endl << endl << "        ";

        for (int c = 0; c<30; c++)                  // ----> The main heading line

        {

            cout << "\*";

        }

        cout << "  \_Welcome to The Course Management System\_  ";

        for (int c = 0; c<30; c++)

        {

            cout << "\*";

        }

        cout << endl;

        cout << endl;

        cout << "\t\t\t\t Please Enter the password to gain access to the system : " << endl << endl;

    }

    void dis1()

    {

        cout << "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n\t\t\t 1. To gain access to admin mode\n\t\t\t 2. To go to the student section\n\t\t\t 3. To go to the teacher section\n\t\t\t 4. To exit the system \n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

        cout << endl << " Please enter your choice (1-4) :   ";

    }

    void dis2()

    {

        cout << "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n\t\t\t\t \*\*\*Admin Mode\*\*\* \n\n\t\t\t 1. To add a student in university database\n\t\t\t 2. To remove a student in university database\n\t\t\t 3. To add a teacher in university database\n\t\t\t 4. To remove a teacher in university database\n\t\t\t 5. To display all Students in university database\n\t\t\t 6. To display all teachers in university database\n\t\t\t 7. To go back to the main menu\n\t\t\t 8. To exit the program \n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

        cout << endl << " Please enter your choice (1-9) :   ";

    }

    void dis3()

    {

        cout << "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n\t\t\t\t \*\*\*Student Section\*\*\* \n\n\t\t\t 1. To see the assigned courses \n\t\t\t 2. To drop an assigned course\n\t\t\t 3. To add a course\n\t\t\t 4. To see the result of all assigned courses \n\t\t\t 5. To go back to the main menu\n\t\t\t 6. To exit the program\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

        cout << endl << " Please enter your choice (1-6) :   ";

    }

    void dis4()

    {

        cout << "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n\t\t\t\t \*\*\*Teacher Section\*\*\* \n\n\t\t\t 1. To enter marks of a student in a course \n\t\t\t 2. To see the list of courses \n\t\t\t 3. To go back to the main menu\n\t\t\t 4. To exit the program\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

        cout << endl << " Please enter your choice (1-5) :   ";

    }

};

class TryPass

{

private:

    string pw;

    int n;

    int a;

public:

    bool trypass()

    {

        n = 0;

        a = 2;

        while (n <= 0)

        {

            cout << "                                              ";

            cin >> pw;                                  // ----> input password string for login to the library

            if (pw == "p")

            {

                system("cls");

                n = 1;

            }

            else

            {

                cout << "\n\t   You entered the wrong password, please try again by pressing another password. " << a << " tries left.\n\n\n";

                n--;

                a--;

            }

            if (n == -3)

            {

                cout << "\n\t\t\t   You tried many times.The system is exiting. " << endl << endl;

                return false;

            }

        }

    }

};

//================================================= MAIN CLASSES SECTION

class Student

{

public:

    char name[30];

    int id;

    char pass[30];

    int batch;

    char degree[10];

    int t\_course;

    char courses[50];

    int sess;

    int mid;

    int final\_m;

};

class Teacher

{

public:

    char name[30];

    int id;

    char pass[30];

    char courses[50];

    int t\_course;

};

class Teacher\_functionality

{

protected:

    class ListNode

    {

    public:

        Teacher data;

        ListNode \*next;

        ListNode(Teacher d)

        {

            data = d;

            next = NULL;

        }

    };

    class linked\_list

    {

    protected:

        Teacher t;

        Student stude;

        ListNode \*head;

        fstream tea, temp, marks, temp1;

        string x[10],x1[10];

    public:

        linked\_list()// This will be called when object of LINKEdLIST is created in start

        {

            head = NULL;

        }

        void appendet(Teacher t)

        {

            tea.open("teacher.txt", ios::app);

            if (head == NULL)

            {

                cout << "Enter the id: ";

                cin >> t.id;

                cin.ignore();

                cout << "Enter the name: ";

                cin.getline(t.name, 30);

                cout << "Set the password for ur id: ";

                cin.getline(t.pass, 30);

                cout << "How many courses: ";

                cin >> t.t\_course;

                cin.ignore();

                cout << "Enter the name of courses u want to assign (courses must be seperated by commas): ";

                cin.getline(t.courses, 60);

                tea << endl << t.id << "," << t.pass << "-" << t.name << "$" << t.t\_course << "." << t.courses;

                head = new ListNode(t);

                tea.close();

                return;

            }

            else

            {

                appendeet(t, head);

            }

        }

        void appendeet(Teacher t, ListNode \*tmp)

        {

            if (tmp->next == NULL)

            {

                cout << "Enter the id: ";

                cin >> t.id;

                cin.ignore();

                cout << "Enter the name: ";

                cin.getline(t.name, 30);

                cout << "Set the password for ur id: ";

                cin.getline(t.pass, 30);

                cout << "How many courses: ";

                cin >> t.t\_course;

                cin.ignore();

                cout << "Enter the name of courses u want to assign (courses must be seperated by commas): ";

                cin.getline(t.courses, 60);

                tea << endl << t.id << "," << t.pass << "-" << t.name << "$" << t.t\_course << "." << t.courses;

                tea.close();

            }

            appendeet(t, tmp->next);

        }

        void displaye()

        {

            cout << endl << endl << "\t\t\t\t\t\t Teacher's Record" << endl << endl;

            cout << endl << endl << right << setw(4) << "ID \t" << left << setw(30) << "Name" << left << setw(20) << "Total courses" << left << setw(60) << "Course names" << endl;

            int f = 0;

            while (f<100)

            {

                cout << "\_";

                f++;

            }

            cout << endl << endl;

            if (head == NULL)

            {

                tea.open("teacher.txt");

                tea >> t.id;

                tea.ignore(10, ',');

                tea.ignore(10, '-');

                tea.getline(t.name, 30, '$');

                tea >> t.t\_course;

                tea.ignore(10, '.');

                tea.getline(t.courses, 50, '\n');

                head = new ListNode(t);

                tea.close();

            }

            display(head);

        }

        void display(ListNode \*tmp)

        {

            tea.open("teacher.txt");

            while (!tea.eof())

            {

                tea >> tmp->data.id;

                tea.ignore(10, ',');

                tea.ignore(10, '-');

                tea.getline(tmp->data.name, 30, '$');

                tea >> tmp->data.t\_course;

                tea.ignore(10, '.');

                tea.getline(tmp->data.courses, 50, '\n');

                cout << left << setw(4) << tmp->data.id << "\t" << left << setw(30) << tmp->data.name  << left << setw(20) << tmp->data.t\_course << left << setw(60) << tmp->data.courses << endl;

                cout << endl;

            }

            tea.close();

            cout << endl << endl;

        }

        void eraseet()

        {

            string s;

            cout << "Name of the teacher which has to deleted: ";

            cin >> s;

            if(head==NULL)

            {

                tea.open("teacher.txt");

                tea >> t.id;

                tea.ignore(10, ',');

                tea.getline(t.pass, 30, '-');

                tea.getline(t.name, 30, '$');

                tea >> t.t\_course;

                tea.ignore(10, '.');

                tea.getline(t.courses, 50, '\n');

                head = new ListNode(t);

                tea.close();

            }

            ListNode \*tmp = head;

            tea.open("teacher.txt");

            temp.open("temp.txt", ios::app);

            int a = 1;

            while (!tea.eof())

            {

                tea >> t.id;

                tea.ignore(10, ',');

                tea.getline(t.pass, 30, '-');

                tea.getline(t.name, 30, '$');

                tea >> t.t\_course;

                tea.ignore(10, '.');

                tea.getline(t.courses, 50, '\n');

                string sr(t.name);

                if (sr == s)

                {

                    tmp->next = new ListNode(t);

                    tmp = tmp->next;

                    a = 0;

                    continue;

                }

                else

                {

                    temp << endl << t.id << "," << t.pass << "-" << t.name << "$" << t.t\_course << "." << t.courses;

                }

                tmp->next = new ListNode(t);

                tmp = tmp->next;

            }

            temp.close();

            tea.close();

            if (a != 0)

            {

                cout << "\n\n\nNot found\n\n";

                return;

            }

            remove("teacher.txt");

            rename("temp.txt", "teacher.txt");

            eraset(s, head);

        }

        void eraset(string s, ListNode \*&tmp)

        {

                string str(tmp->data.name);

                if (str == s)

                {

                    tmp = tmp->next;

                    cout << "\n\n Task done.\n\n ";

                    return;

                }

                eraset(s, tmp->next);

        }

        void see\_assigned1()

        {

            tea.open("teacher\_open.txt");

            tea.ignore(100, '$');

            tea >> t.id;

            tea.ignore(10, '.');

            int j = t.id;

            int i = 0;

            while (j>0)

            {

                getline(tea, x[i], ',');

                i++;

                j--;

            }

            j = 0;

            cout << "\nThe list of assigned courses is following:-\n ";

            for (int i = 0; i < t.id; i++)

                cout << endl << ++j << ". " << x[i] << "\n";

            cout << endl << endl;

            tea.close();

        }

        void enter\_m()

        {

            marks.open("Marks.txt");

            cout << "Enter the name of student you want to enter marks: ";

            string e,f;

            cin >> e;

            int p,tid,i,j,a=0,m;

            while (!marks.eof())

            {

                marks >> t.id;

                marks.ignore(10, ',');

                marks.getline(t.name, 30, '-');

                string str(t.name);

                if (str == e)

                {

                    a = 1;

                    marks.ignore(50, '!');

                    marks >> p;

                    marks.ignore(10, ',');

                }

                marks.ignore(100, '\n');

            }

            marks.close();

            if (a != 1)

            {

                cout << "\n\nNot found\n\n";

            }

            else

            {

                j = p;

                marks.open("Marks.txt");

                while (!marks.eof())

                {

                    marks >> t.id;

                    marks.ignore(10, ',');

                    marks.getline(t.name, 30, '-');

                    string str(t.name);

                    if (str == e)

                    {

                        i = 0;

                        while (j > 0)

                        {

                            getline(marks, x[i], ',');

                            i++;

                            j--;

                        }

                        j = 0;

                        cout << "\nThe list of assigned courses is following:-\n ";

                        for (int i = 0; i < p; i++)

                            cout << endl << ++j << ". " << x[i] << "\n";

                        cout << endl << endl;

                    }

                    marks.ignore(100, '\n');

                }

                marks.close();

                cout << "Enter the name of subject u want to enter marks: ";

                cin >> f;

                tea.open("teacher\_open.txt");

                tea.ignore(100, '$');

                tea >> t.id;

                tea.ignore(10, '.');

                int j = t.id;

                int i = 0;

                while (j>0)

                {

                    getline(tea, x1[i], ',');

                    i++;

                    j--;

                }

                tea.close();

                for (int k = 0; k < t.id; k++)

                {

                    if (f == x1[k])

                    {

                        a = 0;

                        cout << " \n\n yes u r eligible for entering the marks\n\n";

                        int rise = 1,xtra;

                        for (int i = 0; i < p; i++)

                        {

                            if (f == x[i])

                            {

                                break;

                            }

                            rise++;

                        }

                        cout << "Enter sessional marks: ";

                        cin >> stude.sess;

                        cout << "Enter mid marks: ";

                        cin >> stude.mid;

                        cout << "Enter final marks: ";

                        cin >> stude.final\_m;

                        marks.open("Marks.txt");

                        temp.open("temp.txt",ios::app);

                        while (!marks.eof())

                        {

                            marks >> t.id;

                            marks.ignore(10, ',');

                            marks.getline(t.name, 30, '-');

                            marks.getline(t.courses, 60, '!');

                            marks >> tid;

                            marks.ignore(60, '\n');

                            string str(t.name);

                            if (str == e)

                            {

                                temp << endl << t.id << "," << t.name << "-" << t.courses << "!" << tid << ",";

                                rise--;

                                for (int i = 0; i < rise; i++)

                                {

                                    temp << "$";

                                }

                                temp << stude.sess << "," << stude.mid << "," << stude.final\_m << ",";

                                xtra = tid - rise;

                                for (int i = 0; i < xtra; i++)

                                {

                                    temp << "$";

                                }

                            }

                            else

                            {

                                temp << endl << t.id << "," << t.name << "-" << t.courses << "!" << tid << ",";

                                while (tid>0)

                                {

                                    temp << "$";

                                    tid--;

                                }

                            }

                        }

                        marks.close();

                        temp.close();

                        remove("Marks.txt");

                        rename("temp.txt", "Marks.txt");

                        cout << " DONE!!!!!!!!!!!!!!!!!!!!1";

                    }

                }

                if (a != 0)

                {

                    cout << " \n\n u r not eligible for entering the marks\n\n\n";

                    return;

                }

            }

        }

};

    char choice;

    Teacher t;

    linked\_list s;

public:

    void add\_t()

    {

        s.appendet(t);

        cout << " Do u want to see it ? (y/n) ";

        cin >> choice;

        if (choice == 'Y' || choice == 'y')

        {

            s.displaye();

            system("pause");

        }

    }

    void show\_d()

    {

        s.displaye();

        system("pause");

    }

    void del\_t()

    {

        cout << " Do u want to see the list of teachers ? (y/n) ";

        cin >> choice;

        if (choice == 'Y' || choice == 'y')

        {

            s.displaye();

            system("pause");

        }

        s.eraseet();

        cout << " Do u want to see the changes ? (y/n) ";

        cin >> choice;

        if (choice == 'Y' || choice == 'y')

        {

            s.displaye();

            system("pause");

        }

    }

    void see\_assigned\_t()

    {

        s.see\_assigned1();

        system("pause");

    }

    void enter\_marks()

    {

        s.enter\_m();

        system("pause");

    }

};

class Admin

{

protected:

    class ListNode

    {

    public:

        Student data;

        ListNode \*next;

        ListNode(Student d)

        {

            data = d;

            next = NULL;

        }

    };

    class linked\_list {

    protected:

        Student d;

        ListNode \*head;

        fstream stu,temp,marks,temp1;

        string x[10],x1[10];

    public:

        linked\_list()// This will be called when object of LINKEdLIST is created in start

        {

            head = NULL;

        }

        void appende(Student d)

        {

            stu.open("student.txt", ios::app);

            marks.open("Marks.txt", ios::app);

            if (head == NULL)

            {

                cout << "Enter the id: ";

                cin >> d.id;

                cin.ignore();

                cout << "Enter the name: ";

                cin.getline(d.name, 30);

                cout << "Set the password for ur id: ";

                cin.getline(d.pass, 30);

                cout << "Enter the name of degree: ";

                cin.getline(d.degree, 10);

                cout << "Enter the batch number: ";

                cin >> d.batch;

                cout << "How many courses: ";

                cin >> d.t\_course;

                cin.ignore();

                cout << "Enter the name of courses u want to assign (courses must be seperated by commas): ";

                cin.getline(d.courses, 60);

                marks <<endl<< d.id << "," << d.name << "-" << d.courses<<"!"<< d.t\_course << ",";

                stu << endl << d.id << "," << d.pass << "-" << d.name << ","<<d.degree<<","<< d.batch;

                head = new ListNode(d);

                stu.close();

                while (d.t\_course>0)

                {

                    marks << "$";

                    d.t\_course--;

                }

                marks.close();

                return;

            }

            else

            {

                append(d, head);

            }

        }

        void append(Student d, ListNode \*tmp)

        {

            if (tmp->next == NULL)

            {

                cout << "Enter the id: ";

                cin >> d.id;

                cin.ignore();

                cout << "Enter the name: ";

                cin.getline(d.name, 30);

                cout << "Set the password for ur id: ";

                cin.getline(d.pass, 30);

                cout << "Enter the name of degree: ";

                cin.getline(d.degree, 10);

                cout << "Enter the batch number: ";

                cin >> d.batch;

                cout << "How many courses: ";

                cin >> d.t\_course;

                cin.ignore();

                cout << "Enter the name of courses u want to assign (courses must be seperated by commas): ";

                cin.getline(d.courses, 60);

                marks << endl << d.id << "," << d.name << "-" << d.courses << "!" << d.t\_course << ",";

                stu << endl << d.id << "," << d.pass << "-" << d.name << "," << d.degree << "," << d.batch;

                stu.close();

                while (d.t\_course>0)

                {

                    marks << "$";

                    d.t\_course--;

                }

                marks.close();

            }

            append(d, tmp->next);

        }

        void displaye()

        {

            cout << endl << endl << "\t\t\t\t\t\t Students's Record" << endl << endl;

            cout << endl << endl << right << setw(4) << "ID \t" << left << setw(30) << "Name" << left << setw(10) << "Degree" << left << setw(10) << "Batch" << left << setw(20) << "Total courses" << left << setw(60) << "Course names" << endl;

            int t = 0;

            while (t<100)

            {

                cout << "\_";

                t++;

            }

            cout << endl << endl;

            if (head == NULL)

            {

                stu.open("student.txt");

                marks.open("Marks.txt");

                stu >> d.id;

                stu.ignore(10, ',');

                stu.ignore(10, '-');

                stu.getline(d.name, 30, ',');

                stu.getline(d.degree, 10, ',');

                stu >> d.batch;

                stu.ignore();

                marks.ignore(100,'-');

                marks.getline(d.courses, 50, '!');

                marks >> d.t\_course;

                marks.ignore(10, ',');

                marks.ignore(100, '\n');

                head = new ListNode(d);

                marks.close();

                stu.close();

            }

            display(head);

        }

        void display(ListNode \*tmp)

        {

            stu.open("student.txt");

            marks.open("Marks.txt");

            while (!stu.eof())

            {

                stu >> tmp->data.id;

                stu.ignore(10, ',');

                stu.ignore(10, '-');

                stu.getline(tmp->data.name, 30, ',');

                stu.getline(tmp->data.degree, 10, ',');

                stu >> tmp->data.batch;

                stu.ignore();

                marks.ignore(100, '-');

                marks.getline(tmp->data.courses, 60, '!');

                marks >> tmp->data.t\_course;

                marks.ignore(10, ',');

                marks.ignore(100, '\n');

                cout << left << setw(4) << tmp->data.id << "\t" << left << setw(30) << tmp->data.name << left << setw(10) << tmp->data.degree<< left << setw(10) << tmp->data.batch << left << setw(20) << tmp->data.t\_course << left << setw(60) << tmp->data.courses << endl;

            }

            stu.close();

            marks.close();

        }

        void erasee()

        {

            string s;

            cout << "Name of the student which has to deleted: ";

            cin >> s;

            if (head == NULL)

            {

                stu.open("student.txt");

                marks.open("Marks.txt");

                stu >> d.id;

                stu.ignore(10, ',');

                stu.getline(d.pass, 30, '-');

                stu.getline(d.name, 30, ',');

                stu.getline(d.degree, 10, ',');

                stu >> d.batch;

                stu.ignore();

                marks.ignore(100, '-');

                marks.getline(d.courses, 50, '!');

                marks >> d.t\_course;

                marks.ignore(10, ',');

                marks.ignore(100, '\n');

                head = new ListNode(d);

                stu.close();

                marks.close();

            }

            ListNode \*tmp = head;

            stu.open("student.txt");

            marks.open("Marks.txt");

            temp.open("temp.txt", ios::app);

            temp1.open("temp1.txt", ios::app);

            int a = 1;

            while (!stu.eof())

            {

                stu >> d.id;

                stu.ignore(10, ',');

                stu.getline(d.pass, 30, '-');

                stu.getline(d.name, 30, ',');

                stu.getline(d.degree, 10, ',');

                stu >> d.batch;

                stu.ignore();

                marks.ignore(100, '-');

                marks.getline(d.courses, 50, '!');

                marks >> d.t\_course;

                marks.ignore(10, ',');

                marks.ignore(100, '\n');

                string sr(d.name);

                if (sr==s)

                {

                    a = 0;

                    tmp->next = new ListNode(d);

                    tmp = tmp->next;

                    continue;

                }

                else

                {

                    temp << endl << d.id << "," << d.pass << "-" << d.name << "," << d.degree << "," << d.batch;

                    temp1 << endl << d.id << "," << d.name << "-" << d.courses << "!" << d.t\_course << ",";

                    while (d.t\_course > 0)

                    {

                        temp1 << "$";

                        d.t\_course--;

                    }

                }

                tmp->next = new ListNode(d);

                tmp = tmp->next;

            }

            temp.close();

            stu.close();

            temp1.close();

            marks.close();

            if (a != 0)

            {

                cout << "\n\n\nNot found\n\n";

                return;

            }

            remove("Marks.txt");

            rename("temp1.txt", "Marks.txt");

            remove("student.txt");

            rename("temp.txt", "student.txt");

            erase(s, head);

        }

        void erase(string s, ListNode \*&tmp)

        {

            string str(tmp->data.name);

            if (str == s)

            {

                tmp = tmp->next;

                cout << "\n\n Task done.\n\n " ;

                return;

            }

            erase(s,tmp->next);

        }

        void see\_assigned1()

        {

        marks.open("Marksx.txt");

        marks.ignore(100, '!');

        marks >> d.id;

        marks.ignore(10, ',');

        marks.close();

        marks.open("Marksx.txt");

        marks.ignore(100, '-');

        int j = d.id;

        int i = 0;

        while (j>0)

        {

        getline(marks, x[i], ',');

        i++;

        j--;

        }

        j = 0;

        cout << "\nThe list of assigned courses is following:-\n ";

        for (int i = 0; i < d.id; i++)

            cout <<endl<<++j<<". "<< x[i] << "\n";

        cout << endl << endl;

        marks.close();

        }

        void drope()

        {

            see\_assigned1();

            cout << "Which course u want to drop:";

            string s;

            cin >> s;

            marks.open("Marksx.txt");

            temp1.open("temp3.txt",ios::app);

            marks >> d.batch;

            marks.ignore(10, ',');

            marks.getline(d.degree, 30, '-');

            marks >> d.id;

            marks.ignore(10, ',');

            int j = d.id;

            int i = 0;

            int k = d.id;

            int a = 0;

            temp1 << d.batch << "," << d.degree << "-"  ;

            while (j>0)

            {

                getline(marks, x[i], ',');

                if (s == x[i])

                {

                    a = 1;

                    i++;

                    j--;

                    k=k - 1;

                    continue;

                }

                else

                {

                    temp1 << x[i]<<",";

                }

                i++;

                j--;

            }

            temp1 << "!" << k << ",";

            for ( int i = 0;  i < k;  i++)

            {

                temp1 << "$";

            }

            if (a != 1)

            {

                cout << "\n\nnot found.\n\n";

                marks.close();

                temp1.close();

            }

            else

            {

                cout << "\n\nTask completed successfully.\n\n";

                marks.close();

                temp1.close();

                remove("Marksx.txt");

                rename("temp3.txt", "Marksx.txt");

                marks.open("Marks.txt");

                temp.open("temp.txt", ios::app);

                while (!marks.eof())

                {

                    marks >> d.id;

                    marks.ignore(10, ',');

                    marks.getline(d.name, 30, '-');

                    marks.getline(d.courses, 50, '!');

                    marks >> d.t\_course;

                    marks.ignore(10, ',');

                    marks.ignore(100, '\n');

                    if (strcmp(d.name, d.degree) == 0)

                    {

                        temp << endl << d.id << "," << d.name << "-";

                        for (int i = 0; i < d.t\_course; i++)

                        {

                            if (s == x[i])

                            {

                                continue;

                            }

                            temp << x[i] << ",";

                        }

                        temp << "!" << k << ",";

                        for (int i = 0; i < k; i++)

                        {

                            temp << "$";

                        }

                        continue;

                    }

                    else

                    {

                        temp << endl << d.id << "," << d.name << "-" << d.courses << "!" << d.t\_course << ",";

                        while (d.t\_course > 0)

                        {

                            temp << "$";

                            d.t\_course--;

                        }

                    }

                }

                marks.close();

                temp.close();

                remove("Marks.txt");

                rename("temp.txt", "Marks.txt");

            }

        }

        void adde()

        {

            cout << "Enter the name of course you want to add: ";

            string s;

            int z;

            char person[30];

            cin >> s;

            marks.open("Marksx.txt");

            temp.open("temp.txt",ios::app);

            marks >> d.id;

            marks.ignore(10, ',');

            marks.getline(d.name, 30, '-');

            marks.getline(d.courses, 50, '!');

            marks >> d.t\_course;

            marks.ignore(10, ',');

            marks.ignore(100, '\n');

            d.t\_course = d.t\_course + 1;

            z = d.t\_course;

            temp << d.id << "," << d.name << "-" << s<<","<<d.courses << "!" << d.t\_course << ",";

            while (d.t\_course>0)

            {

                temp << "$";

                d.t\_course--;

            }

            marks.close();

            temp.close();

            remove("Marksx.txt");

            rename("temp.txt", "Marksx.txt");

            strcpy\_s(person, d.name);

            marks.open("Marks.txt");

            temp.open("temp.txt", ios::app);

            while (!marks.eof())

            {

                marks >> d.id;

                marks.ignore(10, ',');

                marks.getline(d.name, 30, '-');

                marks.getline(d.courses, 50, '!');

                marks >> d.t\_course;

                marks.ignore(10, ',');

                marks.ignore(100, '\n');

                if (strcmp(d.name, person) == 0)

                {

                    temp << endl << d.id << "," << d.name << "-"<<s << "," << d.courses << "!" << z << ",";

                    for (int i = 0; i < z; i++)

                    {

                        temp << "$";

                    }

                    continue;

                }

                else

                {

                    temp << endl << d.id << "," << d.name << "-" << d.courses << "!" << d.t\_course << ",";

                    while (d.t\_course>0)

                    {

                        temp << "$";

                        d.t\_course--;

                    }

                }

            }

            cout << " \n\n Task completed.\n\n";

            marks.close();

            temp.close();

            remove("Marks.txt");

            rename("temp.txt", "Marks.txt");

        }

        void showe()

        {

            string co;

            char x3[30];

            int rise=0;

            marks.open("Marksx.txt");

            marks.ignore(100, '-');

            getline(marks, co, '!');

            marks.close();

            marks.open("Marks.txt");

            while (!marks.eof())

            {

                marks.ignore(100, '-');

                marks.getline(d.courses, 60, '!');

                string str(d.courses);

                if (co == str)

                {

                    marks >> d.id;

                    marks.ignore(10, ',');

                    for (int i = 0; i < d.id; i++)

                    {

                        marks.getline(d.pass, 30, '$');

                        if (d.pass == NULL)

                        {

                            x1[i] = "No entry";

                            continue;

                        }

                        x1[i] = d.pass;

                    }

                }

                marks.ignore(50, '\n');

            }

            marks.close();

            marks.open("Marksx.txt");

            marks.ignore(100, '!');

            marks >> d.id;

            marks.ignore(10, ',');

            marks.close();

            marks.open("Marksx.txt");

            marks.ignore(100, '-');

            int j = d.id;

            int i = 0;

            while (j>0)

            {

                getline(marks, x[i], ',');

                i++;

                j--;

            }

            j = 0;

            cout << "\nThe marks of assigned courses is following:-\n ";

            cout << endl << "Subjects\t\t\t\tSessional,Mid,Final marks" << endl;

            for (int i = 0; i < 100; i++)

            {

                cout << "\_";

            }

            cout << endl;

            for (int i = 0; i < d.id; i++)

            {

                cout << endl << ++j << ". " << x[i] << "\t\t\t\t\t" << x1[i];

                cout << endl;

            }

            cout << endl << endl;

            marks.close();

        }

    };

    char choice;

    Student d;

    linked\_list s;

public:

    void show\_d()

    {

        s.displaye();

        system("pause");

    }

    void add()

    {

        s.appende(d);

        cout << " Do u want to see it ? (y/n) ";

        cin >> choice;

        if (choice == 'Y' || choice == 'y')

        {

            s.displaye();

            system("pause");

        }

    }

    void del()

    {

        cout << " Do u want to see the list of students ? (y/n) ";

        cin >> choice;

        if (choice == 'Y' || choice == 'y')

        {

            s.displaye();

            system("pause");

        }

        s.erasee();

        cout << " Do u want to see the changes ? (y/n) ";

        cin >> choice;

        if (choice == 'Y' || choice == 'y')

        {

            s.displaye();

            system("pause");

        }

    }

    void see\_assigned()

    {

        s.see\_assigned1();

        system("pause");

    }

    void drop()

    {

        s.drope();

        system("pause");

    }

    void addc()

    {

        s.adde();

        system("pause");

    }

    void showc()

    {

        s.showe();

        system("pause");

    }

};

int main()

{

    TryPass t;

    string n;

    Teacher\_functionality te;

    Password d;

    Admin ad;

    Menu a;

    Student s;

    a.dis();

    if (t.trypass() == 0)

    {

        goto exit;

    }

top1:                           // ----> defining label for first screen

    a.dis1();

    cin >> n;

    if (n == "1")

    {   //----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

        // ADMIN MODE

        if (d.isLogin() == 1)

        {

        top2:

            system("cls");

            a.dis2();

            cin >> n;

            if (n == "1")

            {

                ad.add();

                goto top2;

            }

            if (n == "2")

            {

                ad.del();

                goto top2;

            }

            if (n == "3")

            {

                te.add\_t();

                goto top2;

            }

            if (n == "4")

            {

                te.del\_t();

                goto top2;

            }

            if (n == "5")

            {

                ad.show\_d();

                goto top2;

            }if (n == "6")

            {

                te.show\_d();

                goto top2;

            }

            if (n == "7")

            {

                system("cls");

                goto top1;

            }

            if (n == "8")

            {

                goto exit;

            }

            else

            {

                cout << "\n Invalid Entry. Please try again. \n\n";

                system("pause");

                goto top2;

            }

        }

        else

        {

            if (d.isLoginTry() == 1)

            {

                goto top2;

            }

        }

    }

    if (n == "2")

    {

        if (d.isLogin1() == 1)

        {

        top3:

            system("cls");

            a.dis3();

            cin >> n;

            if (n == "1")

            {

                ad.see\_assigned();

                goto top3;

            }

            if (n == "2")

            {

                ad.drop();

                goto top3;

            }

            if (n == "3")

            {

                ad.addc();

                goto top3;

            }

            if (n == "4")

            {

                ad.showc();

                goto top3;

            }

            if (n == "5")

            {

                system("cls");

                goto top1;

            }

            if (n == "6")

            {

                goto exit;

            }

            else

            {

                cout << "\n Invalid Entry. Please try again. \n\n";

                system("pause");

                goto top3;

            }

        }

        else

        {

            if (d.isLoginTry1() == 1)

            {

                goto top3;

            }

        }

    }

    if (n == "3")

    {

        if (d.isLogin2() == 1)

        {

        top4:

            system("cls");

            a.dis4();

            cin >> n;

            if (n == "1")

            {

                te.enter\_marks();

                goto top4;

            }

            if (n == "2")

            {

                te.see\_assigned\_t();

                goto top4;

            }

            if (n == "3")

            {

                system("cls");

                goto top1;

            }

            if (n == "4")

            {

                goto exit;

            }

            else

            {

                cout << "\n Invalid Entry. Please try again. \n\n";

                system("pause");

                goto top4;

            }

        }

        else

        {

            if (d.isLoginTry2() == 1)

            {

                goto top4;

            }

        }

    }

    if (n == "4")

    {

        goto exit;

    }

    else

    {

        cout << "\n Invalid Entry. Please try again. \n\n";

        system("pause");

        system("cls");

        goto top1;

    }

exit:

    system("pause");

}

## OUTPUT

**FRONT PAGE:**

Text

Description automatically generated

**ADMIN MODE:**

Text

Description automatically generated

**ADMIN CHECKING STUDENT DETAILS:**

Table

Description automatically generated

**ADMIN CHECKING TEACHER DETAILS:**

Graphical user interface, text, application

Description automatically generated

**STUDENT MODE:**

Text

Description automatically generated

**STUDENT FRONT PAGE:**

Text

Description automatically generated

**STUDENT CHECKING ASSIGNED COURSE DETAILS:**

Text

Description automatically generated

**STUDENT CHECKING RESULT DETAILS:**

Graphical user interface, text, application, email

Description automatically generated

**TEACHER LOGIN MODE**

Text

Description automatically generated with low confidence

**TEACHER FRONT PAGE:**

Graphical user interface, text, application

Description automatically generated

**TEACHER ADDING STUDENT MARKS:**

Text, email

Description automatically generated

**TEACHER ADDING STUDENT MARKS:**

Graphical user interface, text, application, email

Description automatically generated

**TEACHER CHECKING OWN ASSIGNED COURSE DETAILS:**

Graphical user interface, text, application

Description automatically generated