A

Seminar Report On "STUDENT ATTENDANCE MANAGEMENT SYSTEM"

Submitted In partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

In

Computer Science & Engineering

(Session 2022-2023)

Submitted to:

Mrs. Madhu Choudhary

Submitted by:

Abhinn Agrawal – 22EJCCS006

Department of Computer Science & Engineering

Jaipur Engineering College & Research Centre, Jaipur

Rajasthan Technical University, Kota

CANDIDATE’S DECLARATION

I hereby declare that the report entitled “ATTENDENCE MANAGEMENT SYSTEM” has been carried out and submitted by the undersigned to the Jaipur Engineering College &amp; Research Centre, Jaipur (Raj.) is an original work, conducted under the guidance and supervision of Ms. Madhu Choudhary.

The empirical findings in this report are based on the data, which has been collected by me. I have not reproduced from any report of the University neither of this year nor of any previous year.

I understand that any such reproducing from an original work by another is liable to be punished in a way the College authorities deed fit.

Date: 10/11/2023 Abhinn Agrawal-22EJCCS006

Place: Jaipur

VISION OF CSE DEPARTMENT

To become renowned Centre of excellence in computer science and engineering and make competent engineers & professionals with high ethical values prepared for lifelong learning.

MISSION OF CSE DEPARTMENT

1. To impart outcome-based education for emerging technologies in the field of computer science and engineering.
2. To provide opportunities for interaction between academia and industry.
3. To provide platform for lifelong learning by accepting the change in technologies
4. To develop aptitude of fulfilling social responsibilities.

PROGRAM EDUCATIONAL OUTCOMES

1. To provide students with the fundamentals of Engineering Sciences with more emphasis in Computer Science & Engineering by way of analyzing and exploiting engineering challenges.
2. To train students with good scientific and engineering knowledge so as to comprehend, analyze, design, and create novel products and solutions for the real-life problems.
3. To inculcate professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach, entrepreneurial thinking and an ability to relate engineering issues with social issues.
4. To provide students with an academic environment aware of excellence, leadership, written ethical codes and guidelines, and the self-motivated life-long learning needed for a successful professional career.
5. To prepare students to excel in Industry and Higher education by Educating Students along with high moral values and knowledge.

PROGRAM OUTCOMES

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and Computer Science &amp; Engineering specialization to the solution of complex Computer Science & Engineering problems.
2. Problem analysis: Identify, formulate, research literature, and analyze complex computer Science & Engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex Computer Science & Engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of Computer Science & Engineering experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern Computer Science& Engineering and IT tools including prediction and modeling to complex computer science engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional Computer Science & Engineering practice.
7. Environment and sustainability: Understand the impact of the professional Computer Science & Engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the Computer Science &amp; Engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings in Computer Science & Engineering.
10. Communication: Communicate effectively on complex Computer Science & Engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the Computer Science & Engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of Computer Science & Engineering change.

COURSE OUTCOMES

Graduates would be able:

1. To understand Software Requirement Analysis, CASE Tools, Software Testing, and other configuration tools.
2. To understand Functional Modeling (DFD), Data Modeling (DFD) - Use work products data dictionary.
3. An ability to understand the Structural and Behavioral UML Diagrams with the use of Project Management Tool – Project Libre.

----------------------------------- CONTENTS: --

1. Introduction

1.1 Purpose of this document

The purpose of this SRS document is to provide a detailed overview of our software product, it's parameters and goals. This document describes the project's target audience and it's user interface, hardware and software requirements. It defines how our client, team and audience see the product and its functionality.

1.2 Scope of the Development Project

The goal is to design a robust software for the management of Student Attendance in the Jaipur Engineering College and Research Centre. In this project we will fully automate the entire process of keeping attendance record of the students. At the end of the lecture when the teacher will be marking attendance he/she will do so directly on the mobile and the students present in the class will be able to mark their presence. In case a student was unable to attend college due to medical reasons or personal reasons he/she can upload a leave application. Students can login to check their attendance record in all the subjects. The admin can register new students into the database and also check the presence of each registered student.

The software must be able to perform the following operations:

I. Mark Student Attendance: It must be able to mark the attendance of the students present in the class.
II. Show Student Attendance: It must be able to show the student of the college his/her attendance in all the subjects.
III. Accept Leave Application: It must allow students to upload their leave applications in pdf form.
IV. Register Students: It must allow the faculty or admin to register new students into the database of attendance record.

1.3 Intended Audience and Document Overview

The project is being designed for the students of Jaipur Engineering College & Research Centre. The students face a lot of problems when it comes to checking their attendance and submitting their leave applications. Thus, we want to automate the entire student attendance management system so that they can have a comfort of checking their attendance anytime in just one click. This document also serves as a contract between the owner of the software and the developers where the owner can clearly see what and how the developers intend to do to make the software.

1.4 Definitions, Acronyms and Abbreviation

I. IEEE – Institute of Electrical and Electronics Engineers
II. RSA – Rational Software Architect
III. UML – Unified Modelling Language
IV. DFD – Data Flow Diagram

2. Overall Description

2.1 Project Overview

Attendance Management System basically has two main modules for proper functioning:

- Admin module has rights for creating any new entry of student details, check attendance of every student roll no. wise, delete student.
- User has a right of making daily attendance, checking attendance in every subject, uploading leave applications. Attendance report can be taken by given details of student details, date, class.

2.2 Project Functions

The product should be able to perform the following operations:

I. It must be able to register new student ids.
II. It must be able to authenticate the login id for students and admin.
III. It must be able to mark and count the attendance of students as entered by students.

2.3 Design and Implementation Constraints

The development of the system will be constrained by the availability of required software Such as web servers, database and development tools. The availability of these tools will be governed by the JECRC Foundation. The hardware constraints include a smartphone or laptop to access the website and make the request.

2.4 Assumptions and Dependencies

The following list prevents the assumptions, dependencies or guidelines that are imposed l upon implementation of the system.

1. The product must have a user-friendly interface that is simple enough for all types.

2. User to understand.
3. Response time should not be longer than 5 seconds
4. A general knowledge of basic computer skills and internet is required to use the product.

3. Specific Requirements

3. 1 External Interface Requirements

3.1.1 User Interfaces

The goal is to design the software used for proper management of attendance and automate the process. The user types are listed followed

I. Students
II. Admin

Our goal is to develop a software that should be easy to use for all types of users. Thus while designing the software one can assume that each user type has the following characteristics:

I. The user is a computer-literate and has little or no difficulty in using the software keeping in mind the software is user friendly.
II. In order to use software a user must be aware of the internal working and expected to know how things work.
III. All the guidelines about the use of software will be informed to the user once the user signs up on the software or web page.

3.1.2 Hardware Interfaces

1. Computer: A computer will be required to open the website and use the software
2. Smartphone: A smartphone can also be required in case there is no availability of computer.
3. Internet: A good internet connection is required to access the website.

3.1.3 Software Interfaces

1. A SQL Database Server will be required to store and retrieve data.
2. A web browser will be required to open the website.

3.2 Functional Requirements

3.2.1 User Authentication:

- **Student Login:** Allow students to log in using a username and password**.**
- **Admin Login:** Provide administrators with a secure login using a predefined username and password.

3.2.2 Student Functionality:

- **Mark Attendance:** Allow students to mark their attendance for the current date.
- **Check Attendance:** Provide students with the ability to check their total attendance count.
- **Submit Leave Application:** Enable students to submit leave applications with a corresponding reason.

3.2.3 Admin Functionality:

- **Register Student:** Allow administrators to register new students by providing necessary details.
- **Delete Student Data:** Provide the capability to delete individual student data.
- **Delete All Students:** Allow administrators to delete all registered students.
- **Check List of Students Registered**: Enable administrators to view a list of all registered students.

3.2.4 Schedule Management:

- **Add Schedule:** Allow administrators to add schedules, including day, time, and subject.
- **Display Schedule:** Provide a way to view the schedule.

3.2.5 Leave Management:

- **Submit Leave Application:** Allow students to submit leave applications.
- **Display Leave Applications**: Enable administrators to view a list of leave applications.
- **Approve Leave Application:** Allow administrators to approve leave applications submitted by students.

3.2.6 Data Persistence:

- **File Handling:** Implement file operations to store and retrieve student data, attendance records, and other relevant information.

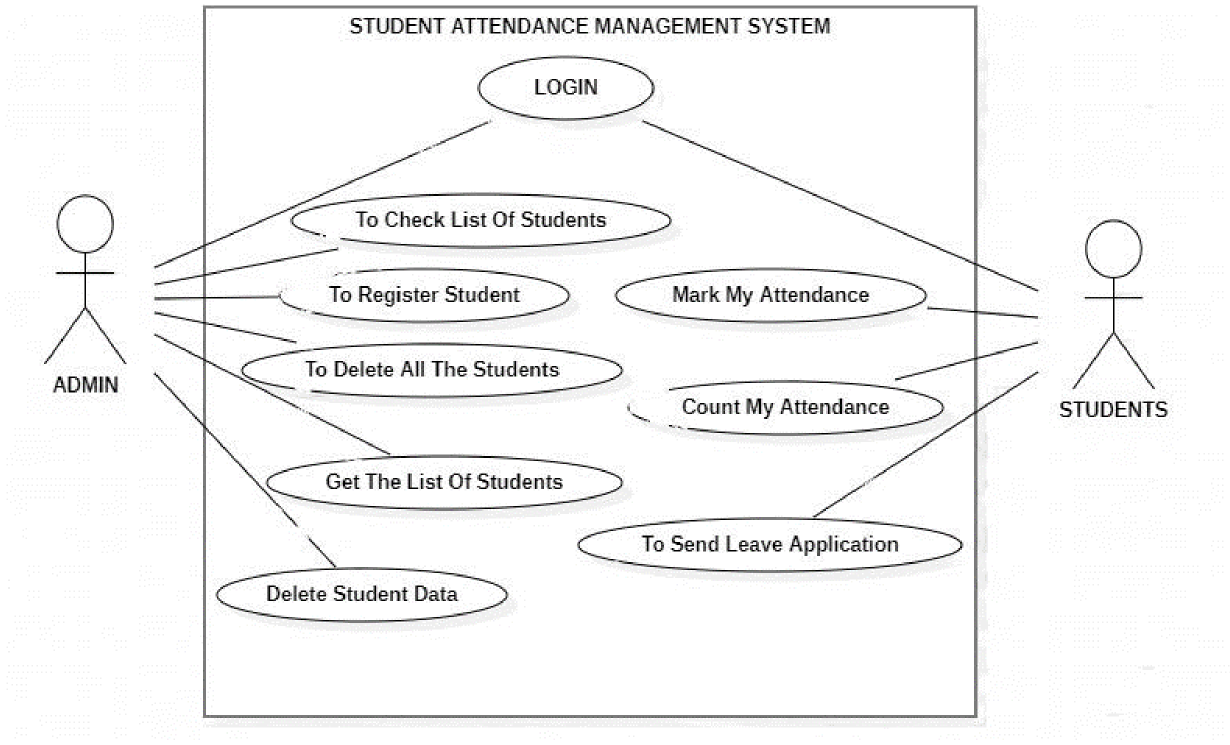
3.2.7 Menu Navigation:

- **User Interface:** Provide an interactive menu system for both students and administrators to navigate through different functionalities.

3.2.8 Exit System:

- **Exit Option:** Allow users to exit the system securely.

3.3 UML Use Case Diagram



4. Other Non-functional requirements:

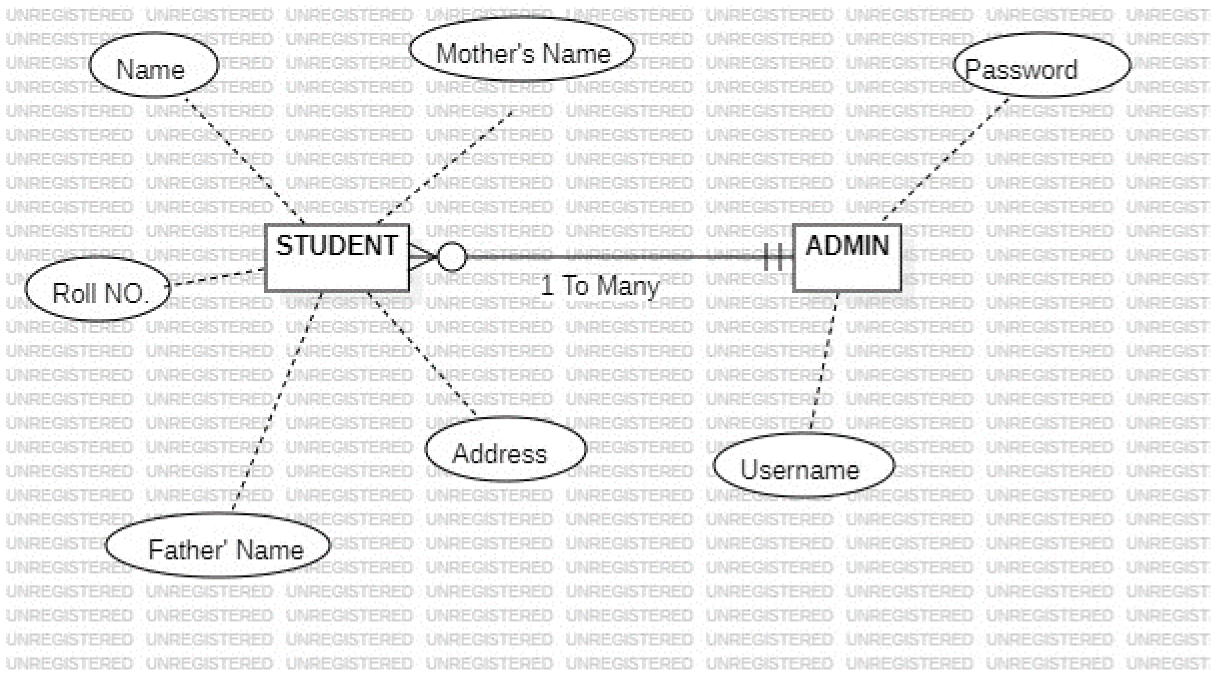
Non-functional requirements are the constraints that must be adhered during development.

The various non-functional requirements are:

- provide rating after request completion.
- Select available time slot for response to issue.
- The system should be able to handle a growing number of users without a significant decrease in performance.
- The system should be able to run on different operating systems without modification.
- The system should comply with relevant laws and regulations.

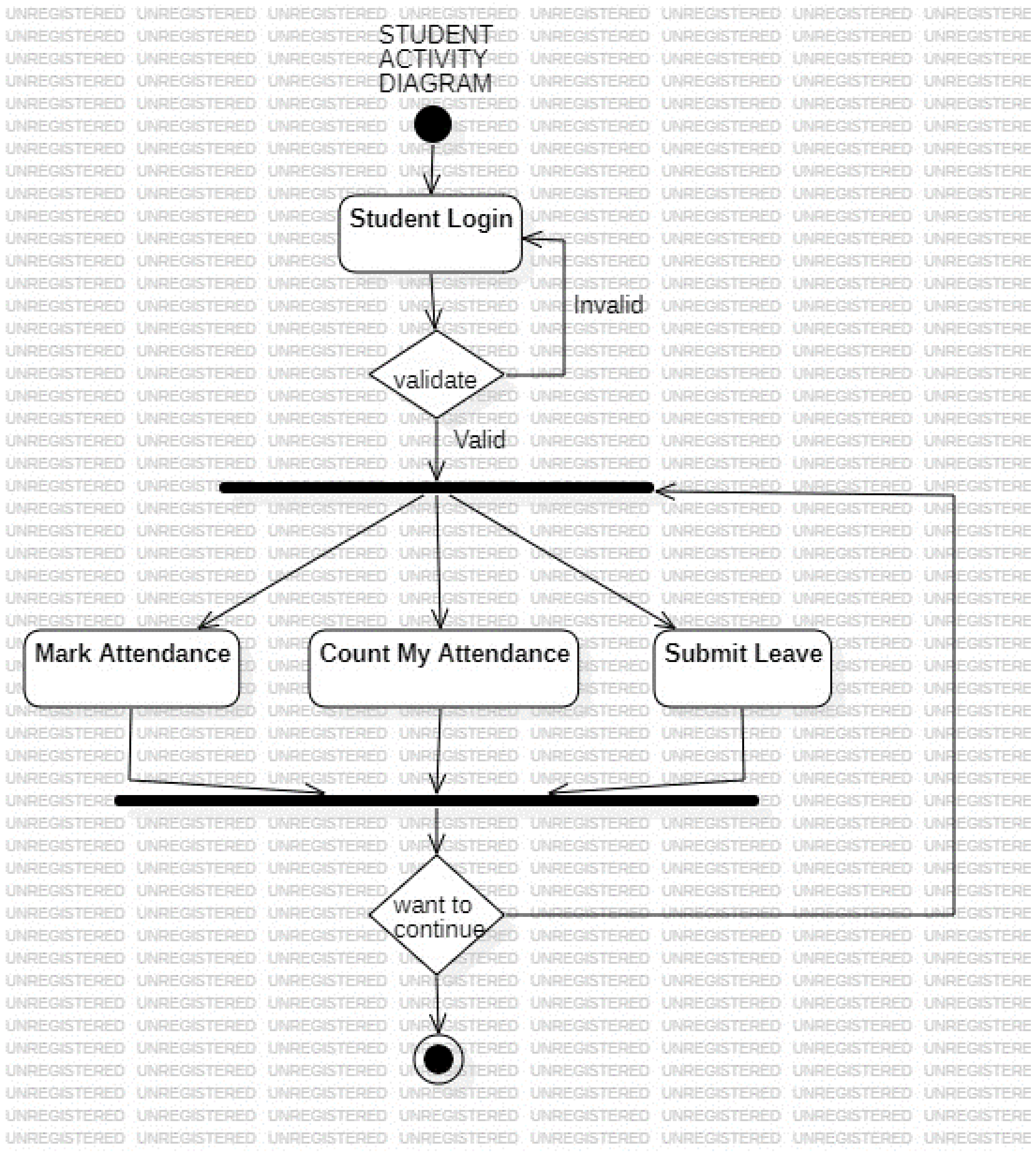
5. Diagrams:

5.1 ER (Entity relation) diagram:

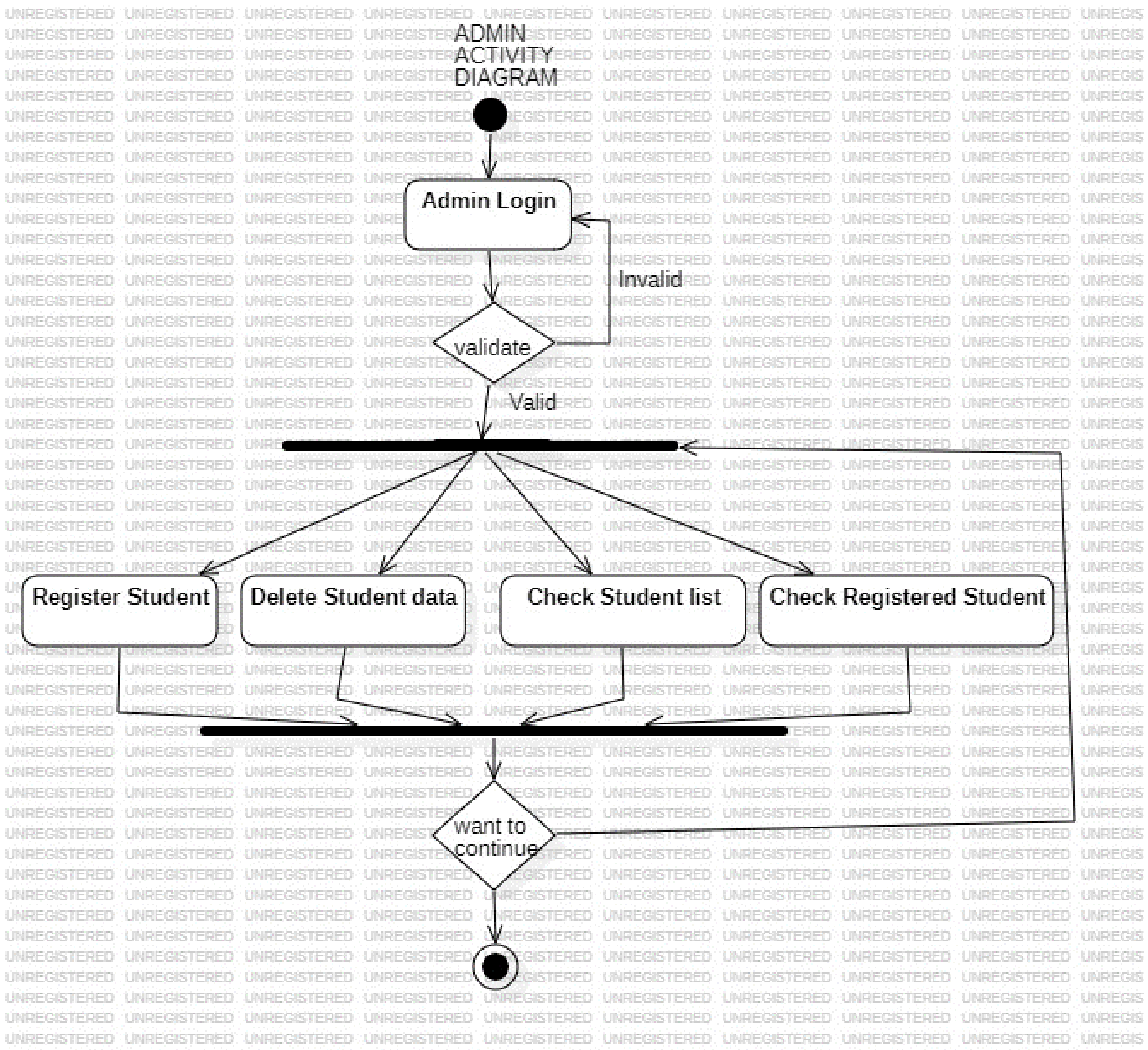


5.2 Activity Diagram:

5.2.1 Student Activity diagram:

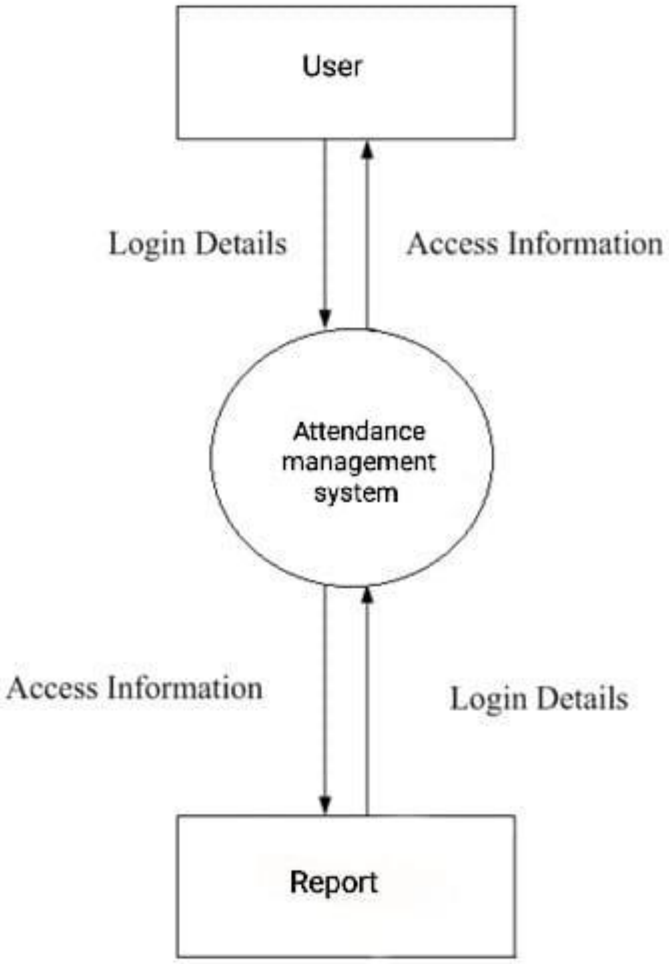


5.2.2 Admin Activity diagram:

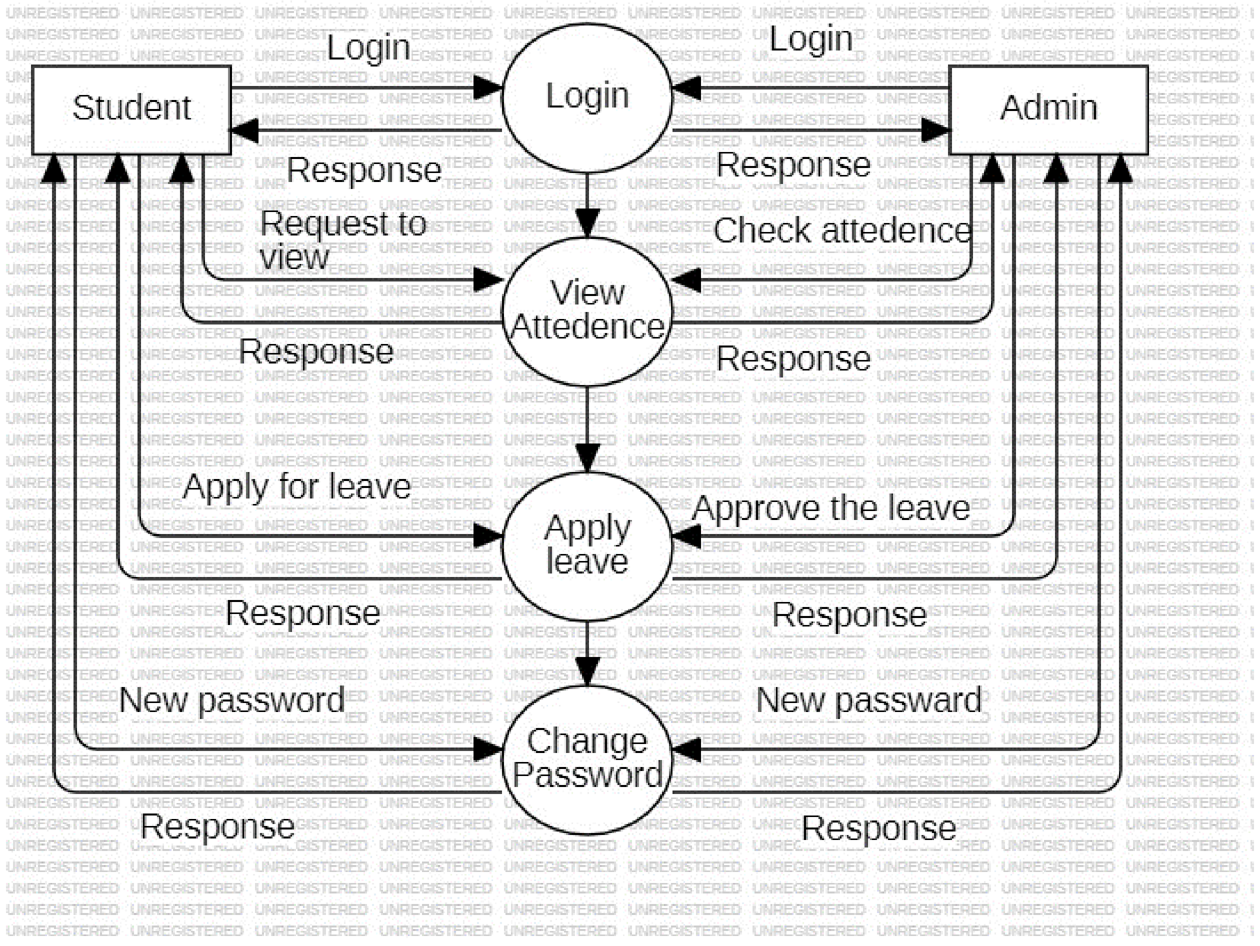


5.3 Data Flow Diagram:

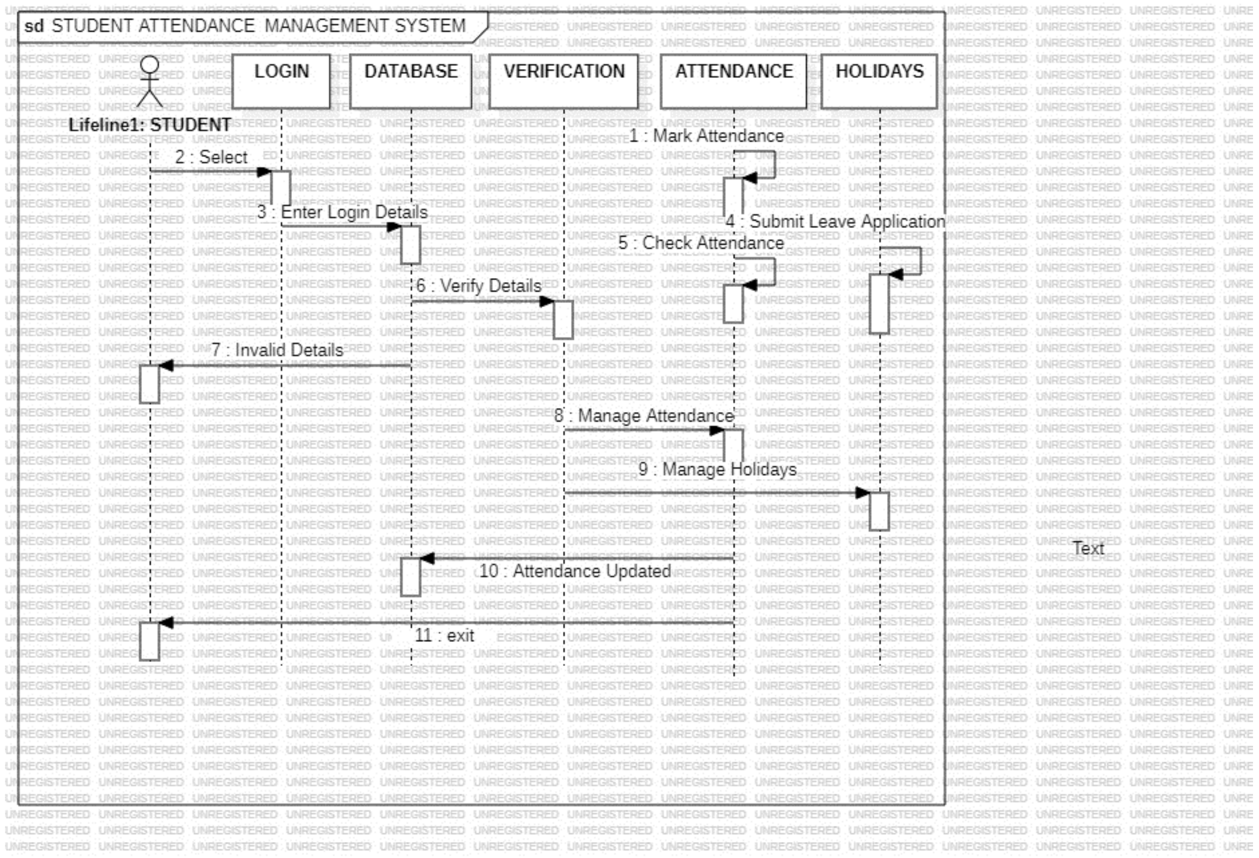
5.3.1 DFD level 0:



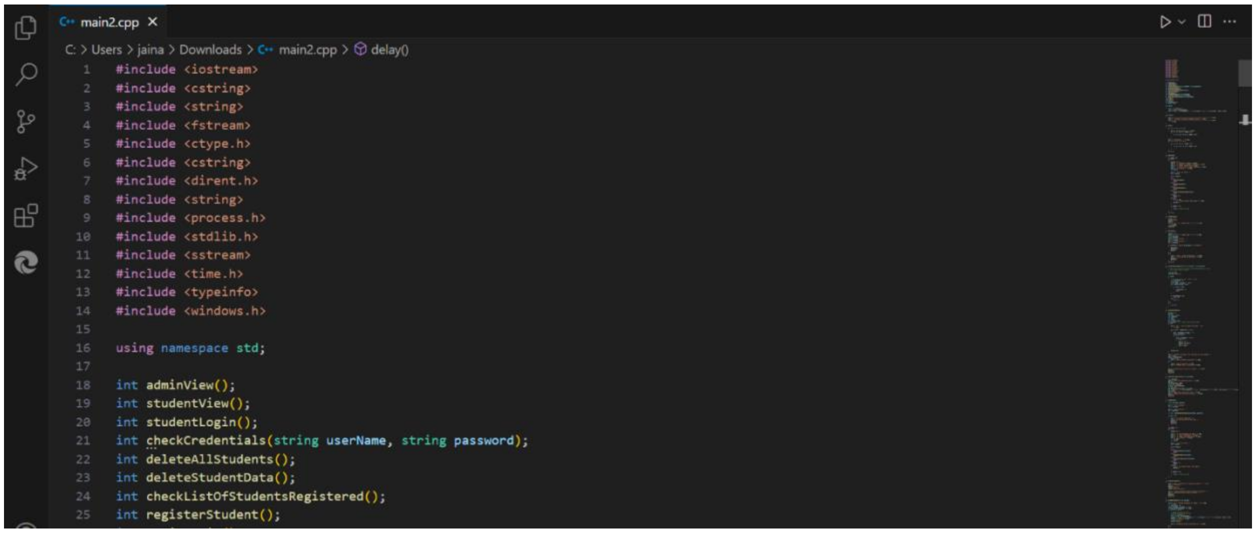
5.3.2 DFD level 1:



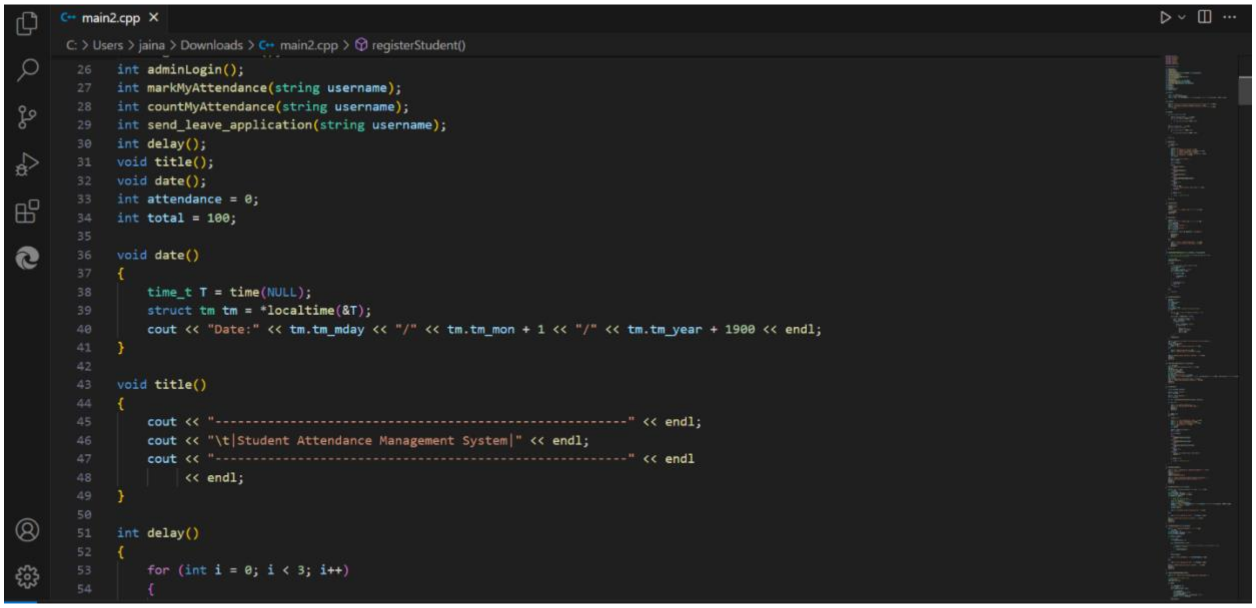
5.4 Sequence Diagram:



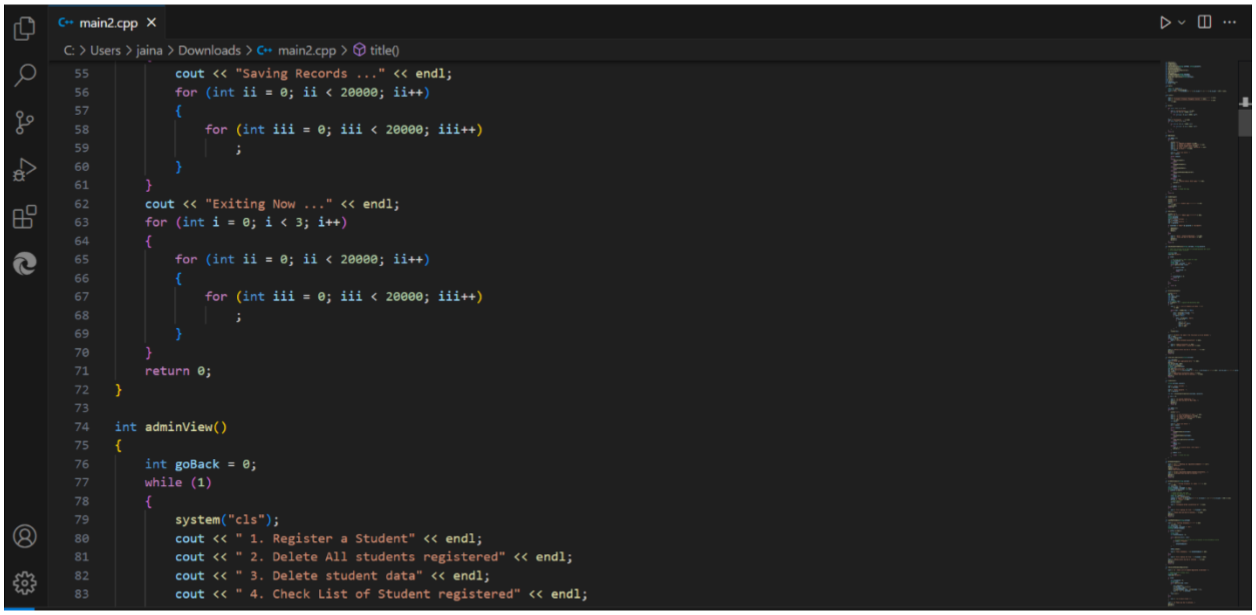
6 Project Snapshot:



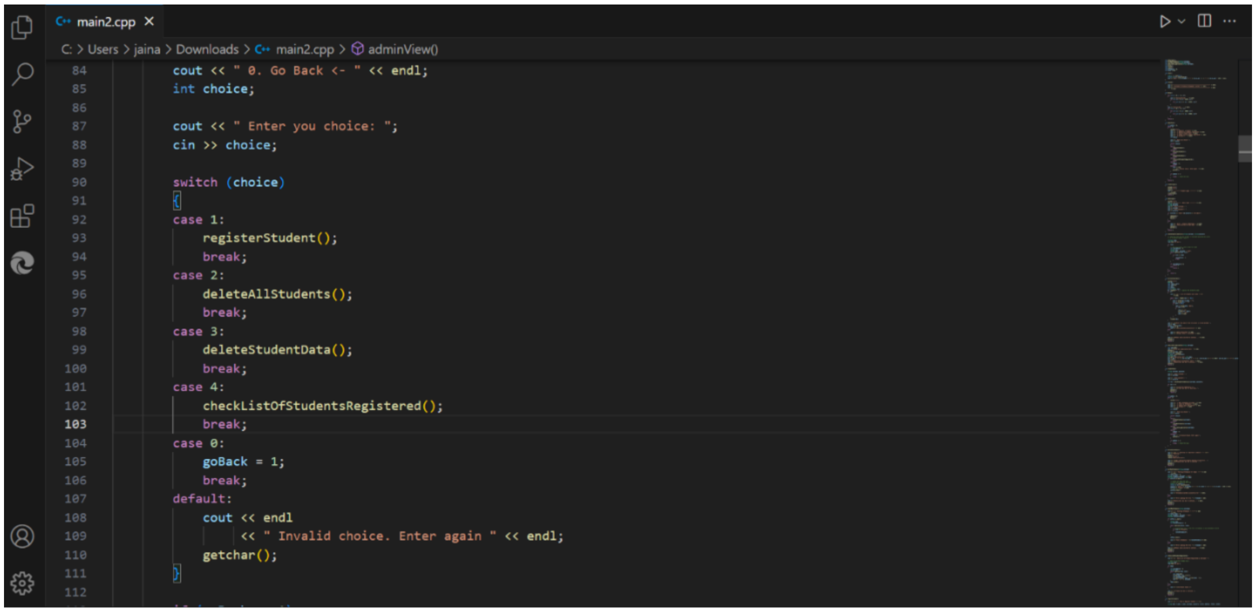
```cpp
#include <iostream>
#include <cstring>
#include <string>
#include <fstream>
#include <ctype.h>
#include <cstring>
#include <dirent.h>
#include <string>
#include <process.h>
#include <stdlib.h>
#include <sstream>
#include <time.h>
#include <typeinfo>
#include <windows.h>

using namespace std;

int adminView();
int studentView();
int studentLogin();
int checkCredentials(string userName, string password);
int deleteAllStudents();
int deleteStudentData();
int checkListOfStudentsRegistered();
int registerStudent();
```


main2.cpp
C: > Users > jaina > Downloads > main2.cpp > registerStudent()

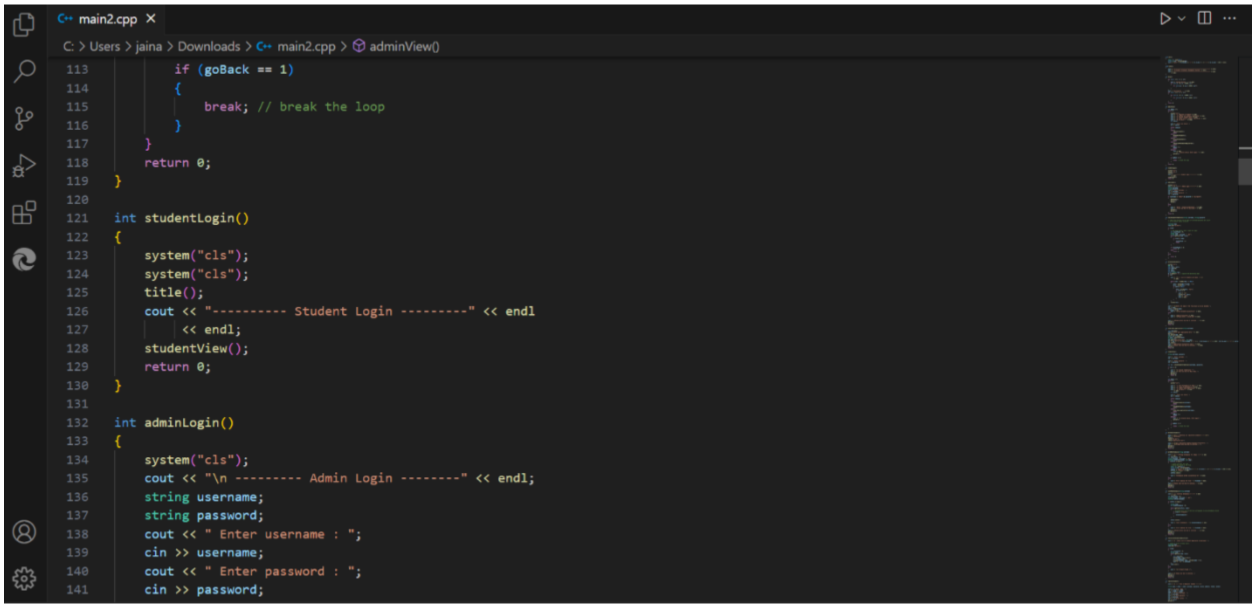
```cpp
int adminLogin();
int markMyAttendance(string username);
int countMyAttendance(string username);
int send_leave_application(string username);
int delay();
void title();
void date();
int attendance = 0;
int total = 100;

void date()
{
    time_t T = time(NULL);
    struct tm tm = *localtime(&T);
    cout << "Date:" << tm.tm_mday << "/" << tm.tm_mon + 1 << "/" << tm.tm_year + 1900 << endl;
}

void title()
{
    cout << "-------------------------------------------------------" << endl;
    cout << "\t|Student Attendance Management System|" << endl;
    cout << "-------------------------------------------------------" << endl
        << endl;
}

int delay()
{
    for (int i = 0; i < 3; i++)
    {
```



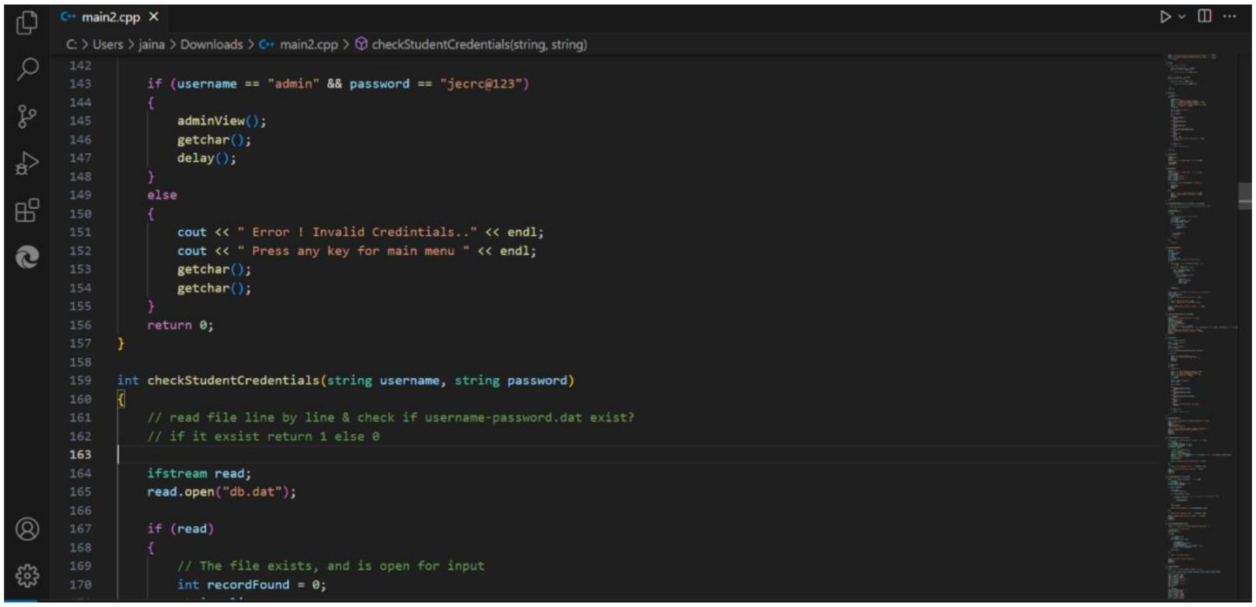
```cpp
        cout << "Saving Records ..." << endl;
        for (int ii = 0; ii < 20000; ii++)
        {
            for (int iii = 0; iii < 20000; iii++)
                ;
        }
    }
    cout << "Exiting Now ..." << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int ii = 0; ii < 20000; ii++)
        {
            for (int iii = 0; iii < 20000; iii++)
                ;
        }
    }
    return 0;
}

int adminView()
{
    int goBack = 0;
    while (1)
    {
        system("cls");
        cout << " 1. Register a Student" << endl;
        cout << " 2. Delete All students registered" << endl;
        cout << " 3. Delete student data" << endl;
        cout << " 4. Check List of Student registered" << endl;
```



```cpp
        cout << " 0. Go Back <- " << endl;
        int choice;

        cout << " Enter you choice: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            registerStudent();
            break;
        case 2:
            deleteAllStudents();
            break;
        case 3:
            deleteStudentData();
            break;
        case 4:
            checkListOfStudentsRegistered();
            break;
        case 0:
            goBack = 1;
            break;
        default:
            cout << endl
                 << " Invalid choice. Enter again " << endl;
            getchar();
        }

```

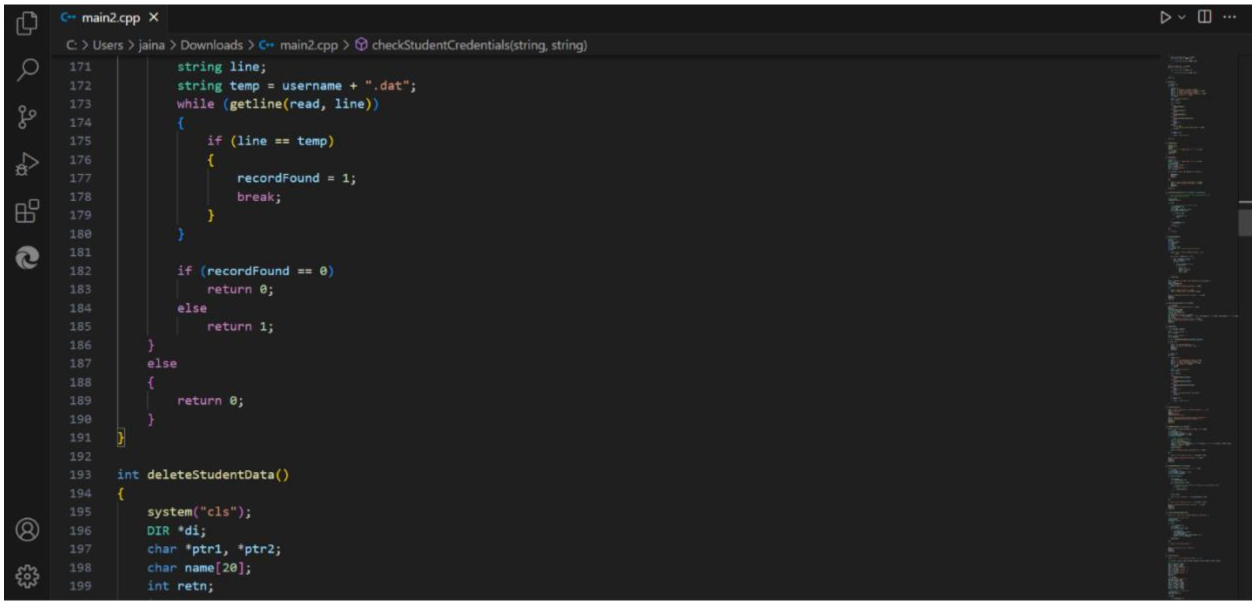


```cpp
            if (goBack == 1)
            {
                break; // break the loop
            }
        }
        return 0;
    }

    int studentLogin()
    {
        system("cls");
        system("cls");
        title();
        cout << "---------- Student Login ---------" << endl
             << endl;
        studentView();
        return 0;
    }

    int adminLogin()
    {
        system("cls");
        cout << "\n --------- Admin Login --------" << endl;
        string username;
        string password;
        cout << " Enter username : ";
        cin >> username;
        cout << " Enter password : ";
        cin >> password;
```

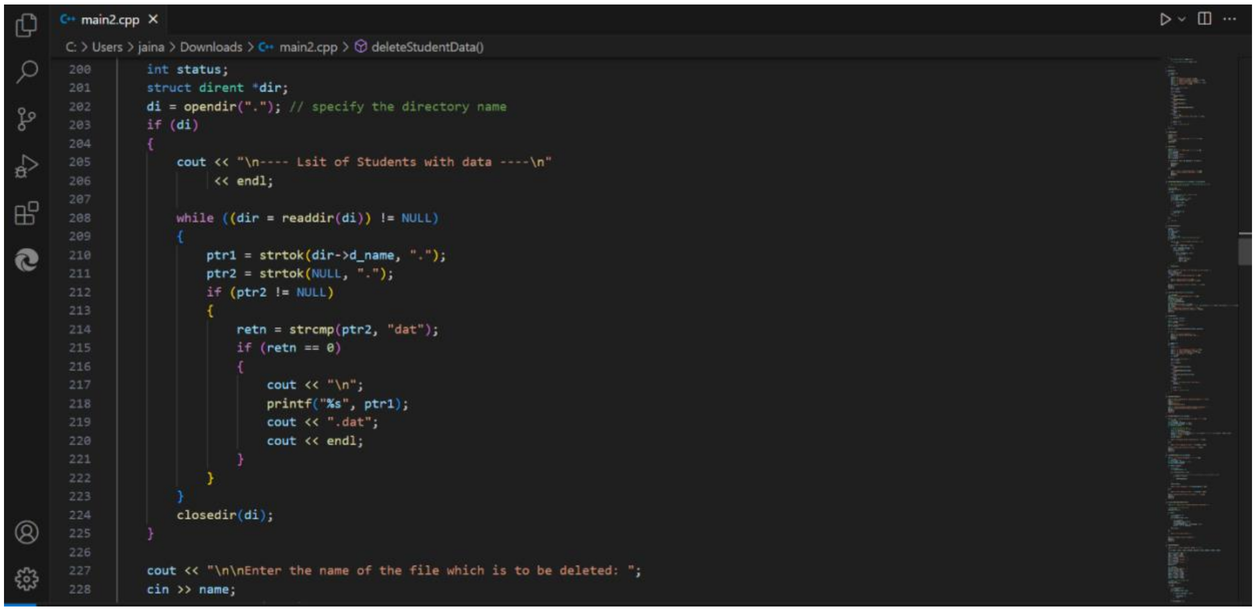


main2.cpp
C: > Users > jaina > Downloads > main2.cpp > checkStudentCredentials(string, string)

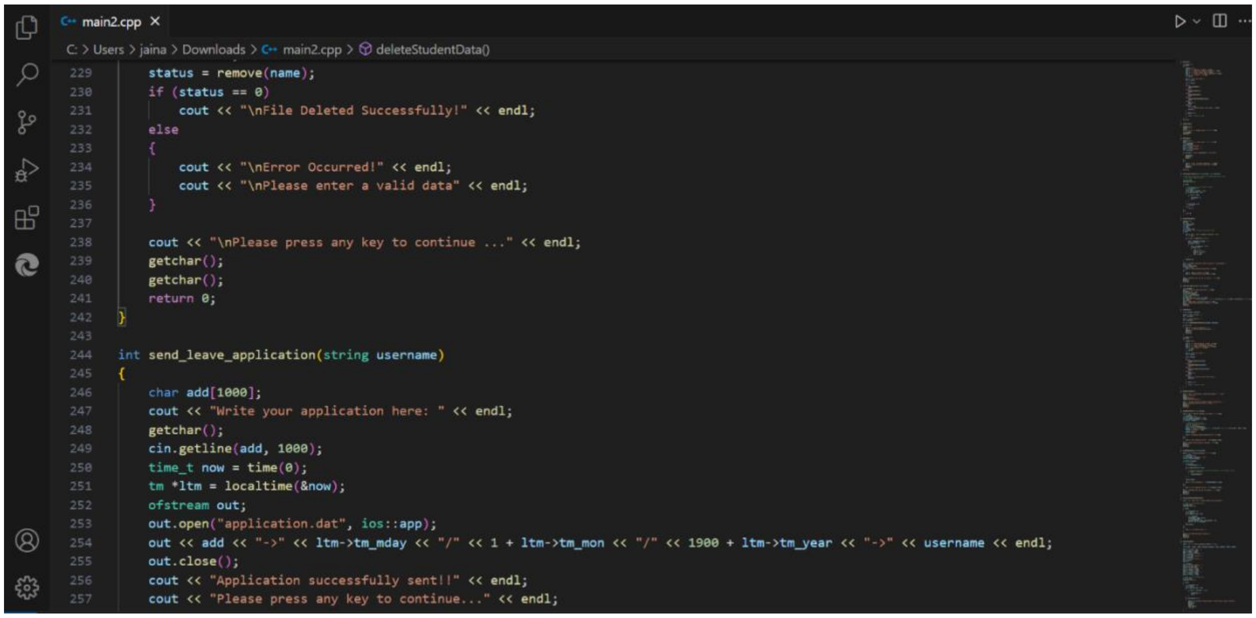
```cpp

        if (username == "admin" && password == "jecrc@123")
        {
            adminView();
            getchar();
            delay();
        }
        else
        {
            cout << " Error ! Invalid Credintials.." << endl;
            cout << " Press any key for main menu " << endl;
            getchar();
            getchar();
        }
        return 0;
    }

    int checkStudentCredentials(string username, string password)
    {
        // read file line by line & check if username-password.dat exist?
        // if it exsist return 1 else 0

        ifstream read;
        read.open("db.dat");

        if (read)
        {
            // The file exists, and is open for input
            int recordFound = 0;
```


main2.cpp
C: > Users > jaina > Downloads > main2.cpp > checkStudentCredentials(string, string)

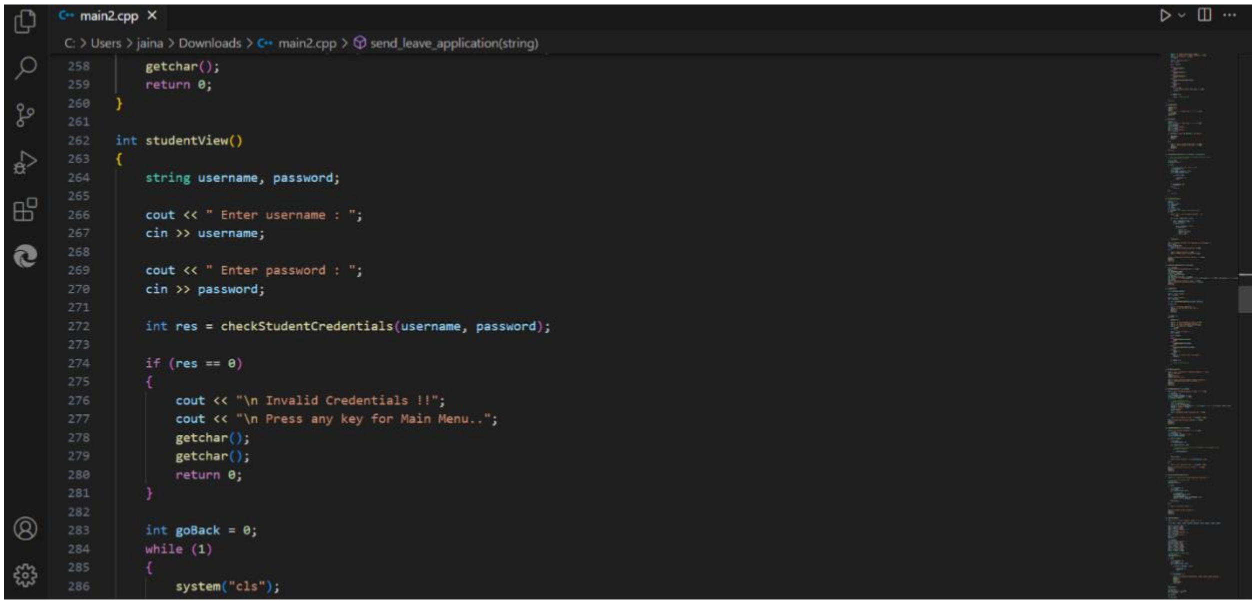
```cpp
        string line;
        string temp = username + ".dat";
        while (getline(read, line))
        {
            if (line == temp)
            {
                recordFound = 1;
                break;
            }
        }

        if (recordFound == 0)
            return 0;
        else
            return 1;
    }
    else
    {
        return 0;
    }
}

int deleteStudentData()
{
    system("cls");
    DIR *di;
    char *ptr1, *ptr2;
    char name[20];
    int retn;
```


main2.cpp
C: > Users > jaina > Downloads > main2.cpp > deleteStudentData()

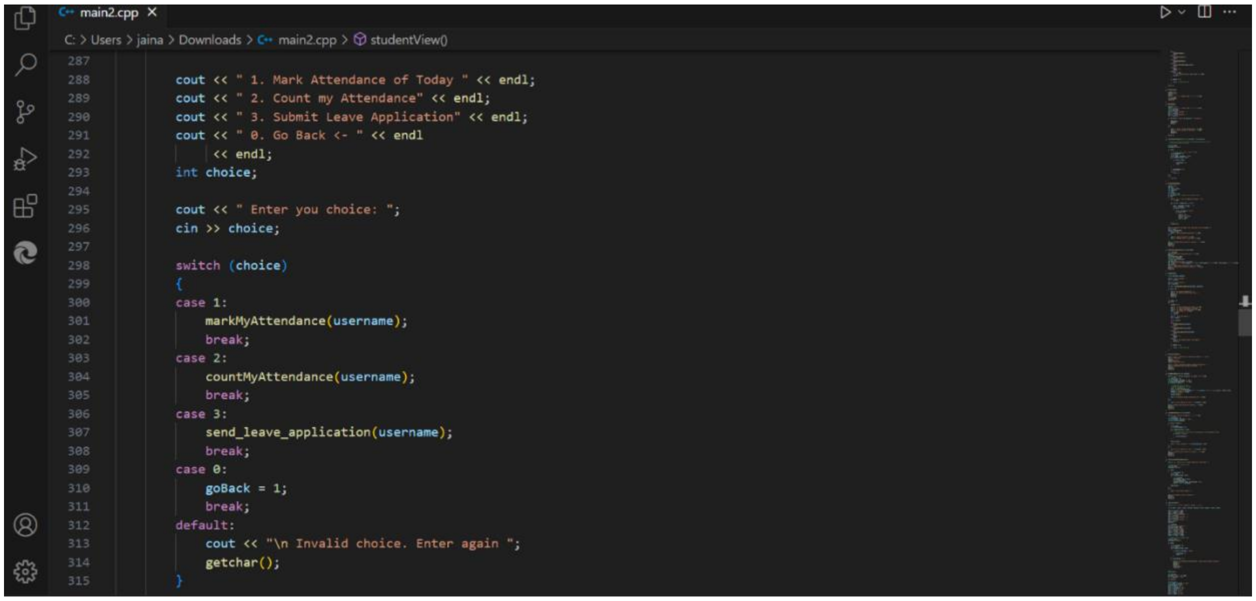
```cpp
    int status;
    struct dirent *dir;
    di = opendir("."); // specify the directory name
    if (di)
    {
        cout << "\n---- Lsit of Students with data ----\n"
             << endl;

        while ((dir = readdir(di)) != NULL)
        {
            ptr1 = strtok(dir->d_name, ".");
            ptr2 = strtok(NULL, ".");
            if (ptr2 != NULL)
            {
                retn = strcmp(ptr2, "dat");
                if (retn == 0)
                {
                    cout << "\n";
                    printf("%s", ptr1);
                    cout << ".dat";
                    cout << endl;
                }
            }
        }
        closedir(di);
    }

    cout << "\n\nEnter the name of the file which is to be deleted: ";
    cin >> name;
```


main2.cpp
C: > Users > jaina > Downloads > main2.cpp > deleteStudentData()

```cpp
    status = remove(name);
    if (status == 0)
        cout << "\nFile Deleted Successfully!" << endl;
    else
    {
        cout << "\nError Occurred!" << endl;
        cout << "\nPlease enter a valid data" << endl;
    }

    cout << "\nPlease press any key to continue ..." << endl;
    getchar();
    getchar();
    return 0;
}

int send_leave_application(string username)
{
    char add[1000];
    cout << "Write your application here: " << endl;
    getchar();
    cin.getline(add, 1000);
    time_t now = time(0);
    tm *ltm = localtime(&now);
    ofstream out;
    out.open("application.dat", ios::app);
    out << add << "->" << ltm->tm_mday << "/" << 1 + ltm->tm_mon << "/" << 1900 + ltm->tm_year << "->" << username << endl;
    out.close();
    cout << "Application successfully sent!!" << endl;
    cout << "Please press any key to continue..." << endl;
```

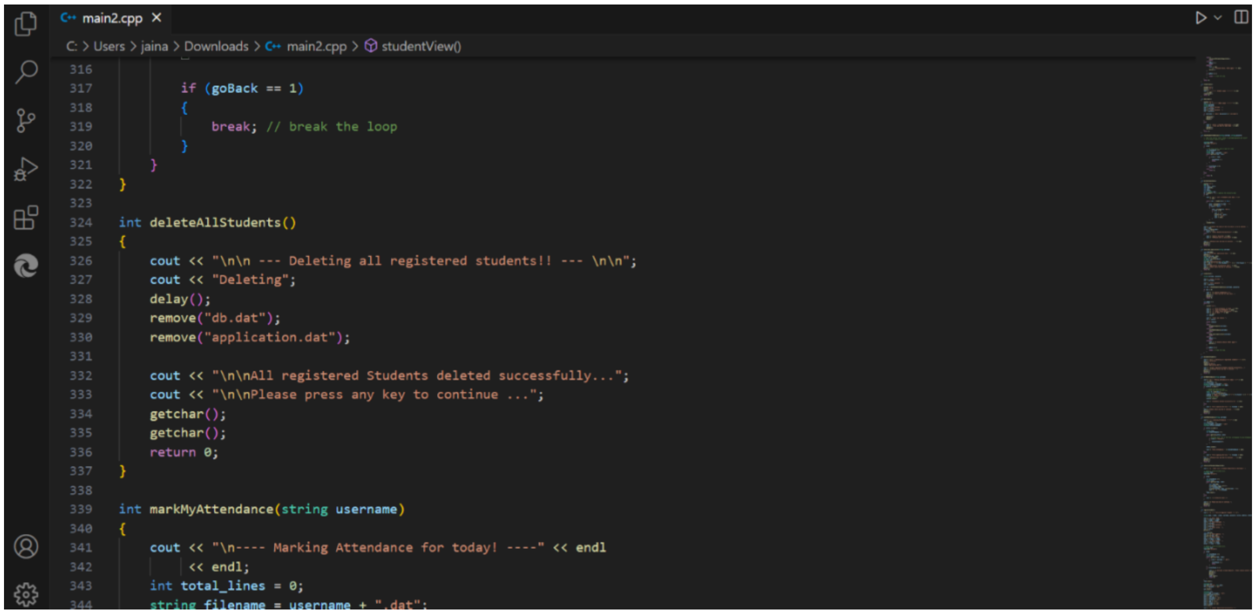

main2.cpp
C: > Users > jaina > Downloads > main2.cpp > send_leave_application(string)

```cpp
    getchar();
    return 0;
}

int studentView()
{
    string username, password;

    cout << " Enter username : ";
    cin >> username;

    cout << " Enter password : ";
    cin >> password;

    int res = checkStudentCredentials(username, password);

    if (res == 0)
    {
        cout << "\n Invalid Credentials !!";
        cout << "\n Press any key for Main Menu..";
        getchar();
        getchar();
        return 0;
    }

    int goBack = 0;
    while (1)
    {
        system("cls");
```


main2.cpp

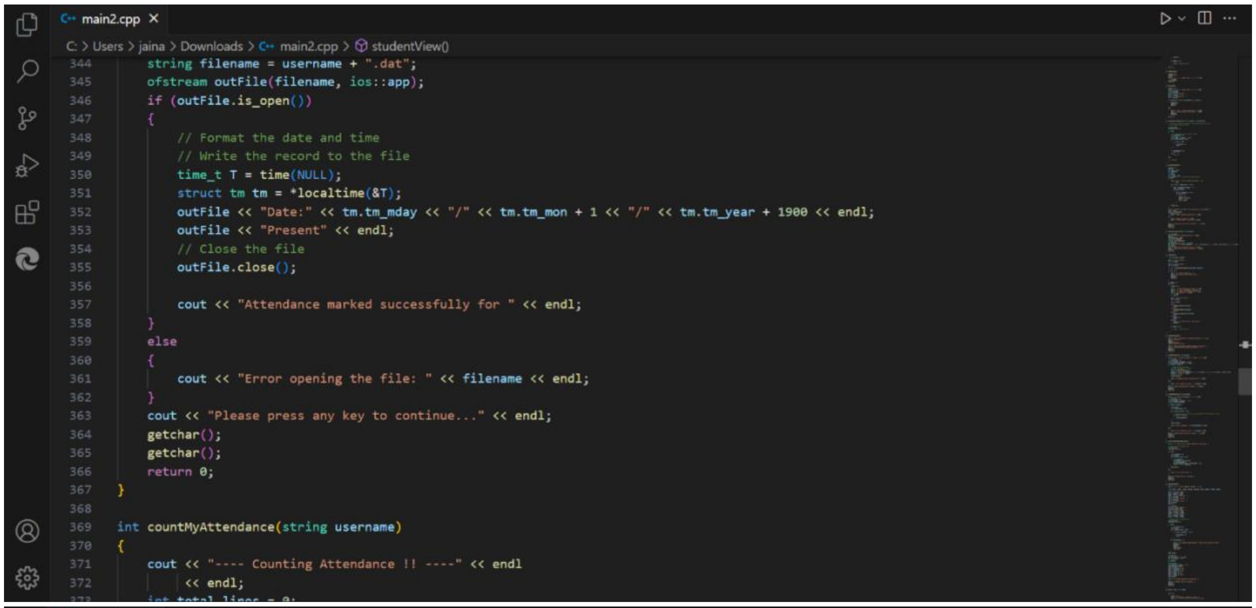
C: > Users > jaina > Downloads > main2.cpp > studentView()

```cpp

        cout << " 1. Mark Attendance of Today " << endl;
        cout << " 2. Count my Attendance" << endl;
        cout << " 3. Submit Leave Application" << endl;
        cout << " 0. Go Back <- " << endl
             << endl;
        int choice;

        cout << " Enter you choice: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            markMyAttendance(username);
            break;
        case 2:
            countMyAttendance(username);
            break;
        case 3:
            send_leave_application(username);
            break;
        case 0:
            goBack = 1;
            break;
        default:
            cout << "\n Invalid choice. Enter again ";
            getchar();
        }
```

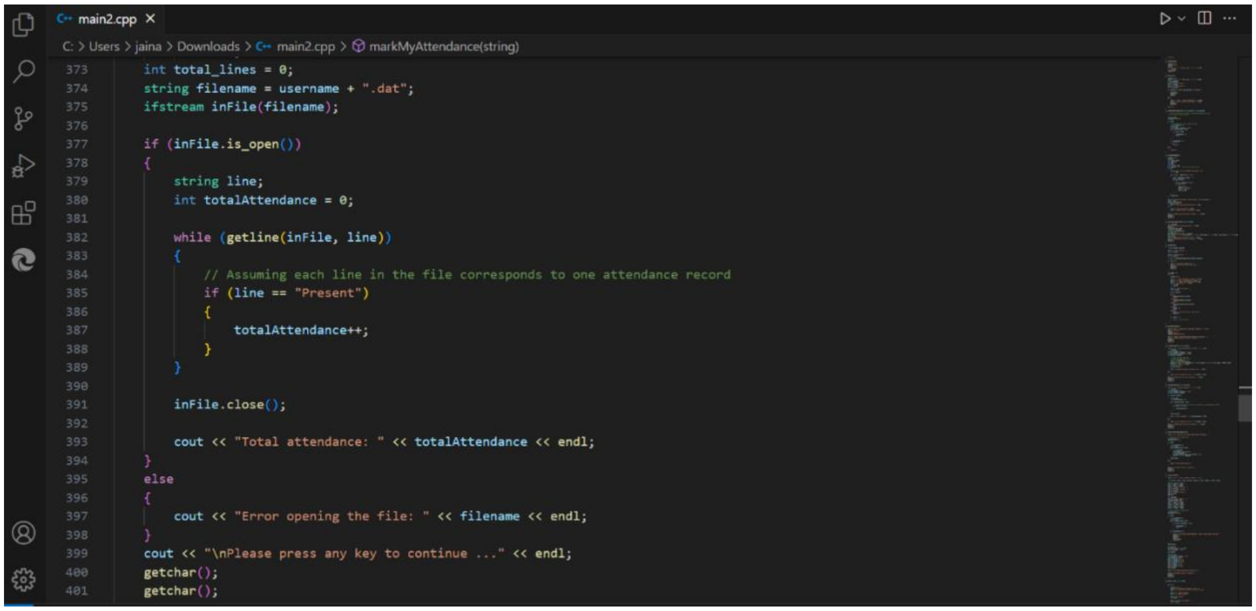

main2.cpp

C: > Users > jaina > Downloads > main2.cpp > studentView()

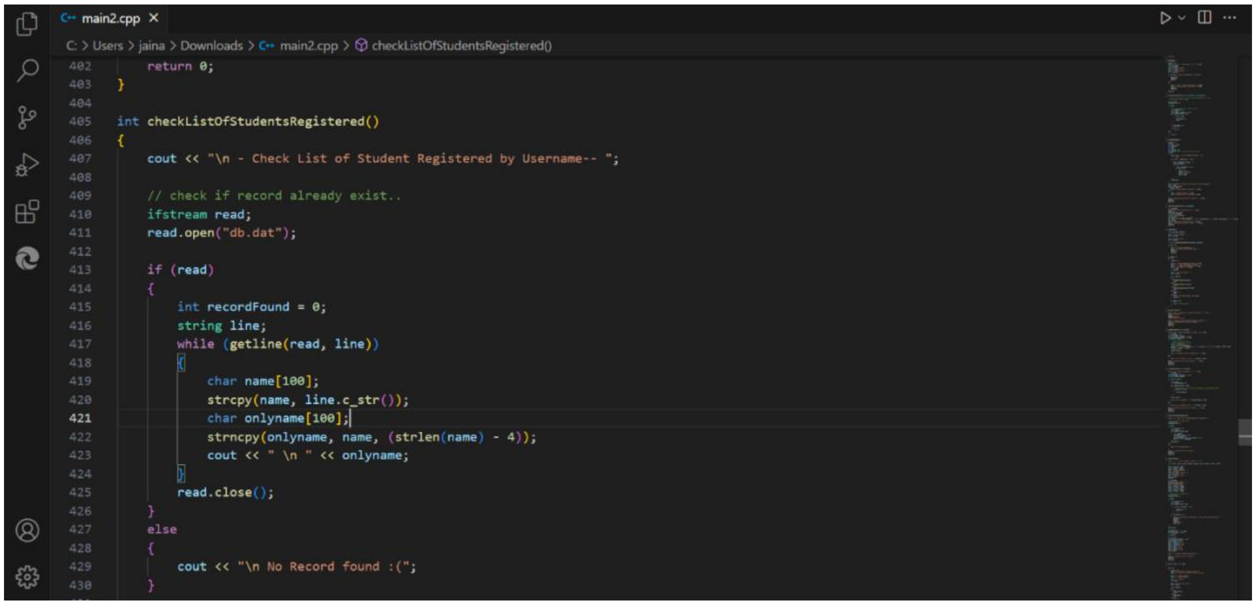
```cpp

        if (goBack == 1)
        {
            break; // break the loop
        }
    }
}

int deleteAllStudents()
{
    cout << "\n\n --- Deleting all registered students!! --- \n\n";
    cout << "Deleting";
    delay();
    remove("db.dat");
    remove("application.dat");

    cout << "\n\nAll registered Students deleted successfully...";
    cout << "\n\nPlease press any key to continue ...";
    getchar();
    getchar();
    return 0;
}

int markMyAttendance(string username)
{
    cout << "\n---- Marking Attendance for today! ----" << endl
         << endl;
    int total_lines = 0;
    string filename = username + ".dat";
```



```cpp
    string filename = username + ".dat";
    ofstream outFile(filename, ios::app);
    if (outFile.is_open())
    {
        // Format the date and time
        // Write the record to the file
        time_t T = time(NULL);
        struct tm tm = *localtime(&T);
        outFile << "Date:" << tm.tm_mday << "/" << tm.tm_mon + 1 << "/" << tm.tm_year + 1900 << endl;
        outFile << "Present" << endl;
        // Close the file
        outFile.close();

        cout << "Attendance marked successfully for " << endl;
    }
    else
    {
        cout << "Error opening the file: " << filename << endl;
    }
    cout << "Please press any key to continue..." << endl;
    getchar();
    getchar();
    return 0;
}

int countMyAttendance(string username)
{
    cout << "---- Counting Attendance !! ----" << endl
         << endl;
```


main2.cpp
C: > Users > jaina > Downloads > main2.cpp > markMyAttendance(string)

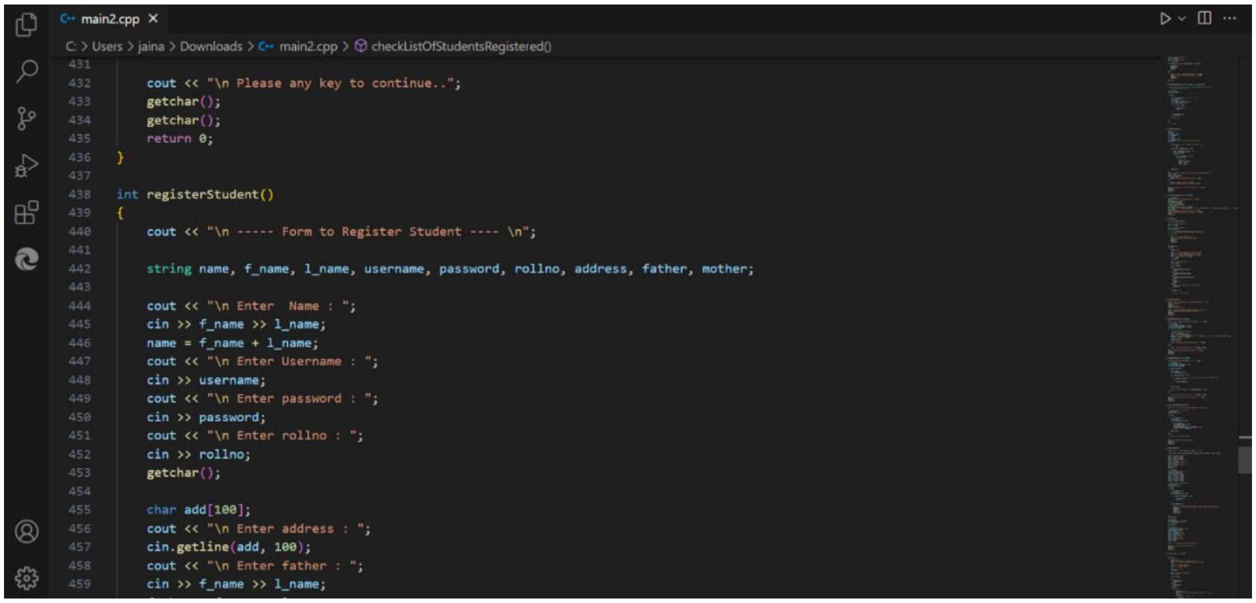
```cpp
    int total_lines = 0;
    string filename = username + ".dat";
    ifstream inFile(filename);

    if (inFile.is_open())
    {
        string line;
        int totalAttendance = 0;

        while (getline(inFile, line))
        {
            // Assuming each line in the file corresponds to one attendance record
            if (line == "Present")
            {
                totalAttendance++;
            }
        }

        inFile.close();

        cout << "Total attendance: " << totalAttendance << endl;
    }
    else
    {
        cout << "Error opening the file: " << filename << endl;
    }
    cout << "\nPlease press any key to continue ..." << endl;
    getchar();
    getchar();
```



main2.cpp

C: > Users > jaina > Downloads > main2.cpp > checkListOfStudentsRegistered()

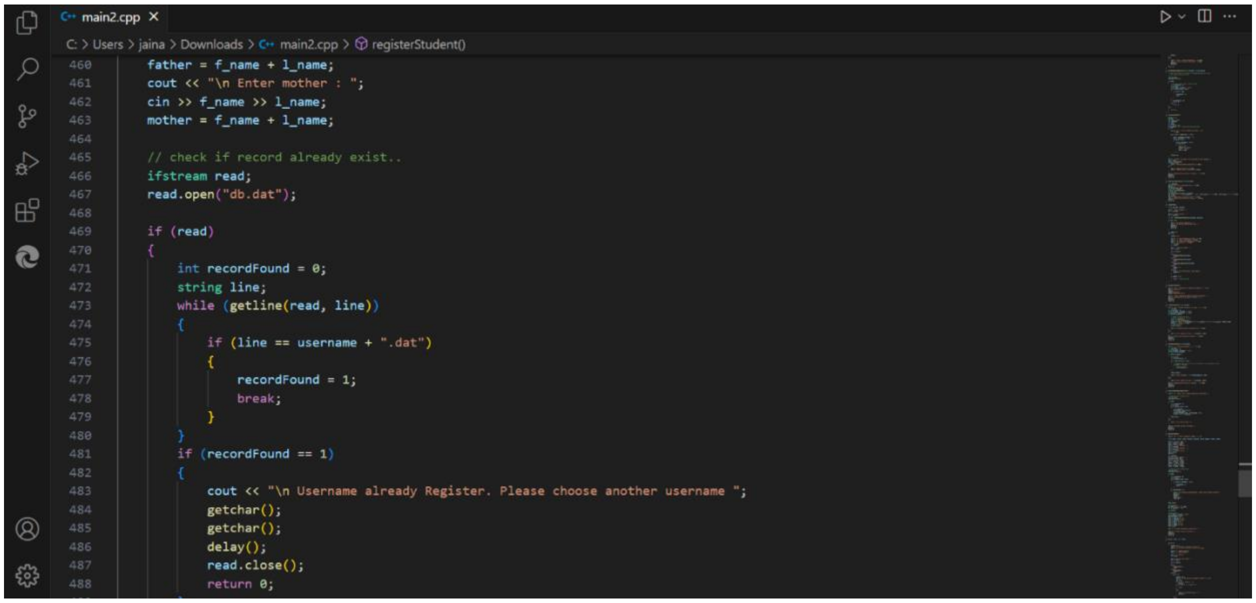
```cpp
    return 0;
}

int checkListOfStudentsRegistered()
{
    cout << "\n - Check List of Student Registered by Username-- ";

    // check if record already exist..
    ifstream read;
    read.open("db.dat");

    if (read)
    {
        int recordFound = 0;
        string line;
        while (getline(read, line))
        {
            char name[100];
            strcpy(name, line.c_str());
            char onlyname[100];
            strncpy(onlyname, name, (strlen(name) - 4));
            cout << " \n " << onlyname;
        }
        read.close();
    }
    else
    {
        cout << "\n No Record found :(";
    }
```



main2.cpp

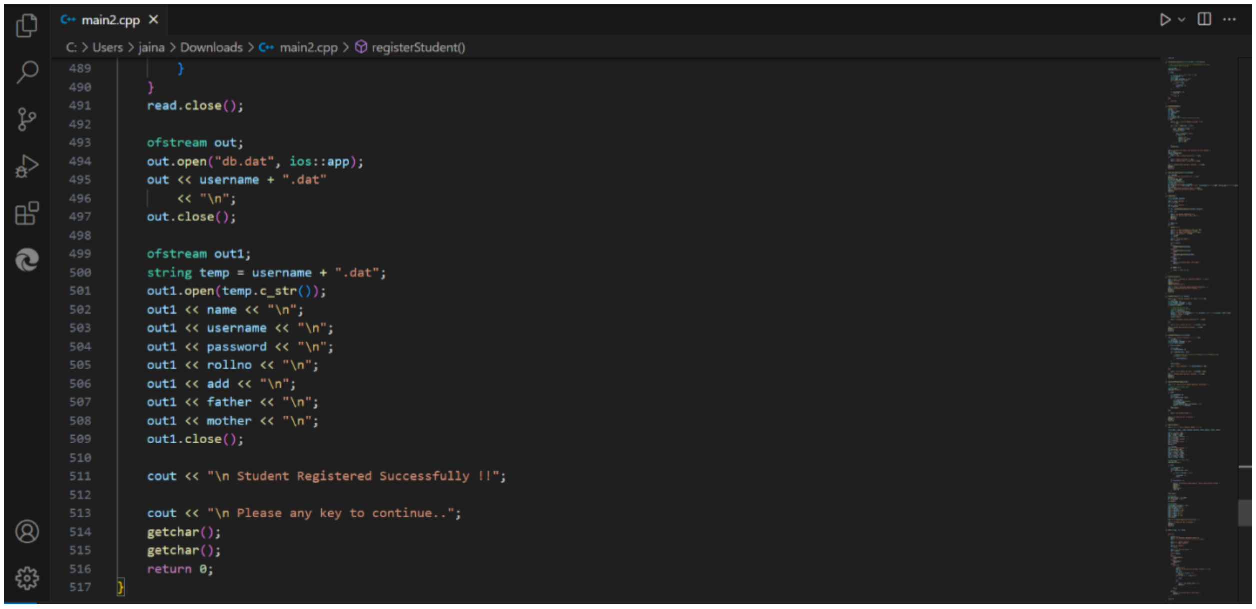
C: > Users > jaina > Downloads > main2.cpp > checkListOfStudentsRegistered()

```cpp

    cout << "\n Please any key to continue..";
    getchar();
    getchar();
    return 0;
}

int registerStudent()
{
    cout << "\n ----- Form to Register Student ---- \n";

    string name, f_name, l_name, username, password, rollno, address, father, mother;

    cout << "\n Enter  Name : ";
    cin >> f_name >> l_name;
    name = f_name + l_name;
    cout << "\n Enter Username : ";
    cin >> username;
    cout << "\n Enter password : ";
    cin >> password;
    cout << "\n Enter rollno : ";
    cin >> rollno;
    getchar();

    char add[100];
    cout << "\n Enter address : ";
    cin.getline(add, 100);
    cout << "\n Enter father : ";
    cin >> f_name >> l_name;
```



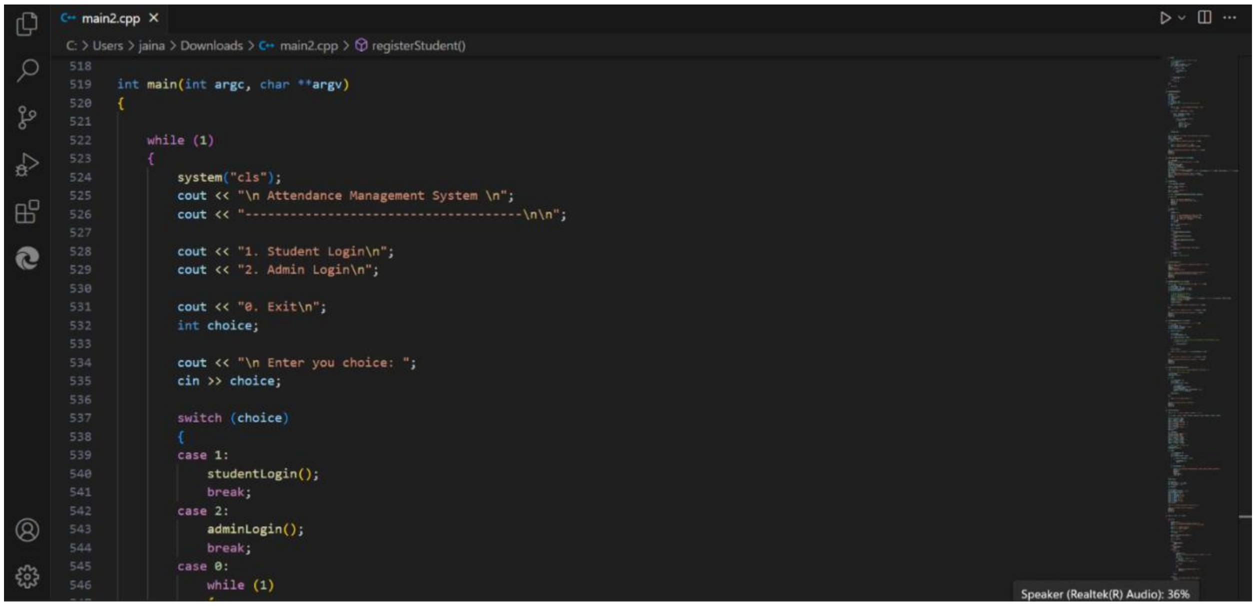
C: > Users > jaina > Downloads > main2.cpp > registerStudent()

```cpp
    father = f_name + l_name;
    cout << "\n Enter mother : ";
    cin >> f_name >> l_name;
    mother = f_name + l_name;

    // check if record already exist..
    ifstream read;
    read.open("db.dat");

    if (read)
    {
        int recordFound = 0;
        string line;
        while (getline(read, line))
        {
            if (line == username + ".dat")
            {
                recordFound = 1;
                break;
            }
        }
        if (recordFound == 1)
        {
            cout << "\n Username already Register. Please choose another username ";
            getchar();
            getchar();
            delay();
            read.close();
            return 0;
```

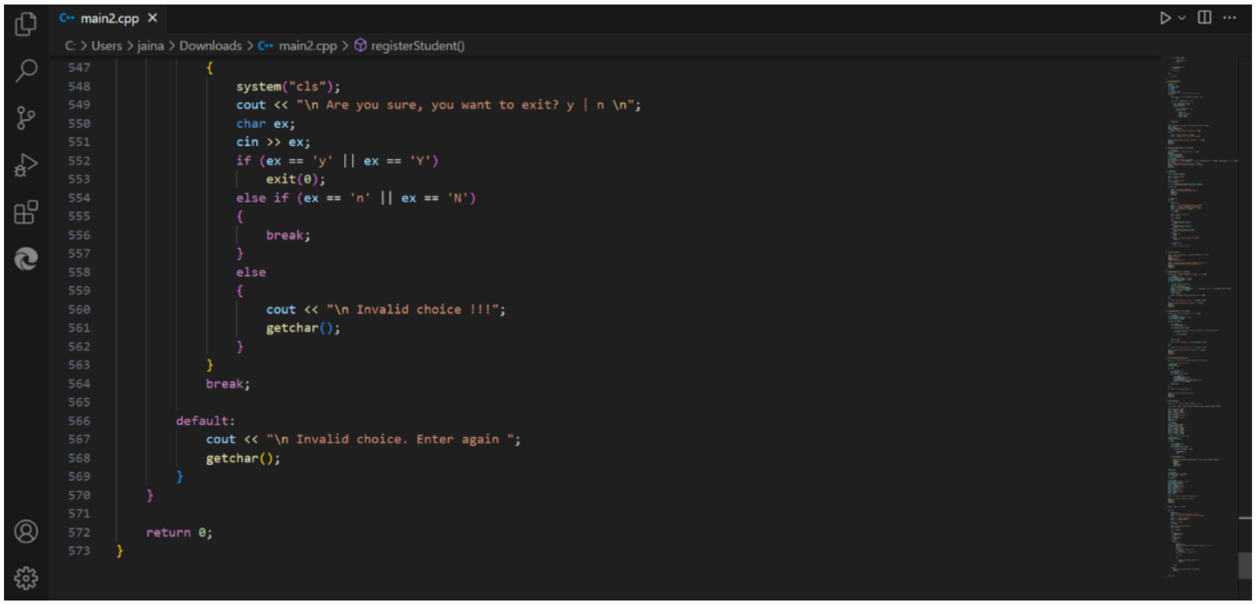


C: > Users > jaina > Downloads > main2.cpp > registerStudent()

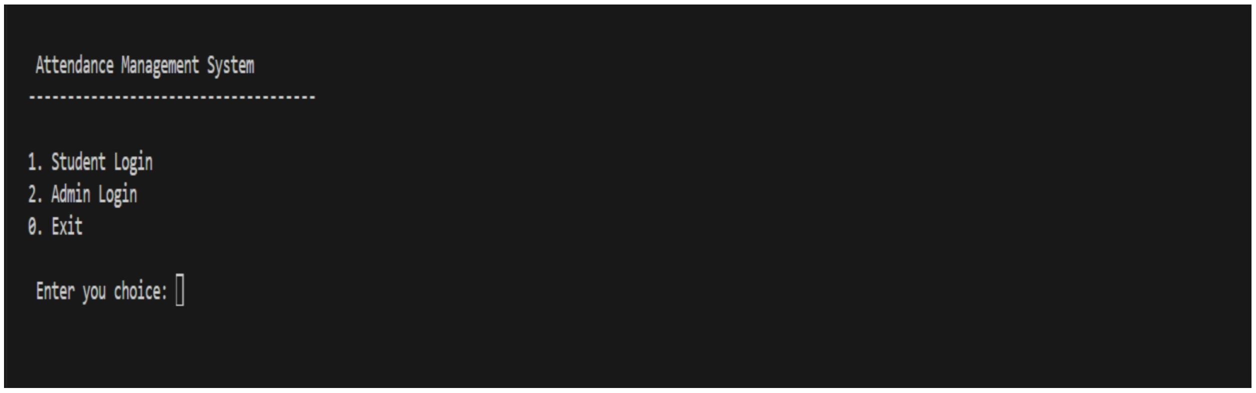
```cpp
        }
    }
    read.close();

    ofstream out;
    out.open("db.dat", ios::app);
    out << username + ".dat"
        << "\n";
    out.close();

    ofstream out1;
    string temp = username + ".dat";
    out1.open(temp.c_str());
    out1 << name << "\n";
    out1 << username << "\n";
    out1 << password << "\n";
    out1 << rollno << "\n";
    out1 << add << "\n";
    out1 << father << "\n";
    out1 << mother << "\n";
    out1.close();

    cout << "\n Student Registered Successfully !!";

    cout << "\n Please any key to continue..";
    getchar();
    getchar();
    return 0;
}
```


main2.cpp
C: > Users > jaina > Downloads > main2.cpp > registerStudent()
Speaker (Realtek(R) Audio): 36%

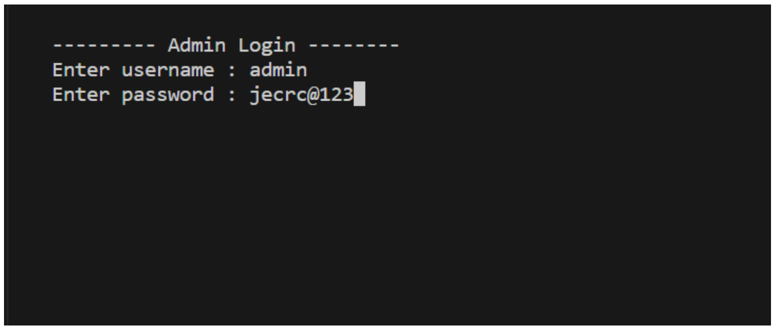
```cpp

int main(int argc, char **argv)
{

    while (1)
    {
        system("cls");
        cout << "\n Attendance Management System \n";
        cout << "-----------------------------------\n\n";

        cout << "1. Student Login\n";
        cout << "2. Admin Login\n";

        cout << "0. Exit\n";
        int choice;

        cout << "\n Enter you choice: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            studentLogin();
            break;
        case 2:
            adminLogin();
            break;
        case 0:
            while (1)
```


main2.cpp
C: > Users > jaina > Downloads > main2.cpp > registerStudent()

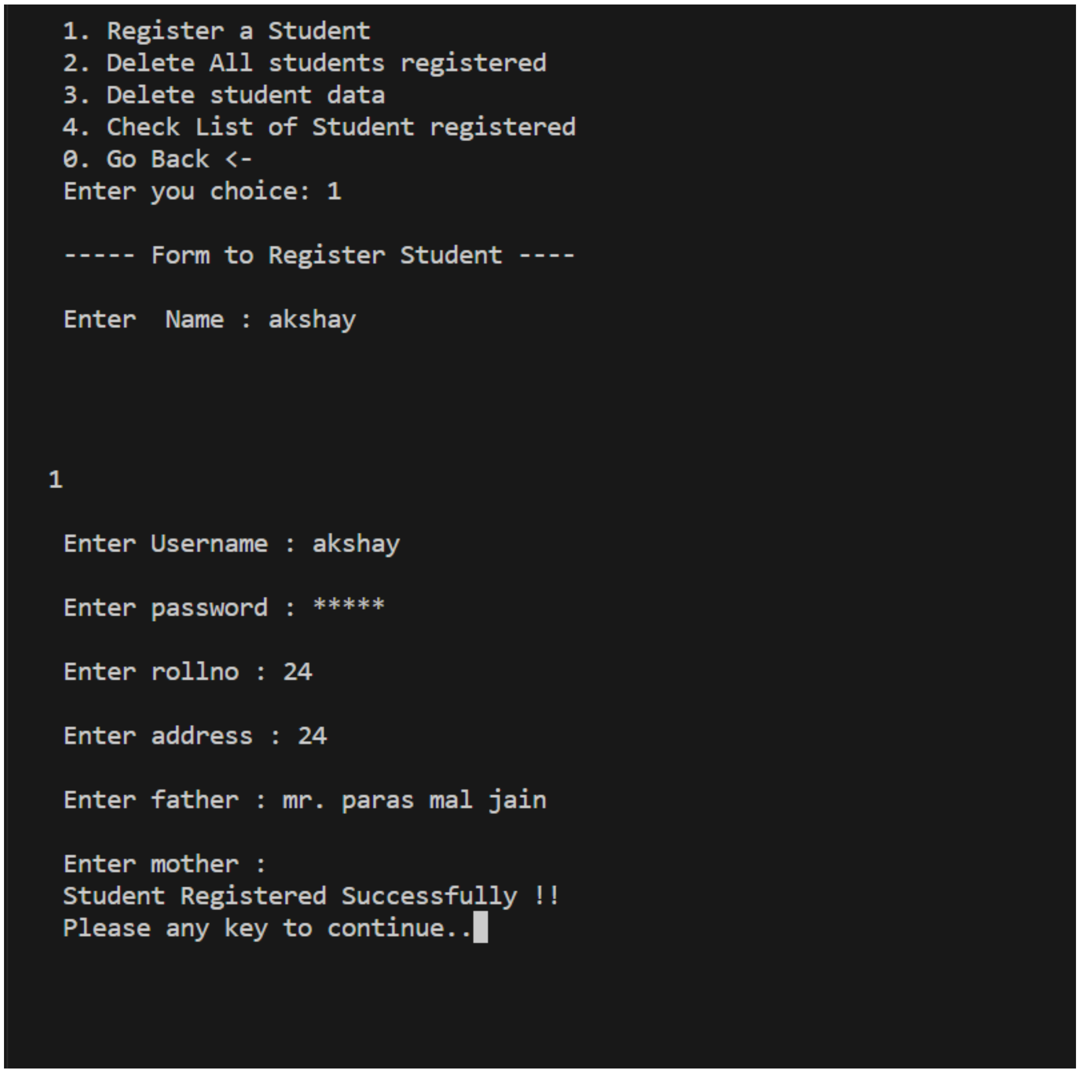
```cpp
            {
                system("cls");
                cout << "\n Are you sure, you want to exit? y | n \n";
                char ex;
                cin >> ex;
                if (ex == 'y' || ex == 'Y')
                    exit(0);
                else if (ex == 'n' || ex == 'N')
                {
                    break;
                }
                else
                {
                    cout << "\n Invalid choice !!!";
                    getchar();
                }
            }
            break;

        default:
            cout << "\n Invalid choice. Enter again ";
            getchar();
        }
    }

    return 0;
}
```



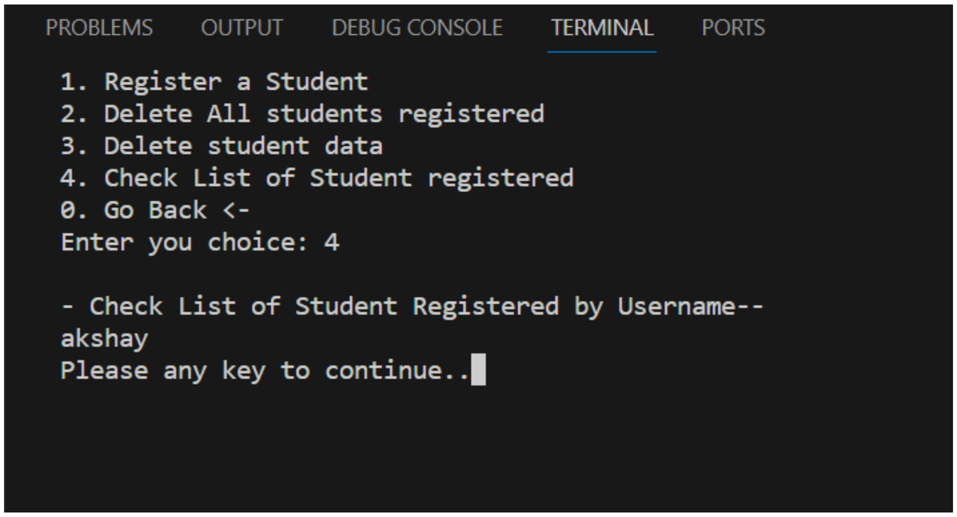
```
 Attendance Management System
-----------------------------------

1. Student Login
2. Admin Login
0. Exit

 Enter you choice:
```



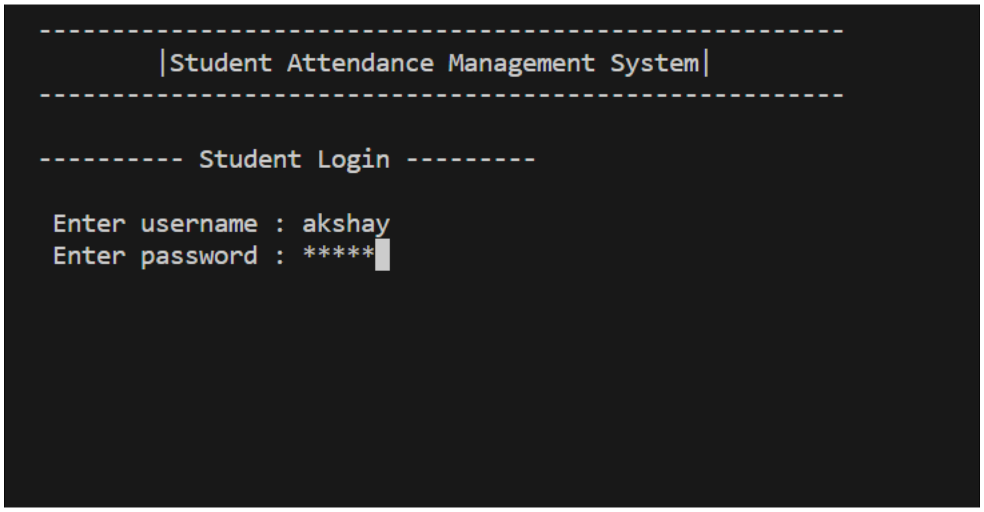
```
--------- Admin Login --------
Enter username : admin
Enter password : jecrc@123
```



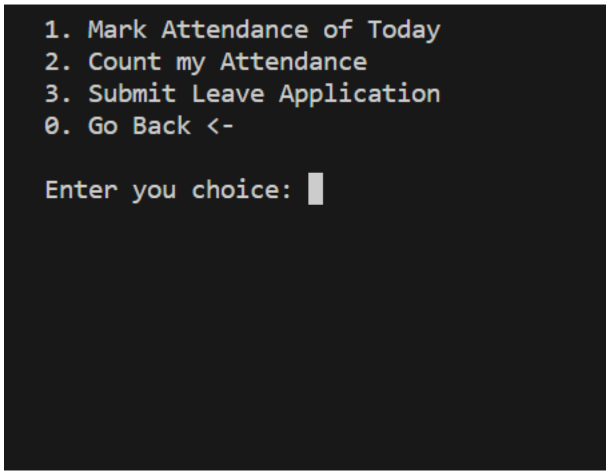
```
 1. Register a Student
 2. Delete All students registered
 3. Delete student data
 4. Check List of Student registered
 0. Go Back <-
 Enter you choice: 1

 ----- Form to Register Student ----

 Enter  Name : akshay




1

 Enter Username : akshay

 Enter password : *****

 Enter rollno : 24

 Enter address : 24

 Enter father : mr. paras mal jain

 Enter mother :
 Student Registered Successfully !!
 Please any key to continue..
```



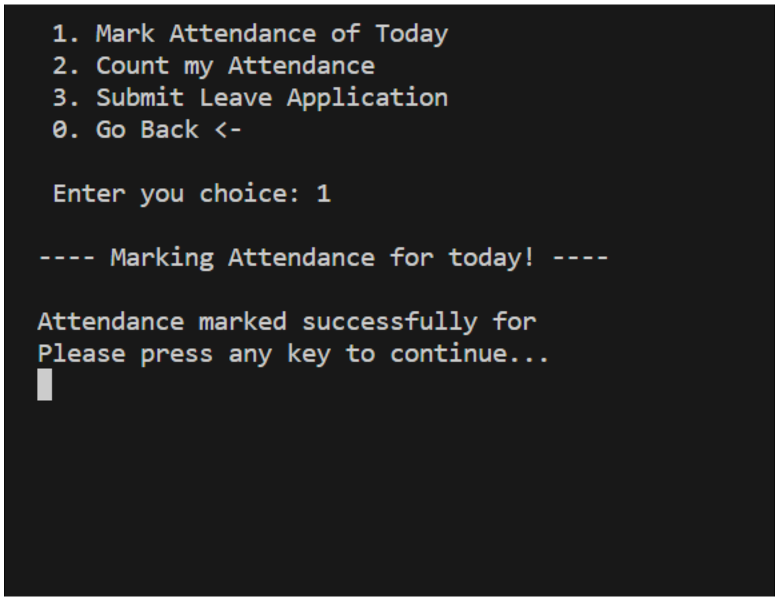
```
1. Register a Student
2. Delete All students registered
3. Delete student data
4. Check List of Student registered
0. Go Back <-
Enter you choice: 4

- Check List of Student Registered by Username--
akshay
Please any key to continue..
```



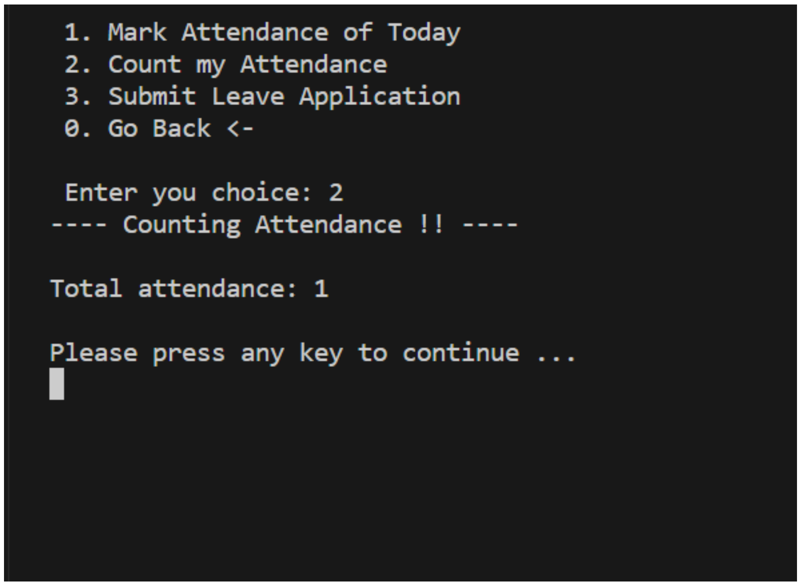
```
-------------------------------------------------------
        |Student Attendance Management System|
-------------------------------------------------------


---------- Student Login ---------

 Enter username : akshay
 Enter password : *****
```



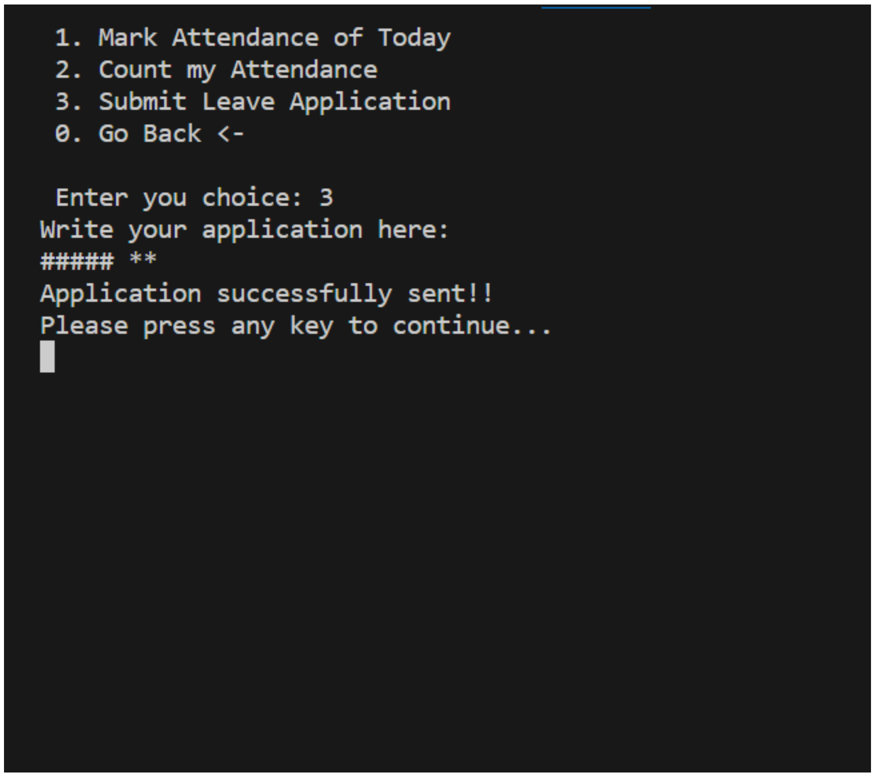
```
1. Mark Attendance of Today
2. Count my Attendance
3. Submit Leave Application
0. Go Back <-

Enter you choice: 
```



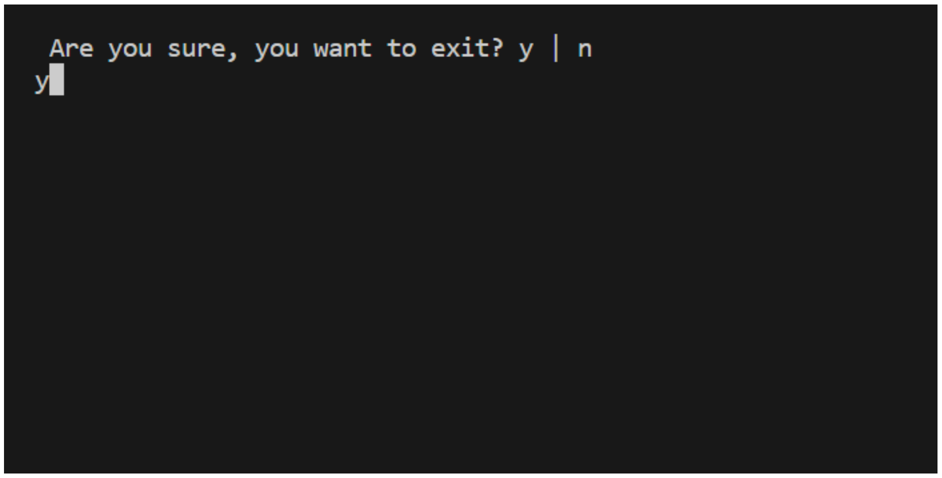
```
 1. Mark Attendance of Today
 2. Count my Attendance
 3. Submit Leave Application
 0. Go Back <-

 Enter you choice: 1

---- Marking Attendance for today! ----

Attendance marked successfully for
Please press any key to continue...
```



```
 1. Mark Attendance of Today
 2. Count my Attendance
 3. Submit Leave Application
 0. Go Back <-

 Enter you choice: 2
---- Counting Attendance !! ----

Total attendance: 1

Please press any key to continue ...
```



```
 1. Mark Attendance of Today
 2. Count my Attendance
 3. Submit Leave Application
 0. Go Back <-

 Enter you choice: 3
Write your application here:
##### **
Application successfully sent!!
Please press any key to continue...
```



```
 Are you sure, you want to exit? y | n
y
```

7 References:

- "C:\Users\jaina\OneDrive\Desktop\srs\main2.cpp"
- "C:\Users\jaina\OneDrive\Desktop\Visual Studio Code. link"
- https://www.bing.com/ck/a?!&&p=be729690ba9525adJmltdHM9MTcwMjQyNTYwMCZpZ3VpZD0xZDBlYzgwNi0yZGI1LTYwYTEtMzM4NC1kYmJhMmNiMzYxY2EmaW5zaWQ9NTIyOA&ptn=3&ver=2&hsh=3&fclid=1d0ec806-2db5-60a1-3384-dbba2cb361ca&psq=staruml&u=a1aHR0cHM6Ly9zdGFydW1sLmlvLw&ntb=1
- https://www.bing.com/ck/a?!&&p=3cb2ead185ff868cJmltdHM9MTcwMjQyNTYwMCZpZ3VpZD0xZDBlYzgwNi0yZGI1LTYwYTEtMzM4NC1kYmJhMmNiMzYxY2EmaW5zaWQ9NTI1OQ&ptn=3&ver=2&hsh=3&fclid=1d0ec806-2db5-60a1-3384-dbba2cb361ca&psq=academia.edu++code+of+attendence+management+system&u=a1aHR0cHM6Ly93d3cuYWNhZGVtaWEuZWR1LzMxMzQxNTE2L0F0dGVuZGFuY2VfTWFuYWdlbWVudF9TeXN0ZW1fQXR0ZW5kYW5jZV9NYW5hZ2VtZW50X1N5c3RlbV8&ntb=1