SALIM HABIB UNIVERSITY
(FORMERLY BARRETT HODGSON 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.

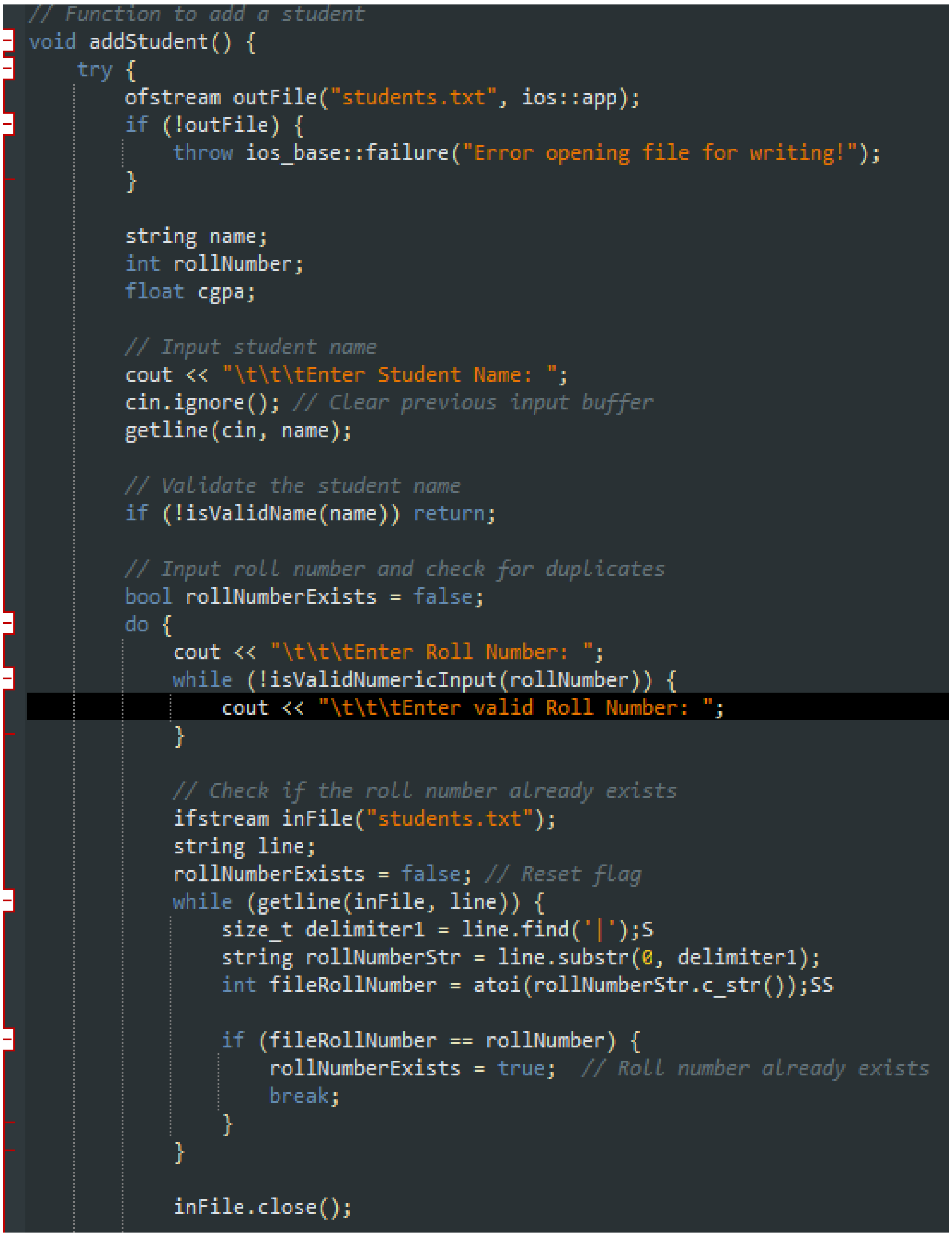
Running the Program

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

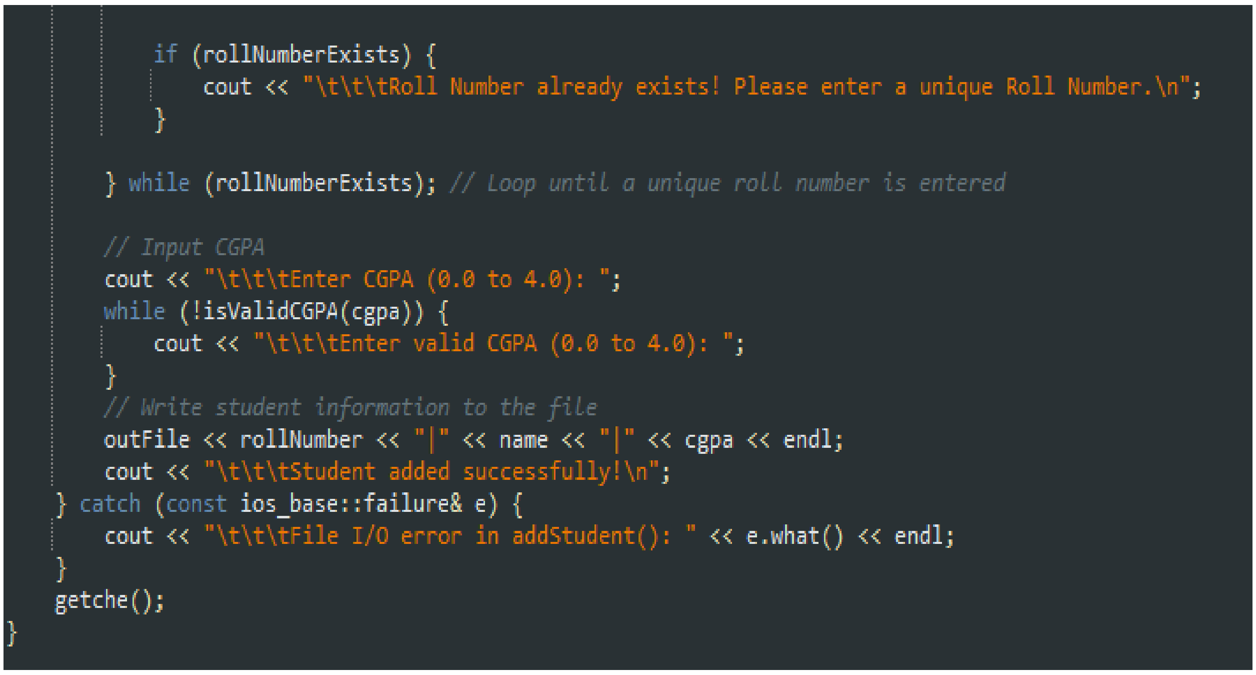
4. *Functionality Overview:*

Student Management

4.1 Add Student



```
// Function to add a student
void addStudent() {
    try {
        ofstream outFile("students.txt", ios::app);
        if (!outFile) {
            throw ios_base::failure("Error opening file for writing!");
        }

        string name;
        int rollNumber;
        float cgpa;

        // Input student name
        cout << "\t\t\tEnter Student Name: ";
        cin.ignore(); // Clear previous input buffer
        getline(cin, name);

        // Validate the student name
        if (!isValidName(name)) return;

        // Input roll number and check for duplicates
        bool rollNumberExists = false;
        do {
            cout << "\t\t\tEnter Roll Number: ";
            while (!isValidNumericInput(rollNumber)) {
                cout << "\t\t\tEnter valid Roll Number: ";
            }

            // Check if the roll number already exists
            ifstream inFile("students.txt");
            string line;
            rollNumberExists = false; // Reset flag
            while (getline(inFile, line)) {
                size_t delimiter1 = line.find('|');S
                string rollNumberStr = line.substr(0, delimiter1);
                int fileRollNumber = atoi(rollNumberStr.c_str());SS

                if (fileRollNumber == rollNumber) {
                    rollNumberExists = true;  // Roll number already exists
                    break;
                }
            }

            inFile.close();
```

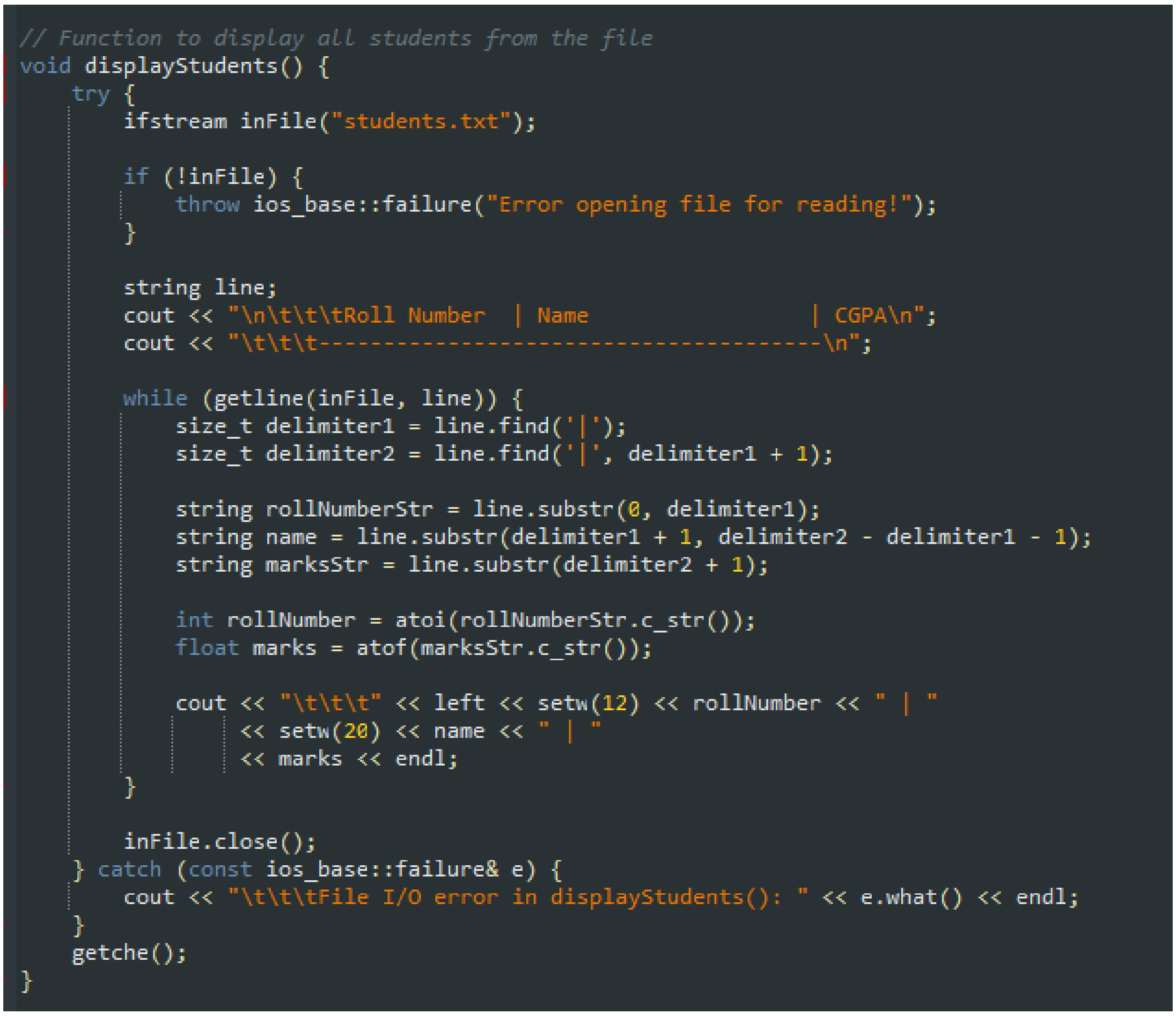


```
            if (rollNumberExists) {
                cout << "\t\t\tRoll Number already exists! Please enter a unique Roll Number.\n";
            }

        } while (rollNumberExists); // Loop until a unique roll number is entered

        // Input CGPA
        cout << "\t\t\tEnter CGPA (0.0 to 4.0): ";
        while (!isValidCGPA(cgpa)) {
            cout << "\t\t\tEnter valid CGPA (0.0 to 4.0): ";
        }
        // Write student information to the file
        outFile << rollNumber << "|" << name << "|" << cgpa << endl;
        cout << "\t\t\tStudent added successfully!\n";
    } catch (const ios_base::failure& e) {
        cout << "\t\t\tFile I/O error in addStudent(): " << e.what() << endl;
    }
    getche();
}
```

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

4.2 Display Students

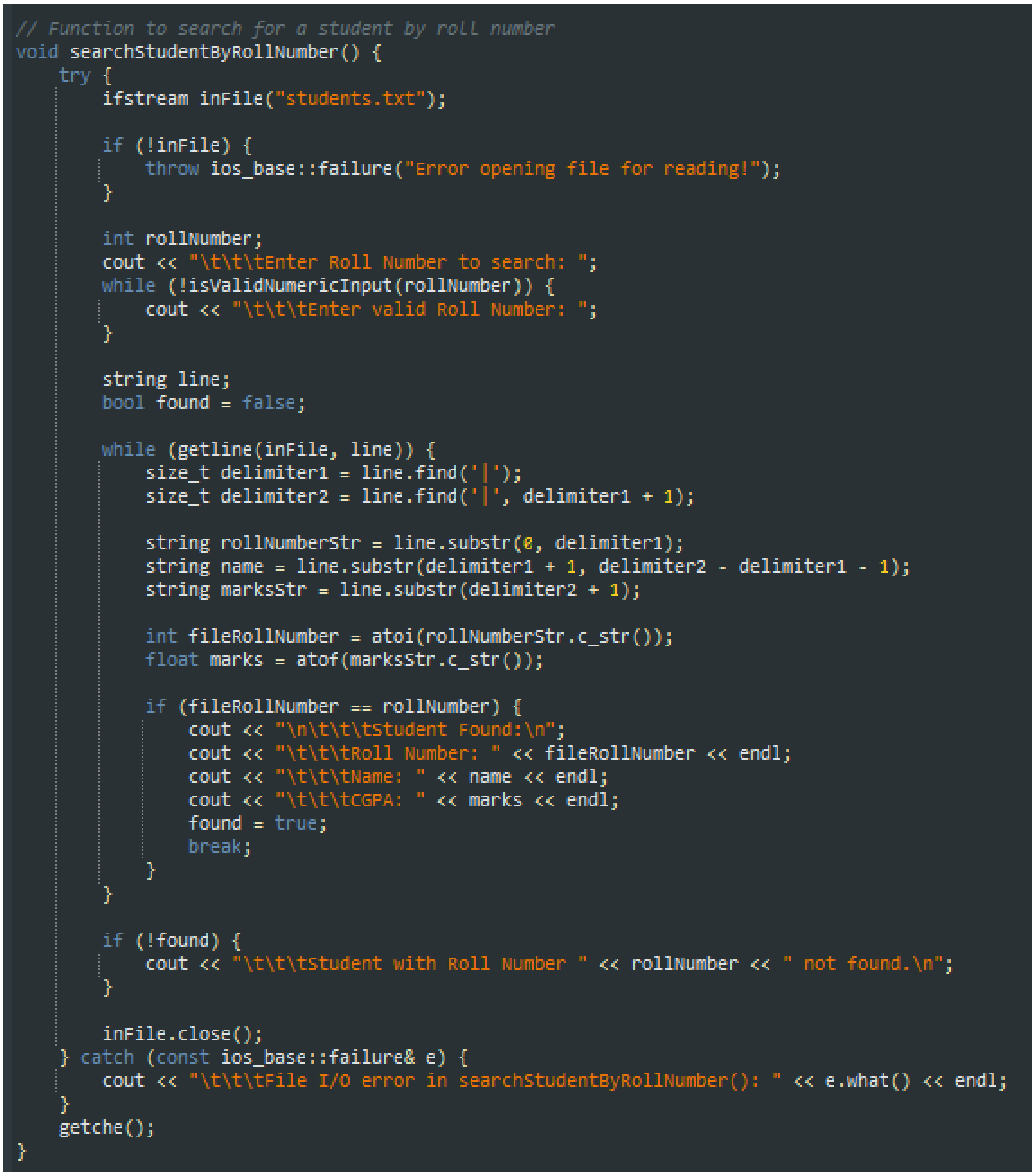


```cpp
// Function to display all students from the file
void displayStudents() {
    try {
        ifstream inFile("students.txt");

        if (!inFile) {
            throw ios_base::failure("Error opening file for reading!");
        }

        string line;
        cout << "\n\t\t\tRoll Number  | Name                 | CGPA\n";
        cout << "\t\t\t---------------------------------------\n";

        while (getline(inFile, line)) {
            size_t delimiter1 = line.find('|');
            size_t delimiter2 = line.find('|', delimiter1 + 1);

            string rollNumberStr = line.substr(0, delimiter1);
            string name = line.substr(delimiter1 + 1, delimiter2 - delimiter1 - 1);
            string marksStr = line.substr(delimiter2 + 1);

            int rollNumber = atoi(rollNumberStr.c_str());
            float marks = atof(marksStr.c_str());

            cout << "\t\t\t" << left << setw(12) << rollNumber << " | "
                 << setw(20) << name << " | "
                 << marks << endl;
        }

        inFile.close();
    } catch (const ios_base::failure& e) {
        cout << "\t\t\tFile I/O error in displayStudents(): " << e.what() << endl;
    }
    getche();
}
```

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.

4.3 Search Student by ID

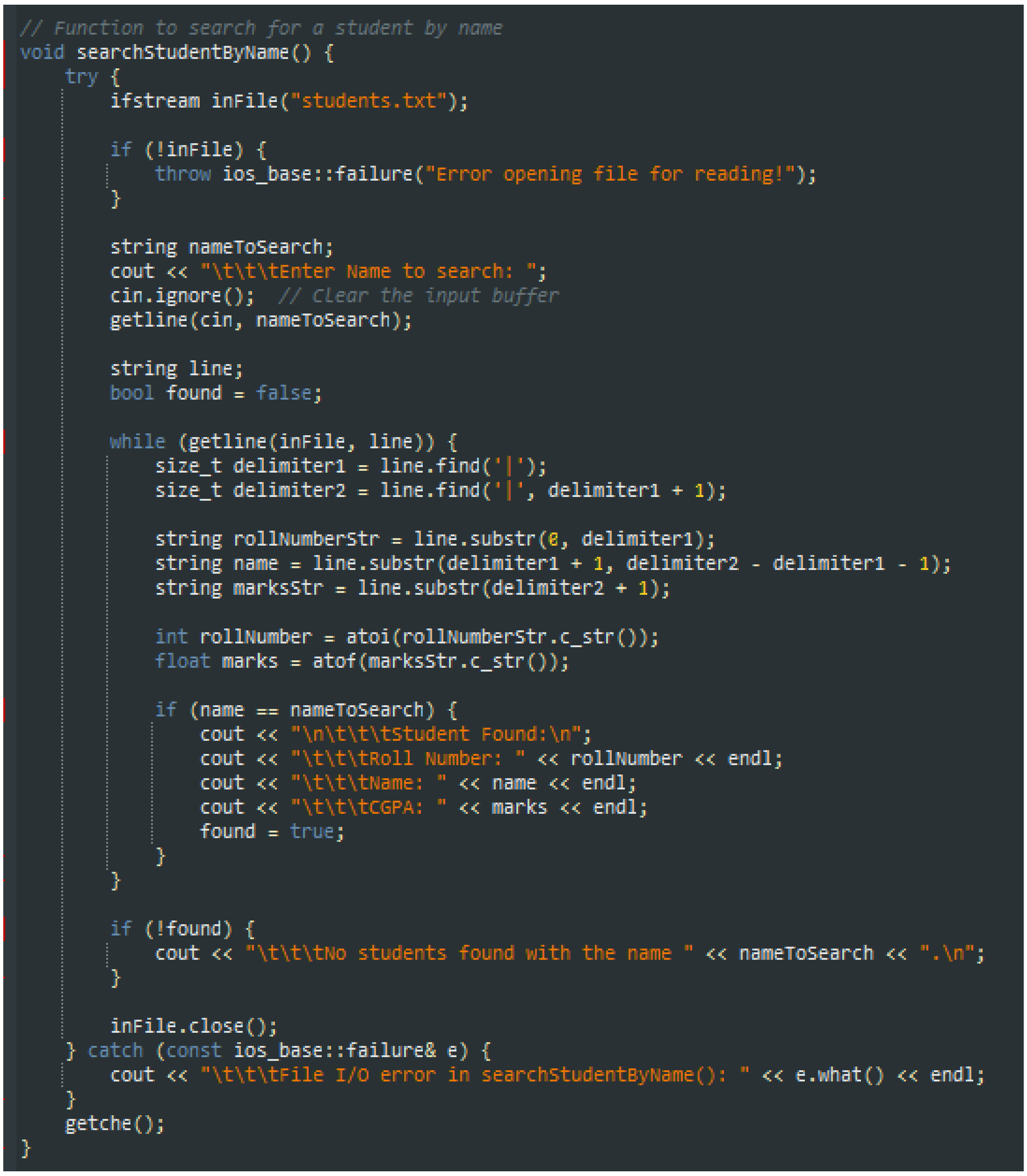


```cpp
// Function to search for a student by roll number
void searchStudentByRollNumber() {
    try {
        ifstream inFile("students.txt");

        if (!inFile) {
            throw ios_base::failure("Error opening file for reading!");
        }

        int rollNumber;
        cout << "\t\t\tEnter Roll Number to search: ";
        while (!isValidNumericInput(rollNumber)) {
            cout << "\t\t\tEnter valid Roll Number: ";
        }

        string line;
        bool found = false;

        while (getline(inFile, line)) {
            size_t delimiter1 = line.find('|');
            size_t delimiter2 = line.find('|', delimiter1 + 1);

            string rollNumberStr = line.substr(0, delimiter1);
            string name = line.substr(delimiter1 + 1, delimiter2 - delimiter1 - 1);
            string marksStr = line.substr(delimiter2 + 1);

            int fileRollNumber = atoi(rollNumberStr.c_str());
            float marks = atof(marksStr.c_str());

            if (fileRollNumber == rollNumber) {
                cout << "\n\t\t\tStudent Found:\n";
                cout << "\t\t\tRoll Number: " << fileRollNumber << endl;
                cout << "\t\t\tName: " << name << endl;
                cout << "\t\t\tCGPA: " << marks << endl;
                found = true;
                break;
            }
        }

        if (!found) {
            cout << "\t\t\tStudent with Roll Number " << rollNumber << " not found.\n";
        }

        inFile.close();
    } catch (const ios_base::failure& e) {
        cout << "\t\t\tFile I/O error in searchStudentByRollNumber(): " << e.what() << endl;
    }
    getche();
}
```

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.

4.4 Search Student by Name

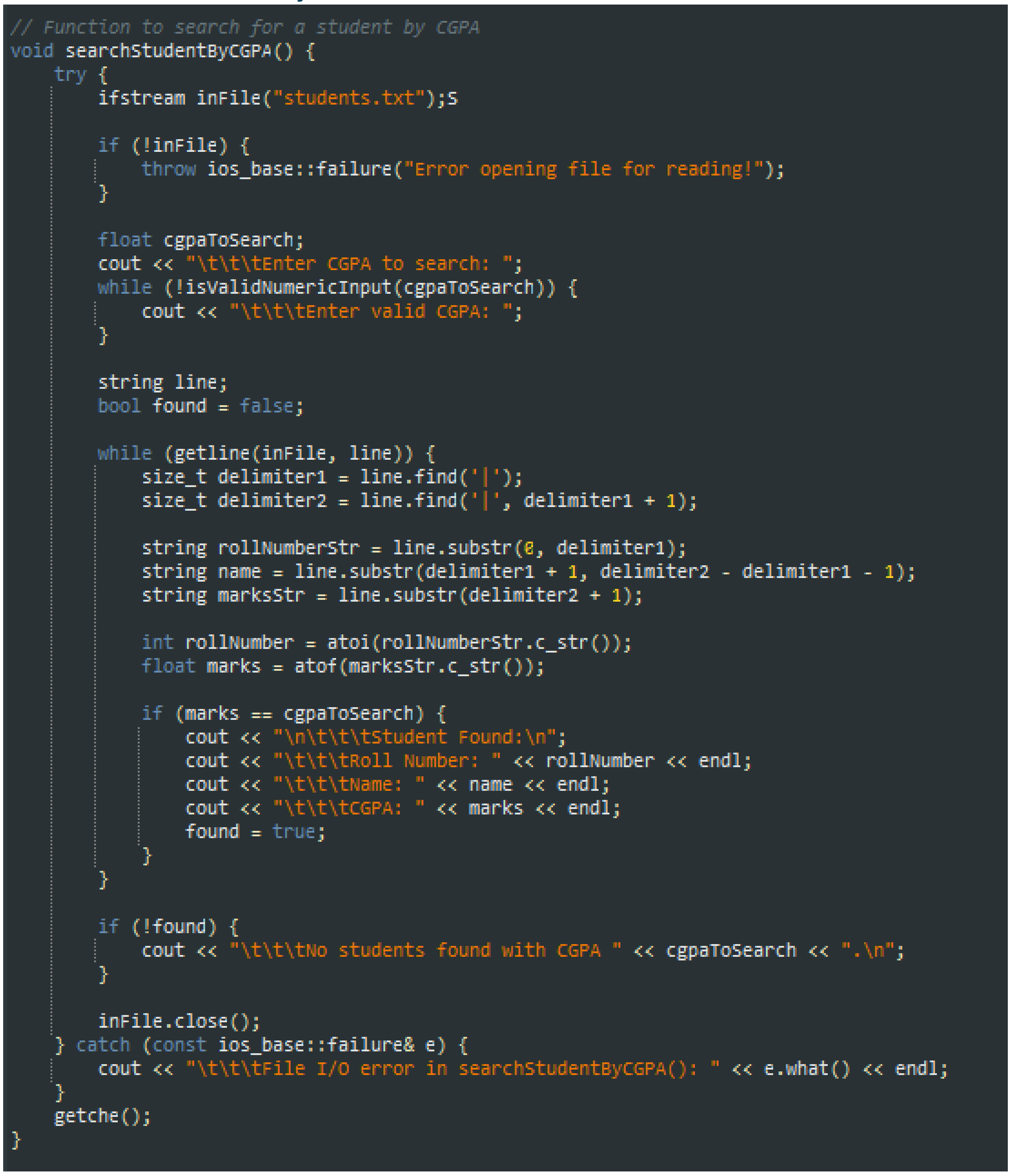


```cpp
// Function to search for a student by name
void searchStudentByName() {
    try {
        ifstream inFile("students.txt");

        if (!inFile) {
            throw ios_base::failure("Error opening file for reading!");
        }

        string nameToSearch;
        cout << "\t\t\tEnter Name to search: ";
        cin.ignore();  // Clear the input buffer
        getline(cin, nameToSearch);

        string line;
        bool found = false;

        while (getline(inFile, line)) {
            size_t delimiter1 = line.find('|');
            size_t delimiter2 = line.find('|', delimiter1 + 1);

            string rollNumberStr = line.substr(0, delimiter1);
            string name = line.substr(delimiter1 + 1, delimiter2 - delimiter1 - 1);
            string marksStr = line.substr(delimiter2 + 1);

            int rollNumber = atoi(rollNumberStr.c_str());
            float marks = atof(marksStr.c_str());

            if (name == nameToSearch) {
                cout << "\n\t\t\tStudent Found:\n";
                cout << "\t\t\tRoll Number: " << rollNumber << endl;
                cout << "\t\t\tName: " << name << endl;
                cout << "\t\t\tCGPA: " << marks << endl;
                found = true;
            }
        }

        if (!found) {
            cout << "\t\t\tNo students found with the name " << nameToSearch << ".\n";
        }

        inFile.close();
    } catch (const ios_base::failure& e) {
        cout << "\t\t\tFile I/O error in searchStudentByName(): " << e.what() << endl;
    }
    getche();
}
```

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.

4.5 *Search Student by CGPA*



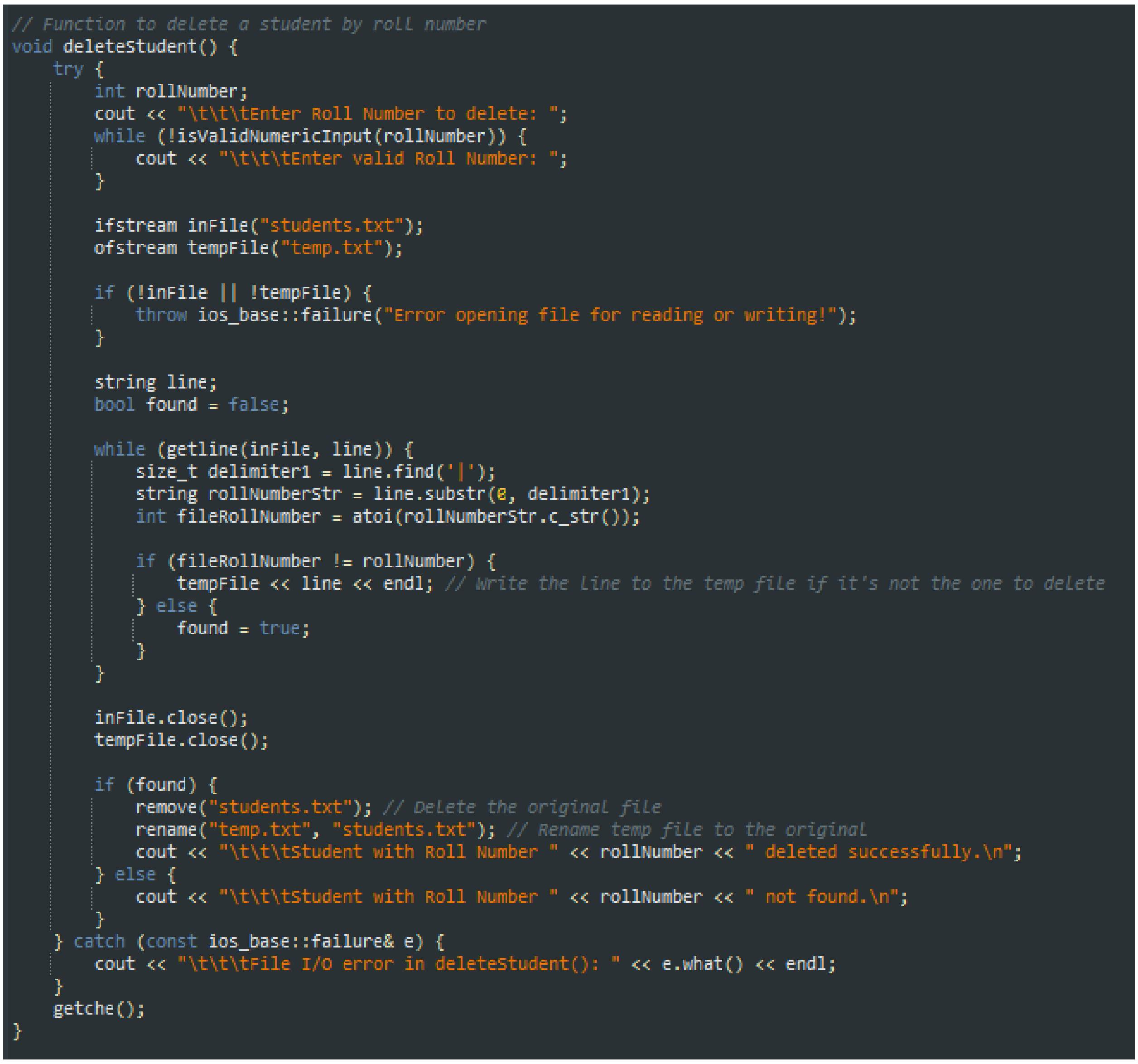
```cpp
// Function to search for a student by CGPA
void searchStudentByCGPA() {
    try {
        ifstream inFile("students.txt");S

        if (!inFile) {
            throw ios_base::failure("Error opening file for reading!");
        }

        float cgpaToSearch;
        cout << "\t\t\tEnter CGPA to search: ";
        while (!isValidNumericInput(cgpaToSearch)) {
            cout << "\t\t\tEnter valid CGPA: ";
        }

        string line;
        bool found = false;

        while (getline(inFile, line)) {
            size_t delimiter1 = line.find('|');
            size_t delimiter2 = line.find('|', delimiter1 + 1);

            string rollNumberStr = line.substr(0, delimiter1);
            string name = line.substr(delimiter1 + 1, delimiter2 - delimiter1 - 1);
            string marksStr = line.substr(delimiter2 + 1);

            int rollNumber = atoi(rollNumberStr.c_str());
            float marks = atof(marksStr.c_str());

            if (marks == cgpaToSearch) {
                cout << "\n\t\t\tStudent Found:\n";
                cout << "\t\t\tRoll Number: " << rollNumber << endl;
                cout << "\t\t\tName: " << name << endl;
                cout << "\t\t\tCGPA: " << marks << endl;
                found = true;
            }
        }

        if (!found) {
            cout << "\t\t\tNo students found with CGPA " << cgpaToSearch << ".\n";
        }

        inFile.close();
    } catch (const ios_base::failure& e) {
        cout << "\t\t\tFile I/O error in searchStudentByCGPA(): " << e.what() << endl;
    }
    getche();
}
```

Description:

- Attempts to open students.txt for reading. If it fails, an error is thrown.
- Prompts the user to enter a CGPA and validates the input.
- Processes each line in the file, extracting the roll number, name, and CGPA.
- Compares the extracted CGPA with the user-provided CGPA.

- If a match is found, displays the student's details (roll number, name, CGPA).
- If no match is found, informs the user that no students were found with the given CGPA.
- Catches file I/O errors and prints an appropriate message.
- Waits for the user to press a key before finishing.

4.6 Delete Student

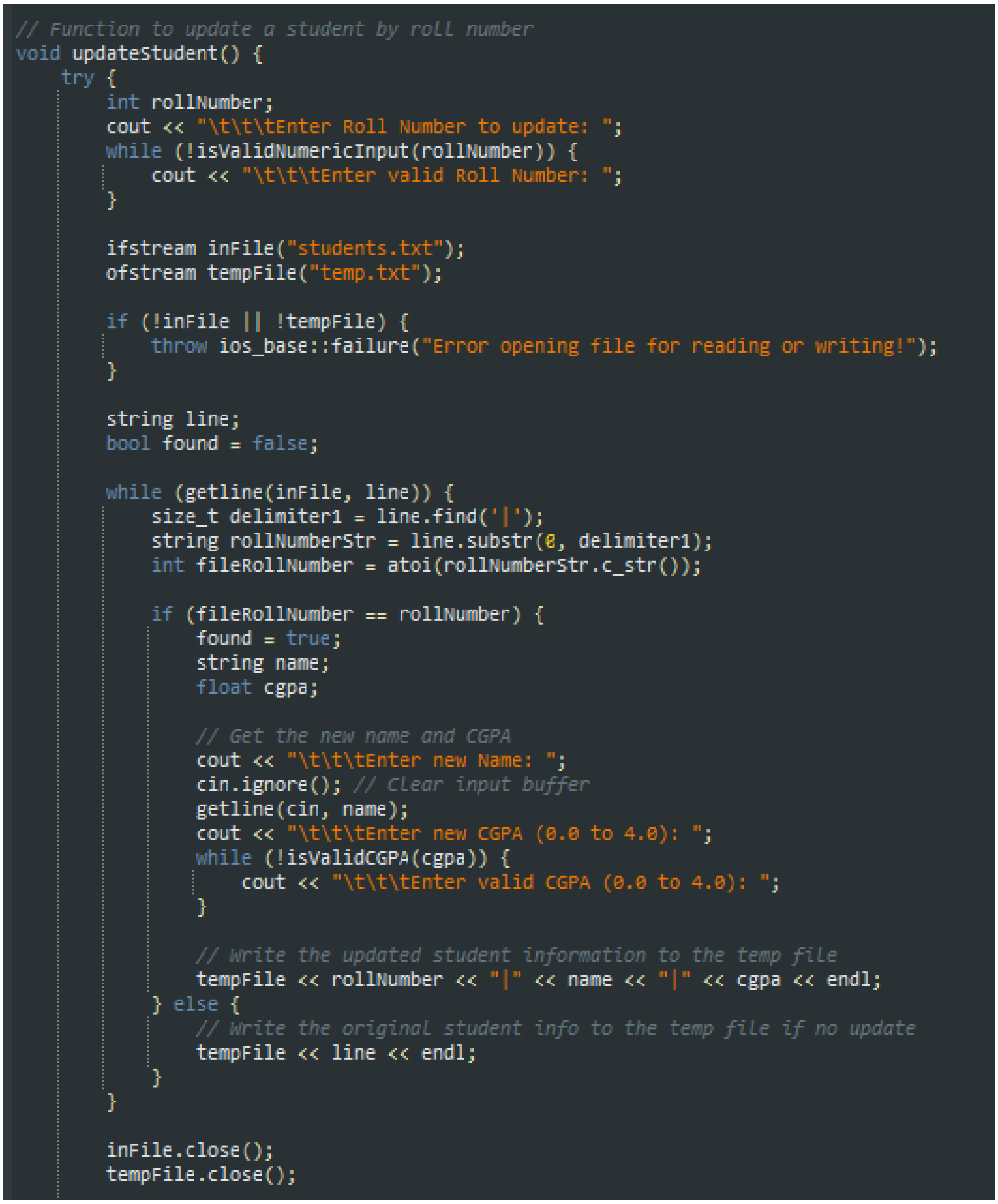


```cpp
// Function to delete a student by roll number
void deleteStudent() {
    try {
        int rollNumber;
        cout << "\t\t\tEnter Roll Number to delete: ";
        while (!isValidNumericInput(rollNumber)) {
            cout << "\t\t\tEnter valid Roll Number: ";
        }

        ifstream inFile("students.txt");
        ofstream tempFile("temp.txt");

        if (!inFile || !tempFile) {
            throw ios_base::failure("Error opening file for reading or writing!");
        }

        string line;
        bool found = false;

        while (getline(inFile, line)) {
            size_t delimiter1 = line.find('|');
            string rollNumberStr = line.substr(0, delimiter1);
            int fileRollNumber = atoi(rollNumberStr.c_str());

            if (fileRollNumber != rollNumber) {
                tempFile << line << endl; // Write the line to the temp file if it's not the one to delete
            } else {
                found = true;
            }
        }

        inFile.close();
        tempFile.close();

        if (found) {
            remove("students.txt"); // Delete the original file
            rename("temp.txt", "students.txt"); // Rename temp file to the original
            cout << "\t\t\tStudent with Roll Number " << rollNumber << " deleted successfully.\n";
        } else {
            cout << "\t\t\tStudent with Roll Number " << rollNumber << " not found.\n";
        }
    } catch (const ios_base::failure& e) {
        cout << "\t\t\tFile I/O error in deleteStudent(): " << e.what() << endl;
    }
    getche();
}
```

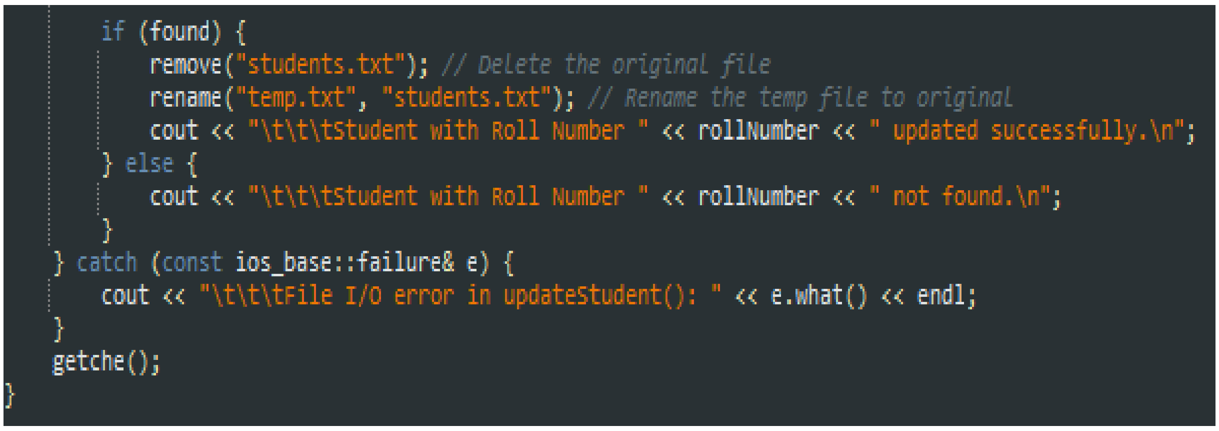
Description:

- 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.

4.7 Update Student



```cpp
// Function to update a student by roll number
void updateStudent() {
    try {
        int rollNumber;
        cout << "\t\t\tEnter Roll Number to update: ";
        while (!isValidNumericInput(rollNumber)) {
            cout << "\t\t\tEnter valid Roll Number: ";
        }

        ifstream inFile("students.txt");
        ofstream tempFile("temp.txt");

        if (!inFile || !tempFile) {
            throw ios_base::failure("Error opening file for reading or writing!");
        }

        string line;
        bool found = false;

        while (getline(inFile, line)) {
            size_t delimiter1 = line.find('|');
            string rollNumberStr = line.substr(0, delimiter1);
            int fileRollNumber = atoi(rollNumberStr.c_str());

            if (fileRollNumber == rollNumber) {
                found = true;
                string name;
                float cgpa;

                // Get the new name and CGPA
                cout << "\t\t\tEnter new Name: ";
                cin.ignore(); // Clear input buffer
                getline(cin, name);
                cout << "\t\t\tEnter new CGPA (0.0 to 4.0): ";
                while (!isValidCGPA(cgpa)) {
                    cout << "\t\t\tEnter valid CGPA (0.0 to 4.0): ";
                }

                // Write the updated student information to the temp file
                tempFile << rollNumber << "|" << name << "|" << cgpa << endl;
            } else {
                // Write the original student info to the temp file if no update
                tempFile << line << endl;
            }
        }

        inFile.close();
        tempFile.close();
```

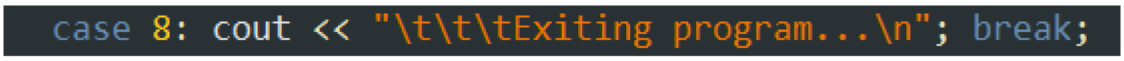


```
        if (found) {
            remove("students.txt"); // Delete the original file
            rename("temp.txt", "students.txt"); // Rename the temp file to original
            cout << "\t\t\tStudent with Roll Number " << rollNumber << " updated successfully.\n";
        } else {
            cout << "\t\t\tStudent with Roll Number " << rollNumber << " not found.\n";
        }
    } catch (const ios_base::failure& e) {
        cout << "\t\t\tFile I/O error in updateStudent(): " << e.what() << endl;
    }
    getche();
}
```

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.

4.8 Exit program



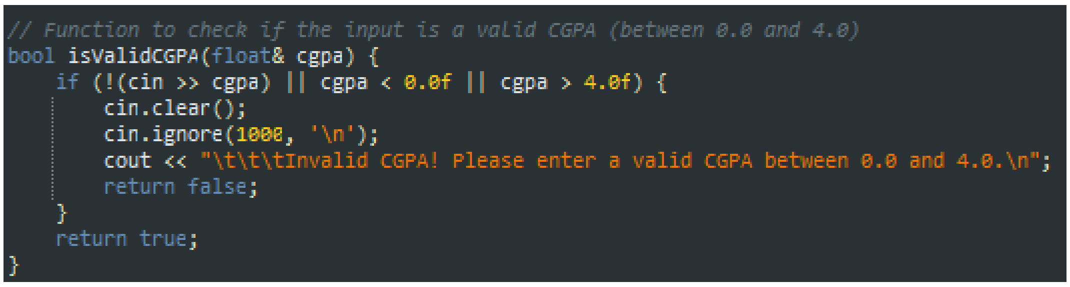
```
case 8: cout << "\t\t\tExiting program...\n"; break;
```

- This will end program.

5. *Input Validation*

Input Validation Functions

5.1 CGPA Validation

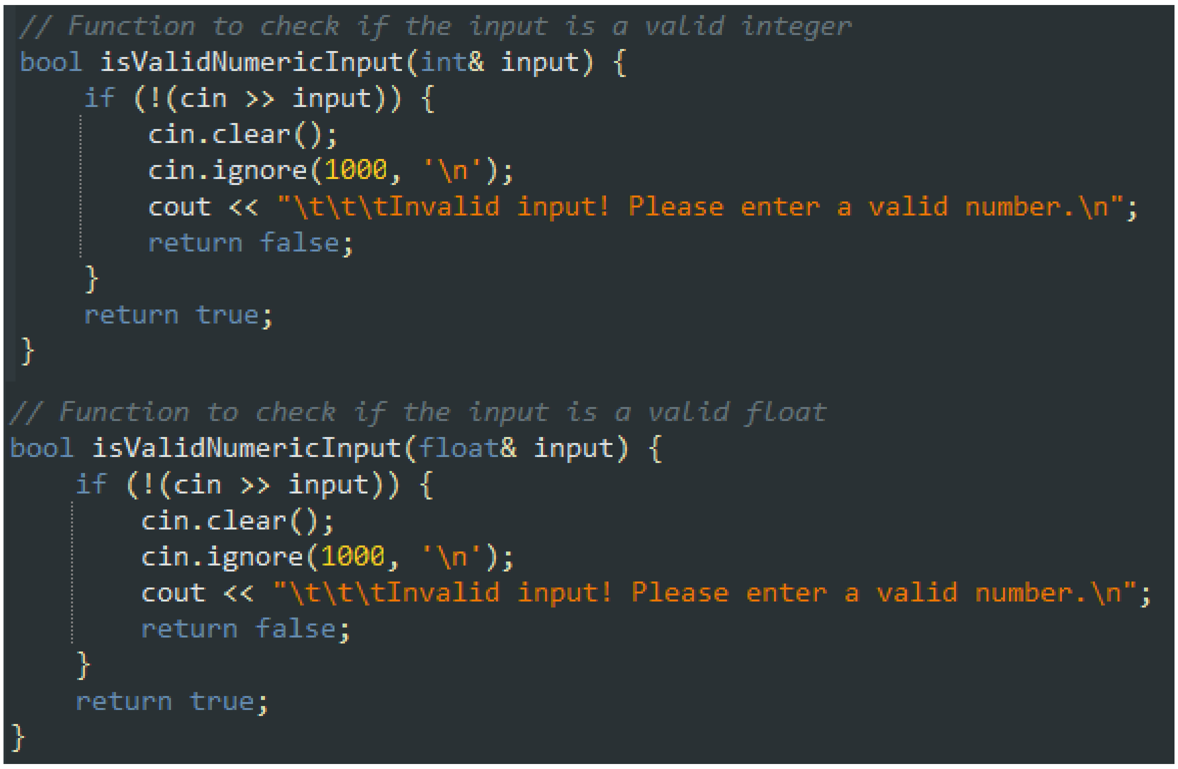


```cpp
// Function to check if the input is a valid CGPA (between 0.0 and 4.0)
bool isValidCGPA(float& cgpa) {
    if (!(cin >> cgpa) || cgpa < 0.0f || cgpa > 4.0f) {
        cin.clear();
        cin.ignore(1000, '\n');
        cout << "\t\t\tInvalid CGPA! Please enter a valid CGPA between 0.0 and 4.0.\n";
        return false;
    }
    return true;
}
```

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.

5.2 Numeric Validation

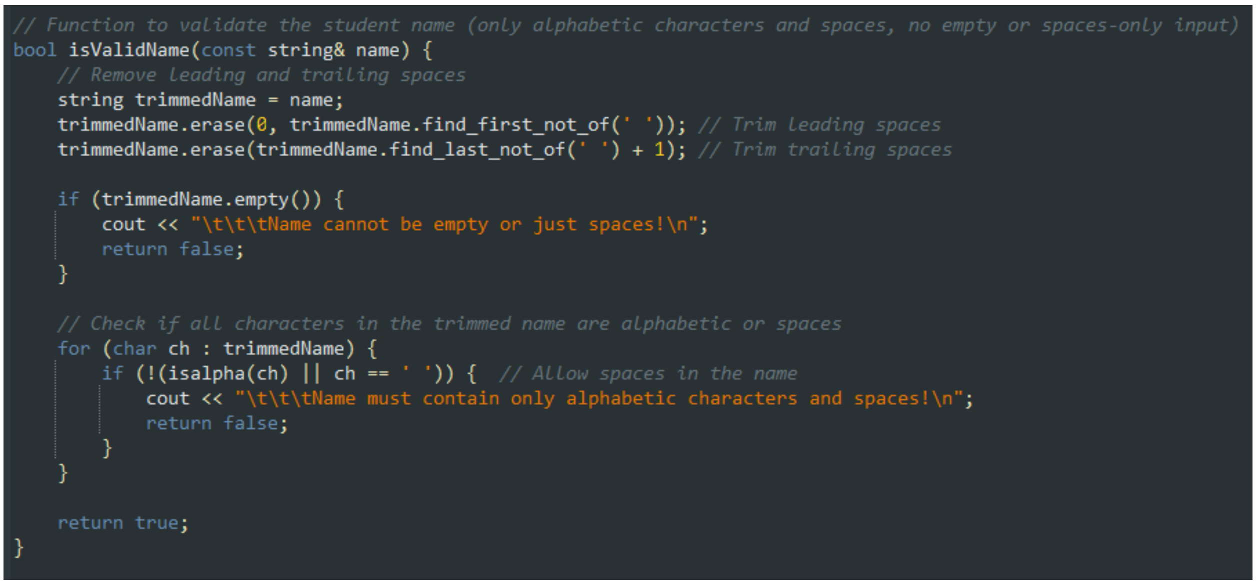


```cpp
// Function to check if the input is a valid integer
bool isValidNumericInput(int& input) {
    if (!(cin >> input)) {
        cin.clear();
        cin.ignore(1000, '\n');
        cout << "\t\t\tInvalid input! Please enter a valid number.\n";
        return false;
    }
    return true;
}

// Function to check if the input is a valid float
bool isValidNumericInput(float& input) {
    if (!(cin >> input)) {
        cin.clear();
        cin.ignore(1000, '\n');
        cout << "\t\t\tInvalid input! Please enter a valid number.\n";
        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.

5.3 Name Validation



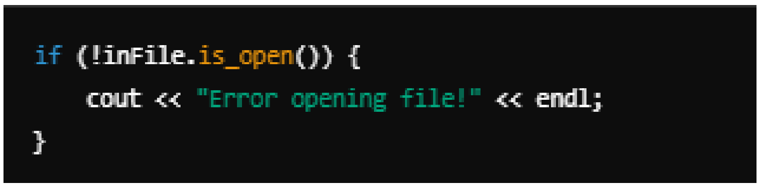
```cpp
// Function to validate the student name (only alphabetic characters and spaces, no empty or spaces-only input)
bool isValidName(const string& name) {
    // Remove leading and trailing spaces
    string trimmedName = name;
    trimmedName.erase(0, trimmedName.find_first_not_of(' ')); // Trim leading spaces
    trimmedName.erase(trimmedName.find_last_not_of(' ') + 1); // Trim trailing spaces

    if (trimmedName.empty()) {
        cout << "\t\t\tName cannot be empty or just spaces!\n";
        return false;
    }

    // Check if all characters in the trimmed name are alphabetic or spaces
    for (char ch : trimmedName) {
        if (!(isalpha(ch) || ch == ' ')) {  // Allow spaces in the name
            cout << "\t\t\tName must contain only alphabetic characters and spaces!\n";
            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

5.1 File I/O Error Handling



```cpp
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.

5.2 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

6.1 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.

6.2 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