```
#include <bits/stdc++.h>
using namespace std;

struct Student {
    int roll;
    string name, email, dept;
    double gpa;
};

class StudentSystem {
    vector<Student> list;
    string fileName = "students.txt";

    // Find student index by roll number
    int findStudent(int roll) {
        for (int i = 0; i < list.size(); i++) {
            if (list[i].roll == roll)
                return i;
        }
        return -1;
    }

    // Simple email check
    bool validEmail(string e) {
        return e.find('@') != string::npos && e.find('.') != string::npos;
    }

    // Load data from file
    void load() {
        ifstream file(fileName);
        if (!file) return;

        string line;
        while (getline(file, line)) {
            stringstream ss(line);
            string r, n, e, g, d;
            getline(ss, r, '|');
            getline(ss, n, '|');
            getline(ss, e, '|');
            getline(ss, g, '|');
            getline(ss, d, '|');

            Student s;
            s.roll = stoi(r);
```

```
            s.name = n;
            s.email = e;
            s.gpa = stod(g);
            s.dept = d;

            list.push_back(s);
        }
    }

    // Save data to file
    void save() {
        ofstream file(fileName);
        for (auto &s : list) {
            file << s.roll << "|" << s.name << "|" << s.email
                 << "|" << s.gpa << "|" << s.dept << "\n";
        }
    }

public:
    StudentSystem() {
        load();
    }

    // Add new student
    void add() {
        Student s;

        cout << "Enter Roll Number: ";
        cin >> s.roll;

        if (findStudent(s.roll) != -1) {
            cout << "Roll already exists!\n";
            return;
        }

        cin.ignore();
        cout << "Enter Name: ";
        getline(cin, s.name);

        cout << "Enter Email: ";
        getline(cin, s.email);
        if (!validEmail(s.email)) {
            cout << "Invalid email!\n";
            return;
```

```
    }

    cout << "Enter GPA (0-4): ";
    cin >> s.gpa;

    cin.ignore();
    cout << "Enter Department: ";
    getline(cin, s.dept);

    list.push_back(s);
    save();
    cout << "Student Added!\n";
}

// Display all students
void display() {
    if (list.empty()) {
        cout << "No students stored.\n";
        return;
    }

    for (auto &s : list) {
        cout << s.roll << " | " << s.name << " | "
             << s.email << " | " << s.gpa
             << " | " << s.dept << "\n";
    }
}

// Search by roll
void searchRoll() {
    int r;
    cout << "Enter Roll Number: ";
    cin >> r;

    int index = findStudent(r);
    if (index == -1) {
        cout << "Student not found.\n";
        return;
    }

    Student s = list[index];
    cout << s.roll << " | " << s.name << " | " << s.email
         << " | " << s.gpa << " | " << s.dept << "\n";
}
```

```
// Search by name
void searchName() {
    cin.ignore();
    string name;
    cout << "Enter name to search: ";
    getline(cin, name);

    bool found = false;
    for (auto &s : list) {
        if (s.name.find(name) != string::npos) {
            cout << s.roll << " | " << s.name << " | "
                 << s.email << " | " << s.gpa << " | " << s.dept << "\n";
            found = true;
        }
    }
    if (!found) cout << "No student found.\n";
}

// Update student
void update() {
    int r;
    cout << "Enter Roll Number to update: ";
    cin >> r;

    int index = findStudent(r);
    if (index == -1) {
        cout << "Not found.\n";
        return;
    }

    Student &s = list[index];
    cin.ignore();

    cout << "New Name: ";
    getline(cin, s.name);

    cout << "New Email: ";
    getline(cin, s.email);

    cout << "New GPA: ";
    cin >> s.gpa;

    cin.ignore();
```

```
    cout << "New Department: ";
    getline(cin, s.dept);

    save();
    cout << "Updated!\n";
}

// Delete
void removeStudent() {
    int r;
    cout << "Roll Number to delete: ";
    cin >> r;

    int index = findStudent(r);
    if (index == -1) {
        cout << "Not found.\n";
        return;
    }

    list.erase(list.begin() + index);
    save();
    cout << "Deleted!\n";
}

// Show statistics
void stats() {
    if (list.empty()) return void(cout << "No data.\n");

    double sum = 0, maxG = list[0].gpa, minG = list[0].gpa;

    for (auto &s : list) {
        sum += s.gpa;
        maxG = max(maxG, s.gpa);
        minG = min(minG, s.gpa);
    }

    cout << "Total Students: " << list.size() << "\n";
    cout << "Average GPA: " << sum / list.size() << "\n";
    cout << "Highest GPA: " << maxG << "\n";
    cout << "Lowest GPA: " << minG << "\n";
}

// Sort by GPA
void sortGPA() {
```

```
        sort(list.begin(), list.end(), [](Student a, Student b) {
            return a.gpa > b.gpa;
        });
        display();
    }
};

int main() {
    StudentSystem sys;
    int choice;

    while (true) {
        cout << "\n1.Add\n2.Display\n3.Search Roll\n4.Search
Name\n5.Update\n6.Delete\n7.Statistics\n8.Sort GPA\n9.Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1: sys.add(); break;
            case 2: sys.display(); break;
            case 3: sys.searchRoll(); break;
            case 4: sys.searchName(); break;
            case 5: sys.update(); break;
            case 6: sys.removeStudent(); break;
            case 7: sys.stats(); break;
            case 8: sys.sortGPA(); break;
            case 9: return 0;
            default: cout << "Invalid choice!\n";
        }
    }
}
```