**School Management System**



Session: 2021 – 2024

**Submitted by:**

Umair Noor 2021-CS-207

**Supervised by:**

Mam Maida

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Description:**

This project will take care of the maximum work done in school in everyday life. SMS has most of the facilities that a modern school requires, computerizing its day-to-day jobs. It provides a facility to track students’ academic records and students can also get in touch with their performance. Principle/Admin has full control over the whole SMS.

This management system can add teachers and students and edit the already added record of teachers and students and delete the records.

**Users of System:**

Users of the SMS are

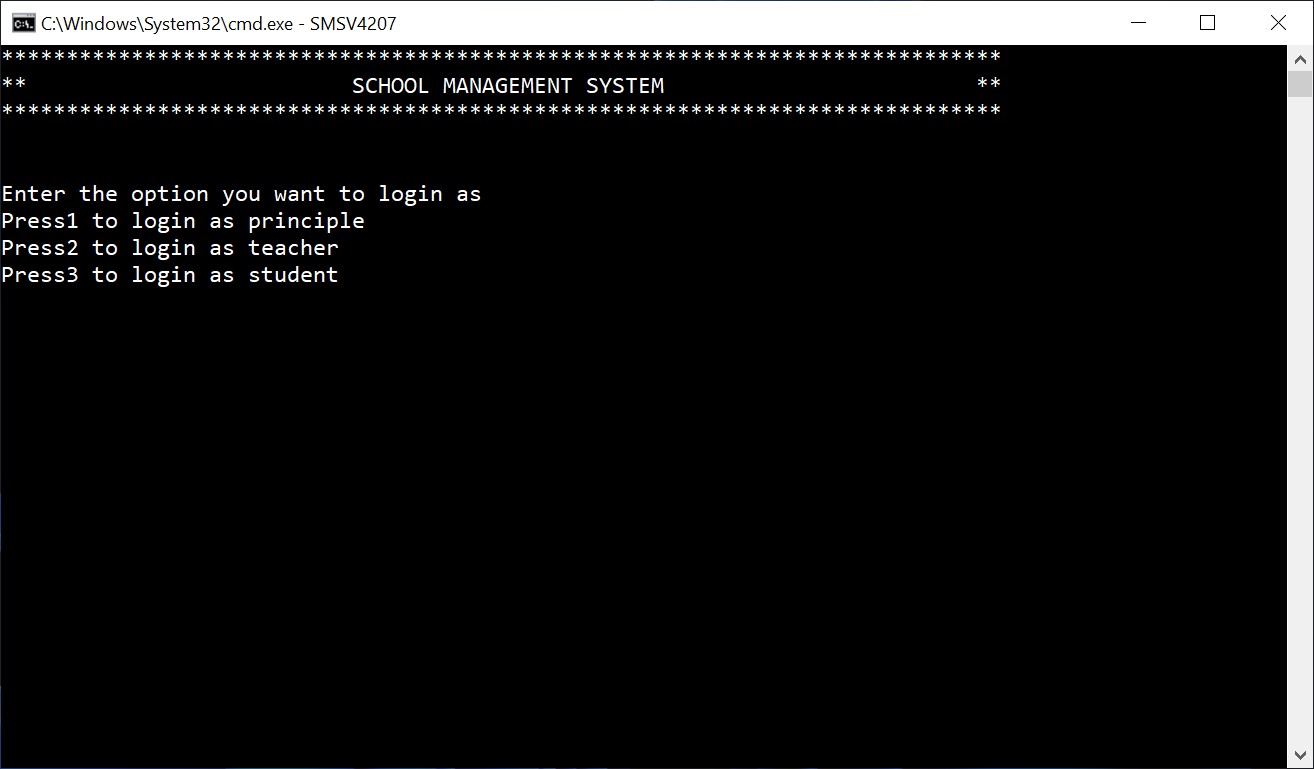
* Principle
* Teachers
* Students

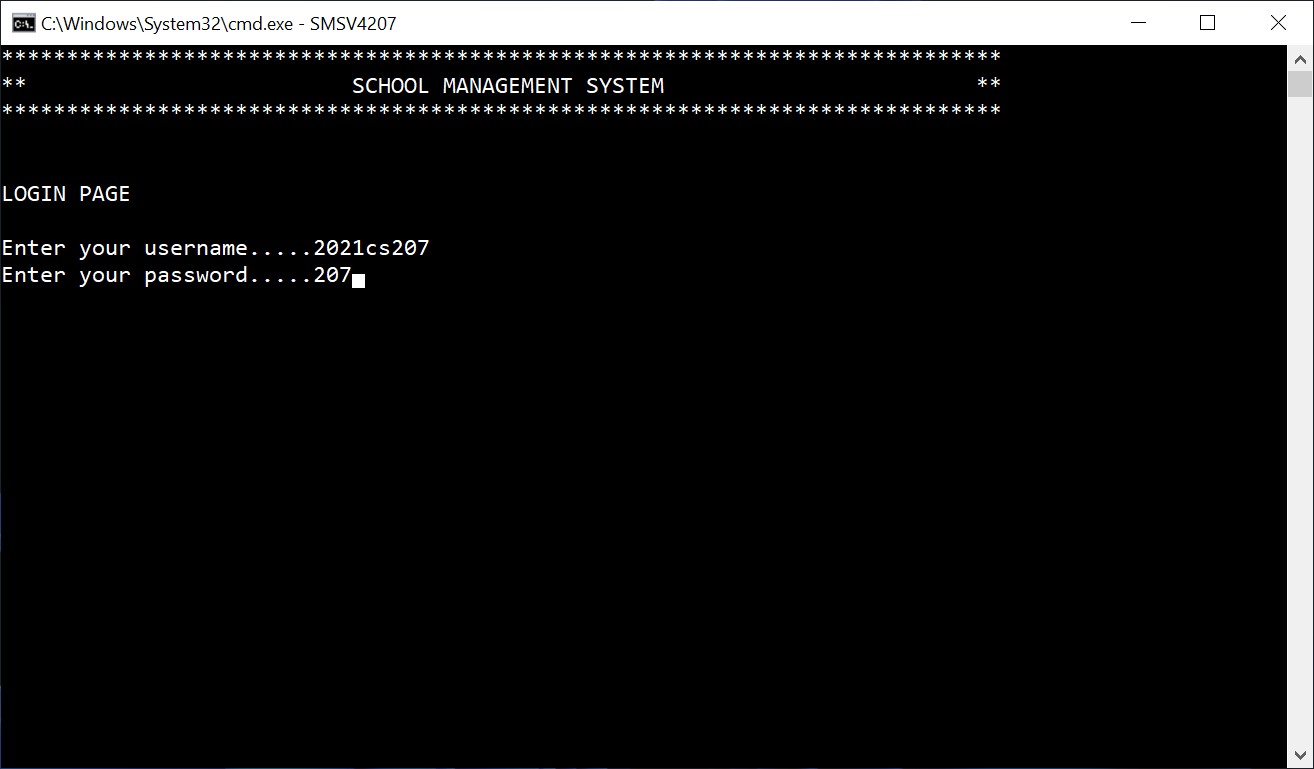
**FUNCTIONAL REQUIREMENTS:**

* Anyone who wants to use the application has to login
* As Principle I can add teachers
* As Principle I can add students
* As Principle I can update the record
* As Principle I can view all the records available
* As Principle I can remove the teacher
* As Principle I can remove students
* As Teacher I can give marks to students
* As Teacher I can see marks of students
* As Teacher I can mark the attendance of students
* As Teacher I can see the attendance of students
* As Student I can see my marks
* As Student I can register subjects
* As Student I can see registered subjects
* As Student I can see my attendance

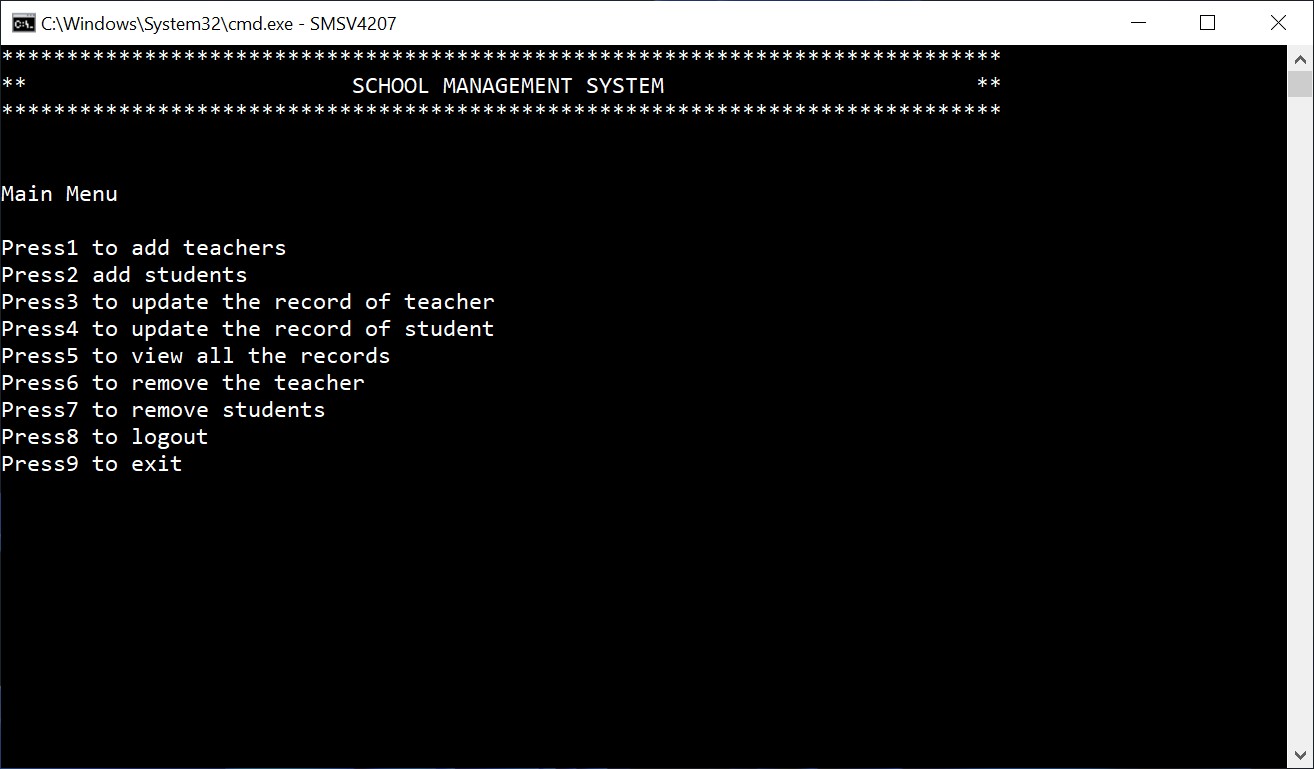
**Wireframes:**

Login Option

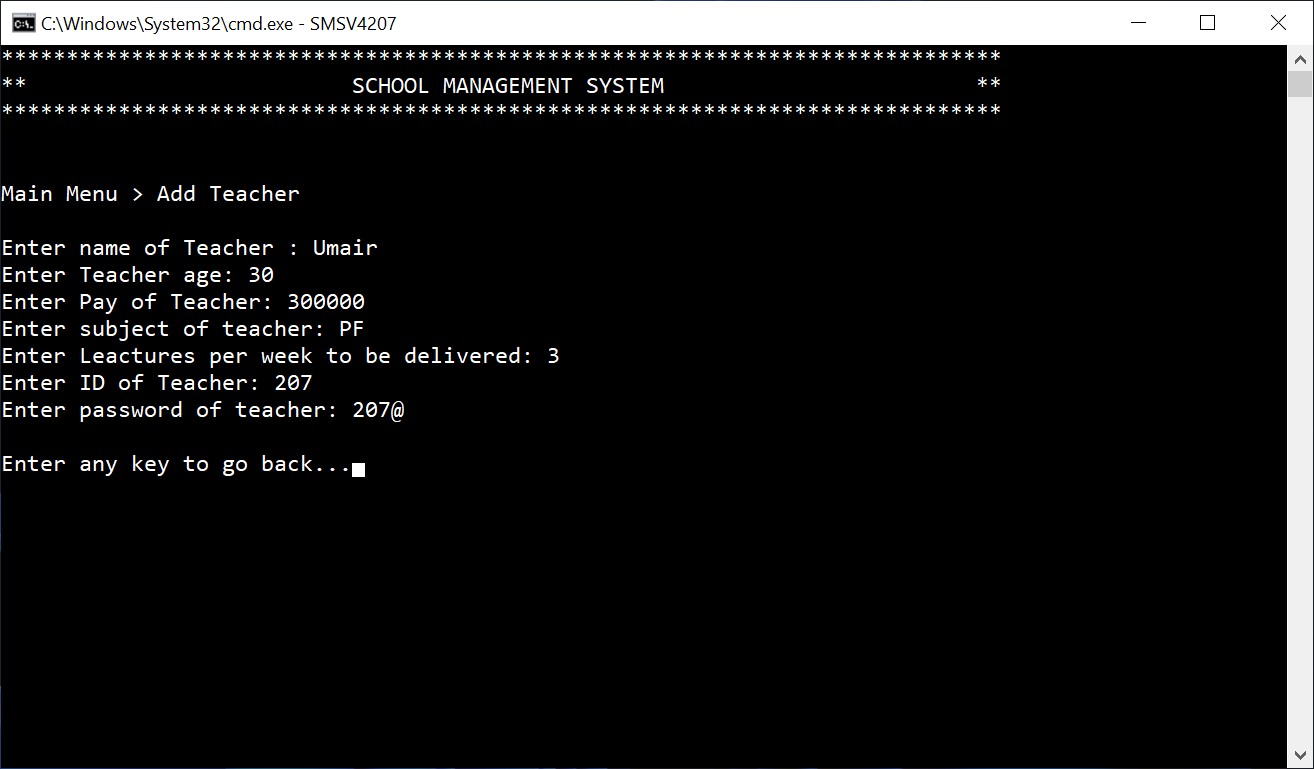


Admin Login Page

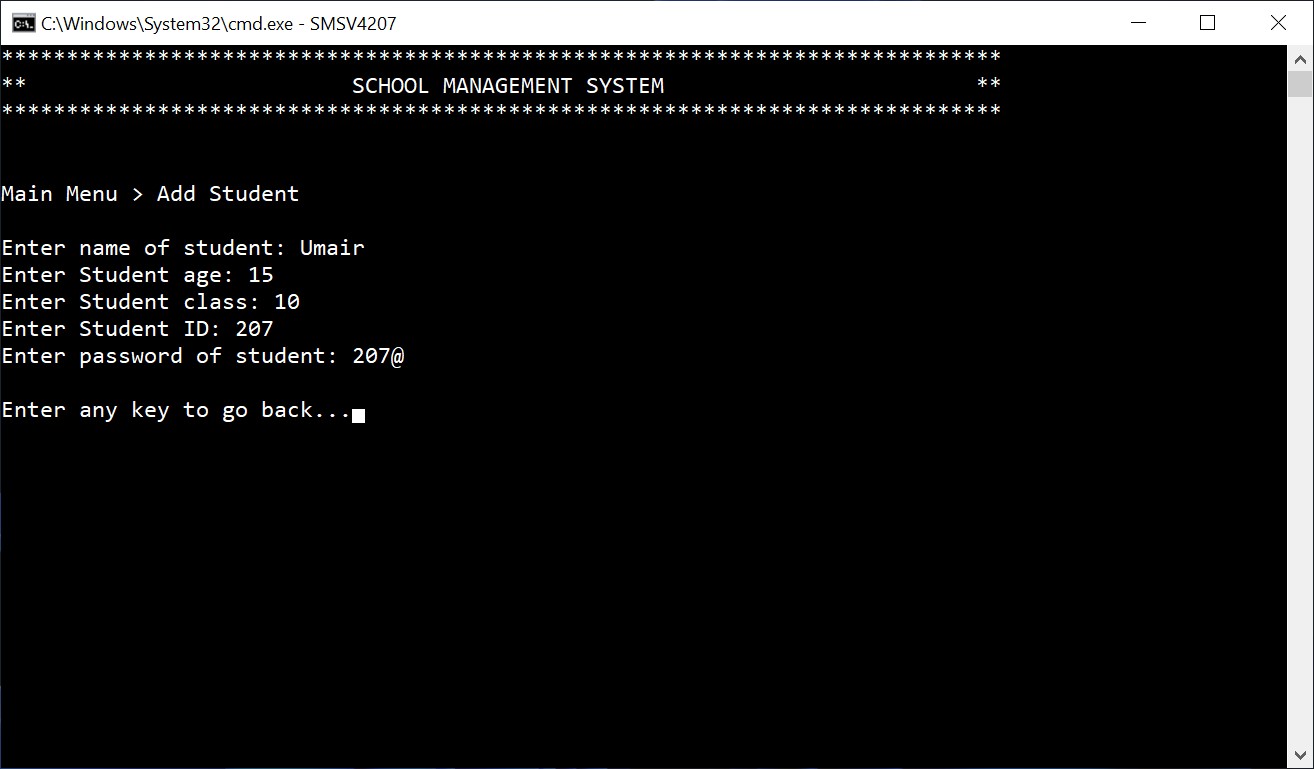
Admin Main Menu



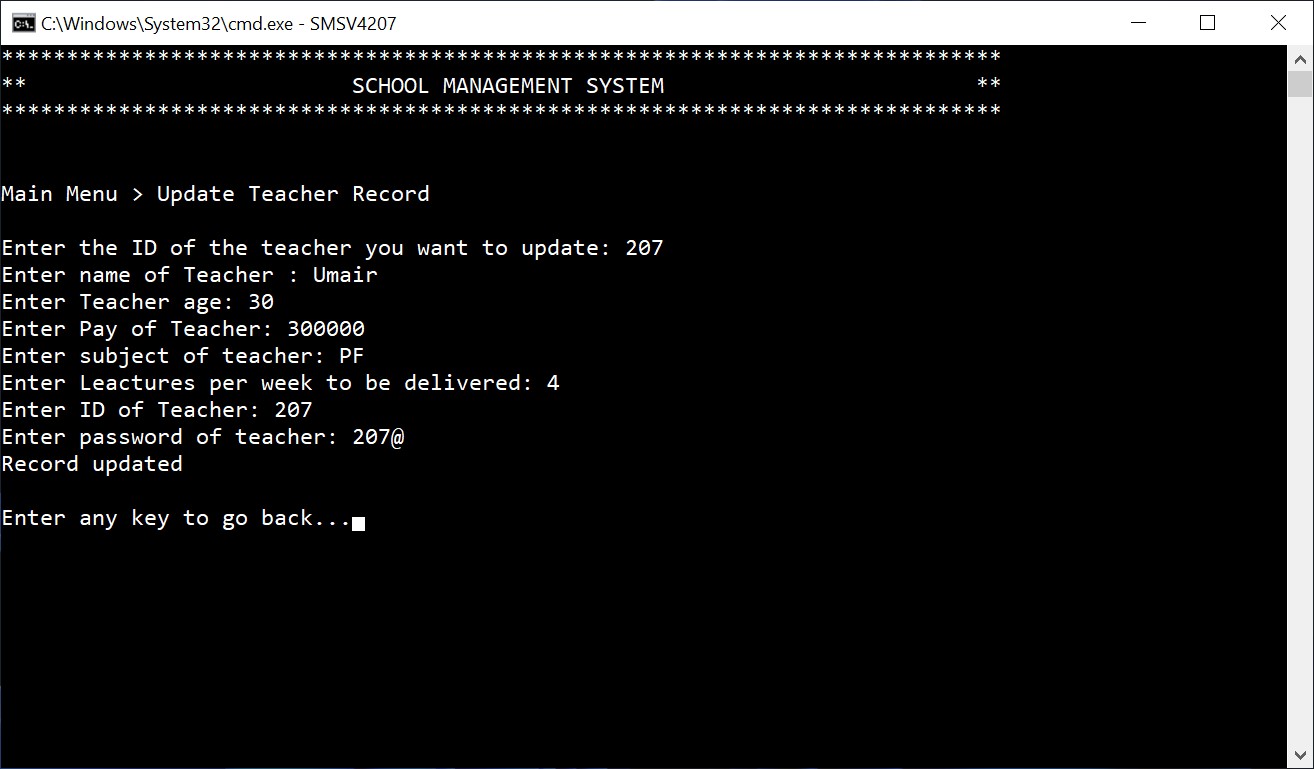
Option 1



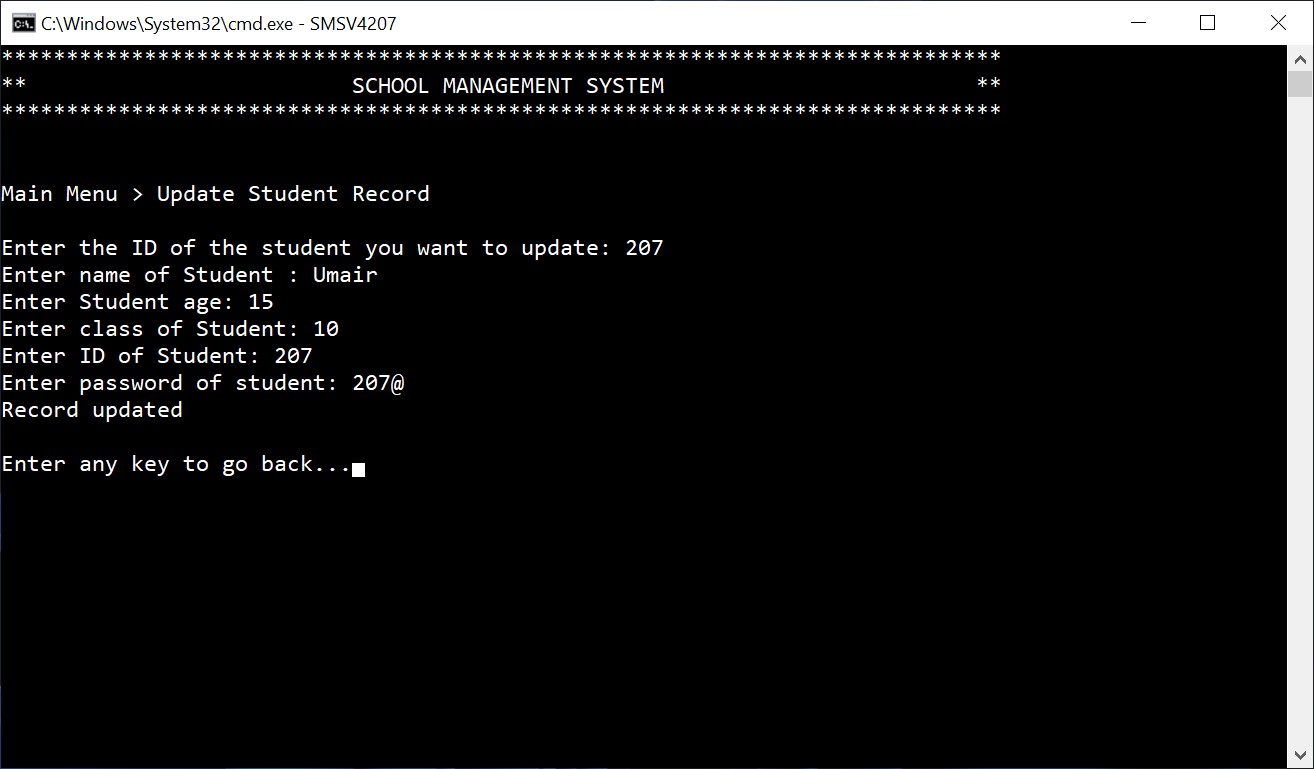
Option 2

****

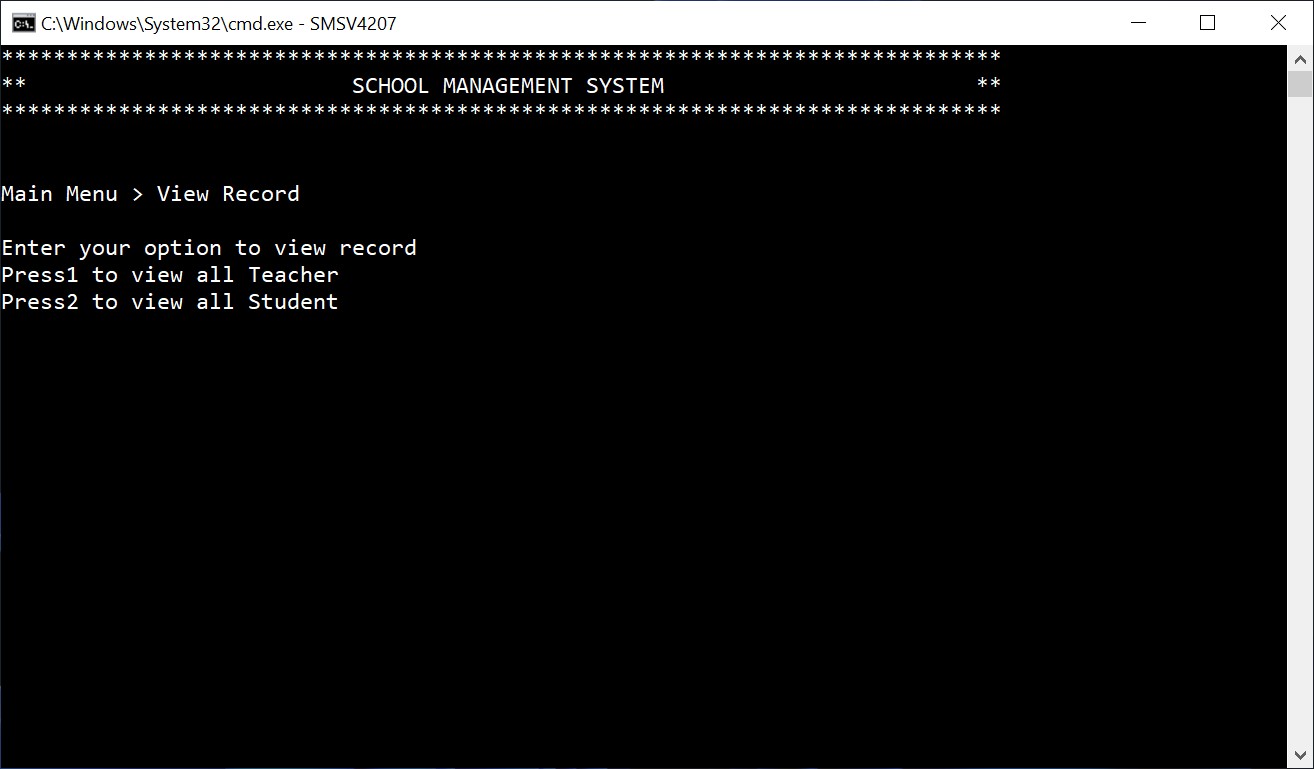
Option 3

****

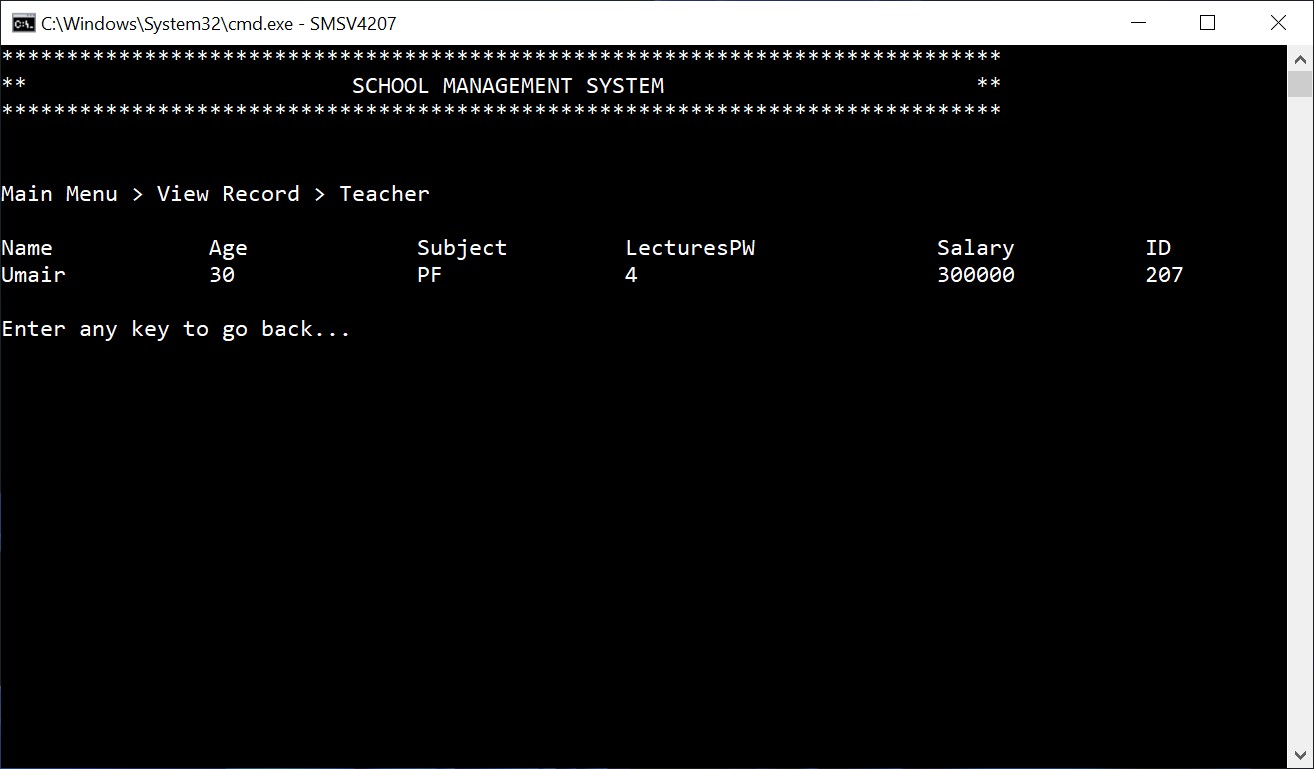
Option 4



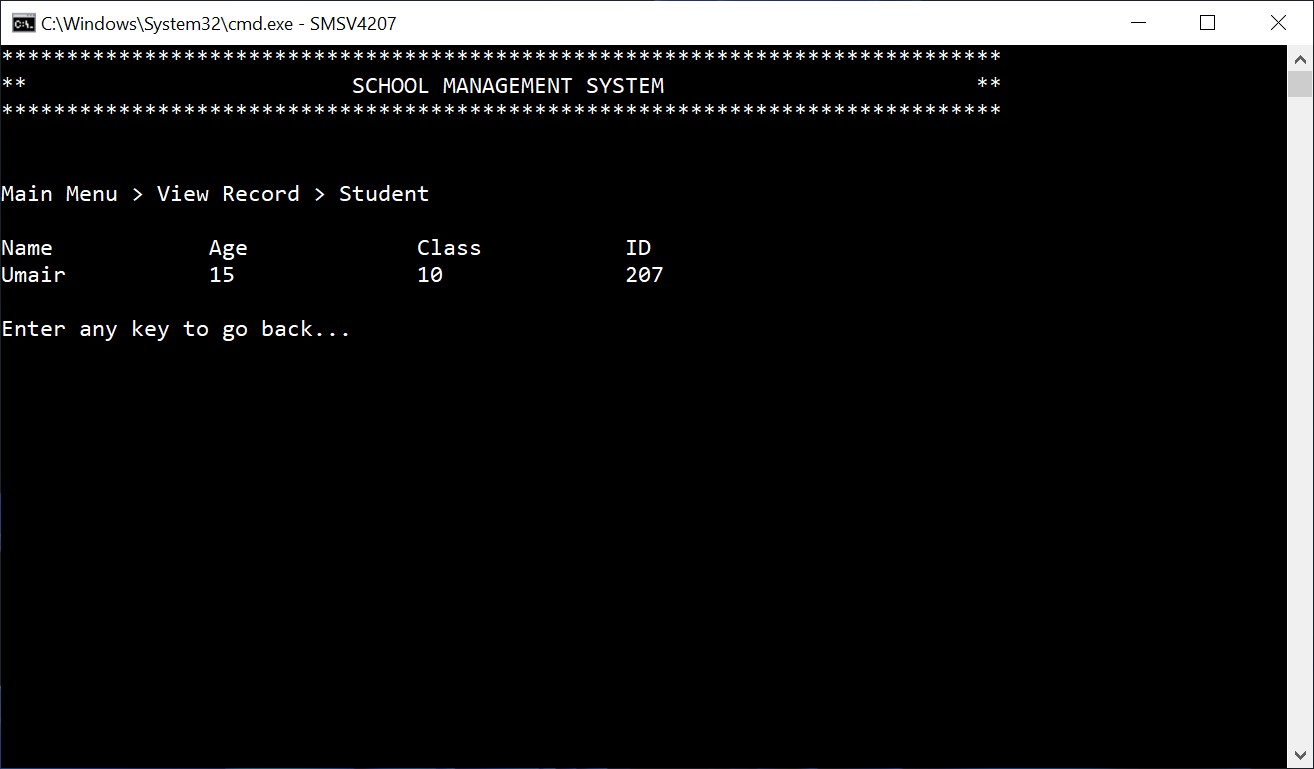
Option 5

****

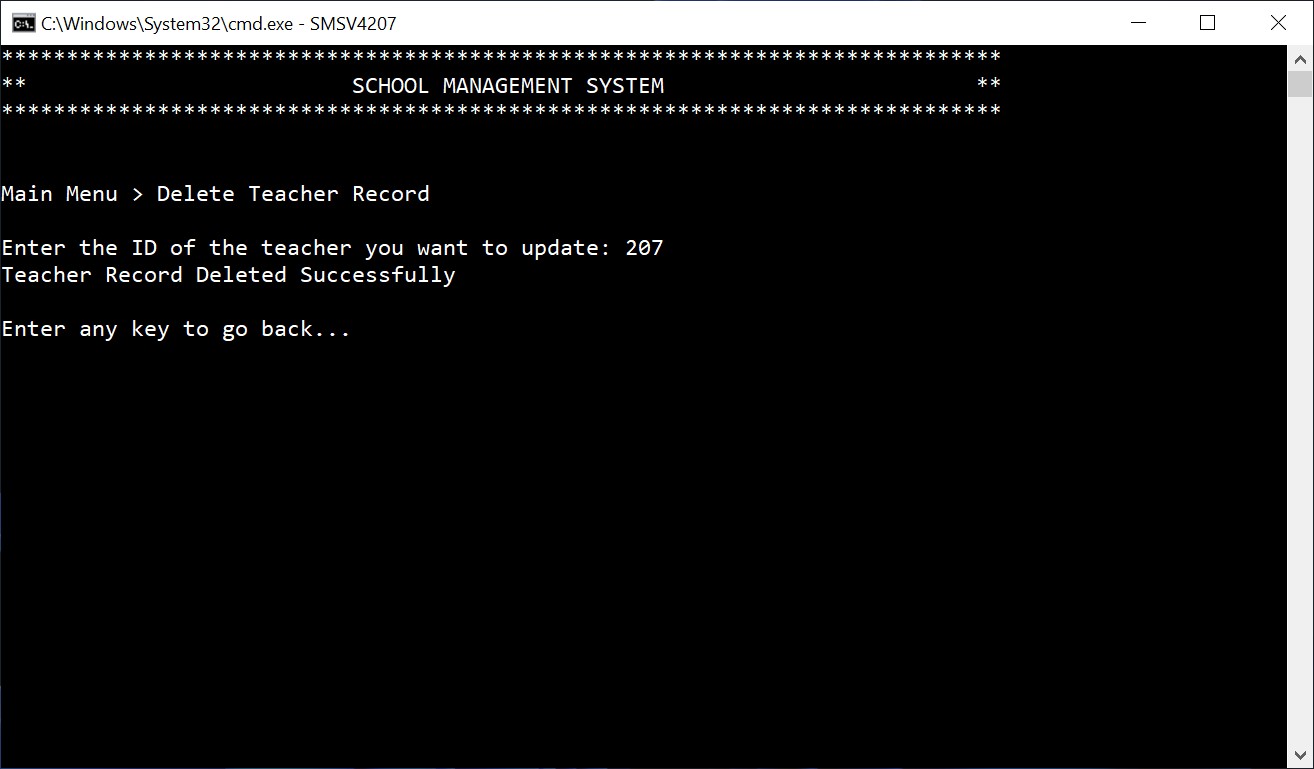
Option 5 (View Teacher’s Record)



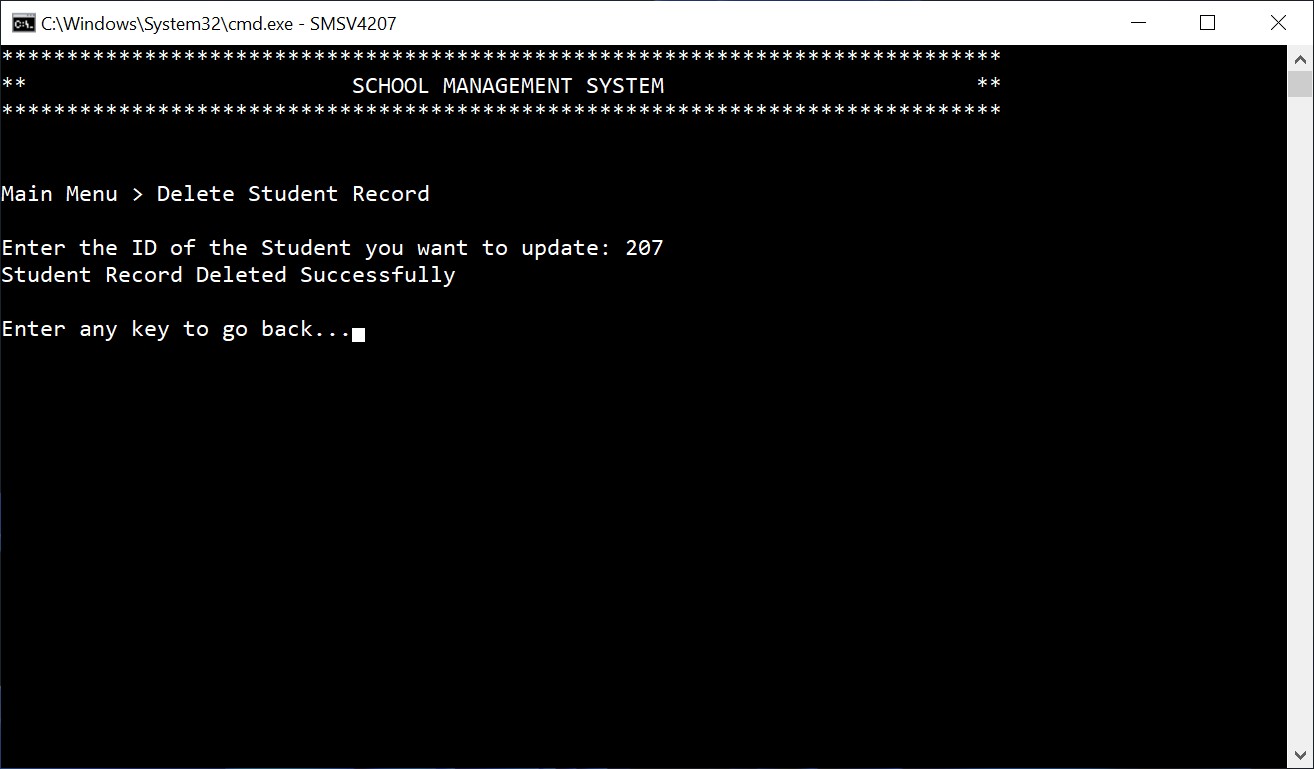
Option 5 (View Student’s Record)



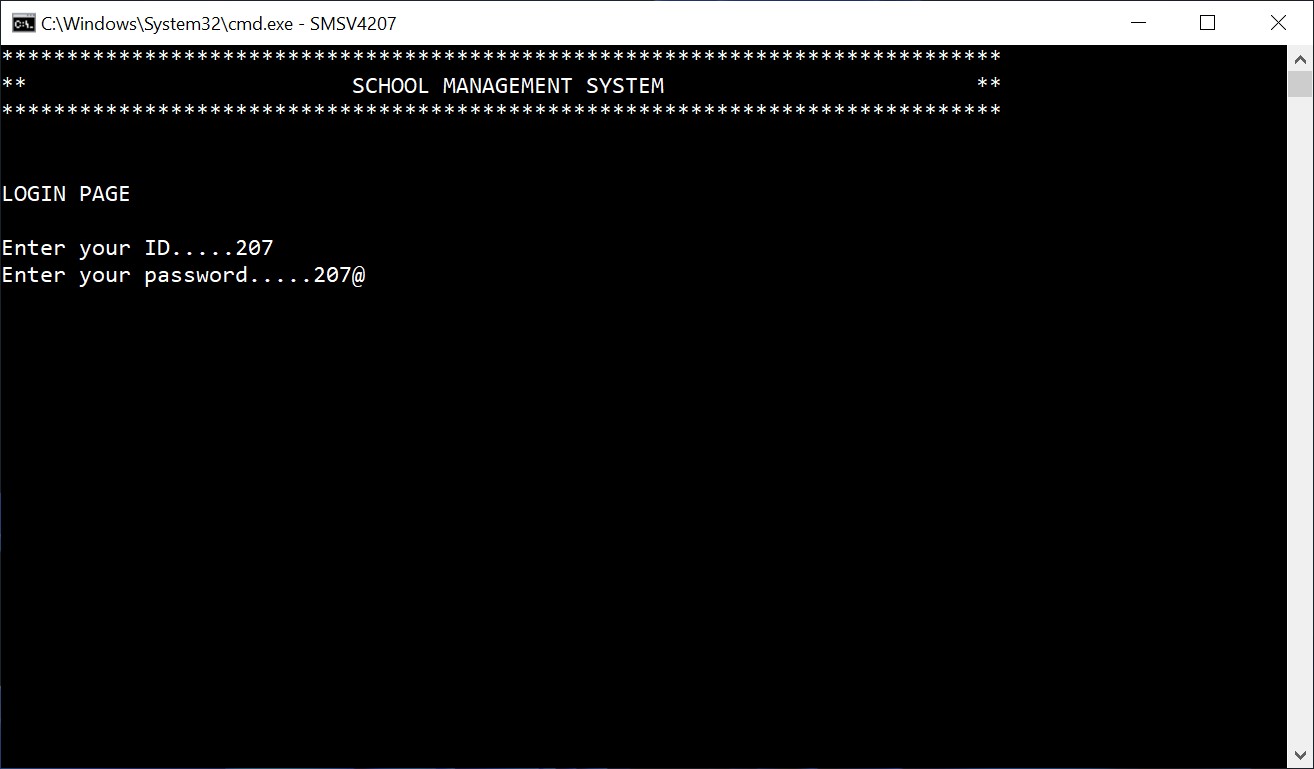
Option 6



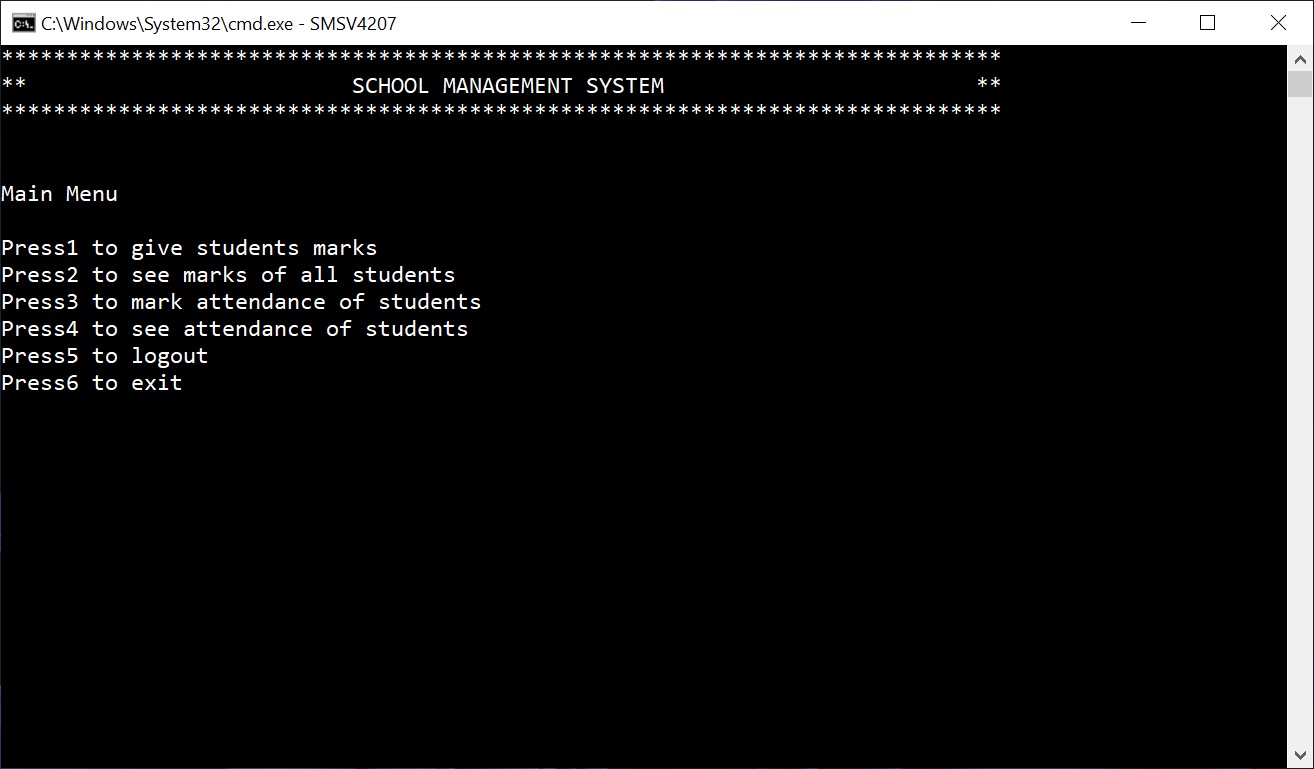
Option 7

****

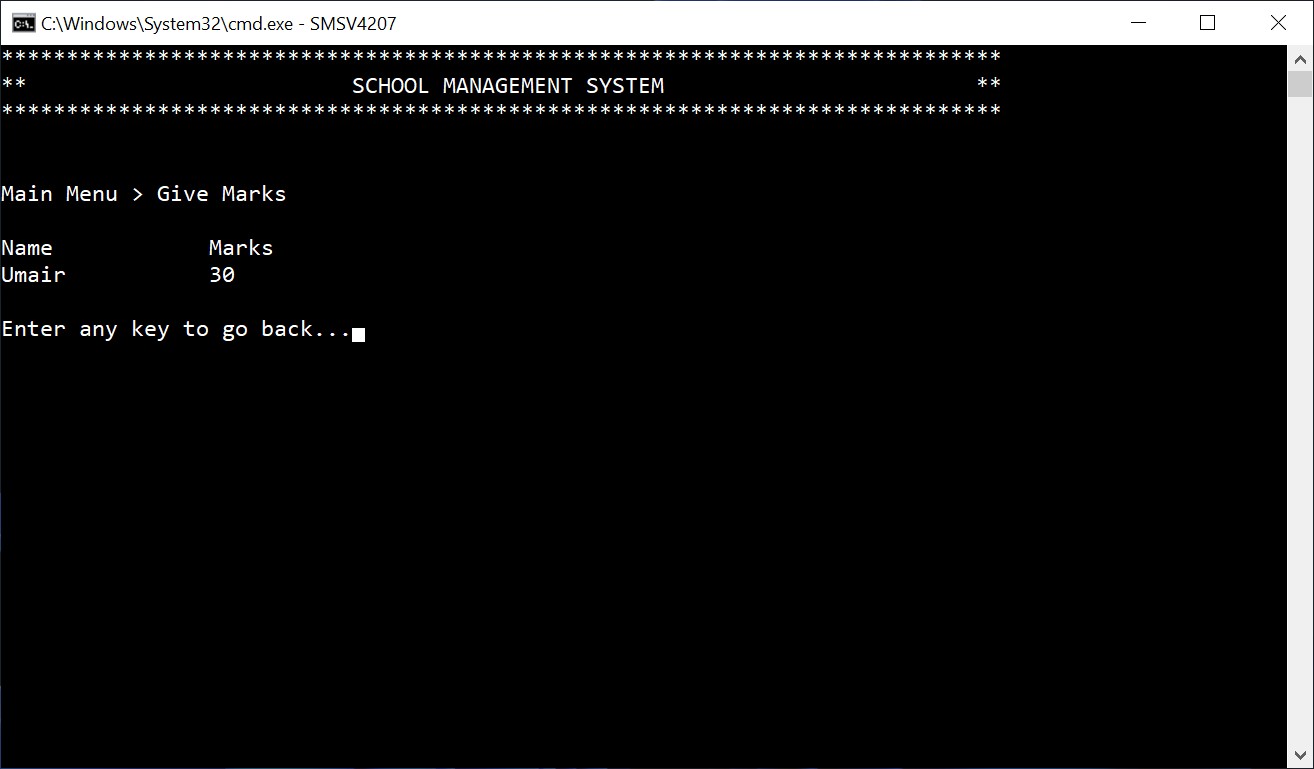
Teacher Login Page



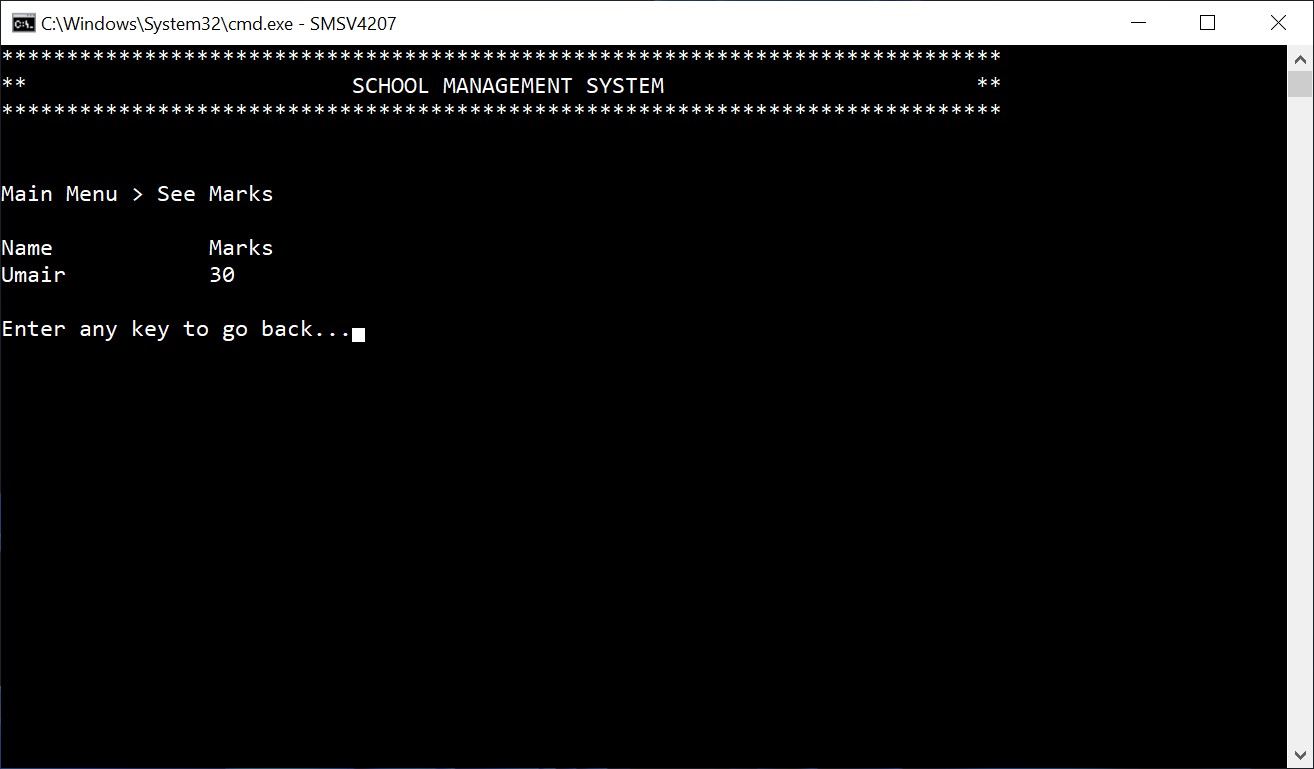
Teacher Main Menu

****

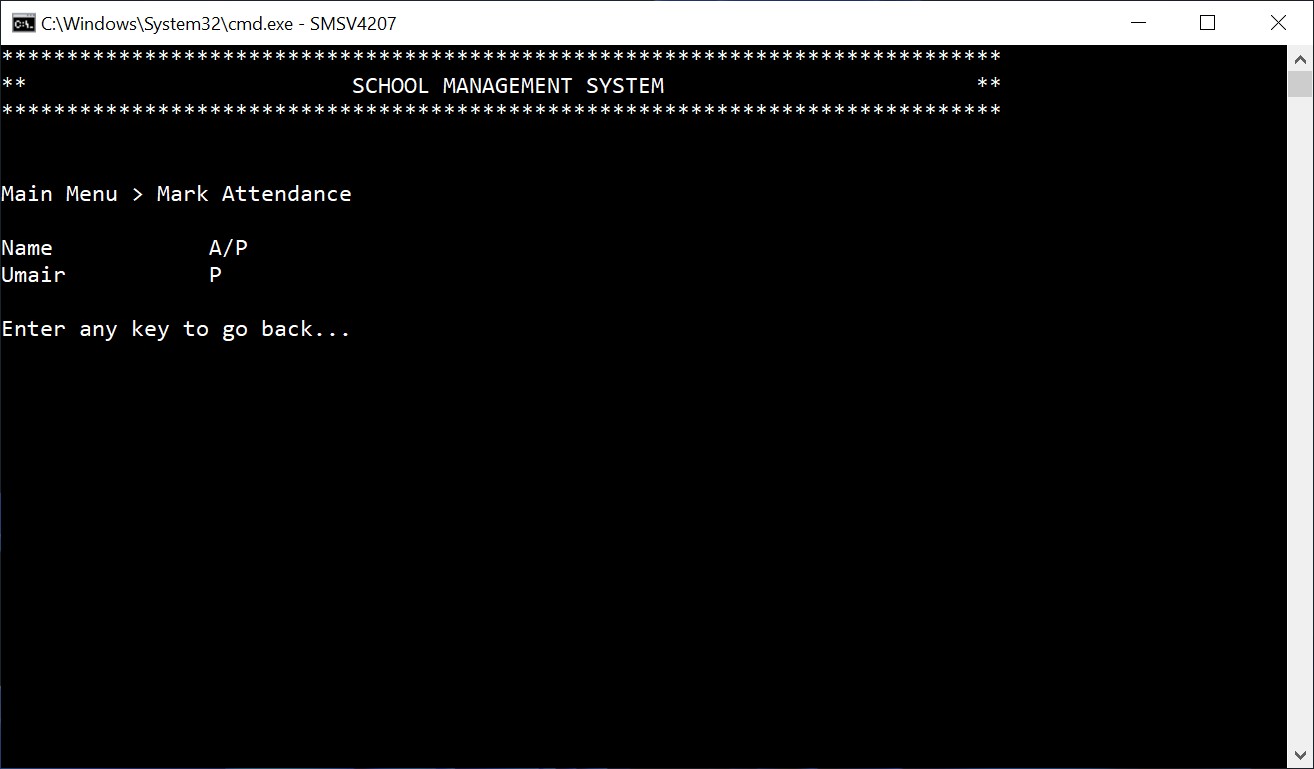
Option 1



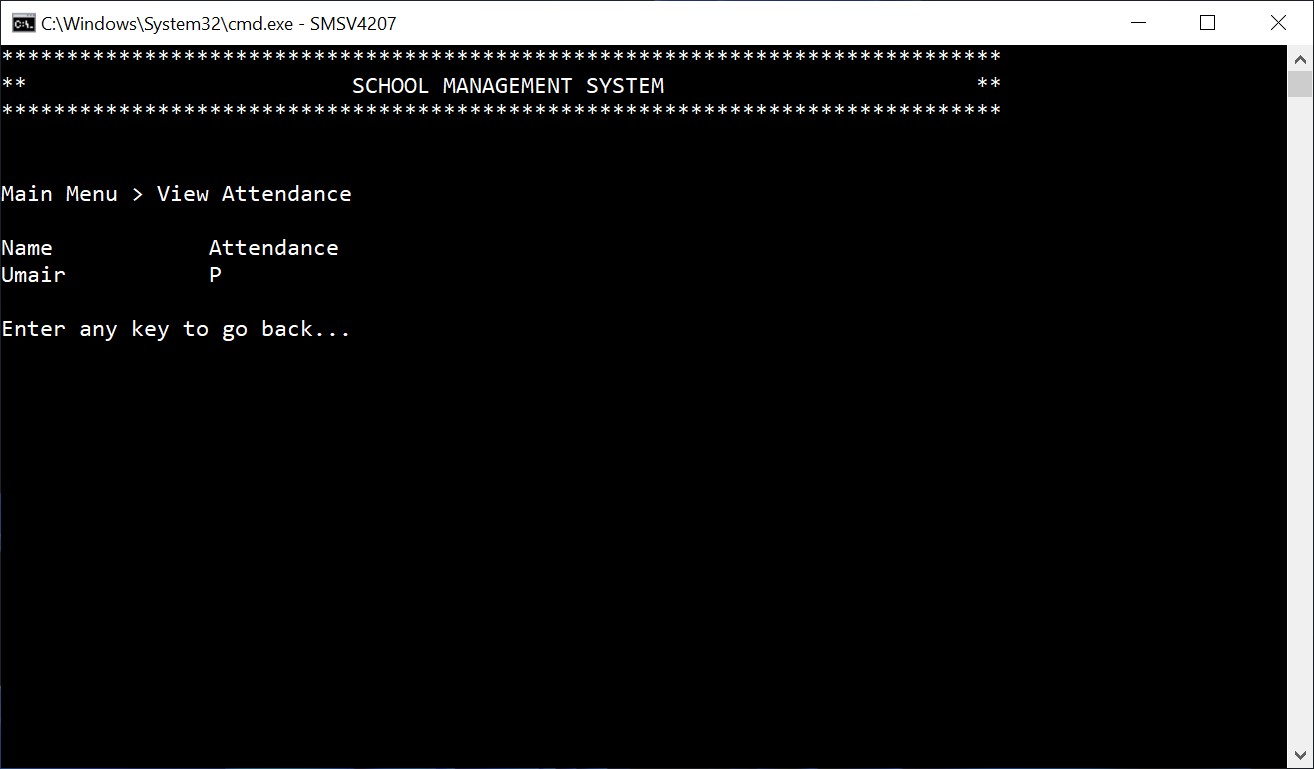
Option 2

****

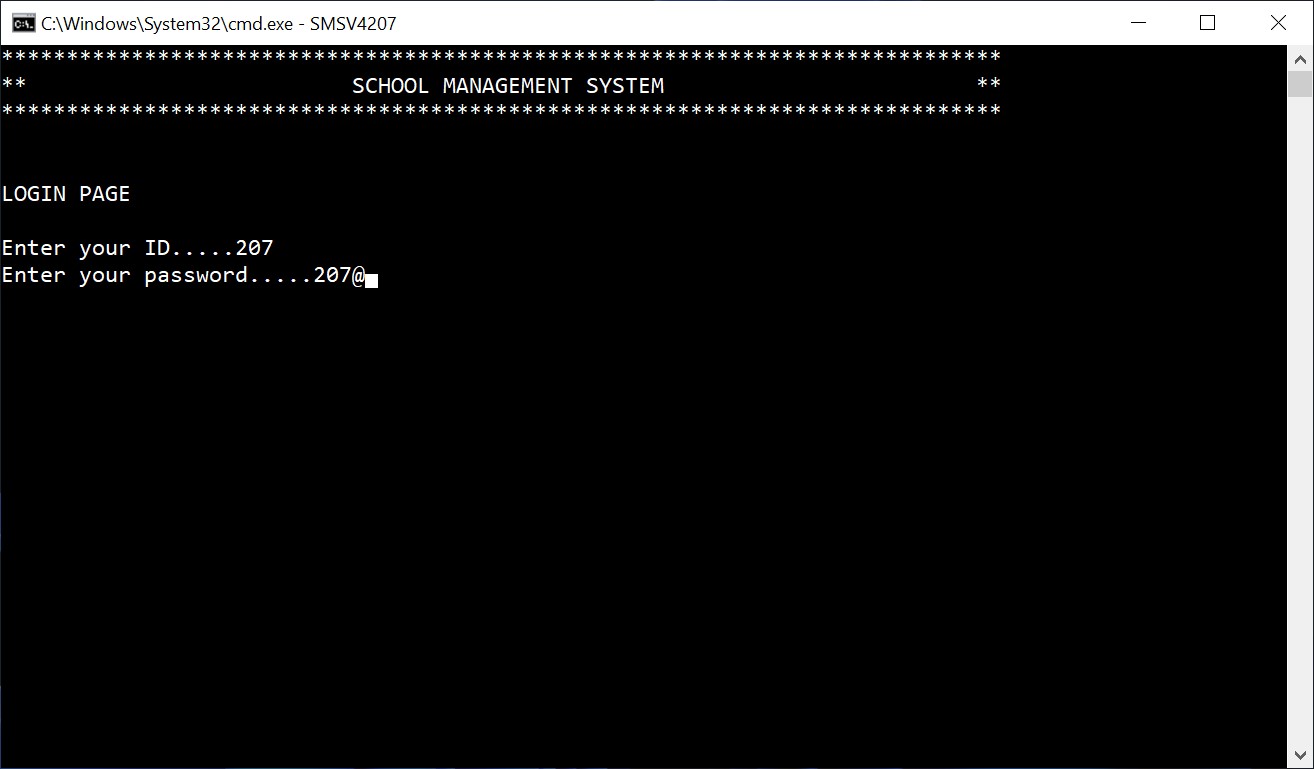
Option 3



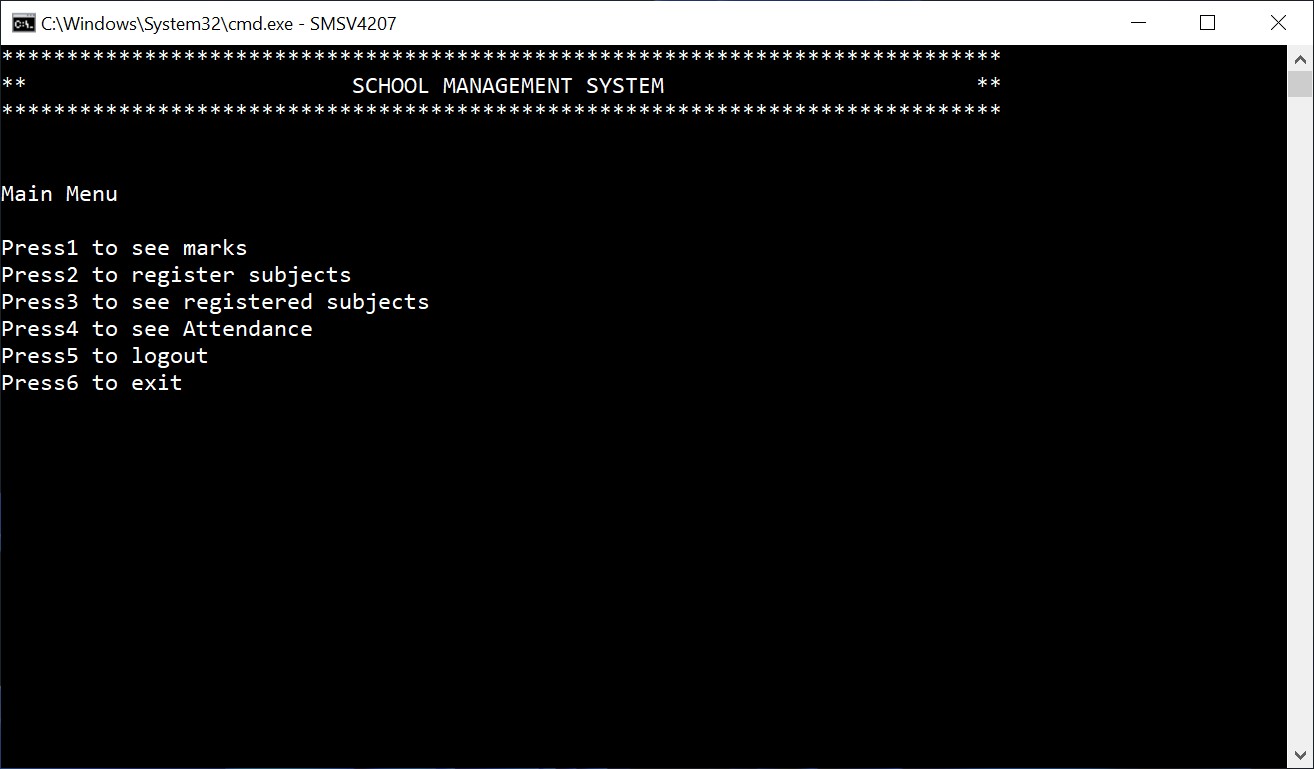
Option 4

****

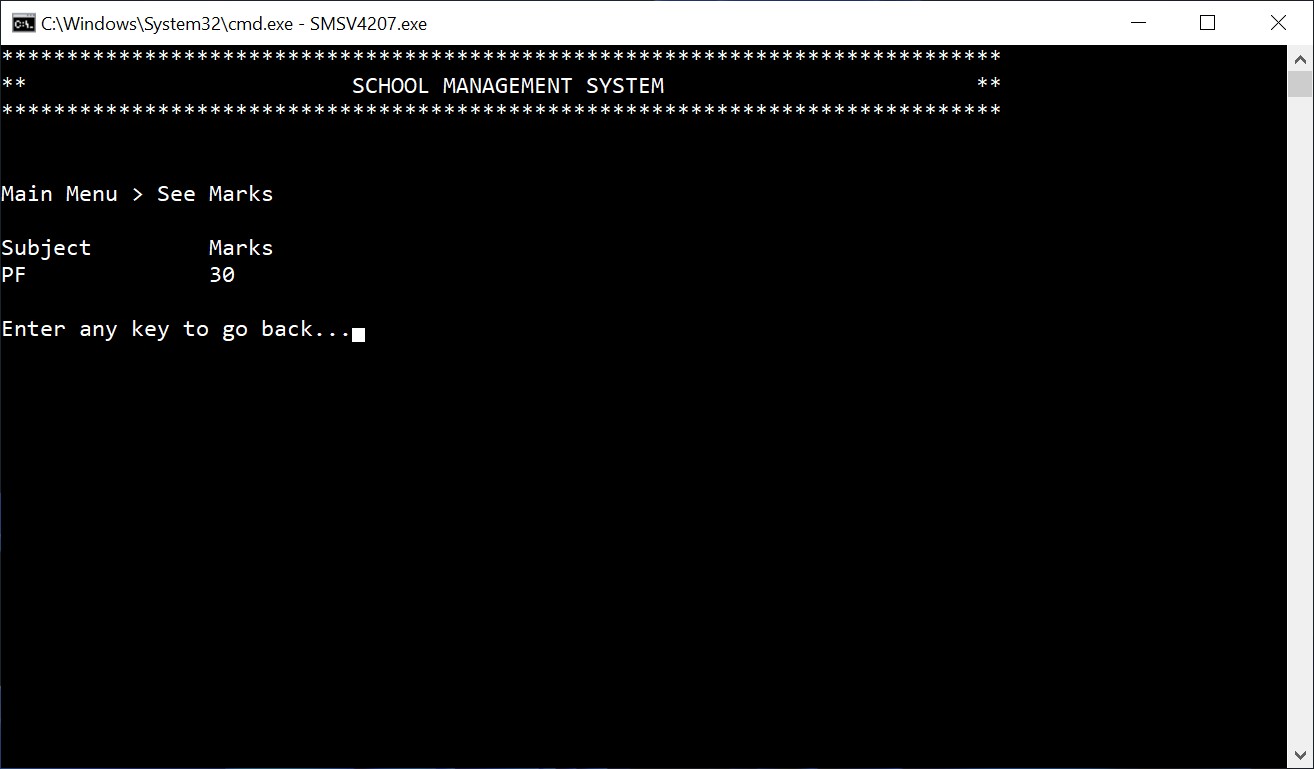
Student Login Page



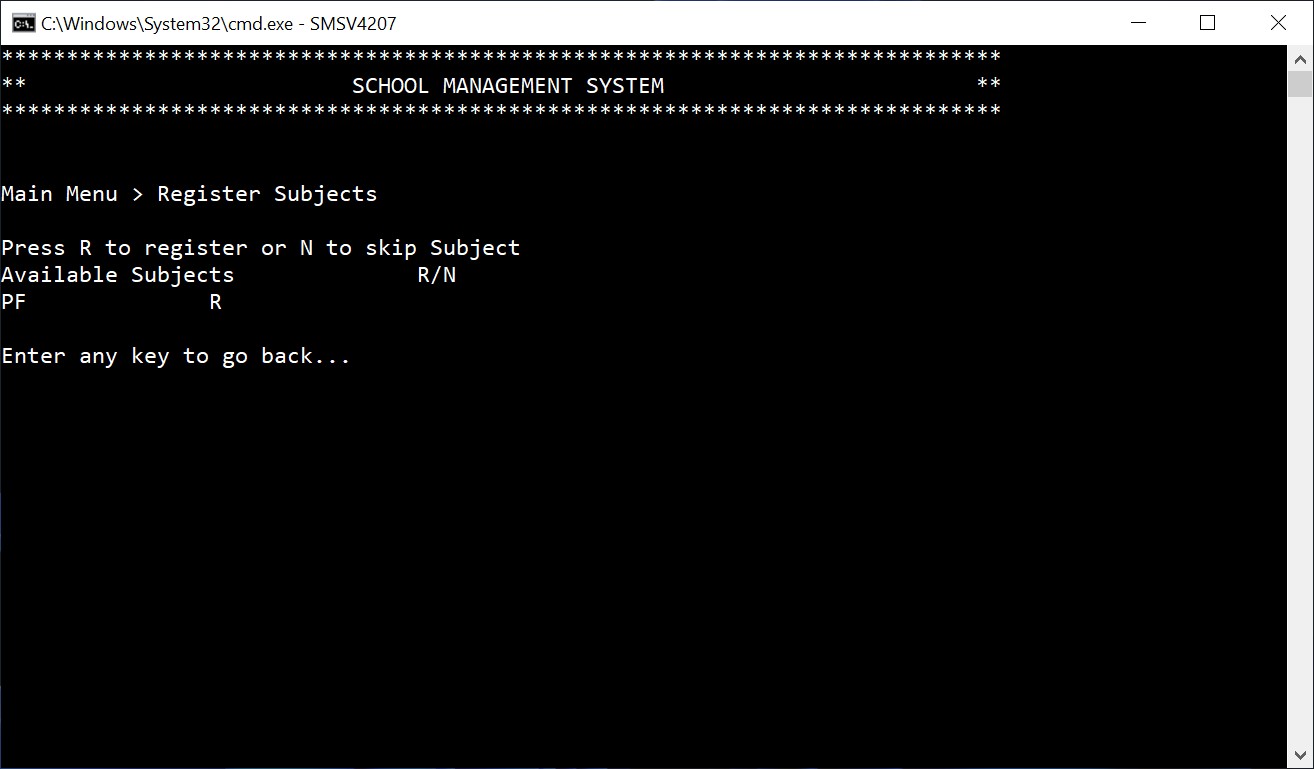
Student Main Menu

****

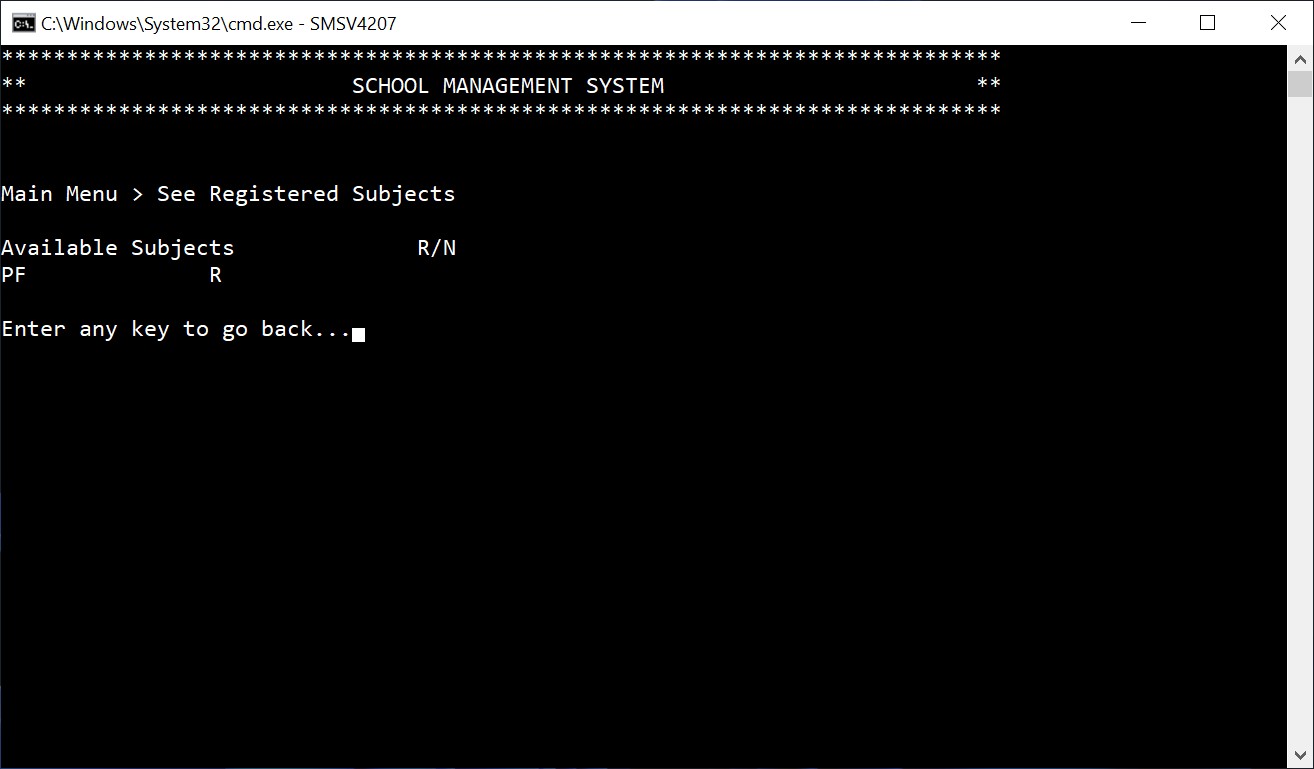
Option 1



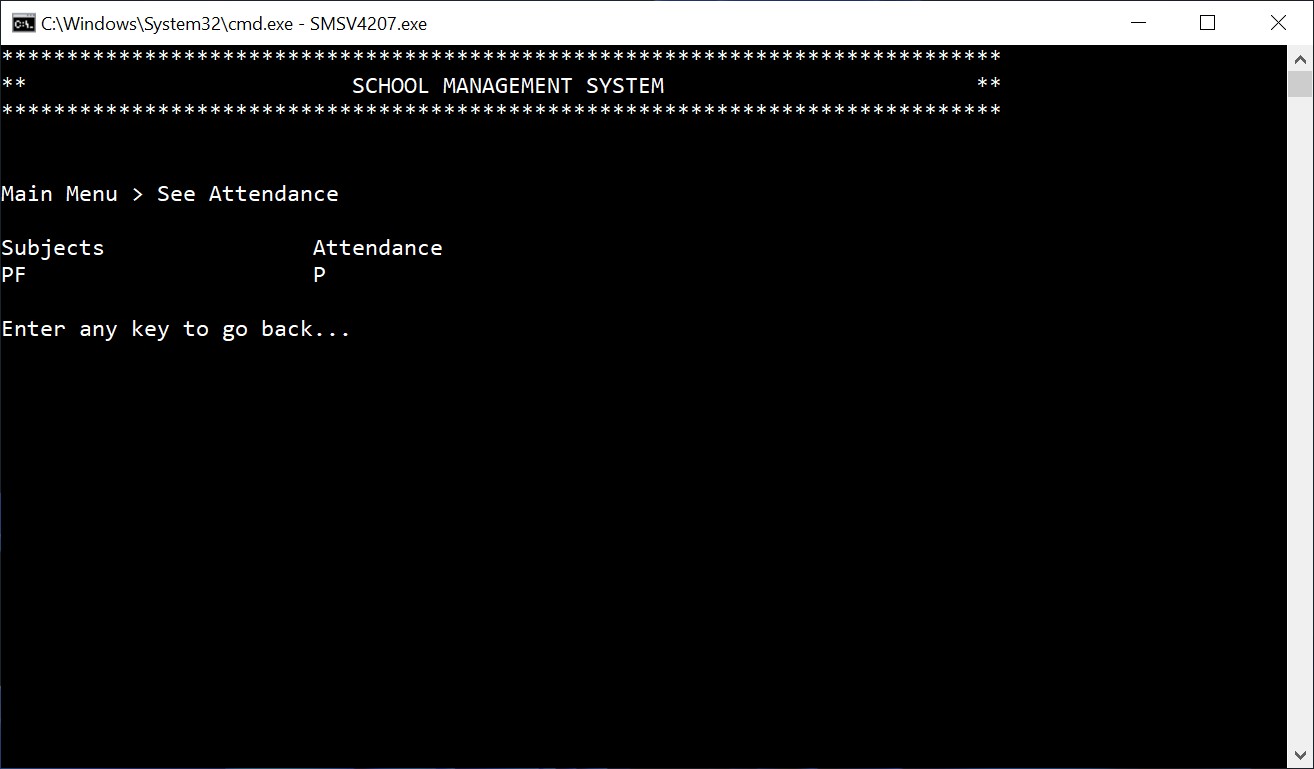
Option 2

****

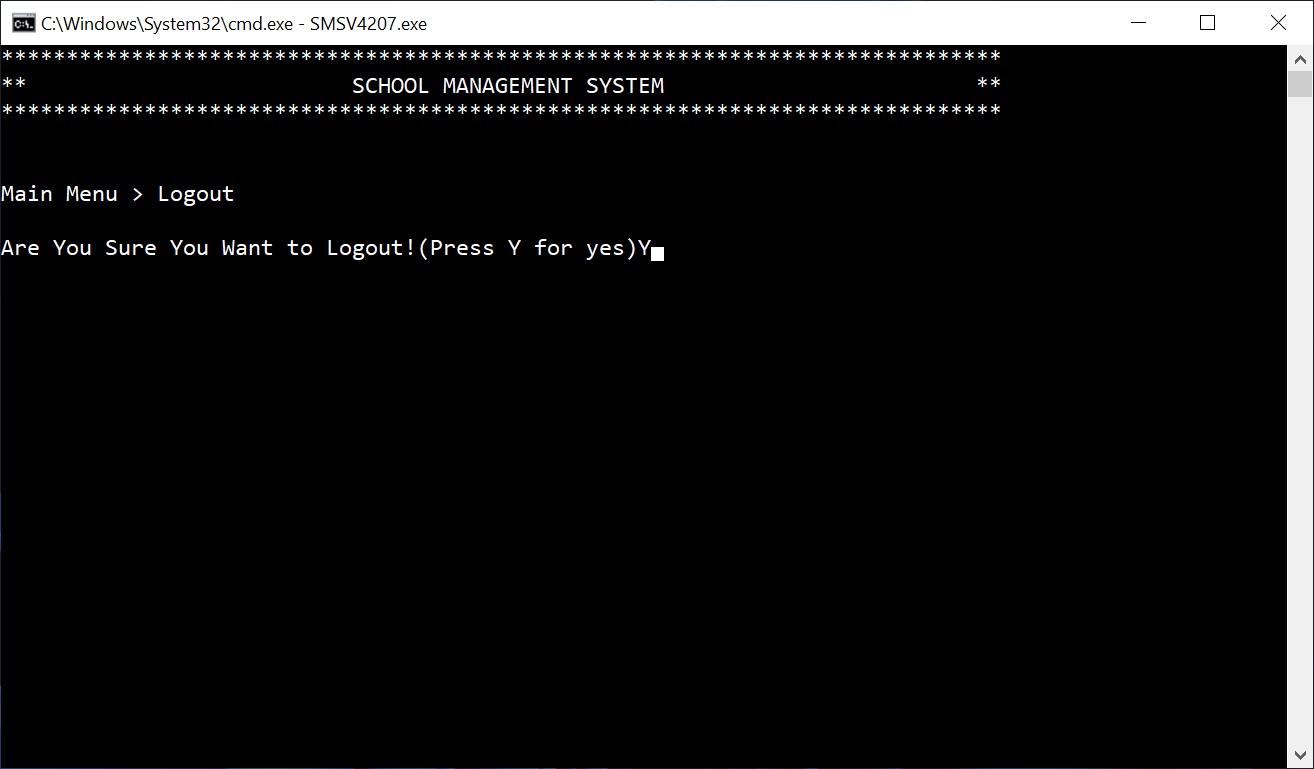
Option 3



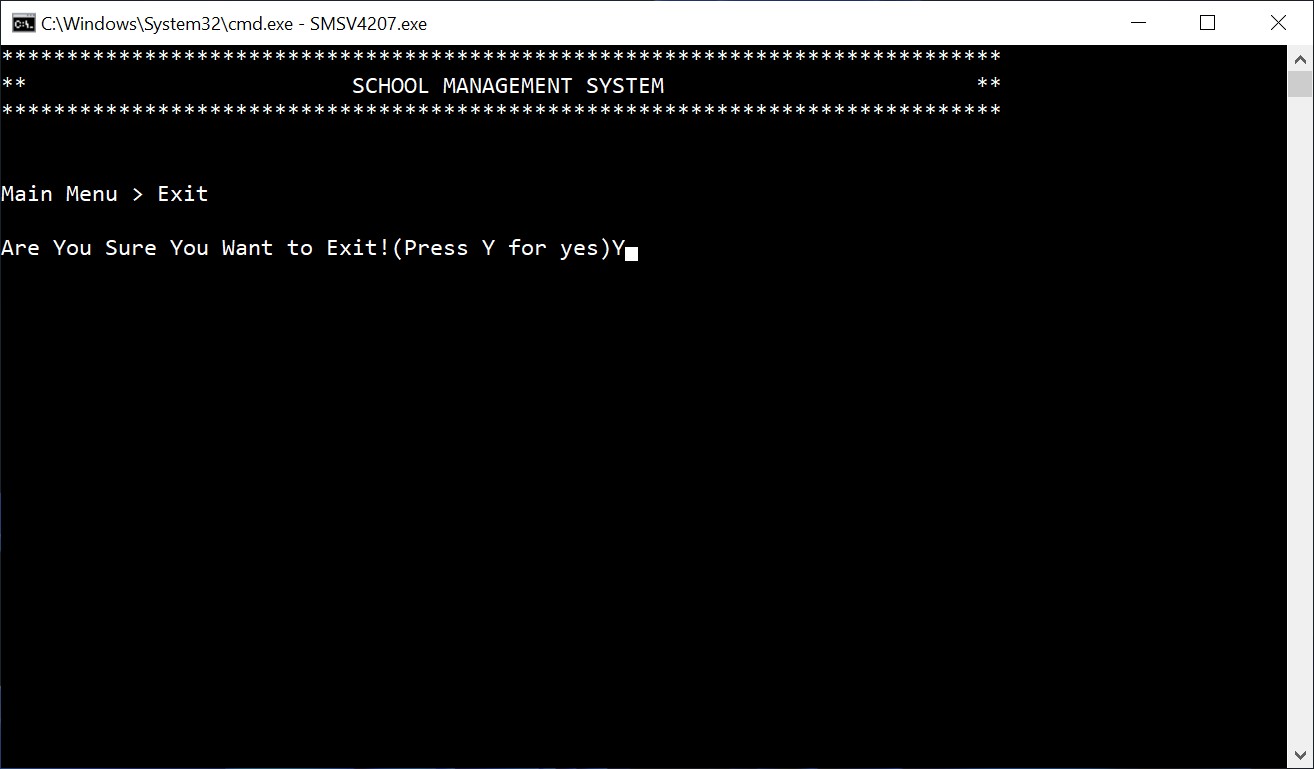
Option 4

****

Logout



Exit



**Data Structures:**

**Admin:**

string adminUser = "2021cs207"; // Default admin username

string adminPass = "207"; // Default admin password

string password; // To login

**Teacher:**

const int MAX\_TEACHER = 5;

string TeacherNameA[MAX\_TEACHER];

string TeacherSubjectA[MAX\_TEACHER];

int TeacherPayA[MAX\_TEACHER];

int TeacherAgeA[MAX\_TEACHER];

int TeacherLecturePerWeekA[MAX\_TEACHER];

int TeacherIDA[MAX\_TEACHER];

string TeacherPasswordA[MAX\_TEACHER];

char MarkAttendanceA[MAX\_STUDENT \* MAX\_TEACHER];

float TeacherMarksA[MAX\_STUDENT \* MAX\_TEACHER];

int TeacherCount = 0;

int LoggedInTeacher;

**Student:**

const int MAX\_STUDENT = 10;

float SortTemp[MAX\_STUDENT];

string SortName[MAX\_STUDENT];

int SortIdx[MAX\_STUDENT];

string StudentNameA[MAX\_STUDENT];

string StudentClassA[MAX\_STUDENT];

int StudentAgeA[MAX\_STUDENT];

int StudentIDA[MAX\_STUDENT];

string StudentPasswordA[MAX\_STUDENT];

char RegisterSubjectsA[MAX\_STUDENT \* MAX\_TEACHER];

int StudentCount = 0;

int LoggedInStudent;

**Functions Prototypes:**

void Header();

char LoginMenu();

char PrincipleMainMenu();

char TeacherMainMenu();

char StudentMainMenu();

void ScreenCls();

void AddTeacher();

void AddTeacherToArray(string name, string subject, int pay, int lecture, int age, int ID, string password);

void AddStudent();

void AddStudentToArray(string name, string Class, int age, int ID, string password);

void TeacherUpdate(int ID);

void StudentUpdate(int ID);

void ViewTeacher();

void ViewStudent();

void DeleteTeacher(int ID);

void DeleteStudent(int ID);

void LoginTeacher();

void LoginStudent();

void GiveMarks(int idx);

void SeeMarks(int idx);

void MarkAttendance(int idx);

void SeeMarkedAttendance(int idx);

void SubjectRegister();

void SeeRegisteredSubjects();

void ViewMarks();

void SeeAttendance();

void SortingOfMarks(int idx);

string parseRecord(string record, int field);

void ReadfromFile();

void LoadinFile();

**Function Working Flow:**

**Code:**

#include <iostream>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

#include <fstream>

using namespace std;

//-----------------------------------------------------------Function Prototypes Starts here

void Header();

char LoginMenu();

char PrincipleMainMenu();

char TeacherMainMenu();

char StudentMainMenu();

void ScreenCls();

void AddTeacher();

void AddTeacherToArray(string name, string subject, int pay, int lecture, int age, int ID, string password);

void AddStudent();

void AddStudentToArray(string name, string Class, int age, int ID, string password);

void TeacherUpdate(int ID);

void StudentUpdate(int ID);

void ViewTeacher();

void ViewStudent();

void DeleteTeacher(int ID);

void DeleteStudent(int ID);

void LoginTeacher();

void LoginStudent();

void GiveMarks(int idx);

void SeeMarks(int idx);

void MarkAttendance(int idx);

void SeeMarkedAttendance(int idx);

void SubjectRegister();

void SeeRegisteredSubjects();

void ViewMarks();

void SeeAttendance();

void SortingOfMarks(int idx);

string parseRecord(string record, int field);

void ReadfromFile();

void LoadinFile();

//-----------------------------------------------------------Function Prototypes Ends here

//-----------------------------------------------------------Variable Declaration Starts here

// When the user wants to enter as admin

string adminUser = "2021cs207"; // Default admin username

string adminPass = "207"; // Default admin password

string password; // To login

int ProgramExit = 0, logout = 0; // Our loops controller

int userName;

int found = 0;

const int MAX\_TEACHER = 5;

const int MAX\_STUDENT = 10;

float SortTemp[MAX\_STUDENT];

string SortName[MAX\_STUDENT];

int SortIdx[MAX\_STUDENT];

//-----------------------------------------Variables for Teachers Data Starts Here

string TeacherNameA[MAX\_TEACHER];

string TeacherSubjectA[MAX\_TEACHER];

int TeacherPayA[MAX\_TEACHER];

int TeacherAgeA[MAX\_TEACHER];

int TeacherLecturePerWeekA[MAX\_TEACHER];

int TeacherIDA[MAX\_TEACHER];

string TeacherPasswordA[MAX\_TEACHER];

char MarkAttendanceA[MAX\_STUDENT \* MAX\_TEACHER];

float TeacherMarksA[MAX\_STUDENT \* MAX\_TEACHER];

int TeacherCount = 0;

int LoggedInTeacher;

//-----------------------------------------Variables for Teachers Data Ends Here

//-----------------------------------------Variables for Students Data Starts Here

string StudentNameA[MAX\_STUDENT];

string StudentClassA[MAX\_STUDENT];

int StudentAgeA[MAX\_STUDENT];

int StudentIDA[MAX\_STUDENT];

string StudentPasswordA[MAX\_STUDENT];

char RegisterSubjectsA[MAX\_STUDENT \* MAX\_TEACHER];

int StudentCount = 0;

int LoggedInStudent;

//-----------------------------------------Variables for Students Data Ends Here

//-----------------------------------------------------------Variable Declaration Ends here

//-----------------------------------------------------------Main Function Starts here

main()

{

for (int i = 0; i < MAX\_STUDENT \* MAX\_TEACHER; i++)

{

RegisterSubjectsA[i] = 'N';

TeacherMarksA[i] = 0;

MarkAttendanceA[i] = 'A';

}

ReadfromFile();

while (true) // Main Loop of whole system Starts Here

{

logout = 0;

char option = LoginMenu();

if (option == '1') // Login option to enter as admin

{

while (true)

{

Header();

cout << "LOGIN PAGE\n\n";

cout << "Enter your username.....";

cin >> adminUser;

cout << "Enter your password.....";

cin >> adminPass;

if (adminUser == "2021cs207" && adminPass == "207")

{

while (true)

{

char PrincipleOption = PrincipleMainMenu();

Header();

if (PrincipleOption == '1') // Add Teacher

{

cout << "Main Menu > Add Teacher\n\n";

AddTeacher();

}

else if (PrincipleOption == '2') // Add Student

{

cout << "Main Menu > Add Student\n\n";

AddStudent();

}

else if (PrincipleOption == '3') // Update Teacher

{

cout << "Main Menu > Update Teacher Record \n\n";

cout << "Enter the ID of the teacher you want to update: ";

int updateID;

cin >> updateID;

TeacherUpdate(updateID);

}

else if (PrincipleOption == '4') // Update Student

{

cout << "Main Menu > Update Student Record \n\n";

cout << "Enter the ID of the student you want to update: ";

int updateID;

cin >> updateID;

StudentUpdate(updateID);

}

else if (PrincipleOption == '5')

{

cout << "Main Menu > View Record\n\n";

int view = 0;

cout << "Enter your option to view record\n";

cout << "Press1 to view all Teacher\n";

cout << "Press2 to view all Student\n";

cin >> view;

if (view == 1)

{

if (TeacherCount != 0)

{

Header();

cout << "Main Menu > View Record > Teacher\n\n";

ViewTeacher();

}

else

{

cout << "You do not have any teacher data to show please add one!" << endl;

}

}

else if (view == 2)

{

if (StudentCount != 0)

{

Header();

cout << "Main Menu > View Record > Student\n\n";

ViewStudent();

}

else

{

cout << "You do not have any Student data to show please add one!" << endl;

}

}

else

{

cout << "You entered wrong option" << endl;

}

}

else if (PrincipleOption == '6') // Delete Teacher Record

{

cout << "Main Menu > Delete Teacher Record \n\n";

cout << "Enter the ID of the teacher you want to update: ";

int DeleteID;

cin >> DeleteID;

DeleteTeacher(DeleteID);

}

else if (PrincipleOption == '7') // Delete Student Record

{

cout << "Main Menu > Delete Student Record\n\n";

cout << "Enter the ID of the Student you want to update: ";

int DeleteID;

cin >> DeleteID;

DeleteStudent(DeleteID);

}

else if (PrincipleOption == '8') // Admin Logout

{

cout << "Main Menu > Logout\n\n";

cout << "Are You Sure You Want to Logout!(Press Y for yes)";

char sure;

cin >> sure;

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

{

logout = 1;

break;

}

}

else if (PrincipleOption == '9') // Terminate Program

{

cout << "Main Menu > Exit\n\n";

cout << "Are You Sure You Want to Exit!(Press Y for yes)";

char sure;

cin >> sure;

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

{

ProgramExit = 1;

break;

}

}

else

{

cout << "You have entered the wrong option";

}

ScreenCls();

}

}

else if (adminUser == "2021cs207" && adminPass != "207")

{

Header();

cout << "You have entered incorrect password";

ScreenCls();

}

else if (adminUser != "2021cs207" && adminPass == "207")

{

Header();

cout << "You have entered incorrect username";

ScreenCls();

}

else if (adminUser != "2021cs207" && adminPass != "207")

{

Header();

cout << "You have entered incorrect username and password";

ScreenCls();

}

if (logout == 1 || ProgramExit == 1)

{

break;

}

}

}

else if (option == '2') // Login option to enter as teacher

{

while (true)

{

found = 0;

LoginTeacher();

if (found == 1)

{

while (true)

{

char option = TeacherMainMenu();

Header();

if (option == '1') // Teacher can give student marks

{

cout << "Main Menu > Give Marks\n\n";

GiveMarks(LoggedInTeacher);

}

else if (option == '2') // Teacher can see the marks of students

{

cout << "Main Menu > See Marks\n\n";

SeeMarks(LoggedInTeacher);

}

else if (option == '3') // Mark Attendance

{

cout << "Main Menu > Mark Attendance\n\n";

MarkAttendance(LoggedInTeacher);

}

else if (option == '4') // See Marked Attendance

{

cout << "Main Menu > View Attendance\n\n";

SeeMarkedAttendance(LoggedInTeacher);

}

else if (option == '5') // Teacher Logout

{

cout << "Main Menu > Logout\n\n";

cout << "Are You Sure You Want to Logout!(Press Y for yes)";

char sure;

cin >> sure;

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

{

logout = 1;

break;

}

}

else if (option == '6') // Terminate Program

{

cout << "Main Menu > Exit\n\n";

cout << "Are You Sure You Want to Exit!(Press Y for yes)";

char sure;

cin >> sure;

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

{

ProgramExit = 1;

break;

}

}

else

{

cout << "You have entered the wrong option";

}

ScreenCls();

}

if (logout == 1 || ProgramExit == 1)

{

break;

}

}

else

{

break;

}

}

}

else if (option == '3') // Login option to enter as student

{

while (true)

{

found = 0;

LoginStudent();

if (found == 1)

{

while (true)

{

char option = StudentMainMenu();

Header();

if (option == '1') // Student can see his/her marks

{

cout << "Main Menu > See Marks\n\n";

ViewMarks();

}

else if (option == '2') // Students Can Register to Available subjects

{

cout << "Main Menu > Register Subjects\n\n";

SubjectRegister();

}

else if (option == '3') // Students can see The subjects they have registered

{

cout << "Main Menu > See Registered Subjects\n\n";

SeeRegisteredSubjects();

}

else if (option == '4') // See Marked Attendance

{

cout << "Main Menu > See Attendance\n\n";

SeeAttendance();

}

else if (option == '5') // Student Logout

{

cout << "Main Menu > Logout\n\n";

cout << "Are You Sure You Want to Logout!(Press Y for yes)";

char sure;

cin >> sure;

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

{

logout = 1;

break;

}

}

else if (option == '6') // Terminate Program

{

cout << "Main Menu > Exit\n\n";

cout << "Are You Sure You Want to Exit!(Press Y for yes)";

char sure;

cin >> sure;

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

{

ProgramExit = 1;

break;

}

}

else

{

cout << "You have entered the wrong option";

}

ScreenCls();

}

if (logout == 1 || ProgramExit == 1)

{

break;

}

}

else

{

break;

}

}

}

else

{

cout << "You have entered the wrong option";

}

if (ProgramExit == 1)

{

break;

}

ScreenCls();

}

LoadinFile();

// Main Loop of whole system Ends Here

}

//-----------------------------------------------------------Main Function Ends here

void Header() // this is the header that will be displayed at the beginning of every screen

{

system("cls");

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\* SCHOOL MANAGEMENT SYSTEM \*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "" << endl;

cout << "" << endl;

}

char LoginMenu()

{

Header();

char option = ' ';

cout << "Enter the option you want to login as" << endl;

cout << "Press1 to login as principle\n";

cout << "Press2 to login as teacher\n";

cout << "Press3 to login as student\n";

cin >> option;

return option;

}

// Admin Function Start

char PrincipleMainMenu() // If the principle is logged in the system then this menu will be displayed

{

Header();

char option;

cout << "Main Menu\n\n";

cout << "Press1 to add teachers\n";

cout << "Press2 add students\n";

cout << "Press3 to update the record of teacher\n";

cout << "Press4 to update the record of student\n";

cout << "Press5 to view all the records\n";

cout << "Press6 to remove the teacher\n";

cout << "Press7 to remove students\n";

cout << "Press8 to logout\n";

cout << "Press9 to exit\n";

cin >> option;

return option;

}

void AddTeacher()

{

if (TeacherCount != MAX\_TEACHER)

{

string name, subject;

int age, pay, lecture, ID;

string password;

cout << "Enter name of Teacher : ";

cin >> name;

cout << "Enter Teacher age: ";

cin >> age;

while (age < 25)

{

cout << "Enter a valid age(Age must be greater than 25)";

cin >> age;

if (age > 25)

{

break;

}

}

cout << "Enter Pay of Teacher: ";

cin >> pay;

cout << "Enter subject of teacher: ";

cin >> subject;

cout << "Enter Leactures per week to be delivered: ";

cin >> lecture;

cout << "Enter ID of Teacher: ";

cin >> ID;

while (true)

{

cout << "Enter password of teacher: ";

cin >> password;

int count = 0;

for (int i = 0; password[i] != '\0'; i++)

{

if (password[i] == '!' || password[i] == '@' || password[i] == '#' || password[i] == '$' || password[i] == '%' || password[i] == '&')

{

count++;

}

}

if (count != 0)

{

break;

}

if (count == 0)

{

cout << "Please Enter a valid Password" << endl;

cout << "Password must contain atleast 1 Special Character" << endl;

}

}

AddTeacherToArray(name, subject, pay, lecture, age, ID, password);

}

else

{

cout << "You do not have enough space" << endl;

}

}

void AddTeacherToArray(string name, string subject, int pay, int lecture, int age, int ID, string password)

{

TeacherNameA[TeacherCount] = name;

TeacherSubjectA[TeacherCount] = subject;

TeacherPayA[TeacherCount] = pay;

TeacherLecturePerWeekA[TeacherCount] = lecture;

TeacherAgeA[TeacherCount] = age;

TeacherIDA[TeacherCount] = ID;

TeacherPasswordA[TeacherCount] = password;

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

{

if (TeacherIDA[i] == ID)

{

cout << "You have set this particular ID to another Teacher So you cannot add this teacher Plz change the ID" << endl;

cout << "Enter a valid ID: ";

cin >> TeacherIDA[TeacherCount];

i = 0;

}

}

TeacherCount++;

}

void AddStudent()

{

if (MAX\_STUDENT != StudentCount)

{

string name, Class;

int age, ID;

string password;

cout << "Enter name of student: ";

cin >> name;

cout << "Enter Student age: ";

cin >> age;

while (age > 20)

{

cout << "Enter a valid age(Age must be Less than 20)";

cin >> age;

if (age < 20)

{

break;

}

}

cout << "Enter Student class: ";

cin >> Class;

cout << "Enter Student ID: ";

cin >> ID;

while (true)

{

cout << "Enter password of student: ";

cin >> password;

int count = 0;

for (int i = 0; password[i] != '\0'; i++)

{

if (password[i] == '!' || password[i] == '@' || password[i] == '#' || password[i] == '$' || password[i] == '%' || password[i] == '&')

{

count++;

}

}

if (count != 0)

{

break;

}

if (count == 0)

{

cout << "Please Enter a valid Password" << endl;

cout << "Password must contain atleast 1 Special Character" << endl;

}

}

AddStudentToArray(name, Class, age, ID, password);

}

else

{

cout << "You do not have enough space" << endl;

}

}

void AddStudentToArray(string name, string Class, int age, int ID, string password)

{

StudentNameA[StudentCount] = name;

StudentClassA[StudentCount] = Class;

StudentAgeA[StudentCount] = age;

StudentIDA[StudentCount] = ID;

StudentPasswordA[StudentCount] = password;

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

{

if (StudentIDA[i] == ID)

{

cout << "You have set this particular ID to another Student So you cannot add this Student Plz change the ID" << endl;

cout << "Enter new ID: ";

cin >> StudentIDA[StudentCount];

}

}

StudentCount++;

}

void TeacherUpdate(int ID)

{

int found = 0;

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

{

if (ID == TeacherIDA[i])

{

found = 1;

cout << "Enter name of Teacher : ";

cin >> TeacherNameA[i];

cout << "Enter Teacher age: ";

cin >> TeacherAgeA[i];

cout << "Enter Pay of Teacher: ";

cin >> TeacherPayA[i];

cout << "Enter subject of teacher: ";

cin >> TeacherSubjectA[i];

cout << "Enter Leactures per week to be delivered: ";

cin >> TeacherLecturePerWeekA[i];

cout << "Enter ID of Teacher: ";

cin >> TeacherIDA[i];

while (true)

{

cout << "Enter password of teacher: ";

cin >> password;

int count = 0;

for (int i = 0; password[i] != '\0'; i++)

{

if (password[i] == '!' || password[i] == '@' || password[i] == '#' || password[i] == '$' || password[i] == '%' || password[i] == '&')

{

count++;

}

}

if (count != 0)

{

break;

}

if (count == 0)

{

cout << "Please Enter a valid Password" << endl;

cout << "Password must contain atleast 1 Special Character" << endl;

}

}

break;

}

}

if (found == 0)

{

cout << "No such record found" << endl;

}

else

{

cout << "Record updated" << endl;

}

}

void StudentUpdate(int ID)

{

int found = 0;

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

{

if (ID == StudentIDA[i])

{

found = 1;

cout << "Enter name of Student : ";

cin >> StudentNameA[i];

cout << "Enter Student age: ";

cin >> StudentAgeA[i];

cout << "Enter class of Student: ";

cin >> StudentClassA[i];

cout << "Enter ID of Student: ";

cin >> StudentIDA[i];

while (true)

{

cout << "Enter password of student: ";

cin >> password;

int count = 0;

for (int i = 0; password[i] != '\0'; i++)

{

if (password[i] == '!' || password[i] == '@' || password[i] == '#' || password[i] == '$' || password[i] == '%' || password[i] == '&')

{

count++;

}

}

if (count != 0)

{

break;

}

if (count == 0)

{

cout << "Please Enter a valid Password" << endl;

cout << "Password must contain atleast 1 Special Character" << endl;

}

}

break;

}

}

if (found == 0)

{

cout << "No such record found" << endl;

}

else

{

cout << "Record updated" << endl;

}

}

void ViewTeacher()

{

int idx[TeacherCount];

int Temp[TeacherCount];

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

{

Temp[i] = TeacherPayA[i];

idx[i] = i;

}

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

{

int temp;

for (int j = 0; j < TeacherCount - 1; j++)

{

temp = Temp[j];

int idx\_temp = idx[j];

if (temp < Temp[j + 1])

{

Temp[j] = Temp[j + 1];

Temp[j + 1] = temp;

idx[j] = idx[j + 1];

idx[j + 1] = idx\_temp;

}

}

}

cout << "Name\t\tAge\t\tSubject\t\tLecturesPW\t\tSalary\t\tID" << endl;

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

{

cout << TeacherNameA[idx[i]] << "\t\t" << TeacherAgeA[idx[i]] << "\t\t" << TeacherSubjectA[idx[i]] << "\t\t" << TeacherLecturePerWeekA[idx[i]] << "\t\t\t" << TeacherPayA[idx[i]] << "\t\t" << TeacherIDA[idx[i]] << endl;

}

}

void ViewStudent()

{

int idx[StudentCount];

int Temp[StudentCount];

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

{

Temp[i] = StudentIDA[i];

idx[i] = i;

}

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

{

int temp;

for (int j = 0; j < StudentCount - 1; j++)

{

temp = Temp[j];

int idx\_temp = idx[j];

if (temp < Temp[j + 1])

{

Temp[j] = Temp[j + 1];

Temp[j + 1] = temp;

idx[j] = idx[j + 1];

idx[j + 1] = idx\_temp;

}

}

}

cout << "Name\t\tAge\t\tClass\t\tID" << endl;

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

{

cout << StudentNameA[idx[i]] << "\t\t" << StudentAgeA[idx[i]] << "\t\t" << StudentClassA[idx[i]] << "\t\t" << StudentIDA[idx[i]] << endl;

}

}

void DeleteTeacher(int ID)

{

int deleted = 0;

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

{

if (TeacherIDA[i] == ID)

{

for (int j = i; j < TeacherCount - 1; j++)

{

TeacherNameA[i] = TeacherNameA[i + 1];

TeacherSubjectA[i] = TeacherSubjectA[i + 1];

TeacherPayA[i] = TeacherPayA[i + 1];

TeacherLecturePerWeekA[i] = TeacherLecturePerWeekA[i + 1];

TeacherAgeA[i] = TeacherAgeA[i + 1];

TeacherIDA[i] = TeacherIDA[i + 1];

TeacherPasswordA[i] = TeacherPasswordA[i + 1];

}

TeacherCount--;

deleted = 1;

break;

}

}

if (deleted == 1)

{

cout << "Teacher Record Deleted Successfully" << endl;

}

else

{

cout << "No such Record Found!!" << endl;

}

}

void DeleteStudent(int ID)

{

int deleted = 0;

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

{

if (StudentIDA[i] == ID)

{

for (int j = i; j < StudentCount - 1; j++)

{

StudentNameA[i] = StudentNameA[i + 1];

StudentAgeA[i] = StudentAgeA[i + 1];

StudentClassA[i] = StudentClassA[i + 1];

StudentIDA[i] = StudentIDA[i + 1];

StudentPasswordA[i] = StudentPasswordA[i + 1];

}

StudentCount--;

deleted = 1;

break;

}

}

if (deleted == 1)

{

cout << "Student Record Deleted Successfully" << endl;

}

else

{

cout << "No such Record Found!!" << endl;

}

}

// Teacher Function Start

void LoginTeacher()

{

Header();

int ID;

string password;

cout << "LOGIN PAGE\n\n";

cout << "Enter your ID.....";

cin >> ID;

cout << "Enter your password.....";

cin >> password;

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

{

if (ID == TeacherIDA[i] && password == TeacherPasswordA[i])

{

LoggedInTeacher = i;

found = 1;

break;

}

}

if (found != 1)

{

cout << "You have entered wrong credentials";

}

}

char TeacherMainMenu() // If a teacher is logged in to the system then this menu will be displayed

{

Header();

char option;

cout << "Main Menu\n\n";

cout << "Press1 to give students marks\n";

cout << "Press2 to see marks of all students\n";

cout << "Press3 to mark attendance of students\n";

cout << "Press4 to see attendance of students\n";

cout << "Press5 to logout\n";

cout << "Press6 to exit\n";

cin >> option;

return option;

}

void MarkAttendance(int idx)

{

if (StudentCount != 0)

{

cout << "Name\t\tA/P" << endl;

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

{

if (RegisterSubjectsA[idx + (MAX\_TEACHER \* i)] == 'R' || RegisterSubjectsA[idx + (MAX\_TEACHER \* i)] == 'r')

{

cout << StudentNameA[i] << "\t\t";

// cout << MarkAttendanceA[idx + (MAX\_TEACHER \* i)];

cin >> MarkAttendanceA[idx + (MAX\_TEACHER \* i)];

}

}

}

else

{

cout << "No Students Added Yet!" << endl;

}

}

void GiveMarks(int idx)

{

int j = 0;

if (StudentCount != 0)

{

cout << "Name\t\tMarks" << endl;

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

{

if (RegisterSubjectsA[idx + (MAX\_TEACHER \* i)] == 'R' || RegisterSubjectsA[idx + (MAX\_TEACHER \* i)] == 'r')

{

cout << StudentNameA[i] << "\t\t";

cin >> TeacherMarksA[idx + (MAX\_TEACHER \* i)];

}

}

}

else

{

cout << "No Students Added Yet!" << endl;

}

}

void SortingOfMarks(int idx)

{

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

{

int tem;

string nameS;

for (int j = 0; j < idx - 1; j++)

{

tem = SortTemp[j];

nameS = SortName[j];

if (tem < SortTemp[j + 1])

{

SortTemp[j] = SortTemp[j + 1];

SortName[j] = SortName[j + 1];

SortTemp[j + 1] = tem;

SortName[j + 1] = nameS;

}

}

}

}

void SeeMarks(int idx)

{

if (StudentCount != 0)

{

int id = 0;

cout << "Name\t\tMarks" << endl;

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

{

if (RegisterSubjectsA[idx + (MAX\_TEACHER \* i)] == 'R' || RegisterSubjectsA[idx + (MAX\_TEACHER \* i)] == 'r')

{

// cout << StudentNameA[i] << "\t\t" << TeacherMarksA[idx + (MAX\_TEACHER \* i)] << endl;

SortName[id] = StudentNameA[i];

SortTemp[id] = TeacherMarksA[idx + (MAX\_TEACHER \* i)];

id++;

}

}

SortingOfMarks(id);

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

{

cout << SortName[i] << "\t\t" << SortTemp[i] << endl;

}

}

else

{

cout << "No Students Added Yet!" << endl;

}

}

void SeeMarkedAttendance(int idx)

{

if (StudentCount != 0)

{

int id = 0;

cout << "Name\t\tAttendance" << endl;

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

{

if (RegisterSubjectsA[idx + (MAX\_TEACHER \* i)] == 'R' || RegisterSubjectsA[idx + (MAX\_TEACHER \* i)] == 'r')

{

cout << StudentNameA[i] << "\t\t" << MarkAttendanceA[idx + (MAX\_TEACHER \* i)] << endl;

}

}

}

else

{

cout << "No Students Added Yet!" << endl;

}

}

// Student Function Start

void LoginStudent()

{

Header();

int ID;

string password;

cout << "LOGIN PAGE\n\n";

cout << "Enter your ID.....";

cin >> ID;

cout << "Enter your password.....";

cin >> password;

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

{

if (ID == StudentIDA[i] && password == StudentPasswordA[i])

{

LoggedInStudent = i;

found = 1;

break;

}

}

if (found != 1)

{

cout << "You have entered wrong credentials";

}

}

char StudentMainMenu() // If a student is logged in to the system then this menu will be displayed

{

char option;

Header();

cout << "Main Menu\n\n";

cout << "Press1 to see marks" << endl;

cout << "Press2 to register subjects" << endl;

cout << "Press3 to see registered subjects" << endl;

cout << "Press4 to see Attendance" << endl;

cout << "Press5 to logout" << endl;

cout << "Press6 to exit" << endl;

cin >> option;

return option;

}

void ViewMarks()

{

int j = 0;

cout << "Subject\t\tMarks\n";

for (int i = LoggedInStudent \* MAX\_TEACHER; i < (LoggedInStudent \* MAX\_TEACHER) + TeacherCount; i++)

{

if (RegisterSubjectsA[i] == 'R' || RegisterSubjectsA[i] == 'r')

{

cout << TeacherSubjectA[j] << "\t\t" << TeacherMarksA[i] << endl;

}

j++;

}

}

void SubjectRegister()

{

int j = 0;

cout << "Press R to register or N to skip Subject" << endl;

cout << "Available Subjects\t\tR/N" << endl;

for (int i = LoggedInStudent \* MAX\_TEACHER; i < (LoggedInStudent \* MAX\_TEACHER) + TeacherCount; i++)

{

cout << TeacherSubjectA[j] << "\t\t";

cin >> RegisterSubjectsA[i];

j++;

}

}

void SeeRegisteredSubjects()

{

int j = 0;

cout << "Available Subjects\t\tR/N" << endl;

for (int i = LoggedInStudent \* MAX\_TEACHER; i < (LoggedInStudent \* MAX\_TEACHER) + TeacherCount; i++)

{

if (RegisterSubjectsA[i] == 'R' || RegisterSubjectsA[i] == 'r')

{

cout << TeacherSubjectA[j] << "\t\t" << RegisterSubjectsA[i] << endl;

}

j++;

}

}

void SeeAttendance()

{

int j = 0;

cout << "Subjects\t\tAttendance" << endl;

for (int i = LoggedInStudent \* MAX\_TEACHER; i < (LoggedInStudent \* MAX\_TEACHER) + TeacherCount; i++)

{

if (RegisterSubjectsA[i] == 'R' || RegisterSubjectsA[i] == 'r')

{

cout << TeacherSubjectA[j] << "\t\t\t" << MarkAttendanceA[i] << endl;

}

j++;

}

}

void ScreenCls()

{

cout << "\nEnter any key to go back...";

getch();

system("cls");

}

string parseRecord(string record, int field)

{

int commaCount = 1;

string item;

for (int x = 0; x < record.length(); x++)

{

if (record[x] == ',')

{

commaCount = commaCount + 1;

}

else if (commaCount == field)

{

item = item + record[x];

}

}

return item;

}

void ReadfromFile()

{

string word;

int i = 0;

fstream ReadStudent;

ReadStudent.open("StudentData.txt", ios::in);

while (!ReadStudent.eof())

{

ReadStudent >> StudentNameA[i] >> StudentClassA[i] >> StudentAgeA[i] >> StudentIDA[i] >> StudentPasswordA[i];

i++;

}

StudentCount = i - 1;

i = 0;

ReadStudent.close();

fstream ReadTeacher;

ReadTeacher.open("TeacherData.txt", ios::in);

while (!ReadTeacher.eof())

{

ReadTeacher >> TeacherNameA[i] >> TeacherSubjectA[i] >> TeacherPayA[i] >> TeacherLecturePerWeekA[i] >> TeacherAgeA[i] >> TeacherIDA[i] >> TeacherPasswordA[i];

i++;

}

TeacherCount = i - 1;

ReadTeacher.close();

fstream ReadRegisterSubject;

ReadRegisterSubject.open("RegisteredSubjects.txt", ios::in);

for (int i = 0; i < MAX\_STUDENT \* MAX\_TEACHER; i++)

{

ReadRegisterSubject >> RegisterSubjectsA[i];

}

ReadRegisterSubject.close();

fstream ReadMarks;

ReadMarks.open("Marks.txt", ios::in);

for (int i = 0; i < MAX\_STUDENT \* MAX\_TEACHER; i++)

{

ReadMarks >> TeacherMarksA[i];

}

ReadMarks.close();

fstream ReadAttend;

ReadAttend.open("Attendance.txt", ios::in);

for (int i = 0; i < MAX\_STUDENT \* MAX\_TEACHER; i++)

{

ReadAttend >> MarkAttendanceA[i];

}

ReadAttend.close();

}

void LoadinFile()

{

// Storing Student's Data in File

fstream SaveStudent;

SaveStudent.open("StudentData.txt", ios::out);

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

{

SaveStudent << StudentNameA[i] << "\t" << StudentClassA[i] << "\t" << StudentAgeA[i] << "\t" << StudentIDA[i] << "\t" << StudentPasswordA[i] << endl;

}

SaveStudent.close();

// Storing Teacher's data in File

fstream SaveTeacher;

SaveTeacher.open("TeacherData.txt", ios::out);

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

{

SaveTeacher << TeacherNameA[i] << "\t" << TeacherSubjectA[i] << "\t" << TeacherPayA[i] << "\t" << TeacherLecturePerWeekA[i] << "\t" << TeacherAgeA[i] << "\t" << TeacherIDA[i] << "\t" << TeacherPasswordA[i] << endl;

}

SaveTeacher.close();

// Store registered Subjects

fstream SaveSubjects;

SaveSubjects.open("RegisteredSubjects.txt", ios::out);

for (int i = 0; i < MAX\_STUDENT \* MAX\_TEACHER; i++)

{

SaveSubjects << RegisterSubjectsA[i] << endl;

}

SaveSubjects.close();

// Store Marks Given To students

fstream SaveMarks;

SaveMarks.open("Marks.txt", ios::out);

for (int i = 0; i < MAX\_STUDENT \* MAX\_TEACHER; i++)

{

SaveMarks << TeacherMarksA[i] << endl;

}

SaveMarks.close();

// Save Attendance

fstream SaveAttend;

SaveAttend.open("Attendance.txt", ios::out);

for (int i = 0; i < MAX\_STUDENT \* MAX\_TEACHER; i++)

{

SaveAttend << MarkAttendanceA[i] << endl;

}

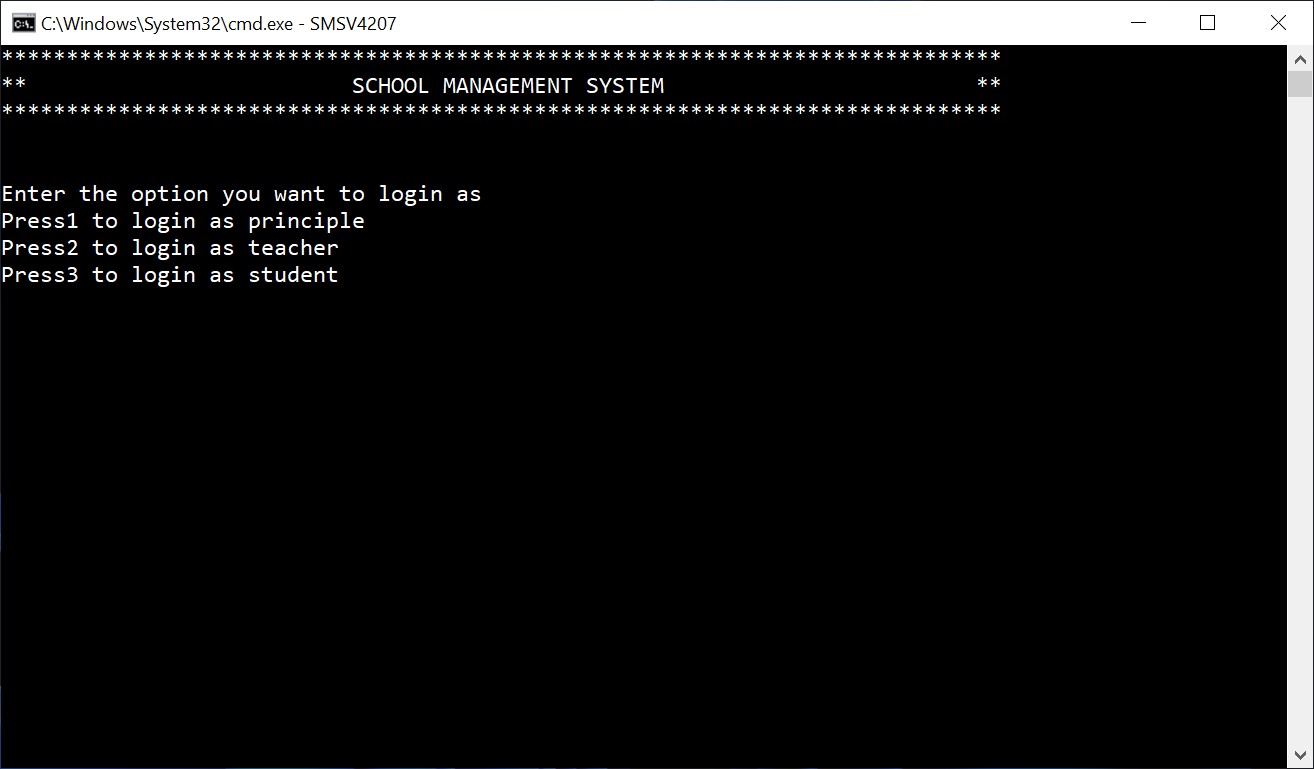
SaveAttend.close();

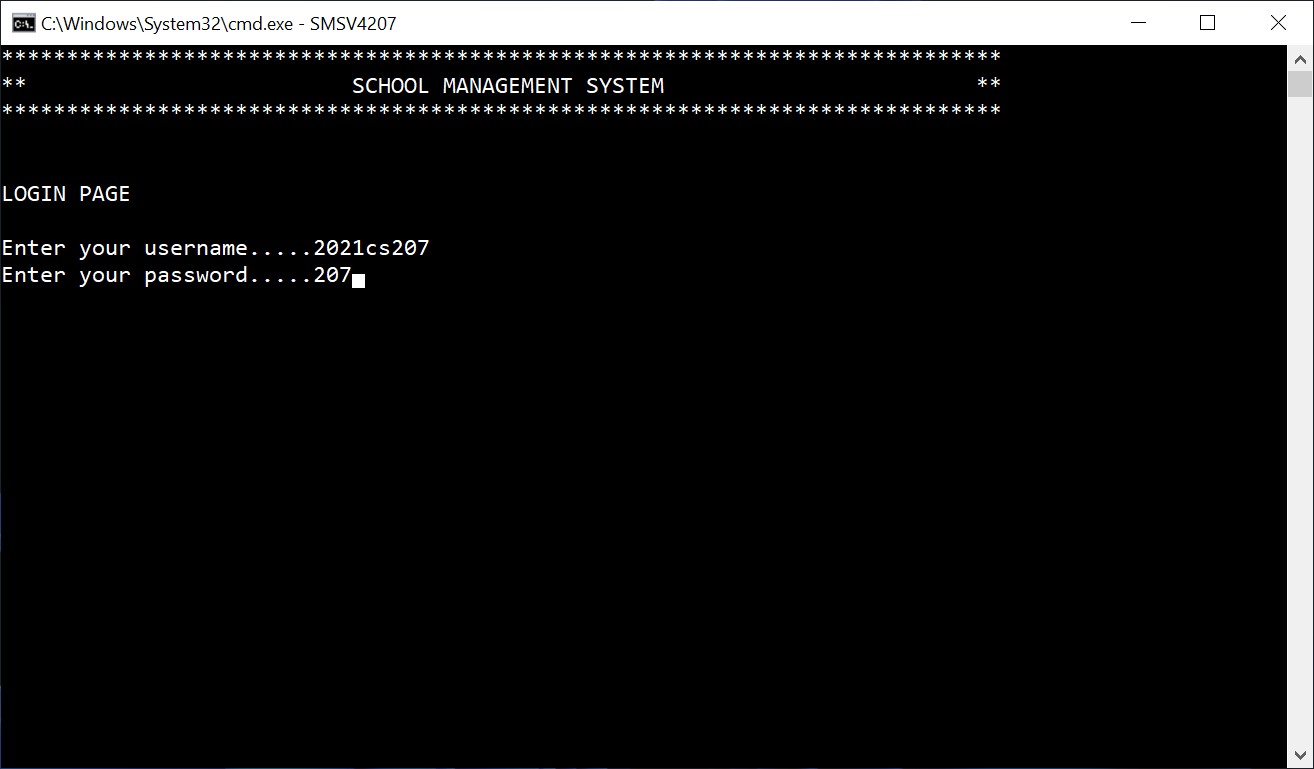
}

//-----------------------------------------------------------Function Implementation Ends here

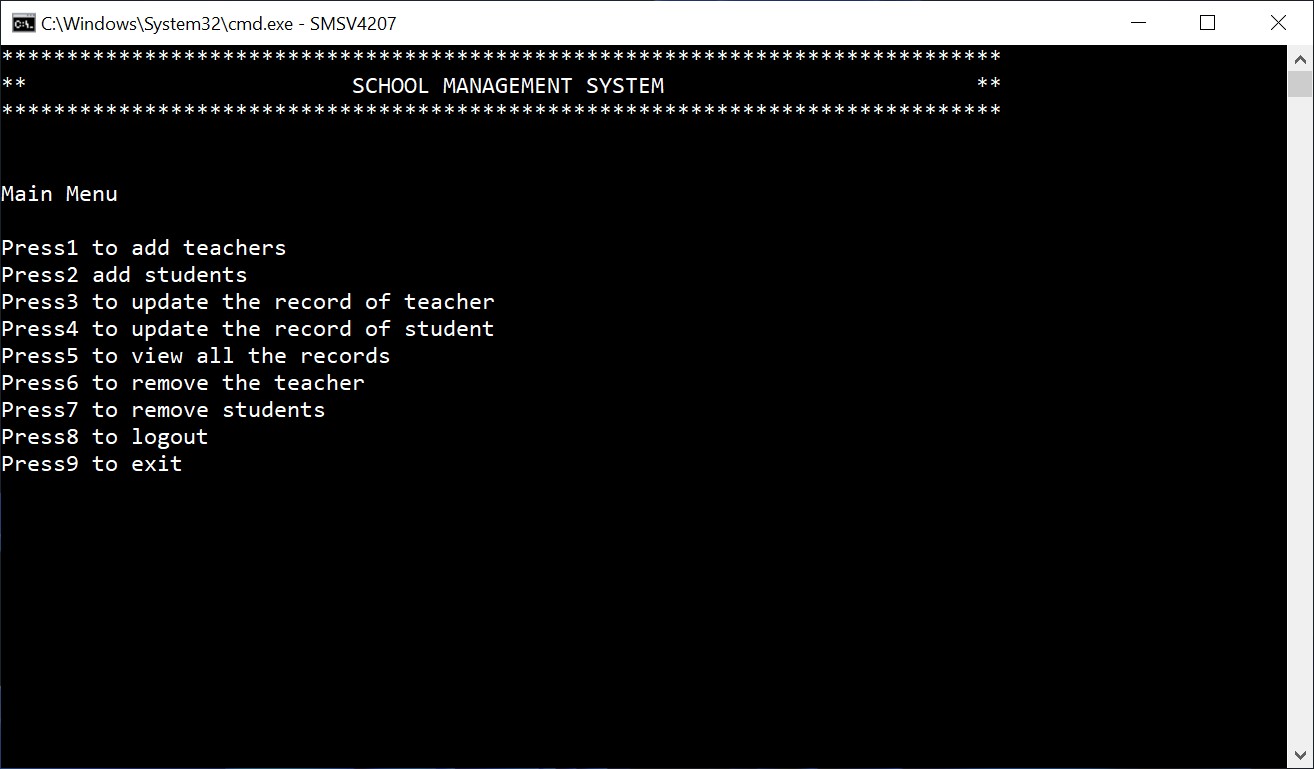
**Test Cases:**

Login Option

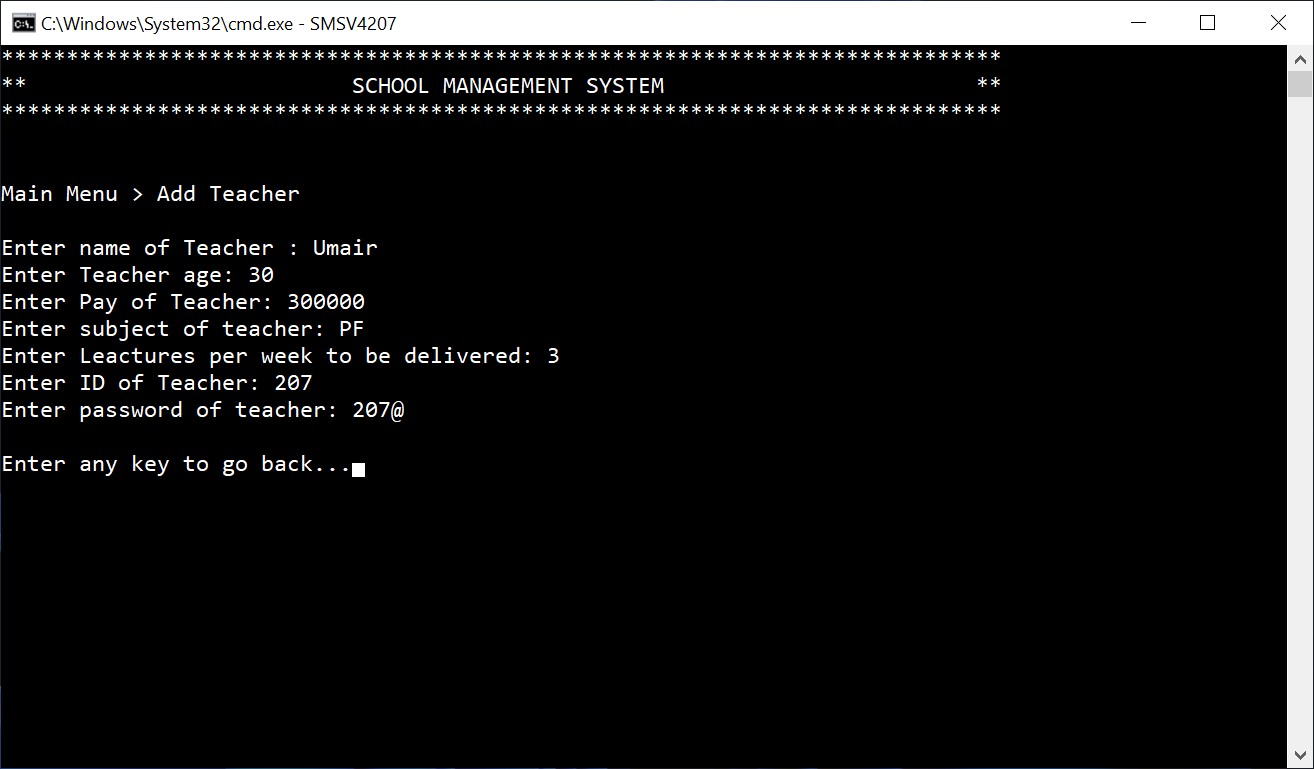


Admin Login Page

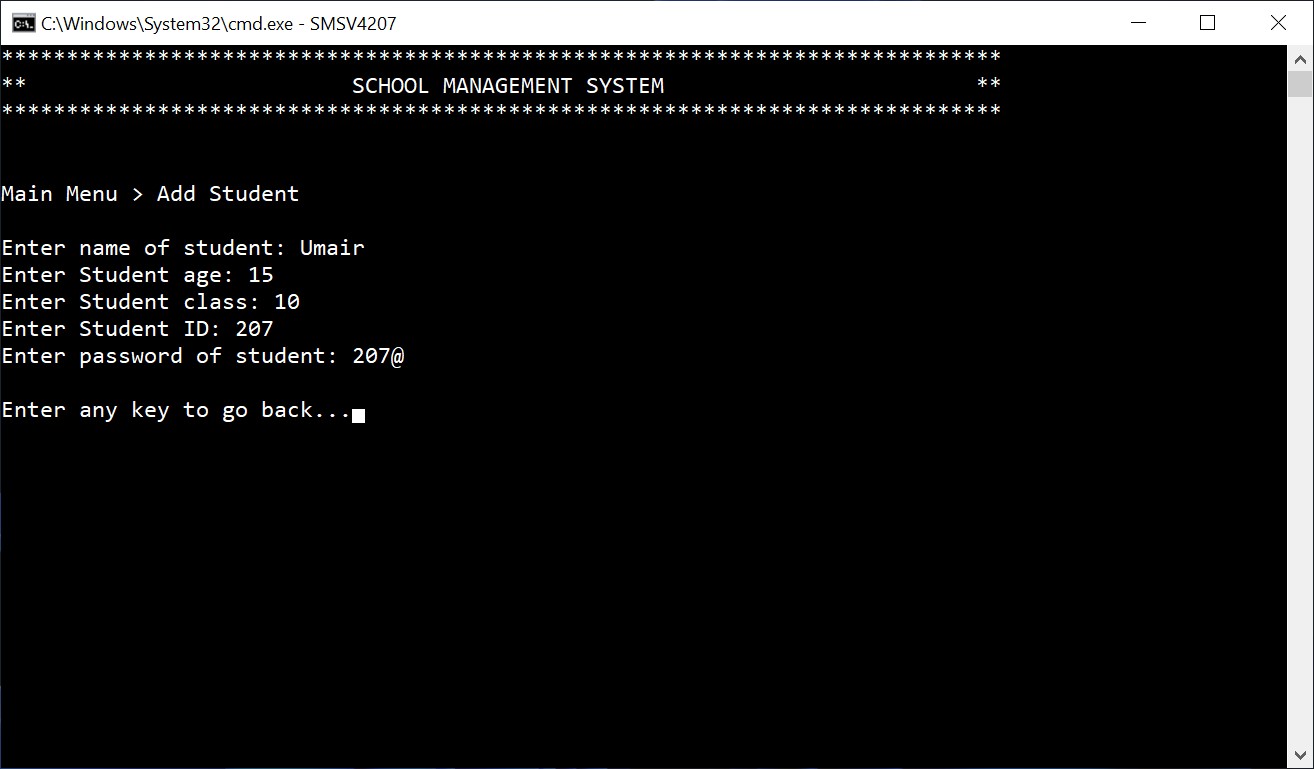
Admin Main Menu



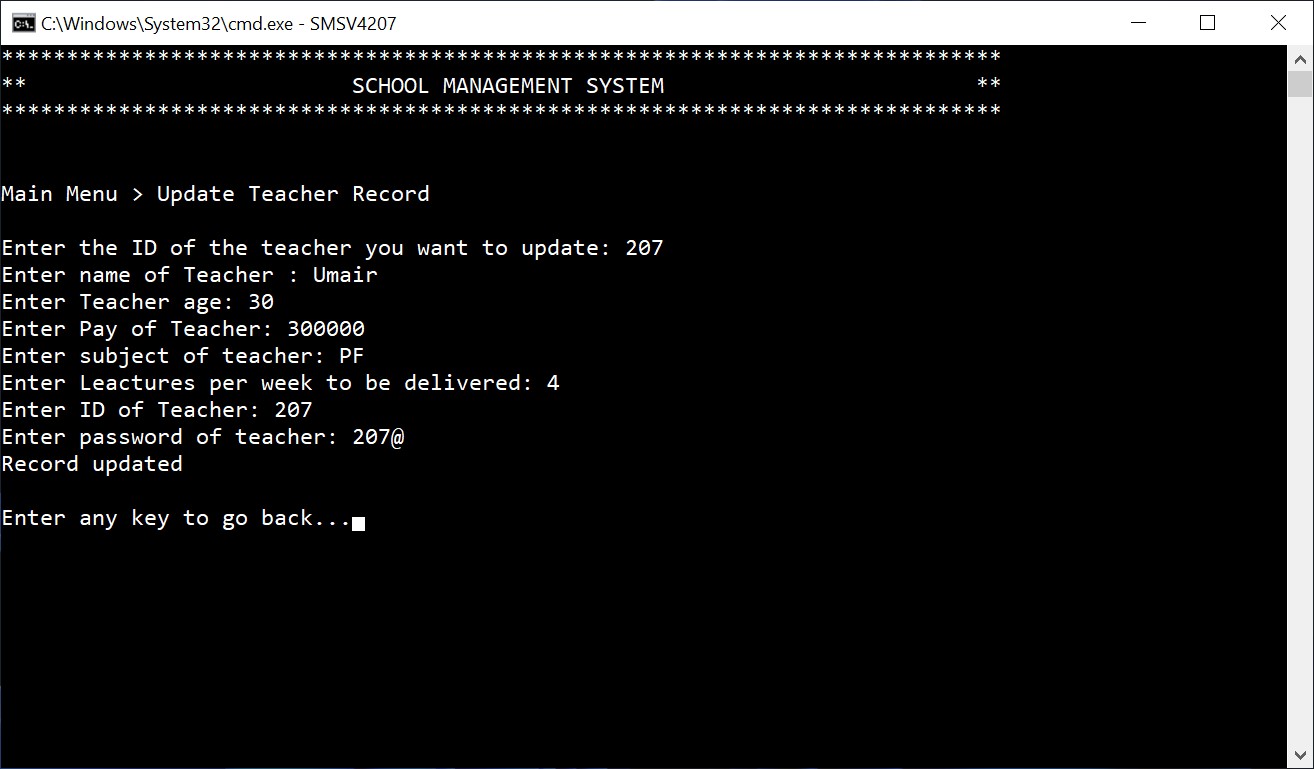
Option 1



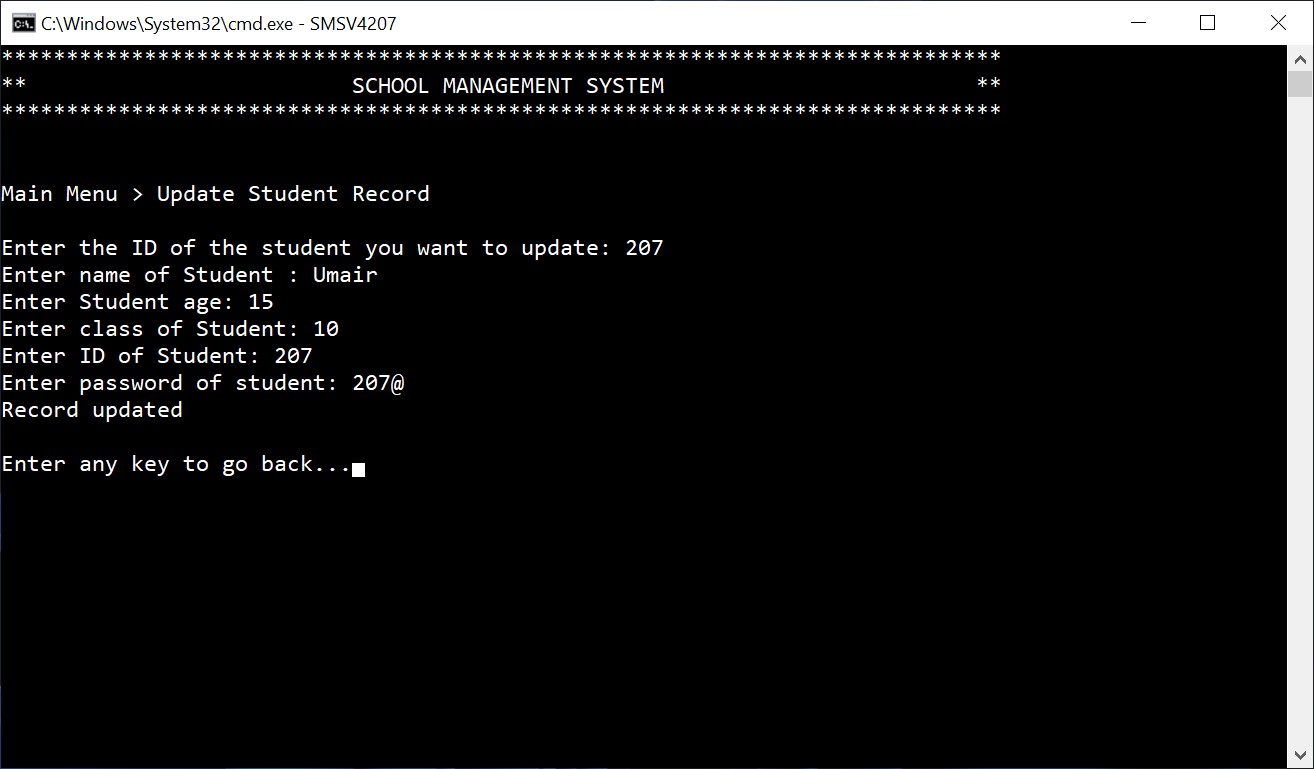
Option 2

****

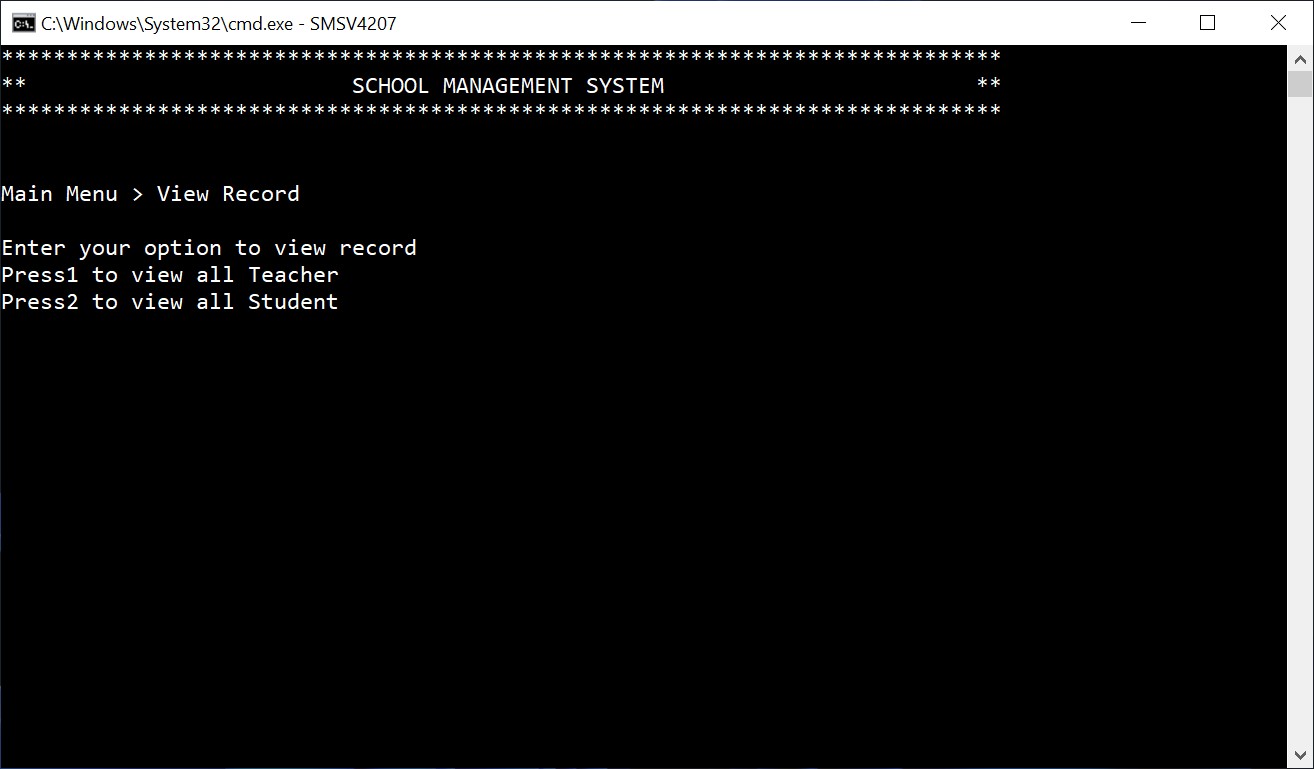
Option 3

****

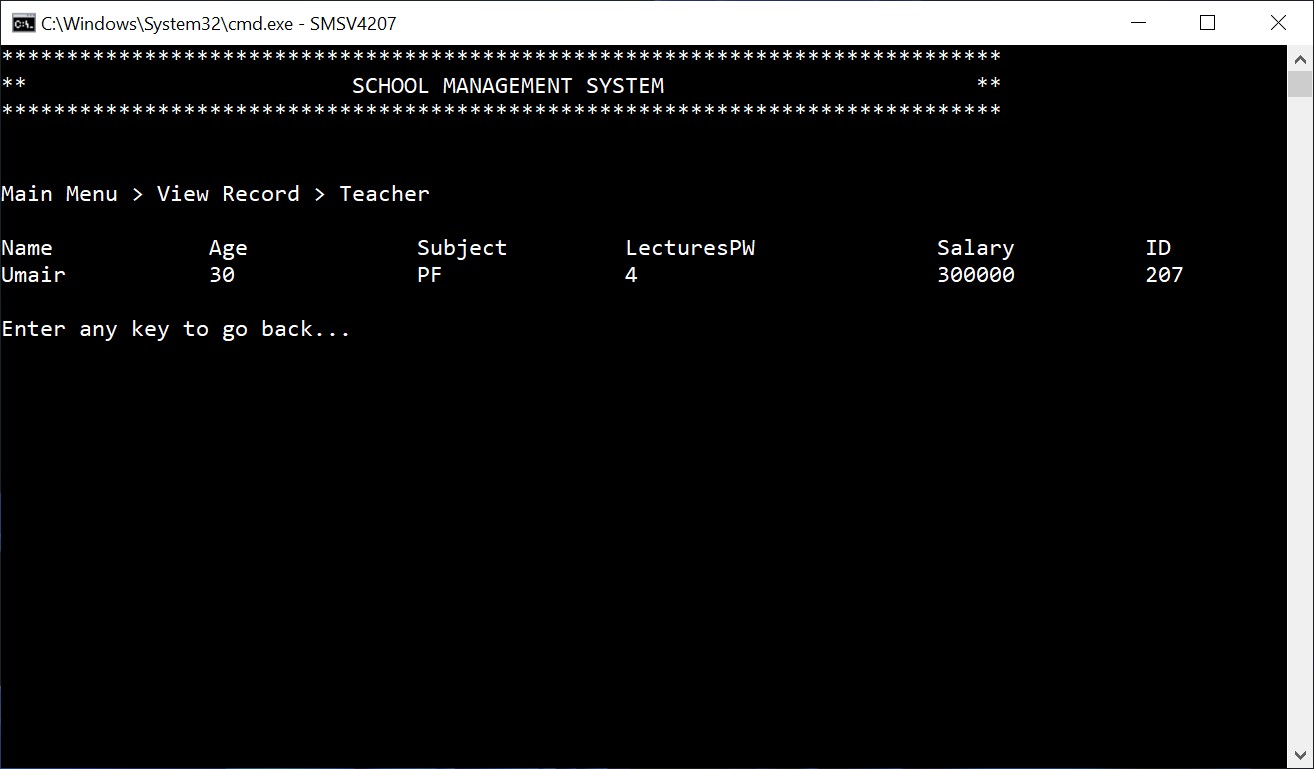
Option 4



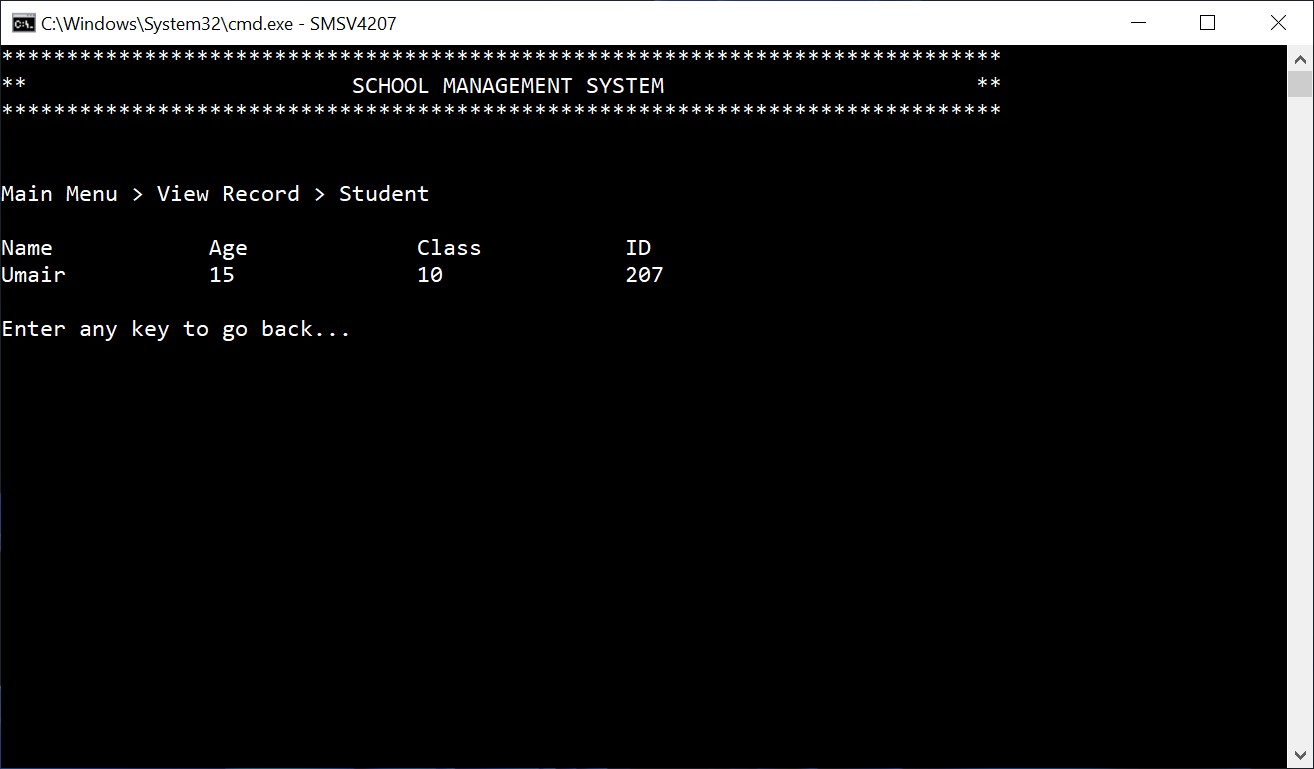
Option 5

****

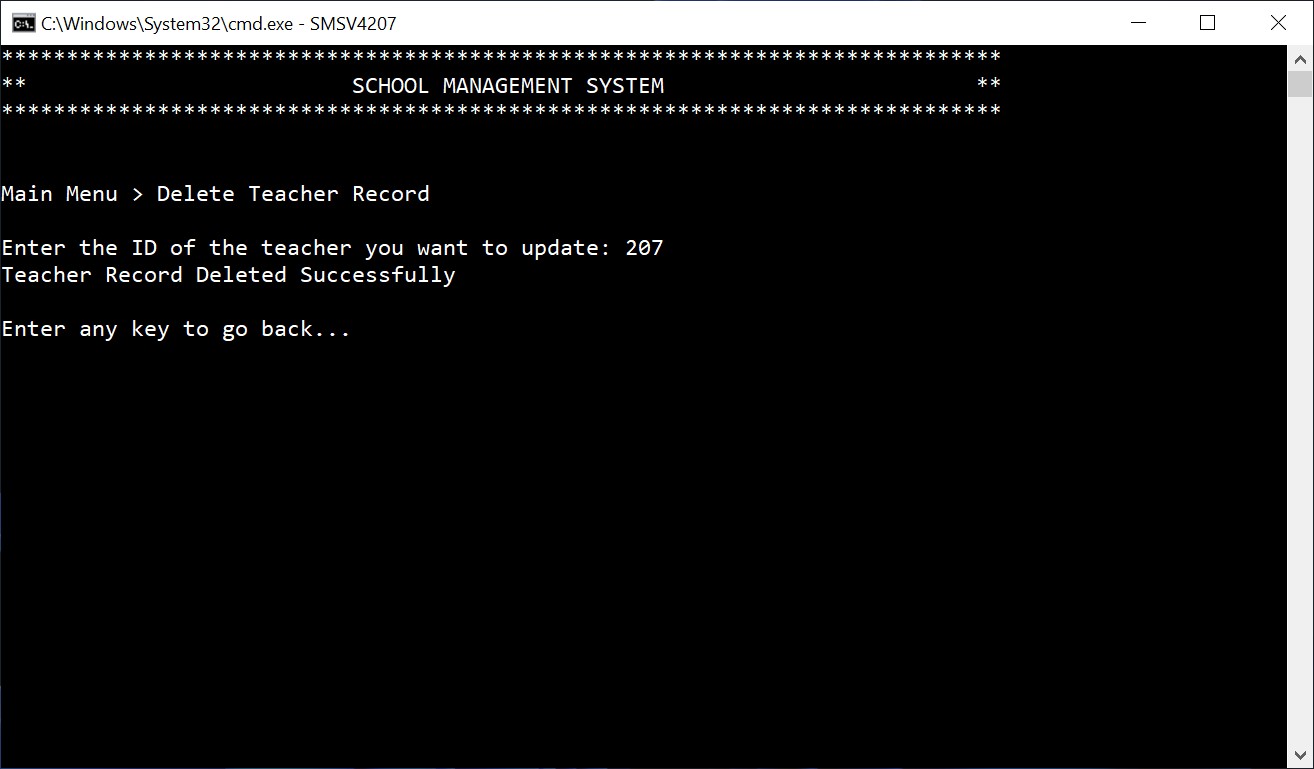
Option 5 (View Teacher’s Record)



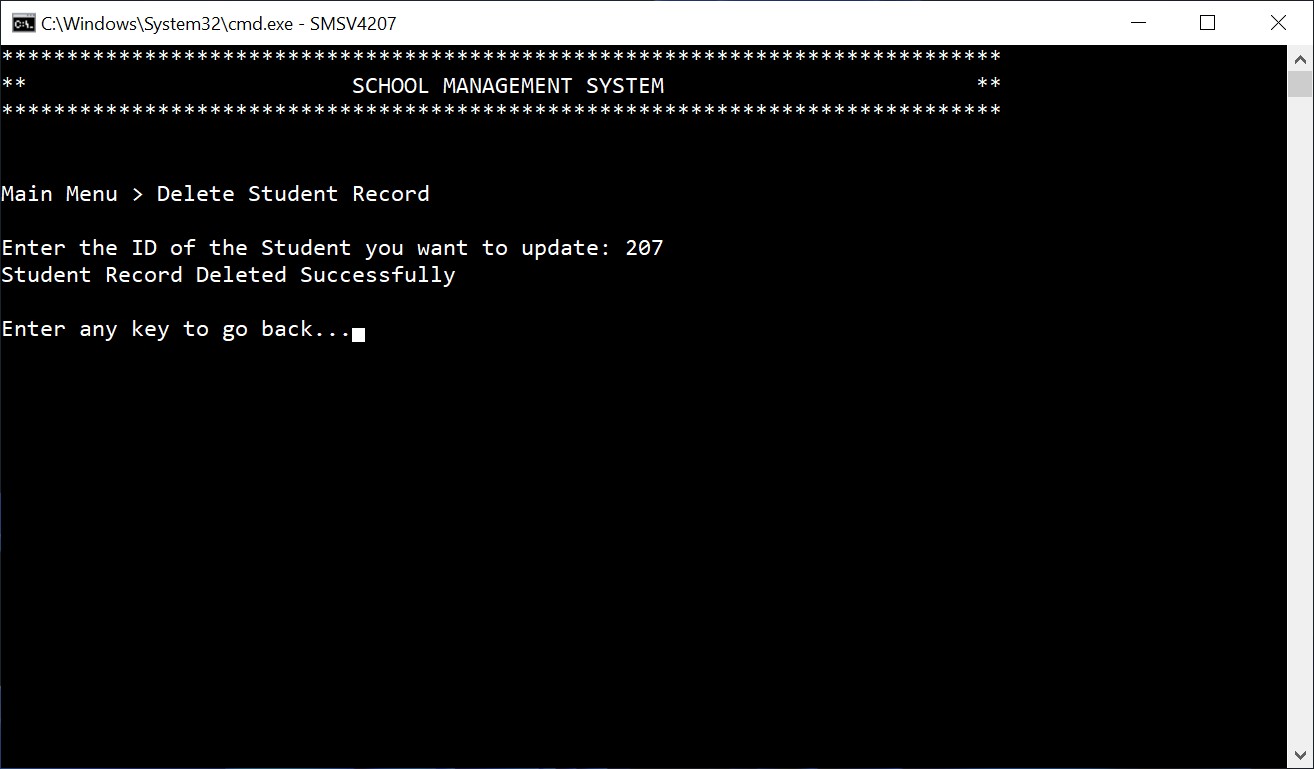
Option 5 (View Student’s Record)



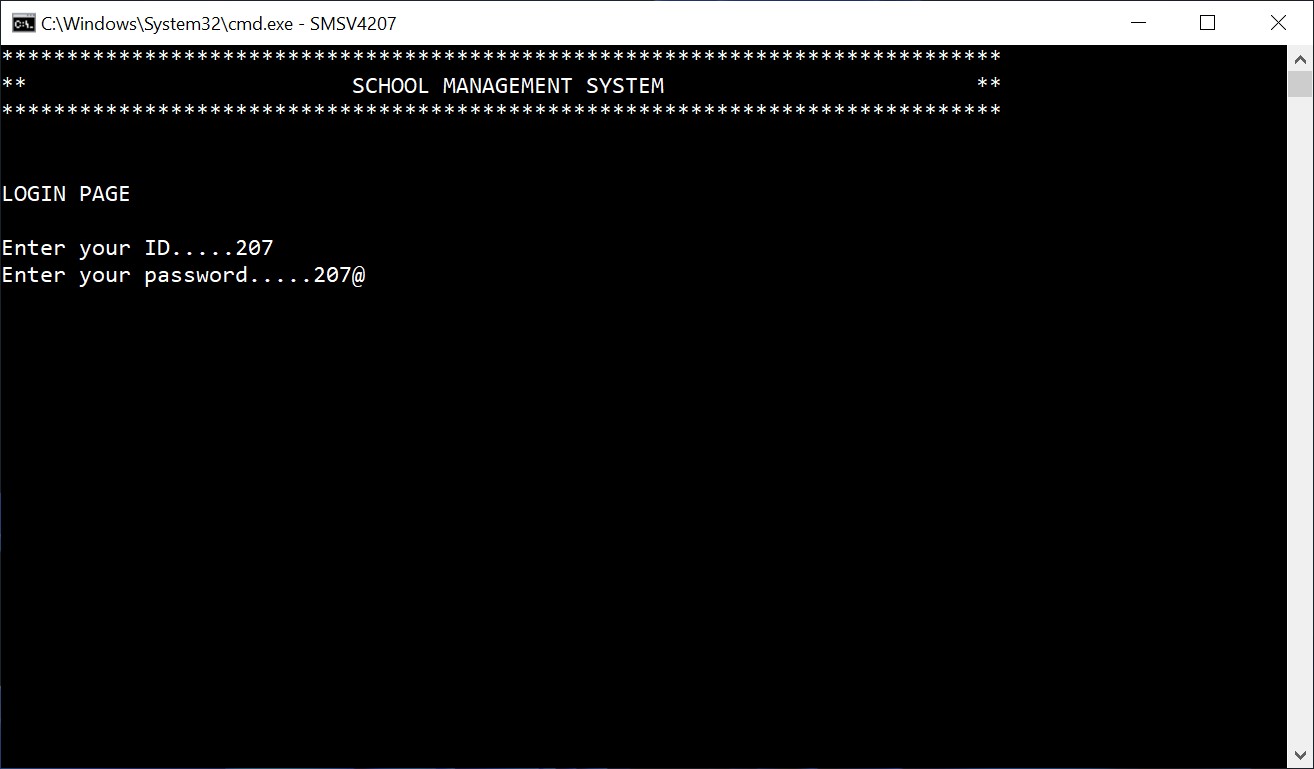
Option 6



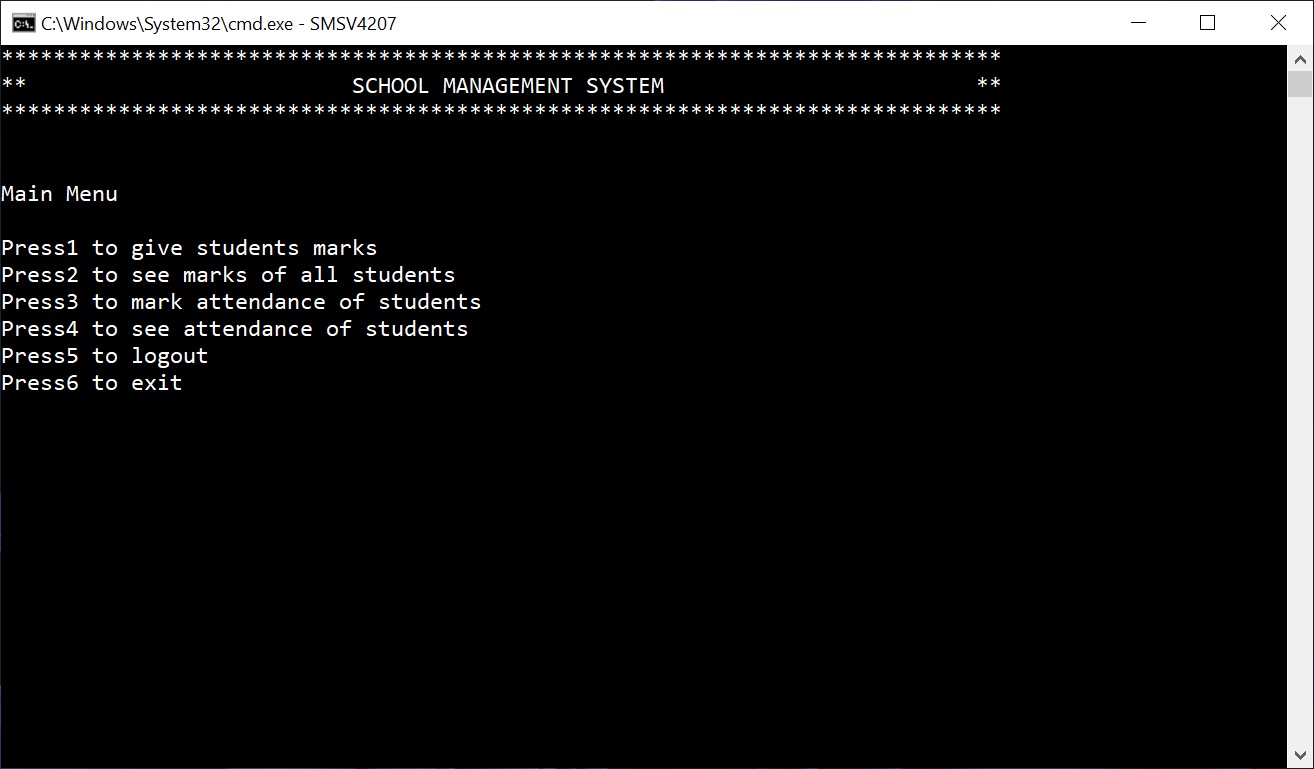
Option 7

****

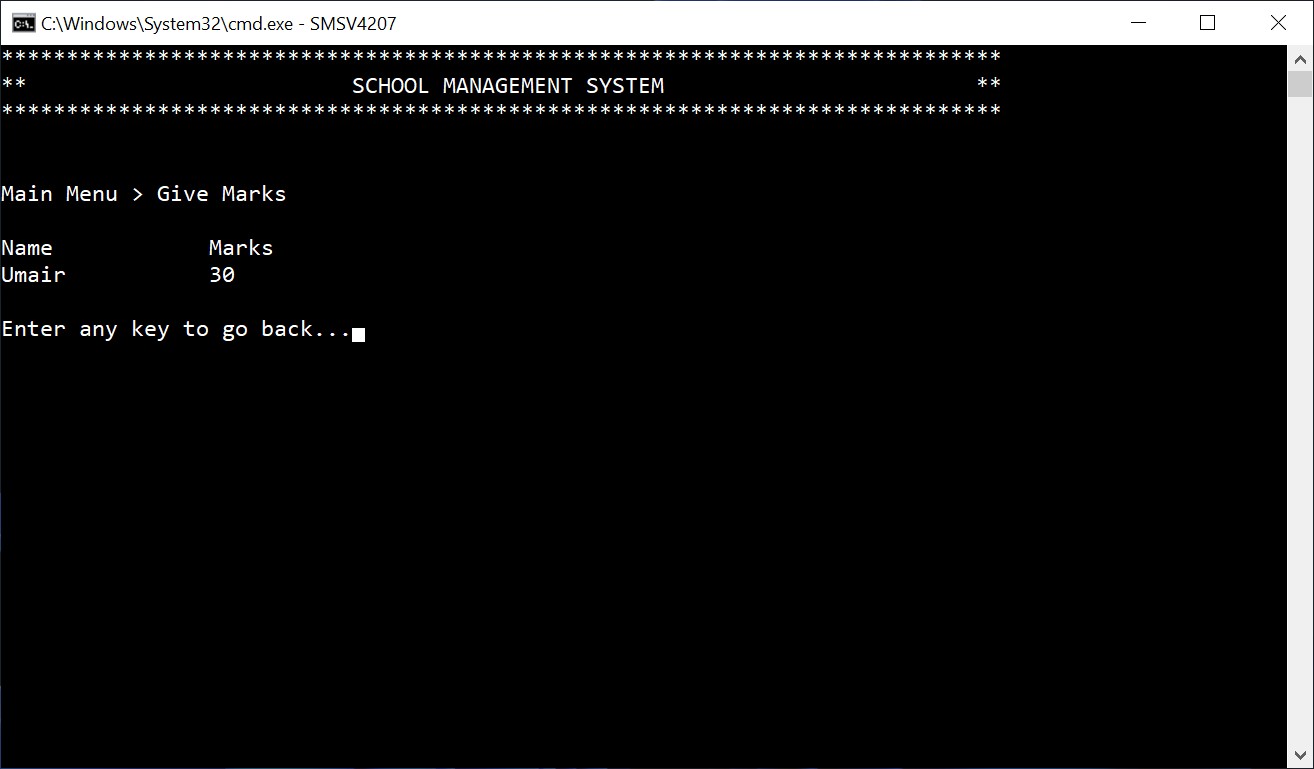
Teacher Login Page



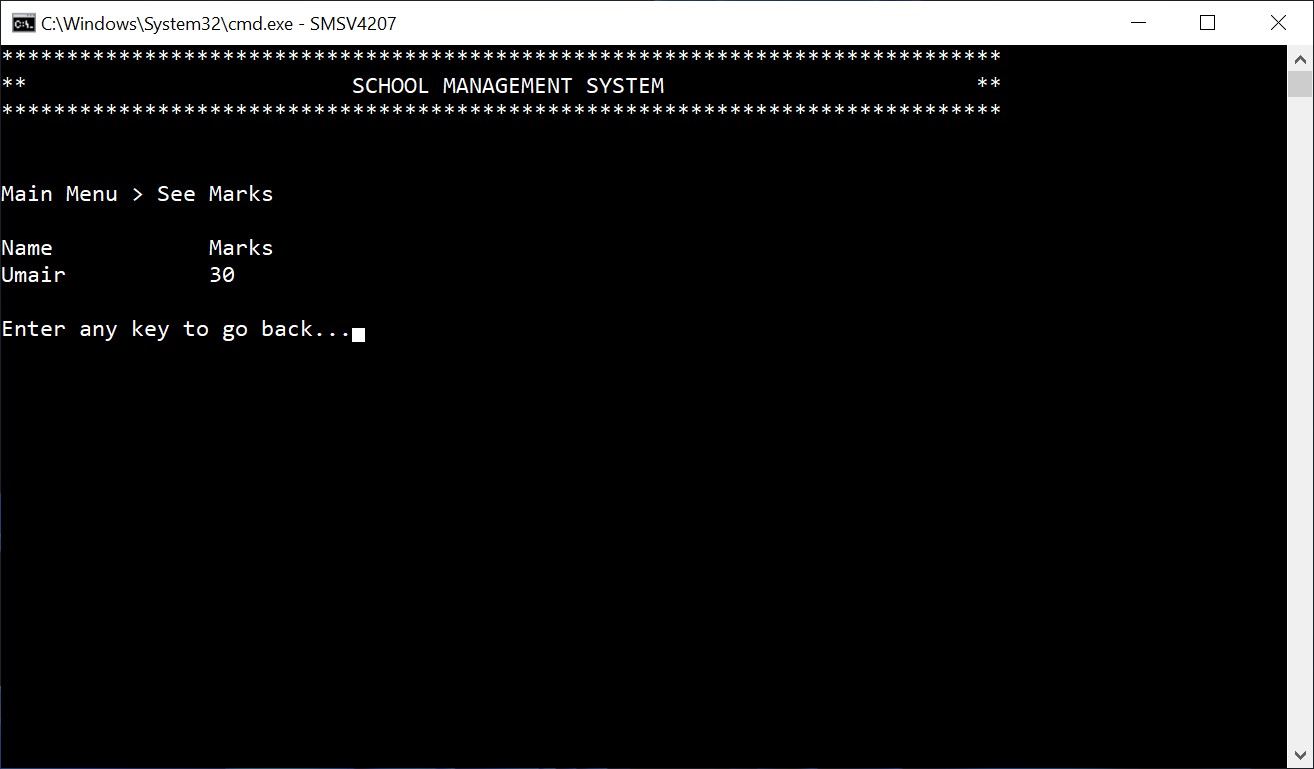
Teacher Main Menu

****

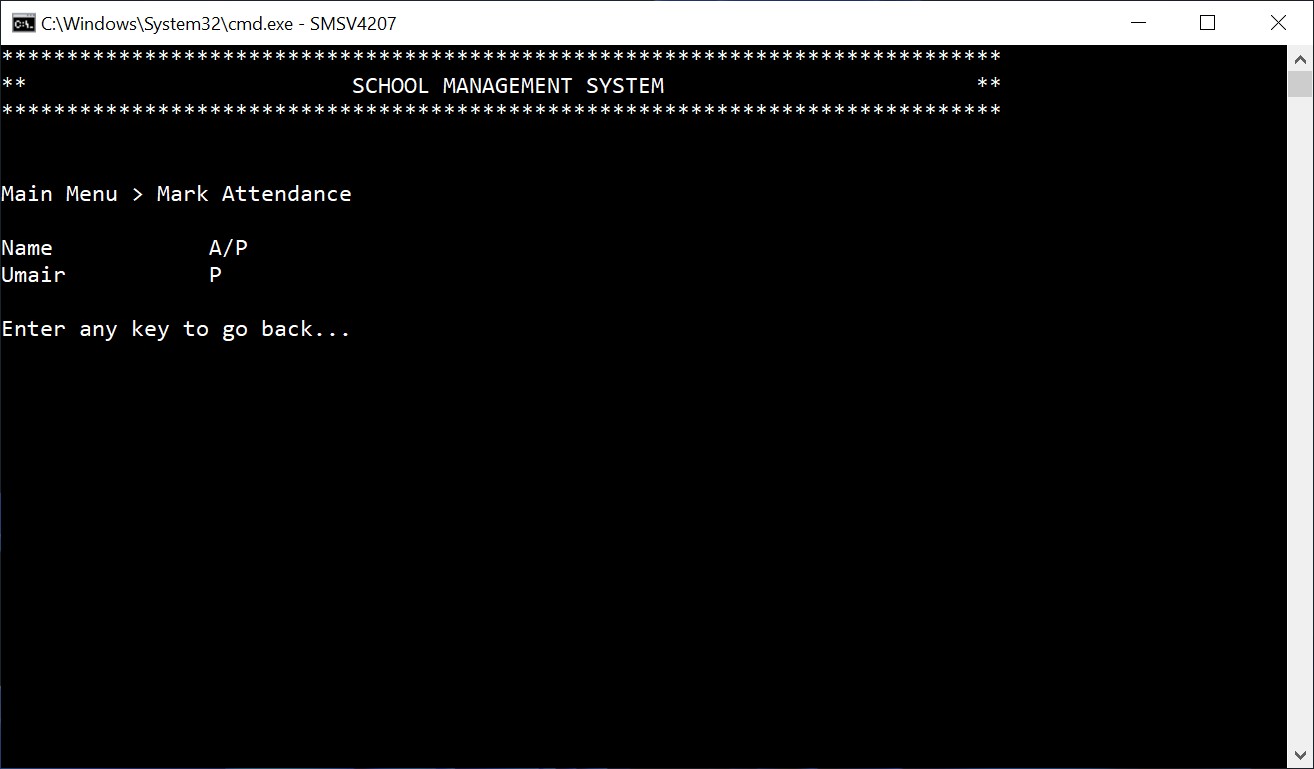
Option 1



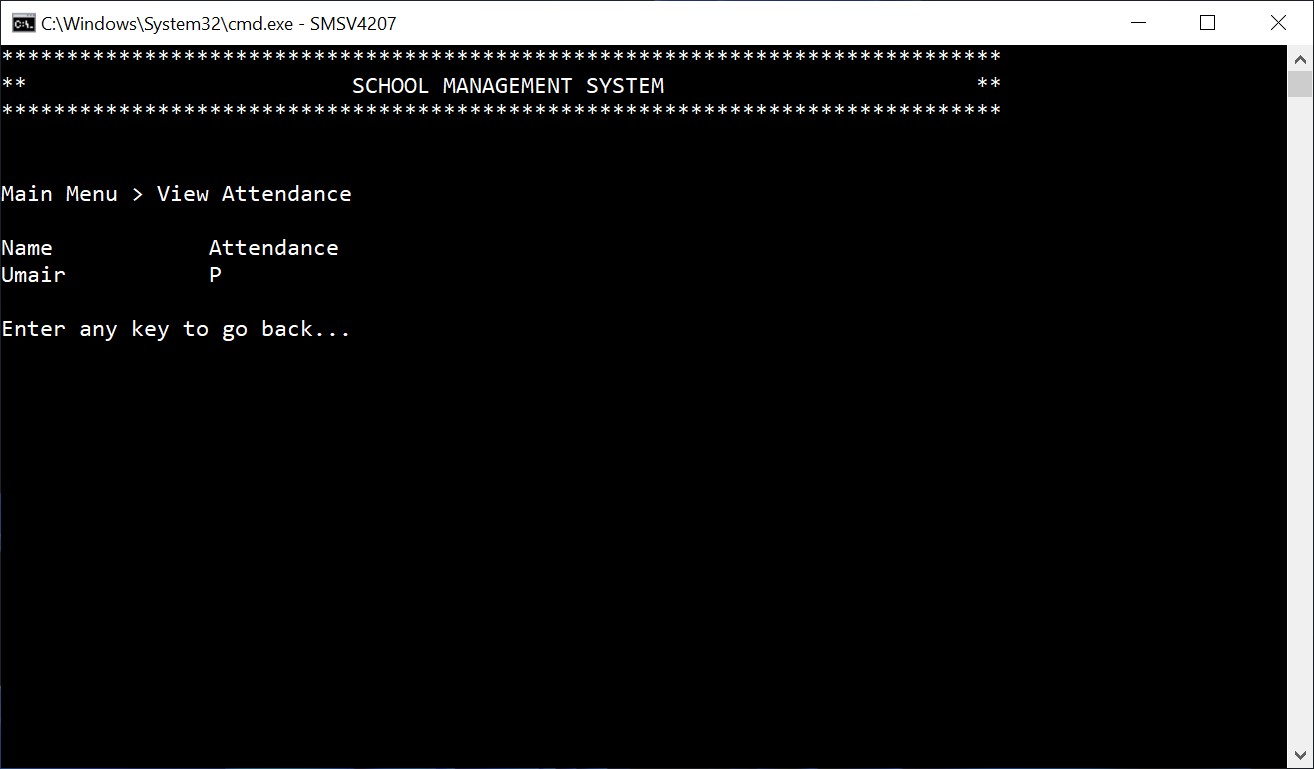
Option 2

****

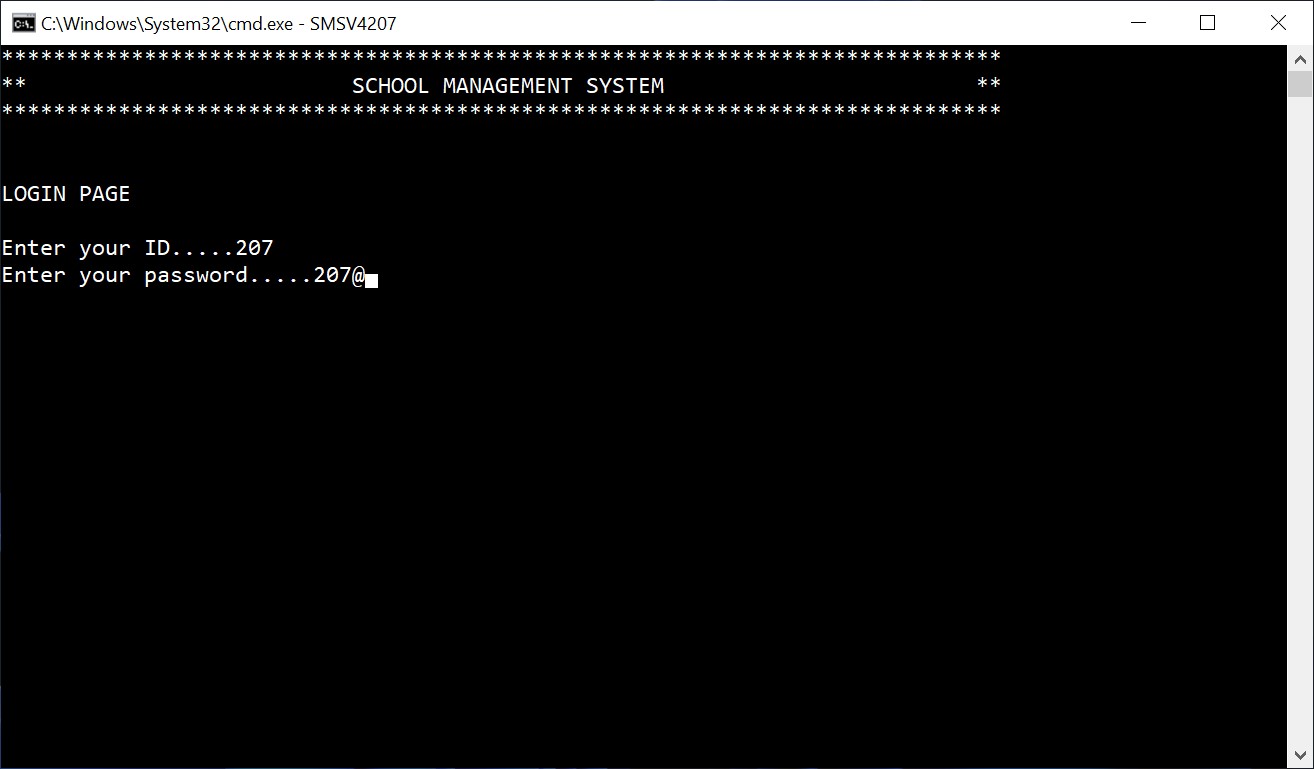
Option 3



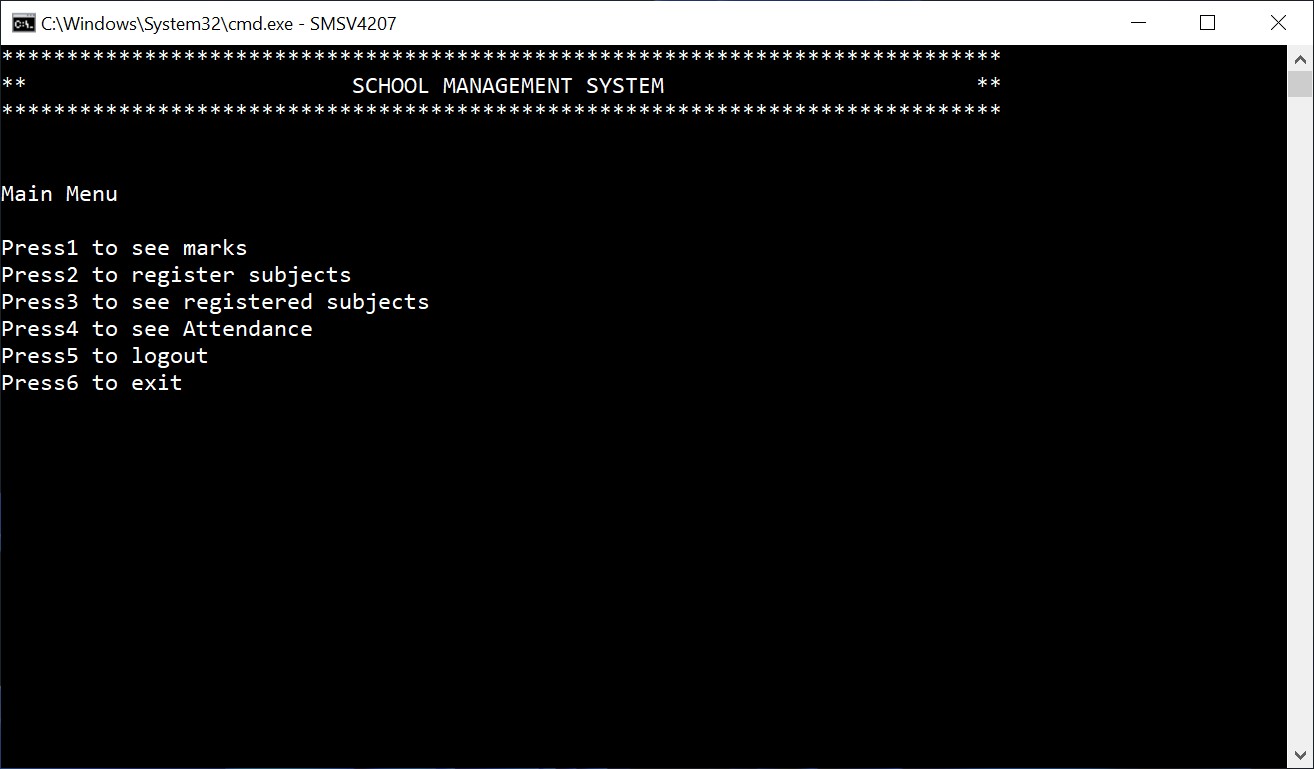
Option 4

****

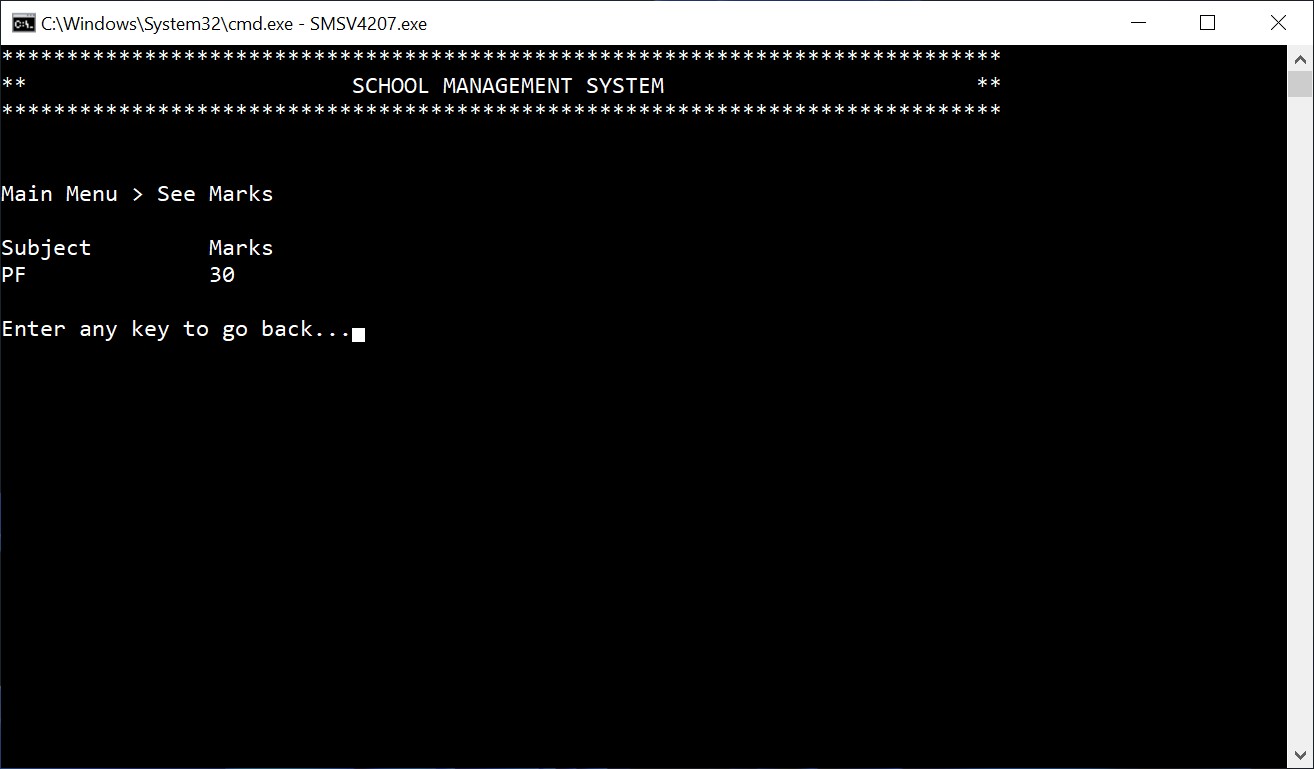
Student Login Page



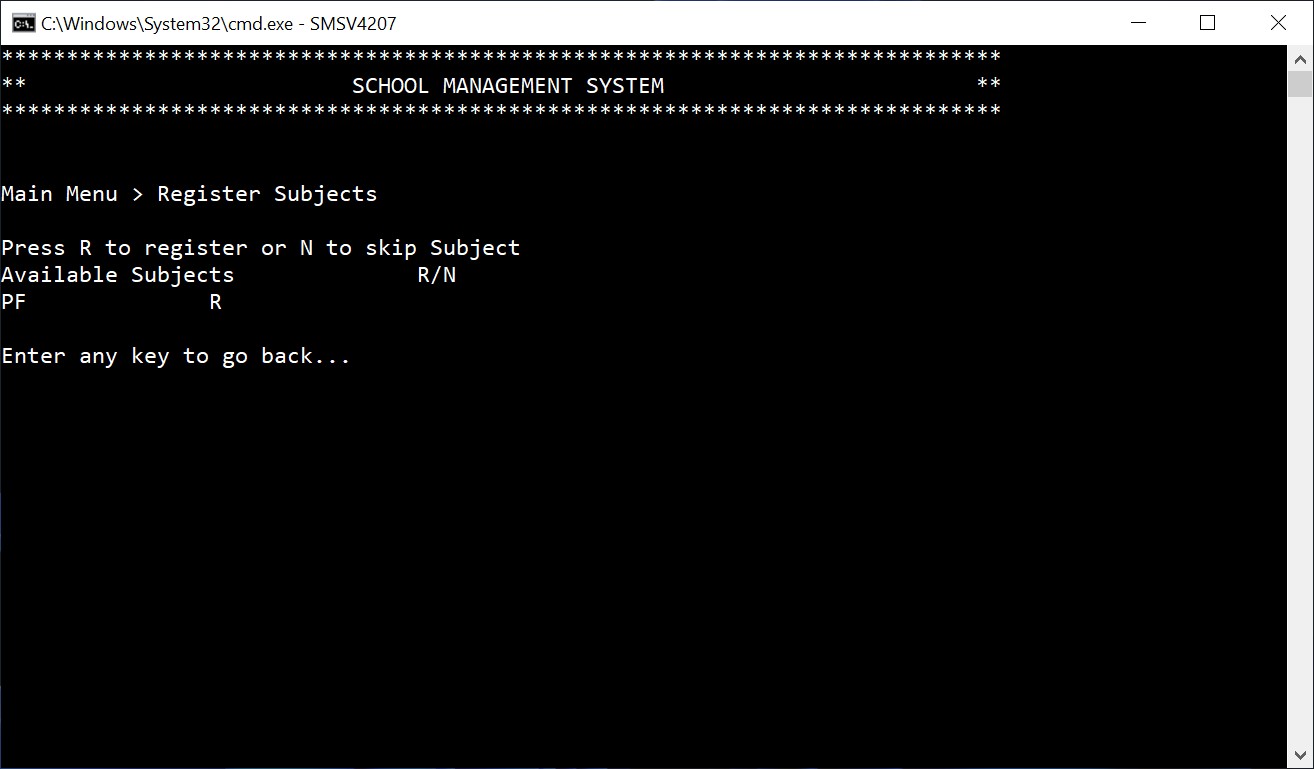
Student Main Menu

****

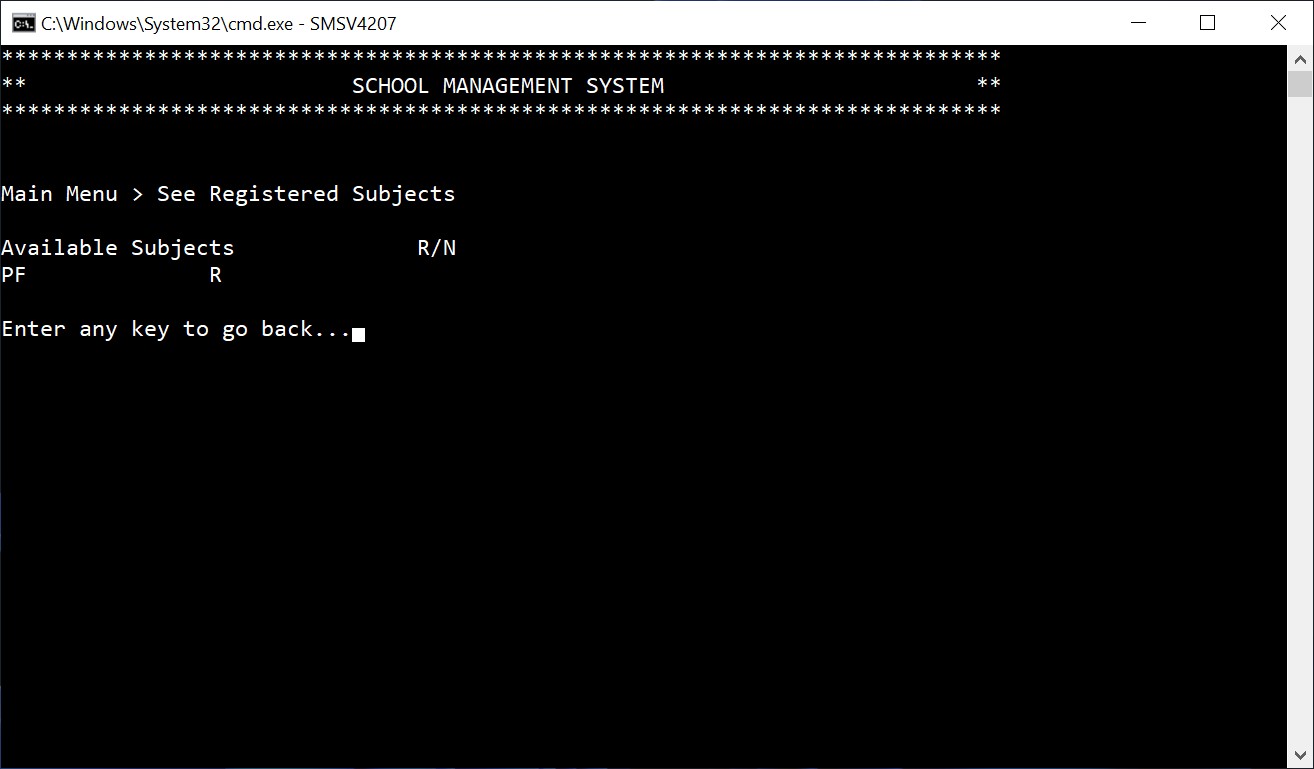
Option 1



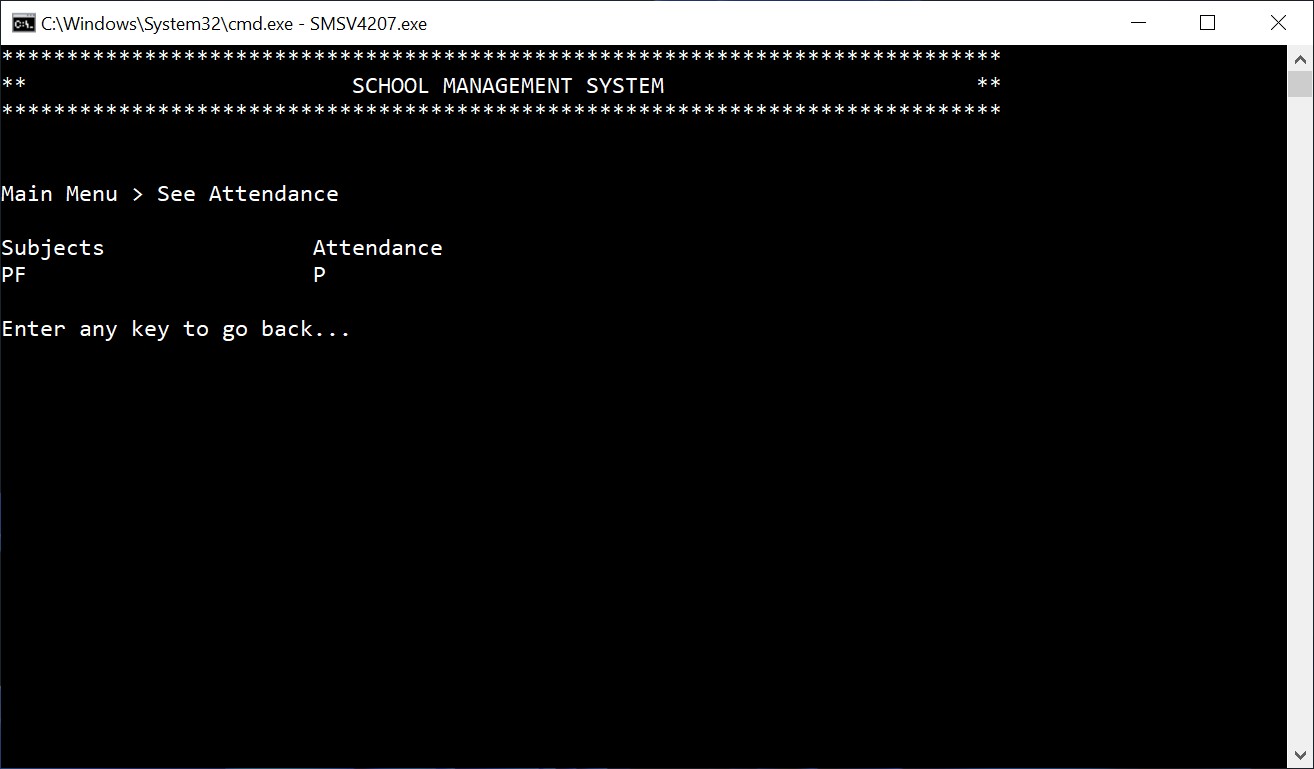
Option 2

****

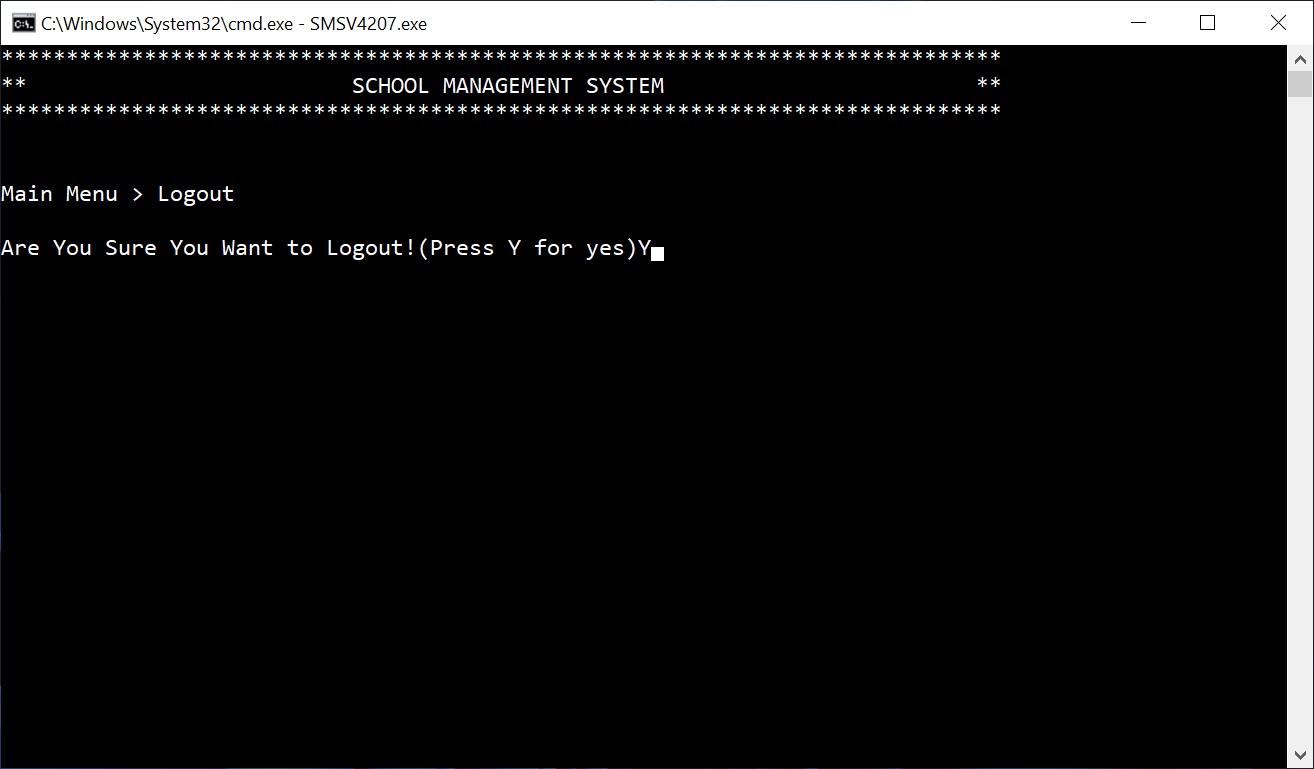
Option 3



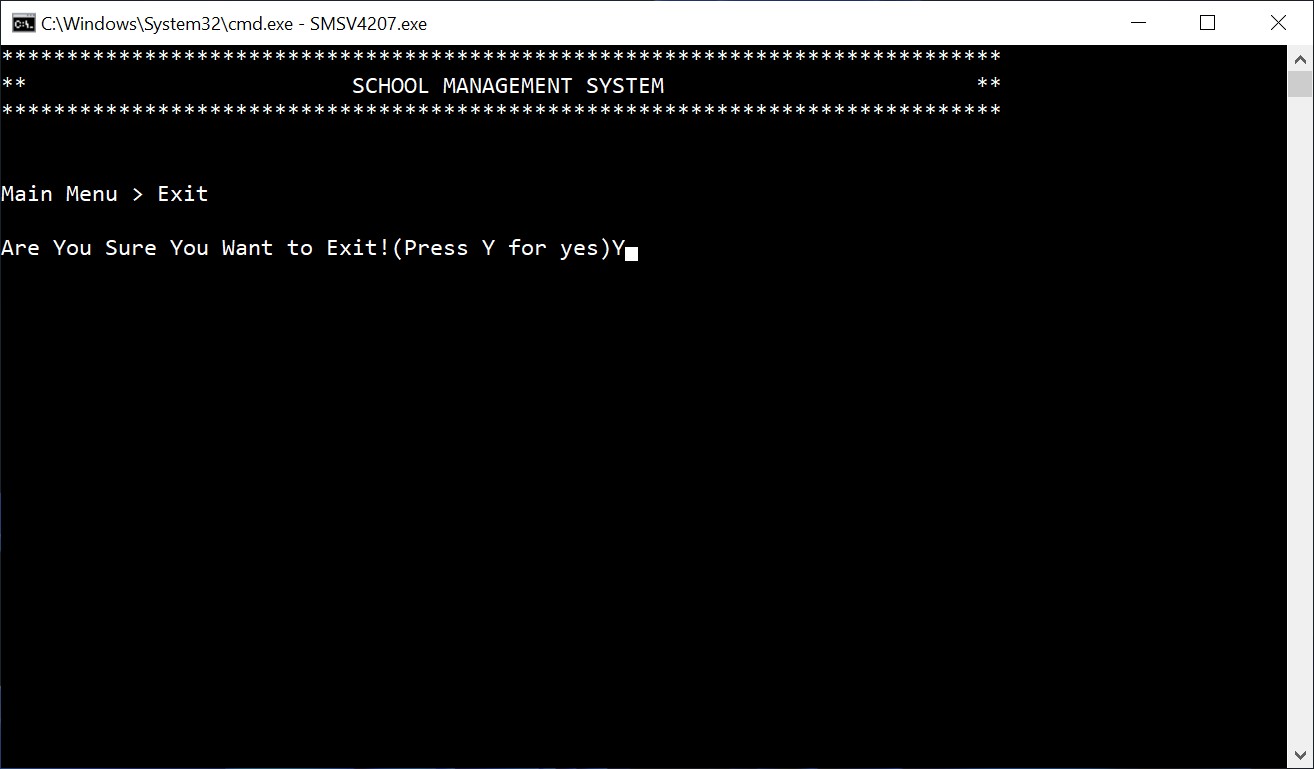
Option 4

****

Logout



Exit



**Student Reg. No. :**  2021-CS-207  **Student Name.**  Umair Noor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria are not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistent and according to given **guidelines**. The project **Poster** is professionally designed and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meets more than 80% of the criteria given. | Documentation meets more than 50% of the criteria. | When the documentation meets less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description - **Algorithms** and Flow Charts of all functions- **Test Cases** are defined -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What you **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | The project has at least 2 user types and each user has at least 5 functionalities. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Code Style  **Grade:** | All Code style criteria are followed | All code style criteria followed but some improvements required | A lot of improvements are required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation are synchronized. | Code and documentation do not synchronize at **some** places | Code and documentation do not synchronize in **many** places | Code and documentation **do not** synchronize. |
| Data Structure (Arrays)  **Grade:** | The data structure is sufficient for the project requirements | Data Structure is sufficient but requires improvement to meet project requirements. | The data structure is not sufficient and needs a lot of improvement | Data Structure is not properly identified and declared. |
| Sorting Features  **Grade:** | Sort working 100% and generating a useful report | The sorting feature is working but sorted data is not useful for the project. | The sorting feature is partial implemented | The project does not contain sorting |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places, it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)- Demo Data Functionality Added-At least Two Unit Tests are defined. | | | | |
| Validations  **Grade:** | Validations on all number type inputs are applied | Validations are applied but at some places, it is missing. | Validations are missing a lot of places | No Validations are used |
| Recommendation Feature | The proper meaning full recommendation is present in the system | Partial Recommendation is implemented | Implemented but not meaning full. | Not implemented |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | The presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | The student has a complete understanding of how the code is working and knows the concept. | The student has good understanding but in some places, he does not know the concepts | The student has very little understanding and lacks the major concepts. | The student does not have any level of understanding of the code. |

|  |  |
| --- | --- |
| **Checked by:** |  |

**Student Reg. No. :**  2021-CS-207  **Student Name.**  Umair Noor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria are not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistent and according to given **guidelines**. The project **Poster** is professionally designed and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meets more than 80% of the criteria given. | Documentation meets more than 50% of the criteria. | When the documentation meets less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description - **Algorithms** and Flow Charts of all functions- **Test Cases** are defined -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What you **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | The project has at least 2 user types and each user has at least 5 functionalities. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Code Style  **Grade:** | All Code style criteria are followed | All code style criteria followed but some improvements required | A lot of improvements are required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation are synchronized. | Code and documentation do not synchronize at **some** places | Code and documentation do not synchronize in **many** places | Code and documentation **do not** synchronize. |
| Data Structure (Arrays)  **Grade:** | The data structure is sufficient for the project requirements | Data Structure is sufficient but requires improvement to meet project requirements. | The data structure is not sufficient and needs a lot of improvement | Data Structure is not properly identified and declared. |
| Sorting Features  **Grade:** | Sort working 100% and generating a useful report | The sorting feature is working but sorted data is not useful for the project. | The sorting feature is partial implemented | The project does not contain sorting |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places, it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)- Demo Data Functionality Added-At least Two Unit Tests are defined. | | | | |
| Validations  **Grade:** | Validations on all number type inputs are applied | Validations are applied but at some places, it is missing. | Validations are missing a lot of places | No Validations are used |
| Recommendation Feature | The proper meaning full recommendation is present in the system | Partial Recommendation is implemented | Implemented but not meaning full. | Not implemented |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | The presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | The student has a complete understanding of how the code is working and knows the concept. | The student has good understanding but in some places, he does not know the concepts | The student has very little understanding and lacks the major concepts. | The student does not have any level of understanding of the code. |

|  |  |
| --- | --- |
| **Checked by:** |  |