Đánh giá và cải tiến thiết kế

1. Phụ thuộc quá nhiều vào database/persistent storage

Nhận xét:

- Trong các unit test, hàm addStudent() và updateStudent() vẫn thực hiện lưu dữ liệu vào CSV/JSON (saveStudentToCSV() và exportStudenToJSON()).
- Điều này không phù hợp với unit test, vì:
    + Chỉ nên kiểm tra logic, không nên phụ thuộc vào I/O như đọc ghi file.
    + Nếu file không tồn tại hoặc bị lỗi quyền truy cập, test có thể thất bại vì lý do không liên quan đến logic.
    + Tốc độ test bị chậm do ghi file.

Giải pháp:

Dùng mock/stub thay vì ghi vào file. VD:

```
void saveStudentToCSV(const std::string&) {} // Mock để test không ghi file void exportStudentToJSON(const std::string&) {} // Mock

 void logEvent(const std::string&) {} // Mock
```

2. Module có quá nhiều trách nhiệm:

Nhận xét:

- Một số module như addStudent() và updateStudent() ở Version 1.0 và 2.0 vừa xử lý logic, vừa tương tác với người dùng (cin/cout)
- Vấn đề:
    + Code không tuân theo nguyên tắc Single Responsibility Principle (SRP).
    + Khó test logic riêng lẻ vì bị phụ thuộc vào cin/cout.

+ Khi cần thay đổi giao diện nhập liệu (ví dụ từ CLI sang giao diện đồ họa), phải sửa lại toàn bộ hàm.

Giải pháp:

Tách thành 2 hàm con, 1 hàm interact với người dùng và tiếp nhận dữ liệu, 1 hàm xử lý dữ liệu được truyền vào:

Ví dụ:

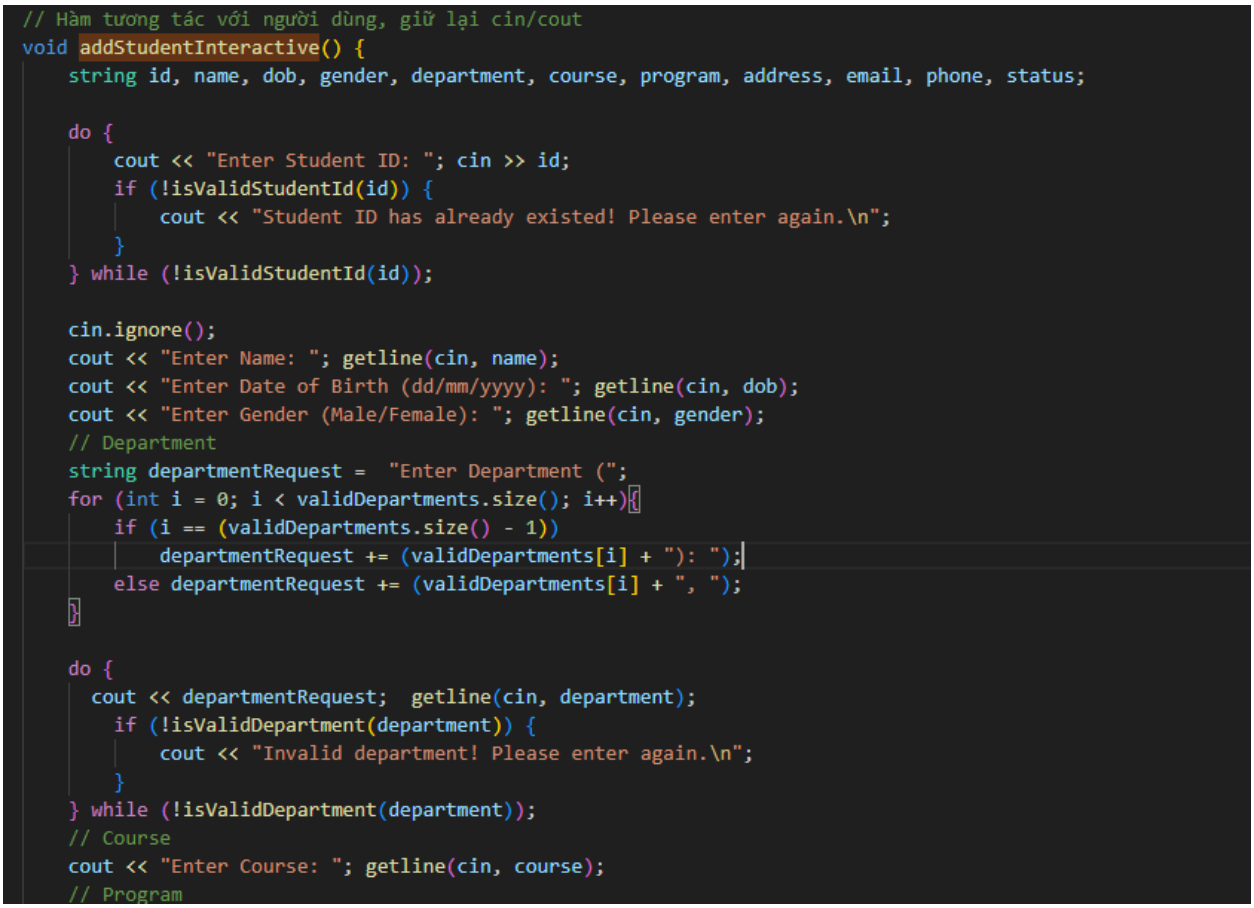
```
// Hàm xử lý logic thêm sinh viên, không có cin/cout (dễ test)
bool addStudent(const string& id, const string& name, const string& dob, const string& gender,
                const string& department, const string& course, const string& program,
                const string& address, const string& email, const string& phone,
                const string& status) {

    if (!isValidStudentId(id)) return false;
    if (!isValidDepartment(department)) return false;
    if (!isValidPrograms(program)) return false;
    if (!isValidEmail(email)) return false;
    if (!isValidPhone(phone)) return false;
    if (!isValidStatus(status)) return false;

    students.emplace_back(id, name, dob, gender, department, course, program, address, email, phone, status);

    saveStudentToCSV("students.csv");
    exportStudentToJSON("student.json");

    // Log event
    logEvent("Added new student: ID = " + id + ", Name = " + name);
    return true;
}
```



```
// Hàm tương tác với người dùng, giữ lại cin/cout
void addStudentInteractive() {
    string id, name, dob, gender, department, course, program, address, email, phone, status;

    do {
        cout << "Enter Student ID: "; cin >> id;
        if (!isValidStudentId(id)) {
            cout << "Student ID has already existed! Please enter again.\n";
        }
    } while (!isValidStudentId(id));

    cin.ignore();
    cout << "Enter Name: "; getline(cin, name);
    cout << "Enter Date of Birth (dd/mm/yyyy): "; getline(cin, dob);
    cout << "Enter Gender (Male/Female): "; getline(cin, gender);
    // Department
    string departmentRequest =  "Enter Department (";
    for (int i = 0; i < validDepartments.size(); i++){
        if (i == (validDepartments.size() - 1))
            departmentRequest += (validDepartments[i] + "): ");
        else departmentRequest += (validDepartments[i] + ", ");
    }

    do {
      cout << departmentRequest;  getline(cin, department);
        if (!isValidDepartment(department)) {
            cout << "Invalid department! Please enter again.\n";
        }
    } while (!isValidDepartment(department));
    // Course
    cout << "Enter Course: "; getline(cin, course);
    // Program
```

3. Không test được business logic quan trọng:

Nhận xét:

– Một số quy tắc nghiệp vụ (business logic) chưa được test rõ ràng, ví dụ:

+ Cập nhật trạng thái sinh viên: Chỉ có 1 test isValidUpdateStatus(), nhưng chưa kiểm tra kỹ tất cả các trạng thái hợp lệ.
+ Tránh trùng ID: Test AddStudent_DuplicateID chưa đảm bảo ID trùng được kiểm tra đúng ở mọi trường hợp.
+ Kiểm tra đầu vào sai: Các test về isValidEmail(), isValidPhone(), isValidDepartment() chưa bao phủ nhiều trường hợp nhập sai.

Giải pháp:

+ Tạo nhiều test case hơn để bao quát tất cả các trường hợp.
+ Dùng parameterized test của Google Test để kiểm tra nhiều dữ liệu đầu vào nhanh chóng.

Ví dụ cải tiến test cập nhật trạng thái sinh viên:

```
TEST(StudentTest, UpdateStudentStatus) {
   students.clear();
   students.emplace_back("S12345", "Alice", "01/01/2000", "Female",
                  "IT", "2024", "Software Engineering",
                  "123 Street", "alice@clc.fitus.edu.vn", "0123456789", "Dang hoc");

   EXPECT_TRUE(updateStudent("S12345", 11, "Bao luu")); // Hợp lệ
   EXPECT_FALSE(updateStudent("S12345", 11, "Dang hoc cao hoc")); // Không hợp lệ
   EXPECT_FALSE(updateStudent("S12345", 11, "Expelled")); // Không hợp lệ
}
```