SALIM HABIB UNIVERSITY

Name : ***Nasir Hussain*** ***ID:*** ***F24CSC020***

Instructors: ***Dr.Sheeraz Arif / Sir Mansoor Ahmed***

Student Management System - Comprehensive Documentation :

This document outlines the Student Management System (SMS), developed in C++ to manage student records efficiently. It supports operations such as adding, displaying, updating, and deleting student information. The system stores data in a text file for persistence and includes input validation and error handling for smooth operation. This documentation provides code snippets and descriptions of key functionalities.

Table of Contents :

- LinkedIn

1. Introduction:

Purpose of the Program

The **Student Management System** is a C++ program designed to manage student records. The system provides features such as adding, displaying, searching, updating, and deleting student records. Each student record consists of a roll number, name, and CGPA. The system also supports user authentication (login/signup) for secure access to these features.

Features

- **Student Management**: Add, display, search, delete, and update student records.
- **User Authentication**: Secure login and signup functionality.
- **Input Validation**: Ensures that inputs (roll number, name, CGPA) are valid.
- **Error Handling**: Handles errors related to file input/output and invalid inputs.

2. Getting Started:

Installation Instructions

1. Clone or download the repository.
2. Compile the program using a C++ compiler.
3. Run the compiled executable from the terminal or command prompt.

Running the Program

Once the program is executed, users are prompted to either sign up or log in. After logging in successfully, they can manage student records.

4.Functionality Overview:

Student Management

Add Student

```
void addStudent() {
    ofstream outFile("students.txt", ios::app);
    Student s;
    cout << "Enter Roll Number: ";
    cin >> s.rollNumber;
    cout << "Enter Name: ";
    cin.ignore();
    getline(cin, s.name);
    cout << "Enter CGPA: ";
    cin >> s.cgpa;

    outFile << s.rollNumber << "|" << s.name << "|" << s.cgpa << endl;
    outFile.close();
}
```

Description:

- This function allows the user to add a new student record (roll number, name, and CGPA).
- An ofstream object (outFile) is used to append the data to the students.txt file.
- The cin.ignore() function is used to clear the buffer before reading the name, as the user may have previously entered numeric data.
- The student's details are written in the file in the format: rollNumber|name|cgpa.

Display Students

```
void displayStudents() {
    ifstream inFile("students.txt");
    string line;
    while (getline(inFile, line)) {
        cout << line << endl;
    }
    inFile.close();
}
```

Description:

- This function reads the students.txt file using an ifstream object (inFile) and displays each student record on the console.
- Each line in the file represents a student's record (roll number, name, CGPA).
- It uses getline() to read the file line by line and prints it to the console.

Search Student by Roll Number

```
void searchByRollNumber() {
    int rollNumber;
    cout << "Enter Roll Number: ";
    cin >> rollNumber;
    ifstream inFile("students.txt");
    string line;
    bool found = false;
    while (getline(inFile, line)) {
        if (line.find(to_string(rollNumber)) != string::npos) {
            cout << line << endl;
            found = true;
            break;
        }
    }
    if (!found) {
        cout << "Student not found." << endl;
    }
    inFile.close();
}
```

Description:

- This function allows the user to search for a student by their roll number.
- It takes the roll number as input, then iterates through the students.txt file line by line.
- The find() function is used to search for the roll number in each line (student record). If found, the corresponding student record is displayed.
- If the roll number is not found, an appropriate message is displayed.

Search Student by Name

```
void searchByName() {
    string name;
    cout << "Enter Name: ";
    cin.ignore();
    getline(cin, name);
    ifstream inFile("students.txt");
    string line;
    bool found = false;
    while (getline(inFile, line)) {
        if (line.find(name) != string::npos) {
            cout << line << endl;
            found = true;
        }
    }
    if (!found) {
        cout << "Student not found." << endl;
    }
    inFile.close();
}
```

Description:

- This function allows the user to search for a student by their name.
- It prompts the user to enter the student's name, then searches for it in the students.txt file.
- The find() function checks if the entered name appears in any student record. All matching records are displayed.
- If no match is found, a message is displayed indicating that the student was not found.

Delete Student

```
void deleteStudent() {
    int rollNumber;
    cout << "Enter Roll Number to Delete: ";
    cin >> rollNumber;
    ifstream inFile("students.txt");
    ofstream tempFile("temp.txt");
    string line;
    bool deleted = false;
    while (getline(inFile, line)) {
        if (line.find(to_string(rollNumber)) == string::npos) {
            tempFile << line << endl;
        } else {
            deleted = true;
        }
    }
    inFile.close();
    tempFile.close();
    remove("students.txt");
    rename("temp.txt", "students.txt");

    if (deleted) {
        cout << "Student deleted successfully." << endl;
    } else {
        cout << "Student not found." << endl;
    }
}
```

Description:

- This function deletes a student record by roll number.
- It creates a temporary file (temp.txt) to store the records that are not deleted.
- It reads through the students.txt file line by line. If the roll number is found, the record is not copied to the temporary file; otherwise, it is.
- After processing the file, the original file (students.txt) is removed, and the temporary file is renamed to students.txt.

Update Student

```
void updateStudent() {
    int rollNumber;
    cout << "Enter Roll Number to Update: ";
    cin >> rollNumber;
    ifstream inFile("students.txt");
    ofstream tempFile("temp.txt");
    string line;
    bool updated = false;
    while (getline(inFile, line)) {
        if (line.find(to_string(rollNumber)) != string::npos) {
            Student s;
            cout << "Enter New Name: ";
            cin.ignore();
            getline(cin, s.name);
            cout << "Enter New CGPA: ";
            cin >> s.cgpa;
            tempFile << rollNumber << "|" << s.name << "|" << s.cgpa << endl;
            updated = true;
        } else {
            tempFile << line << endl;
        }
    }
    inFile.close();
    tempFile.close();
    remove("students.txt");
    rename("temp.txt", "students.txt");

    if (updated) {
        cout << "Student updated successfully." << endl;
    } else {
        cout << "Student not found." << endl;
    }
```

Description:

- This function allows the user to update a student's record (name and CGPA) using their roll number.
- It reads through the students.txt file and looks for the matching roll number. If found, the program prompts the user to enter the new name and CGPA.
- The updated record is written to the temporary file. After processing, the original file is replaced by the updated one.

5. Input Validation

Input Validation Functions

CGPA Validation

```
bool isValidCGPA(float cgpa) {
    return (cgpa >= 0.0 && cgpa <= 4.0);
}
```

Description:

- This function checks if the entered CGPA is within the valid range (0.0 to 4.0).
- It returns true if the CGPA is valid, otherwise returns false.

Numeric Validation

```
bool isNumeric(string str) {
    for (char c : str) {
        if (!isdigit(c)) return false;
    }
    return true;
}
```

Description:

- This function checks if the provided string contains only numeric characters.
- It loops through each character in the string and checks if it is a digit. If any non-digit character is found, it returns false.

Name Validation

```
bool isValidName(string name) {
    for (char c : name) {
        if (!isalpha(c) && c != ' ') return false;
    }
    return true;
}
```

Description:

- This function validates if the name contains only alphabetic characters and spaces.
- It checks each character of the name. If any character is neither an alphabet nor a space, it returns false.

5. Error Handling

File I/O Error Handling

```
if (!inFile.is_open()) {
    cout << "Error opening file!" << endl;
}
```

Description:

- This condition checks if the file was successfully opened.
- If the file could not be opened, it displays an error message.

Invalid Input Handling

```
if (invalidInput) {
    cout << "Invalid input! Please enter valid data." << endl;
}
```

Description:

- This condition checks if the user input is invalid (e.g., entering a non-numeric value when a number is expected).
- If the input is invalid, it prompts the user to enter valid data.

6. Code Structure

Program Flow and Function Calls

- The program starts by prompting the user to sign up or log in. After successful authentication, the user is given options to manage student records.
- Based on the user's choice, functions like addStudent(), displayStudents(), searchByRollNumber(), etc., are called to perform specific actions.

Data Storage (Text File Format)

The data for students is stored in a text file (students.txt). Each line in the file represents a student's record in the following format:

```
Roll Number|Name|CGPA
```

7. Additional Information

User Authentication

The program supports user authentication by prompting for a username and password. These credentials are stored in a file called `credentials.txt`.

8. Conclusion

Limitations and Future Enhancements

- **Limitations**:
  - Storing data in text files is not scalable for large datasets.
  - Lack of password encryption and secure authentication.
- **Future Enhancements**:
  - Use of a database system for storing student records securely.
  - Implementation of password encryption for user authentication.

Summary

The system allows for easy management of student records with features like adding, searching, deleting, and updating records. It includes secure login functionality and ensures valid inputs.

9. Contact Information

For any inquiries, feedback, or support related to the Student Management System, please contact:

- **Name**: Nasir Hussain
- **Email**: nasirhussian.asadi@gmail.com
- **Phone**: 03462477680
- **GitHub**: NasirHussain10
- **LinkedIn**: Nasir Hussain