

ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA KHOA HỌC VÀ CÔNG NGHỆ GIÁO DỤC



BÁO CÁO MÔN CẤU TRÚC DỮ LIỆU G.THUẬT
ĐỀ TÀI:
QUẢN LÝ KẾT QUẢ HỌC TẬP SINH VIÊN

Sinh viên thực hiện:

Đinh Mạnh Tiến - 20231638

Hoàng Hoài Thương - 20231636

Trần Phương Anh - 20231558

Hoàng Thị Thảo - 20231630

Giảng viên hướng dẫn:

Nguyễn Việt Tùng

Hà Nội, 2024

PHÂN CÔNG CÔNG VIỆC

Chức năng 1, 2, 3,... ứng với yêu cầu 1, 2, 3,... theo đề bài

STT	Họ tên	MSSV	Phân công chi tiết	
1	Đinh Mạnh Tiến (Nhóm trưởng)	20231638	Làm báo cáo	- Phần I. Tóm tắt đề tài, II. Giới thiệu đề tài - Phần IV. Phân tích và thiết kế ứng dụng - Phần VII. Kết luận
			Code	- Chức năng 3: Thay đổi CPA của sinh viên - Chức năng 4: Tìm n sinh viên CPA cao nhất - File chương trình chính: file main.js
2	Hoàng Hoài Thương	20231636	Làm báo cáo	- Phần II. Giới thiệu đề tài - Phần III. Phương pháp lựa chọn - Phần IV. Phân tích và thiết kế ứng dụng
			Code	- Chức năng 1: In danh sách sinh viên đang học - Chức năng 2: Tìm sinh viên theo MSSV
3	Trần Phương Anh	20231558	Làm báo cáo	- Phần IV. Phân tích và thiết kế ứng dụng - Phần V. Triển khai cài đặt và đóng gói - Phần VI. Kết quả thực nghiệm
			Code	- Chức năng 7: Đếm số sinh viên có CPA trong đoạn [a, b] - Chức năng 8: Tính số lượng sinh viên đình chỉ học
4	Hoàng Thị Thảo	20231630	Làm báo cáo	- Phần IV. Phân tích và thiết kế ứng dụng - Phần VI. Kết quả thực nghiệm - Phần VIII. Danh mục tài liệu tham khảo
			Code	- Chức năng 5: Tìm n sinh viên CPA thấp nhất - Chức năng 6: Tìm các sinh viên đang bị cảnh cáo, kèm mức cảnh cáo - Chức năng 9: Phân loại học lực theo điểm

MỤC LỤC

PHẦN I: TÓM TẮT ĐỀ TÀI	
PHẦN II: GIỚI THIỆU ĐỀ TÀI	
2.1. Lý do chọn đề tài.....	
2.2. Mục tiêu đề tài	
2.3. Tổng quan bài toán.....	
2.4. Kết cấu đề tài.....	
PHẦN III: PHƯƠNG PHÁP LỰA CHỌN.....	
PHẦN IV: PHÂN TÍCH VÀ THIẾT KẾ ỨNG DỤNG.....	
4.1. Tổng quan chức năng	
4.2. Phân tích và thiết kế.....	
4.2.1. Chức năng 1	
4.2.2. Chức năng 2	
4.2.3. Chức năng 3	
4.2.4. Chức năng 4	
4.2.5. Chức năng 5	
4.2.6. Chức năng 6	
4.2.7. Chức năng 7	
4.2.8. Chức năng 8	
4.2.9. Chức năng 9 (Tự phát triển)	
PHẦN V: TRIỂN KHAI CÀI ĐẶT VÀ ĐÓNG GÓI	
PHẦN VI: KẾT QUẢ THỰC NGHIỆM	
6.1. Bộ dữ liệu.....	
6.2. Kết quả thực nghiệm	
PHẦN VII: KẾT LUẬN	
7.1. Đánh giá mức độ hoàn thành	
7.2. Kiến nghị, giải pháp.....	
PHẦN VII. DANH MỤC TÀI LIỆU THAM KHẢO	

PHẦN I. TÓM TẮT ĐỀ TÀI

Xã hội ngày càng hiện đại và phát triển, tin học đang ngày càng có nhiều ứng dụng trong đời sống, từ lĩnh vực: khoa học, y tế, xây dựng,... và đặc biệt là giáo dục. Trong ngành giáo dục, việc xây dựng một hệ thống quản lý sinh viên là điều rất cần thiết, bởi nó giúp tiết kiệm thời gian, công sức và đầy mạnh hiệu quả quản lý so với việc quản lý điểm qua sổ điểm, viết tay truyền thống.

Đề tài: “**Quản lý kết quả học tập của sinh viên Đại học Bách Khoa Hà Nội**” được nhóm chúng em lựa chọn, trong khoảng 1 tháng vừa phân tích, vừa thiết kế, nghiên cứu, nhóm đã áp dụng các kiến thức đã học để hoàn thiện đề tài này một cách chính chu nhất, với 9 chức năng quản lý kết quả sinh viên.

Để hoàn thành bài tập lớn lần này, nhóm chúng em xin được gửi lời cảm ơn chân thành đến thầy **Nguyễn Việt Tùng** đã giúp đỡ trong quá trình làm đề tài này.

PHẦN II. GIỚI THIỆU ĐỀ TÀI

2.1 Lý do chọn đề tài:

- Với mục tiêu môn học Cấu trúc dữ liệu và giải thuật là nắm được cách phân tích, xây dựng, thiết kế một chương trình phần mềm để ứng dụng giải quyết bài toán thực tế, cùng với bài toán: **“Quản lý kết quả học tập của sinh viên Đại học Bách Khoa Hà Nội”** là 1 đề tài vô cùng thiết thực:

+ **Tính thực tiễn cao:** Hệ thống quản lý điểm có ứng dụng trực tiếp trong cuộc sống, bởi đây là hoạt động không thể thiếu trong trường học khi lượng thông tin về kết quả sinh viên lớn và xu hướng áp dụng công nghệ vào giáo dục.

+ **Giải quyết vấn đề thực tế:** Quản lý điểm thủ công thường gặp nhiều hạn chế như sai sót, mất thời gian, khó truy xuất dữ liệu nên việc xây dựng hệ thống quản lý điểm tự động hóa sẽ giúp khắc phục những khó khăn này.

+ **Áp dụng vào kiến thức đã học:** Xây dựng hệ thống dựa trên các kiến thức về ngôn ngữ lập trình, xây dựng thuật toán, dữ liệu giải thuật,... đây đều là các kiến thức sinh viên đã được học.

+ **Tính khả thi và phát triển:** Đây là vấn đề cần thiết trong xã hội, phù hợp với thời gian và nhân lực cũng như có khả năng mở rộng, phát triển như tích hợp các tính năng mới, ứng dụng cho môn học khác,...

- Ngoài các tiêu chí trên thì đề tài này là một chủ đề mở, có rất nhiều hướng phát triển, nên chúng em đã lựa chọn đề tài này.

2.2 Mục tiêu đề tài:

- Mục tiêu chính của đề tài này là xây dựng một hệ thống quản lý điểm sinh viên giúp tự động hóa, tối ưu hóa công tác lưu trữ và tra cứu điểm số, đồng thời hỗ trợ công tác quản lý và phân tích kết quả học tập một cách dễ dàng và chính xác, cụ thể như sau:

- + Nâng cao hiệu quả quản lý kết quả sinh viên trong quản lý thông tin, điểm số
- + Tự động hóa quá trình nhập liệu và tính toán
- + Dễ dàng truy xuất và cập nhật dữ liệu
- + Tiết kiệm và tối ưu thời gian, công sức

2.3 Tổng quan bài toán:

- Input:

a) Thông tin sinh viên:

- + Mã số sinh viên
- + Họ và tên
- + CPA
- + Mức cảnh báo

b) Các lệnh nhập vào từ người dùng qua giao diện dòng lệnh.

- Output: Những kết quả xử lý:

- + Danh sách sinh viên đang theo học, theo thứ tự lưu trữ
- + Sinh viên được tìm kiếm
- + Thông tin sinh viên sau khi thay đổi CPA
- + Danh sách sinh viên có CPA cao nhất
- + Danh sách sinh viên có CPA thấp nhất
- + Danh sách sinh viên đang bị cảnh cáo kèm mức cảnh cáo
- + Số lượng sinh viên bị đình chỉ học

2.4 Kết cấu đề tài:

- Bản báo cáo đề tài của nhóm chúng em gồm 8 phần:
 - + Phần I. Tóm tắt đề tài
 - + Phần II. Giới thiệu đề tài
 - + Phần III. Phương pháp lựa chọn
 - + Phần IV. Phân tích và thiết kế bài toán
 - + Phần V. Triển khai cài đặt và đóng gói
 - + Phần VI. Kết quả thực nghiệm
 - + Phần VII. Kết luận
 - + Phần VIII. Danh mục tài liệu tham khảo

PHẦN III. PHƯƠNG PHÁP LỰA CHỌN

3.1 Nền tảng sử dụng:

- **Ngôn ngữ lập trình:** Nhóm chúng em sử dụng ngôn ngữ Javascript (JS)

+ JavaScript được tạo ra bởi kỹ sư máy tính Brendan Eich và cộng sự tại công ty Netscape, xuất hiện lần đầu vào tháng 5 năm 1995.

+ JavaScript có những đặc điểm như:

- **Tính đơn giản:** JavaScript có cấu trúc đơn giản, dễ học và triển khai.
- **Tính phổ biến:** JavaScript là một trong những ngôn ngữ lập trình phổ biến trên thế giới. JS thực thi các tập lệnh trực tiếp trong trình duyệt web, giúp tăng tốc độ tải trang.
- **Tính linh hoạt:** JavaScript tương thích với các ngôn ngữ khác và có thể được sử dụng trong nhiều loại ứng dụng.
- **Hướng đối tượng:** JavaScript hỗ trợ lập trình hướng đối tượng, cho phép tạo ra các đối tượng và lớp để tổ chức mã một cách hiệu quả.
- **Hỗ trợ đa nền tảng và dễ dàng tương thích:** Khác với đa số ngôn ngữ lập trình hiện nay, JavaScript có thể được chèn vào mọi trang web cũng như sử dụng với các khung phát triển web và ngôn ngữ khác nhau. Ngoài ra, với Node.js, JavaScript còn có thể chạy trên server

3.2 Thư viện:

a) *fs (File System)*

- **Mô tả:** fs là một module (mô-đun) được tích hợp sẵn trong Node.js, cung cấp một bộ các phương thức (methods) cho phép các nhà phát triển tương tác trực tiếp với hệ thống tệp tin của máy tính.

- **Chức năng chính:** Đọc và ghi dữ liệu vào tệp tin, như việc đọc dataStudent.json và lưu lại sau khi thay đổi dữ liệu.

- **Các phương thức:** readFile (Đọc tệp), writeFile(Ghi tệp), stat(Lấy thông tin về tệp),..

b) *prompt-sync:*

- **Mô tả:** Thư viện này giúp tạo giao diện dòng lệnh đồng bộ (synchronous) để nhận đầu vào từ người dùng. Nó được sử dụng để yêu cầu người dùng nhập các lệnh trong chương trình.

- **Chức năng chính:** Cung cấp một hàm đồng bộ để nhập dữ liệu từ bàn phím.

- **Các phương thức:** Cung cấp một phương thức chính để lấy đầu vào từ người dùng thông qua câu lệnh prompt('message');

PHẦN IV. PHÂN TÍCH VÀ THIẾT KẾ ỨNG DỤNG

4.1 Tổng quan chức năng:

- Sơ đồ khối chương trình:

Hình 1. Sơ đồ khối chương trình

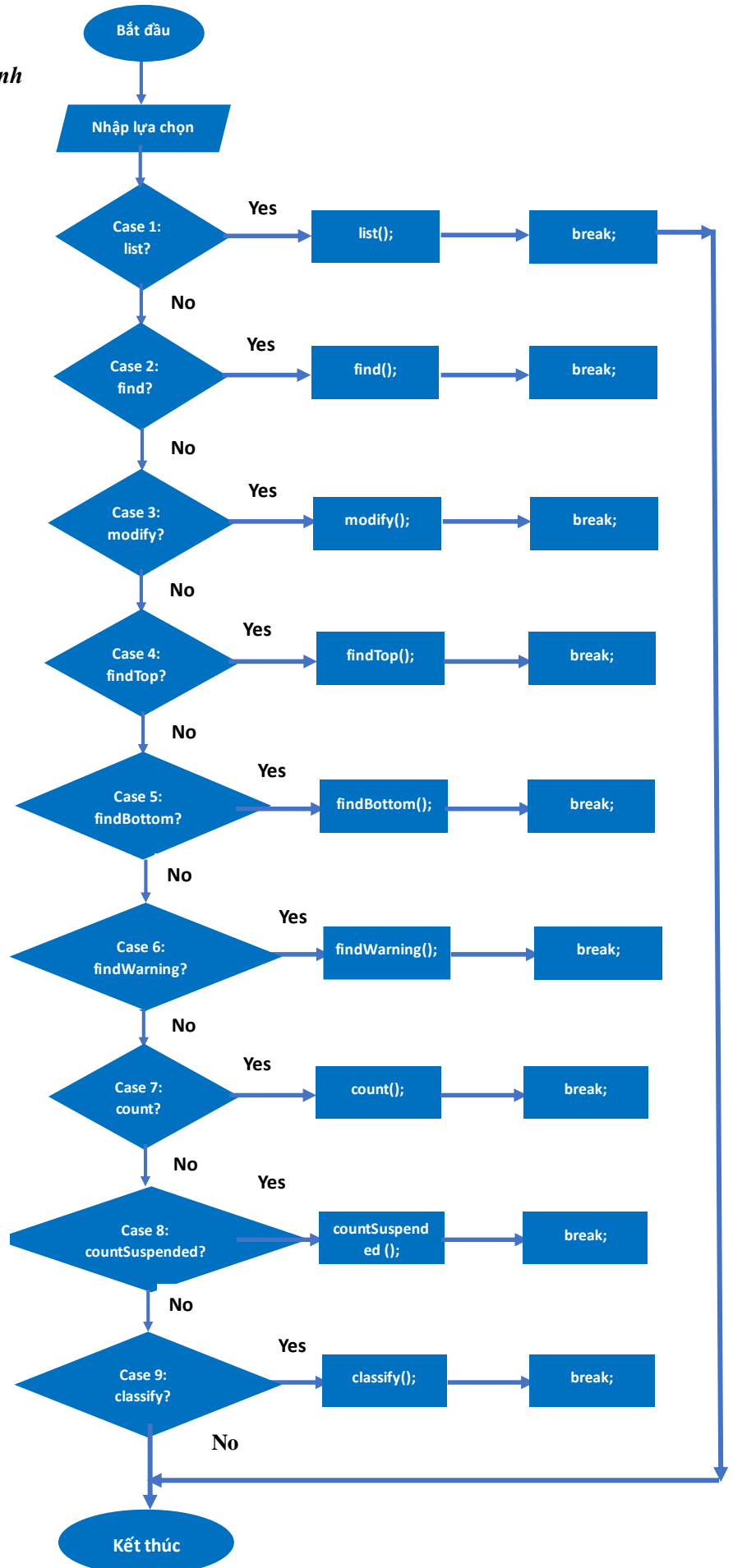
a) Ý tưởng:

- Với đề bài cho dataStudent gồm các thông tin như: Họ tên sinh viên, MSSV, CPA, Mức cảnh cáo,..., yêu cầu quản lý kết quả học tập của sinh viên ĐHBK Hà Nội, giới hạn chỉ xét trong 01 kì (kì 2023.2)

- Trong bài toán này, chúng ta cần xử lý các lệnh yêu cầu thao tác trên dữ liệu sinh viên (như tìm kiếm, thay đổi CPA, liệt kê sinh viên, v.v.). Để quản lý các lệnh này một cách hiệu quả, chúng ta có thể sử dụng cấu trúc điều khiển **switch** trong JavaScript.

- Cấu trúc **switch** rất hữu ích khi chúng ta có nhiều lựa chọn (cases) và muốn kiểm tra giá trị của một biến. Khi đó, chúng ta có thể xác định hành động tương ứng cho mỗi trường hợp mà không cần phải sử dụng quá nhiều câu lệnh if-else.

- Người dùng có thể nhập một chuỗi lệnh yêu cầu hệ thống thực hiện các thao tác trên dữ liệu sinh viên. Các lệnh như list, find, modify, findtop, findbottom, findcanhcao, cnt, và suspended sẽ được xử lý thông qua cấu trúc switch trong JavaScript. Chúng ta sẽ sử dụng prompt-sync để nhận lệnh từ người dùng, sau đó sử dụng switch để xác định hành động cần thực hiện dựa trên lệnh người dùng nhập vào, sau đó thực thi lệnh tương ứng và trả kết quả.



b) Phân tích các chức năng:

Chương trình sử dụng mảng (struct money) để lưu trữ các phần tử nhập vào. Sau khi nhập vào dữ liệu, chương trình cho phép xử lý các chức năng sau:

Lệnh list

Lệnh này sẽ liệt kê tất cả sinh viên trong danh sách, chỉ hiển thị MSSV và Tên của họ.

Lệnh find

Lệnh này yêu cầu người dùng nhập MSSV, sau đó tìm kiếm sinh viên có MSSV đó và in ra thông tin của sinh viên đó (MSSV, tên, CPA, và mức cảnh cáo), trả về undefine nếu không tìm thấy

Lệnh modify

Lệnh này cho phép người dùng cập nhật CPA của một sinh viên dựa trên MSSV, sau đó tính toán lại mức cảnh cáo của sinh viên đó.

Lệnh findtop

Lệnh này yêu cầu người dùng nhập một số n và tìm ra n sinh viên có CPA cao nhất. trả về mỗi mssv trên một dòng, mssv có cpa cao nhất đứng trước, n là số nguyên ≥ 1

Lệnh findbottom

Lệnh này yêu cầu người dùng nhập một số n và tìm ra n sinh viên có CPA thấp nhất. trả về mỗi mssv trên một dòng, mssv có cpa thấp nhất đứng trước, n là số nguyên ≥ 1

Lệnh findWarning

Lệnh này tìm các sinh viên đang bị cảnh cáo, kèm mức cảnh cáo, 1, 2, 3. Việc tính cảnh cáo chỉ dựa trên điểm cpa hiện có, và sử dụng luật của ĐHBK Hà Nội, mức 3 cpa ≤ 0.5 , mức 2: $0.5 < \text{cpa} \leq 1.0$, mức 1: $1.0 < \text{cpa} \leq 1.5$.

Lệnh count

Lệnh này sẽ đếm số sinh viên có điểm cpa là nằm trong đoạn [a b] ($a \leq \text{cpa} \leq b$)

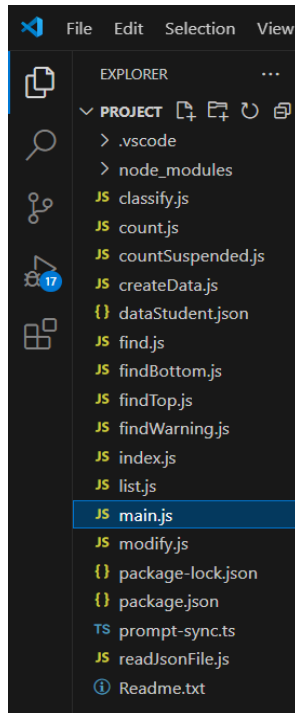
Lệnh findSuspended

Lệnh này tính số lượng sinh viên phải đình chỉ học. Điều kiện đình chỉ học là mức cảnh cáo dựa trên cpa và thời gian học tối đa cho phép (> 5 năm bị đình chỉ) (được tính dựa trên năm vào trường (dựa trên mssv) và thời điểm hiện tại mm/yyyy)

Lệnh classify

Lệnh này Phân loại học lực dựa trên CPA của từng sinh viên theo các mức: "Xuất sắc", "Giỏi", "Khá", "Trung bình", và "Yếu" và tính toán số lượng sinh viên thuộc từng mức học lực. Lệnh này do nhóm chúng em tự phát triển thêm tính năng mới.

c) Cấu trúc dự án:



- Thư mục **Project**: Đây là thư mục chứa toàn bộ các file của dự án.
- Thư mục **node_modules**: Thư mục này thường chứa các thư viện bên thứ ba mà dự án sử dụng.
- Các file JavaScript: Đây là các file chứa mã nguồn JavaScript. Tên các file cho thấy các chức năng khác nhau mà chúng thực hiện, chẳng hạn như:
 - + main.js: Thường là file khởi động chính của chương trình.
 - + createData.js: Có thể dùng để tạo dữ liệu ban đầu chương trình.
 - + find.js, findTop.js, findBottom.js, modify.js, list.js,...: Các file chức năng
- **package.json**: File này chứa thông tin về dự án, bao gồm tên, phiên bản, các thư viện phụ thuộc, các script để chạy chương trình, v.v.
- **package-lock.json**: File này chứa thông tin chi tiết về các phiên bản thư viện đã cài đặt
- **readme.txt**: File này chứa thông tin giới thiệu về dự án, danh sách thành viên

Hình 2. Cấu trúc các tệp và thư mục

4.2 Phân tích và thiết kế:

File main.js:

a) Cấu trúc

```
1  import { list } from './list.js';
2  import { find } from './find.js';
3  import { modifyCPA } from './modify.js';
4  import { findTop } from './findTop.js';
5  import { findBottom } from './findBottom.js';
6  import { findWarning } from './findWarning.js';
7  import { count } from './count.js';
8  import { countsSuspended } from './countsSuspended.js';
9  import promptSync from 'prompt-sync';
10 import { classify } from './classify.js';
11 // Sử dụng thư viện prompt-sync để nhận đầu vào từ người dùng
12 const prompt = promptSync();
13 let exit = false;
14
15 function error(){
16     //console.clear();
17     console.log("Lệnh này không hợp lệ");
18     let comand = prompt("Nhấn phím Enter để tiếp tục");
19     return;
20 }
21
22 while (!exit) {
23     //console.clear();
24     console.log("Quản lý kết quả sinh viên Đại học Bách Khoa Hà Nội");
25     console.log("Danh sách chức năng: ");
26     console.log("1.list: In ra danh sách sinh viên");
27     console.log("2.find <mssv>: Tìm sinh viên với MSSV");
28     console.log("3.modify cpa <mssv> <cpa>: Cập nhật điểm CPA của sinh viên");
29     console.log("4.findTop n: Tìm n sinh viên có CPA cao nhất");
30     console.log("5.findBottom n: Tìm n sinh viên có CPA thấp nhất");
31     console.log("6.findWarning: Tìm các sinh viên đang bị cảnh cáo, kèm mức cảnh cáo");
32     console.log("7.count a b: Đếm số sinh viên có điểm cpa nằm trong đoạn [a,b]");
33     console.log("8.countsuspended: Đếm số sinh viên phải đình chỉ học kèm điều kiện");
34     console.log("9.classify: [Tự phát triển] Đếm số sinh viên theo học lực");
35     console.log("0.exit: Thoát chương trình, nhập 0 để thoát");
36 }
```

Hình 3: Một phần code file main.js

- Mã JavaScript trên thực hiện quản lý kết quả học tập của sinh viên Đại học Bách Khoa Hà Nội thông qua một giao diện dòng lệnh. Chương trình nhận các lệnh từ người dùng và thực hiện các thao tác tương ứng với mỗi lệnh. Dưới đây là phần phân tích chi tiết các dòng lệnh:

- Phần Import:

<pre>import { list } from "./list.js"; import { find } from "./find.js"; import { modifyCPA } from "./modify.js"; import { findTop } from "./findTop.js"; import { findBottom } from "./findBottom.js"; import { findWarning } from "./findWarning.js"; import { count } from "./count.js"; import { countSuspended } from "./countSuspended.js"; import promptSync from "prompt-sync"; import { classify } from "./classify.js";</pre>	<ul style="list-style-type: none"> Các dòng import này sử dụng tính năng ES6 của JavaScript để nhập các hàm hoặc module từ các file khác. Mỗi file như list.js, find.js, modify.js, ... chứa các hàm xử lý dữ liệu sinh viên cụ thể theo yêu cầu của bài toán. promptSync được dùng để nhận dữ liệu đầu vào từ người dùng trong môi trường Node.js, thông qua câu lệnh: let command = prompt("Nhập lệnh theo cú pháp như trên: ");
---	---

- Khởi tạo Prompt và Biến Điều Kiện:

<pre>const prompt = promptSync(); let exit = false;</pre>	<ul style="list-style-type: none"> promptSync() là một thư viện hỗ trợ nhận đầu vào từ người dùng trên console (dòng lệnh). exit là biến điều khiển để dừng vòng lặp và thoát chương trình khi người dùng nhập lệnh exit.
--	---

- Hàm error():

<pre>function error() { console.log("Lệnh này không hợp lệ"); let comand = prompt("Nhấn phím Enter để tiếp tục"); return; }</pre>	<p>Hàm error() được sử dụng để hiển thị thông báo lỗi khi người dùng nhập lệnh không hợp lệ. Sau khi hiển thị thông báo lỗi, chương trình yêu cầu người dùng nhấn phím Enter để tiếp tục.</p>
---	---

- Vòng Lặp Chính và Menu:

<pre>while (!exit) { console.log("Quản lý kết quả sinh viên Đại học Bách Khoa Hà Nội"); console.log("Danh sách chức năng: "); console.log("1.list: In ra danh sách sinh viên"); console.log("2.find <mssv>: Tìm sinh viên với MSSV"); console.log("3.modify cpa <mssv> <cpa>: Cập nhật điểm CPA của sinh viên"); </pre>	<p>Menu: Mỗi lần vòng lặp thực hiện, một menu các chức năng sẽ được hiển thị. Các lệnh này chính là đề bài của bài toán, liên quan đến việc quản lý sinh viên (liệt kê sinh viên, tìm kiếm, cập nhật CPA, v.v).</p>
--	---

- Kiểm Tra Lệnh Thoát Chương Trình:

<pre>if (commandArr[0] === "exit" commandArr[0] === "0") { exit = true; console.clear(); console.log("Đang thoát chương trình..."); break; }</pre>	<p>Nếu lệnh nhập vào là exit hoặc 0, chương trình sẽ dừng vòng lặp và thoát khỏi chương trình. <i>thì phần tử thứ hai trong mảng commandArr (commandArr[1]) sẽ được gán cho biến mssv.</i></p>
---	--

Giao diện file chương trình chính

Hình 4: Giao diện chương trình chính: Gồm thông báo menu và dòng câu lệnh cho người dùng nhập lệnh vào

b) Nguyên lý hoạt động, vận hành:

<p>- Bước 1: Người dùng mở file Project bằng Visual Studio Code, chạy New Terminal -> gõ lệnh <code>node main.js</code></p>	
<p>- Bước 2: Danh sách menu câu lệnh hiện ra, người dùng nhập câu lệnh theo cú pháp đã được ghi trên menu. Giả sử có 2 trường hợp xảy ra:</p> <ul style="list-style-type: none"> + Nhập đúng cú pháp: Ví dụ nhập <code>list</code>, chương trình sẽ thực thi câu lệnh đó và trả về màn hình kết quả tương ứng + Nhập sai cú pháp: Ví dụ nhập <code>lit</code>, chương trình sẽ báo lỗi “Lệnh không hợp lệ, nhấn phím Enter để tiếp tục”, sau đó trở về giao diện menu để người dùng nhập lại. 	

4.2.1 Chức năng 1: In ra danh sách sinh viên đang học (chỉ in MSSV và tên)

a) Ý tưởng:

- Đọc dữ liệu từ file JSON chứa danh sách sinh viên.
- Chuyển đổi dữ liệu thành mảng đối tượng.
- Lọc những sinh viên có MSSV thuộc khoảng năm nhập học từ 2019 đến 2023.
- In danh sách sinh viên đã lọc ra màn hình..
- **Mục đích:** In ra danh sách sinh viên đang học (chỉ in MSSV và tên)
- **Đầu vào:** Không yêu cầu đầu vào từ người dùng, chỉ cần lệnh hoặc yêu cầu từ người dùng

- **Đầu ra:** Danh sách sinh viên sẽ được hiển thị, mỗi dòng sẽ chứa mã số sinh viên (MSSV) và tên của một sinh viên.

b) Cấu trúc:

```
const students = [];
let temp = '';
let inObject = false;

for (let i = 0; i < data.length; i++) {
  if (data[i] === '{') {
    inObject = true;
    temp = '';
  }
  if (inObject) {
    temp += data[i];
  }
  if (data[i] === '}') {
    inObject = false;
    students.push(eval('(' + temp + ')'));
  }
}
```

```
// Thuật toán lọc sinh viên bằng cách duyệt từng phần tử
const filteredStudents = [];

for (let i = 0; i < students.length; i++) {
  const student = students[i];

  // Trích xuất năm từ MSSV của sinh viên
  let year = '';
  for (let j = 0; j < 4; j++) {
    year += student.mssv.toString()[j];
  }

  // Chuyển đổi chuỗi năm thành số và kiểm tra điều kiện
  const studentYear = parseInt(year);
  if (studentYear >= 2019 && studentYear <= 2023) {
    filteredStudents.push(student);
  }
}
```

Hình 5: Một phần source code file list.js

Chuyển đổi dữ liệu JSON thành mảng đối tượng

- Duyệt từng ký tự trong chuỗi JSON để tách từng đối tượng {...}.
- Sử dụng eval để chuyển chuỗi JSON thành đối tượng JavaScript và thêm vào mảng students.
- Cách hoạt động:
 - Khi gặp ký tự {, bắt đầu xây dựng chuỗi đối tượng temp.
 - Khi gặp ký tự }, kết thúc chuỗi và dùng eval để chuyển chuỗi thành đối tượng.
 - Thêm đối tượng vào mảng students.

Lọc sinh viên theo năm nhập học

- Trích xuất 4 ký tự đầu tiên từ MSSV để lấy năm nhập học.
- Chuyển đổi năm nhập học sang số nguyên.
- Nếu năm nằm trong khoảng 2019 đến 2023, thêm sinh viên vào mảng filteredStudents.

Các phép toán ứng với thuật toán

Đọc file: $O(n)O(n)$, với nn là kích thước file JSON.

- Chuyển đổi JSON: $O(m)O(m)$, với mm là số lượng sinh viên.
- Lọc sinh viên: $O(m \cdot k)O(m \cdot k)$, với $k=4k = 4$ (số ký tự để trích xuất năm từ MSSV).
- In danh sách: $O(p)O(p)$, với pp là số sinh viên đã lọc.

Độ phức tạp:

Thời gian:

Đọc file: $O(n)O(n)$.

Chuyển đổi dữ liệu: $O(m)O(m)$.

Lọc sinh viên: $O(m \cdot k)O(m \cdot k)$, với $k=4k = 4$ (dài MSSV cố định).

Tổng cộng: $O(n+m \cdot k)O(n + m \cdot k)$.

Không gian: Cần bộ nhớ để lưu toàn bộ danh sách sinh viên ($O(m)O(m)$).

Đánh giá mức độ sử dụng:

- Phù hợp với ứng dụng nhỏ hoặc dữ liệu không quá lớn.
- Có thể cải thiện bằng cách sử dụng thư viện xử lý JSON tiêu chuẩn như JSON.parse để tăng hiệu quả và bảo mật.

c) Nguyên lý hoạt động, vận hành:

<p>- Bước 1: Người dùng nhập lệnh list ở dòng thông báo “Nhập lệnh theo cú pháp như trên”, nếu gõ sai chương trình hiện thông báo không hợp lệ và bắt người dùng nhập lại</p>	<div> <div>PROBLEMS OUTPUT DEBUG CONSOLE</div> <div> <p>Lệnh này không hợp lệ</p> <p>Nhấn phím Enter để tiếp tục</p> </div> </div>
<p>- Bước 2: Nếu gõ đúng cú pháp, chương trình sẽ chạy và in ra list (danh sách) sinh viên gồm MSSV và họ và tên ở mỗi dòng, đồng thời có câu lệnh: <i>Nhấn phím Enter để tiếp tục để thoát chương trình và trở về giao diện menu</i></p>	<div> <div> MSSV: 20238441, họ và tên: wV0mFwoe MSSV: 20238449, họ và tên: KtGsiLfZ MSSV: 20238454, họ và tên: l1EWlyoa MSSV: 20238456, họ và tên: UiNqgpFE MSSV: 20238464, họ và tên: XHUCnPhu MSSV: 20238474, họ và tên: FbuyVTYB MSSV: 20238477, họ và tên: QSRCwJXN MSSV: 20238484, họ và tên: yEkFS1l MSSV: 20238496, họ và tên: hIpycgUQ Nhấn phím Enter để tiếp tục... </div> </div>

4.2.2 Chức năng 2: Tìm sinh viên có mssv. Trả về đối tượng sinh viên nếu tìm thấy (in ra màn hình <mssv> "<hoten>" <cpa> <canhcao>, trả về undefined nếu không tìm thấy

a) Ý tưởng:

- Đọc file JSON chứa danh sách sinh viên.
- Chuyển đổi dữ liệu từ file thành một mảng đối tượng sinh viên.
- Lần lượt duyệt qua từng sinh viên, so sánh MSSV với giá trị được nhập.
- Nếu tìm thấy, in thông tin sinh viên; nếu không, thông báo không tìm thấy.
- **Mục đích:** Cho phép người dùng nhập MSSV để tìm kiếm nhanh thông tin chi tiết của một sinh viên trong hệ thống.

- **Đầu vào:** MSSV người dùng nhập vào file main.js

- **Đầu ra:**

+ Nếu tìm thấy sinh viên:

MSSV: <mssv> - Tên: <name> - CPA: <cpa> - Cảnh cáo: <canhcao>

+ Nếu không tìm thấy sinh viên

undefined

b) Cấu trúc:



```
export function find(mssv) {
  for (let i = 0; i < arrlength; i++) {
    let currentMSSV = '';
    let targetMSSV = '';

    // Chuyển đổi MSSV của sinh viên hiện tại thành số
    for (let j = 0; j < arrSV[i].mssv.toString().length; j++) {
      currentMSSV += arrSV[i].mssv.toString()[j];
    }

    // Chuyển đổi MSSV mục tiêu thành số
    for (let k = 0; k < mssv.toString().length; k++) {
      targetMSSV += mssv.toString()[k];
    }

    if (parseInt(currentMSSV) === parseInt(targetMSSV)) { // Kiểm tra xem MSSV có khớp không
      console.log('MSSV: ${arrSV[i].mssv} - Tên: ${arrSV[i].name} - CPA: ${arrSV[i].cpa} - Mức cảnh cáo: ${arrSV[i].canhcao}');
      found = true; // Đánh dấu đã tìm thấy sinh viên
    }
  }
}
```

Hình 6: Phần source code file find.js

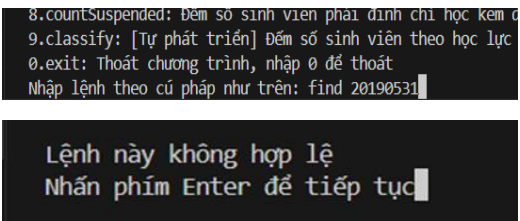
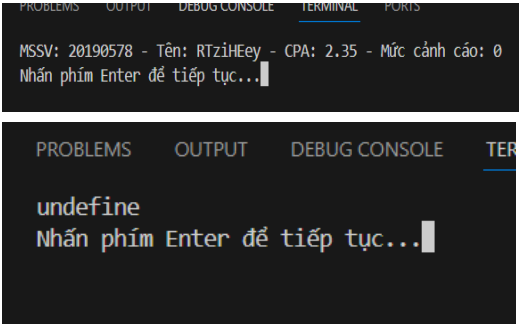
Ý nghĩa:

- Vòng lặp duyệt từng ký tự trong chuỗi data.
- Khi gặp {, đánh dấu bắt đầu một đối tượng JSON và khởi tạo temp.
- Thêm từng ký tự vào temp khi inObject là true.
- Khi gặp }, đánh dấu kết thúc một đối tượng JSON.
- Dùng eval để chuyển chuỗi JSON trong temp thành đối tượng.
- Thêm đối tượng vào mảng arrSV.

Độ phức tạp:

- Thời gian:
- Đọc file: $O(n)O(n)$, với nn là kích thước file.
- Chuyển đổi dữ liệu: $O(m)O(m)$, với mm là số lượng sinh viên.
- Tìm kiếm: $O(m)O(m)$. Tổng cộng: $O(n+m)O(n+m)$.
- Không gian:
- Tồn bộ nhớ để lưu toàn bộ mảng sinh viên.
- Đánh giá mức độ sử dụng:
- Phù hợp với các ứng dụng nhỏ hoặc khi cần xử lý dữ liệu JSON đơn giản.
- Không tối ưu cho hệ thống lớn, nên sử dụng các thư viện JSON tiêu chuẩn (như JSON.parse) và cải thiện cách xử lý file.

c) Nguyên lý hoạt động, vận hành:

<p>- Bước 1: Người dùng nhập lệnh <code>find <mssv></code> ở dòng thông báo “Nhập lệnh theo cú pháp như trên”, nếu gõ sai chương trình hiện thông báo không hợp lệ và bắt người dùng nhập lại</p>	
<p>- Bước 2: Nếu gõ đúng cú pháp, chương trình sẽ chạy và hiện ra dòng thông tin sinh viên gồm MSSV và họ và tên, cpa, mức cảnh cáo đồng thời có câu lệnh: <i>Nhấn phím Enter để tiếp tục để thoát chương trình và trở về giao diện menu</i></p> <p><i>Nếu thông tin mssv người dùng nhập không có trong dataStudent.json, kết quả báo undefine</i></p>	

4.2.3 Chức năng 3: Thay đổi cpa của sinh viên

a) Ý tưởng:

- **Ý tưởng:** Cập nhật điểm CPA của 1 sinh viên, biết người dùng nhập mssv và điểm cpa thay đổi qua file main.js. Chức năng này được triển khai thông qua tìm kiếm sinh viên trong tệp dataStudent.json dựa trên MSSV, cập nhật CPA và 1 thuộc tính canhcao tùy theo CPA để kiểm tra CPA người dùng nhập hợp lệ, sau đó lưu dữ liệu trở lại vào tệp JSON, chuyển dữ liệu JSON thành danh sách sinh viên `fromJSON(jsonData)` và thông báo kết quả ra màn hình

- **Mục đích:** Cho phép người dùng nhập MSSV và cpa muốn cập nhật của sinh viên đó, sau đó lưu kết quả vừa cập nhật, ghi đè lên file dataStudent.json

- **Đầu vào:** MSSV và cpa muốn thay đổi của sinh viên ấy mà người dùng nhập vào file main.js.

- **Đầu ra:**

+ Nếu tìm thấy sinh viên:

Đã cập nhật CPA của sinh viên MSSV: `<mssv>` từ `<oldcpa>` thành `<newcpa>`

Dữ liệu đã được lưu vào file: dataStudent.json

+ Nếu không tìm thấy sinh viên

undefined

b) Cấu trúc:

Đọc dữ liệu JSON:

- Đọc nội dung tệp `dataStudent.json` vào bộ nhớ dưới dạng chuỗi.
- Phân tích chuỗi thành một mảng các đối tượng sinh viên mà không sử dụng `JSON.parse()`.
Đoạn mã này dùng một phương thức thủ công để "parse" chuỗi thành đối tượng JavaScript.

Tìm kiếm sinh viên bằng thuật toán tìm kiếm nhị phân (Binary Search):

- Tìm kiếm sinh viên theo MSSV (Mã số sinh viên) bằng thuật toán tìm kiếm nhị phân.
- Mảng `students` phải được sắp xếp theo MSSV để đảm bảo rằng tìm kiếm nhị phân có thể hoạt động chính xác.
- Nếu tìm thấy sinh viên có MSSV trùng khớp, cập nhật điểm CPA của sinh viên đó và cập nhật mức cảnh cáo (`canhcao`) dựa trên điểm CPA.

Lưu lại dữ liệu:

- Sau khi cập nhật dữ liệu, toàn bộ mảng sinh viên được chuyển lại thành chuỗi JSON và lưu vào tệp `dataStudent.json`.

Độ phức tạp thuật toán:

1. Đọc và phân tích dữ liệu JSON:

- Đoạn mã xử lý chuỗi JSON theo cách thủ công, trong đó bạn phải duyệt qua toàn bộ tệp dữ liệu để phân tích các đối tượng. Thao tác này có độ phức tạp là $O(n)$, với n là độ dài của chuỗi dữ liệu trong tệp JSON.
- Cần phải phân tích tất cả các đối tượng trong tệp, do đó thời gian thực thi tỷ lệ với kích thước tệp dữ liệu.

2. Tìm kiếm nhị phân (Binary Search):

- Thuật toán tìm kiếm nhị phân yêu cầu mảng đã được sắp xếp. Nếu mảng `students` đã được sắp xếp theo MSSV, thì thuật toán tìm kiếm nhị phân có độ phức tạp là $O(\log n)$, với n là số lượng sinh viên trong mảng.
- Điều này có nghĩa là việc tìm kiếm sinh viên có MSSV cụ thể sẽ mất thời gian rất nhanh đối với mảng lớn.

3. Cập nhật dữ liệu và lưu vào tệp:

- Sau khi tìm thấy sinh viên và cập nhật điểm CPA, bạn cần phải ghi lại toàn bộ mảng `students` vào tệp `dataStudent.json`. Việc này có độ phức tạp $O(n)$, vì bạn cần phải chuyển đổi mảng thành chuỗi JSON và ghi vào tệp.

```
// Chuyển đổi JSON thành đối tượng JavaScript mà không dùng JSON.parse
const students = [];
let temp = '';
let inObject = false;

for (let i = 0; i < data.length; i++) {
    if (data[i] === '{') {
        inObject = true;
        temp = '';
    }
    if (inObject) {
        temp += data[i];
    }
    if (data[i] === '}') {
        inObject = false;
        students.push(eval('(' + temp + ')'));
    }
}
```

```
function binarySearch(arr, mssv) {
    let low = 0;
    let high = arr.length - 1;

    while (low <= high) {
        const mid = Math.floor((low + high) / 2);

        // So sánh MSSV dưới dạng chuỗi
        if (arr[mid].mssv.toString() === mssv.toString()) {
            return mid; // Tìm thấy sinh viên
        } else if (arr[mid].mssv.toString() < mssv.toString()) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    return -1; // Không tìm thấy
```

Hình 7: Phần source code file `modify.js`

Tổng độ phức tạp:

- **Đọc và phân tích dữ liệu JSON:** $O(n)$ (nếu tệp dữ liệu lớn).
- **Tìm kiếm nhị phân (Binary Search):** $O(\log n)$ (với n là số lượng sinh viên, nếu mảng đã được sắp xếp).
- **Cập nhật dữ liệu và ghi vào tệp:** $O(n)$ (vì phải ghi lại toàn bộ mảng vào tệp sau khi cập nhật).

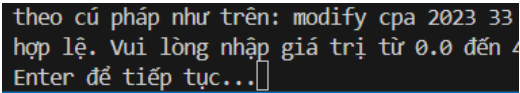
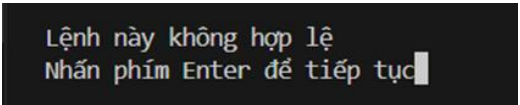
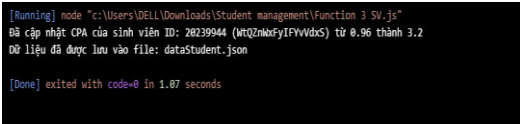
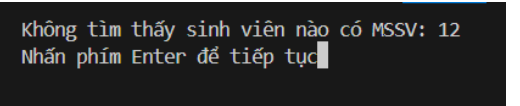
Do đó, tổng độ phức tạp của thuật toán này là:

$$O(n) + O(\log n) + O(n) = O(n).$$

Ý nghĩa code:

Chuyển đổi JSON thành đối tượng JavaScript mà không dùng JSON.parse	Hàm tìm kiếm nhị phân (Binary Search) để tìm sinh viên theo MSSV
<ul style="list-style-type: none"> • Khởi tạo biến: temp: Dùng để chứa các ký tự của một đối tượng JSON tạm thời khi duyệt qua chuỗi. isObject: Cờ (flag) giúp xác định xem ta có đang trong một đối tượng JSON không. Khi giá trị này là true, có nghĩa là ta đang trong một đối tượng JSON, và cần phải ghi lại các ký tự vào temp. • Duyệt qua chuỗi data: for (let i = 0; i < data.length; i++) { ... }: Lặp qua từng ký tự trong chuỗi data. • Khi gặp ký tự {: if (data[i] === '{') { ... }: Khi gặp dấu {, có nghĩa là một đối tượng JSON bắt đầu, do đó: isObject = true; Đặt cờ isObject thành true, báo hiệu rằng ta đang trong một đối tượng JSON. temp = ''; Khởi tạo lại biến temp để bắt đầu lưu trữ các ký tự của đối tượng JSON mới. • Ghi lại ký tự vào temp: if (isObject) { temp += data[i]; }: Nếu cờ isObject đang là true, ta tiếp tục ghi lại ký tự vào biến temp cho đến khi gặp dấu } (kết thúc đối tượng JSON). • Khi gặp ký tự }: if (data[i] === '}') { ... }: Khi gặp dấu }, có nghĩa là đối tượng JSON đã kết thúc. Lúc này: isObject = false; Đặt lại cờ isObject thành false, báo hiệu kết thúc đối tượng JSON. students.push(eval('(' + temp + ')')); Dùng hàm eval() để chuyển đổi chuỗi trong temp thành một đối tượng JavaScript và thêm đối tượng này vào mảng students. <p><i>Đoạn mã này đang thực hiện việc phân tích thủ công chuỗi JSON thành các đối tượng. Cụ thể, nó tìm kiếm các đoạn chuỗi giữa dấu { } (được xem như các đối tượng trong JSON), sau đó chuyển các chuỗi này thành đối tượng và thêm vào mảng students.</i></p>	<ul style="list-style-type: none"> • Khởi tạo biến: • low: Chỉ số của phần tử đầu tiên trong mảng (ban đầu là 0). • high: Chỉ số của phần tử cuối cùng trong mảng (ban đầu là arr.length - 1). • Vòng lặp while: Vòng lặp này tiếp tục cho đến khi chỉ số low vượt qua chỉ số high, nghĩa là khi không còn phần tử nào trong mảng để kiểm tra. Trong mỗi lần lặp, ta sẽ chia mảng thành hai nửa và kiểm tra phần tử giữa của mảng. • Tính chỉ số phần tử giữa (mid): mid là chỉ số của phần tử ở giữa mảng con hiện tại (giữa chỉ số low và high). Math.floor() được sử dụng để làm tròn kết quả chia, giúp mid là một số nguyên. • So sánh phần tử giữa với mssv: Chúng ta so sánh mssv (mã số sinh viên) với arr[mid].mssv (mã số sinh viên của phần tử ở vị trí mid trong mảng). toString() được gọi để đảm bảo rằng so sánh là với chuỗi, tránh các vấn đề về kiểu dữ liệu nếu mssv là số và arr[mid].mssv là chuỗi. Nếu chúng khớp, tức là đã tìm thấy sinh viên, hàm trả về chỉ số mid, tức là vị trí của sinh viên trong mảng. • Điều chỉnh chỉ số tìm kiếm: Nếu phần tử ở giữa nhỏ hơn mssv: Điều này có nghĩa là mssv có thể ở nửa bên phải của mảng, vì mảng đã được sắp xếp. Do đó, ta điều chỉnh low thành mid + 1 để tìm kiếm trong nửa phải. Nếu phần tử ở giữa lớn hơn mssv: Điều này có nghĩa là mssv có thể ở nửa bên trái của mảng, do đó ta điều chỉnh high thành mid - 1 để tìm kiếm trong nửa trái. • Điều chỉnh chỉ số tìm kiếm: <p><i>Hàm binarySearch thực hiện tìm kiếm nhị phân trong mảng arr, nơi mỗi phần tử của mảng là một sinh viên với một thuộc tính mssv. Hàm này sẽ tìm một sinh viên trong mảng có mã số sinh viên khớp với mssv. Nếu tìm thấy, nó trả về vị trí của sinh viên trong mảng; nếu không, nó trả về -1. độ phức tạp thời gian là O(log n)</i></p>

c) Nguyên lý hoạt động, vận hành:

<p>- Bước 1: Người dùng nhập lệnh modify cpa <mssv> <cpa> ở dòng thông báo “Nhập lệnh theo cú pháp như trên”, nếu gõ sai chương trình hiện thông báo không hợp lệ và bắt người dùng nhập lại.</p> <p>Đồng thời kiểm tra CPA mới mà người dùng nhập vào có hợp lệ không</p>	 
<p>- Bước 2: Nếu gõ đúng cú pháp và CPA hợp lệ, chương trình sẽ chạy và hiện ra dòng thông tin sinh viên câu lệnh: Đã cập nhật điểm CPA của sinh viên</p> <p>Đồng thời lưu dữ liệu mới cập nhật vào file JSON</p> <p>Nếu thông tin mssv người dùng nhập không có trong dataStudent.json, kết quả báo Không tìm thấy sinh viên có MSSV trên</p>	 

4.2.4 Chức năng 4: Tìm n sinh viên có cpa cao nhất, trả về mỗi mssv trên một dòng, mssv có cpa cao nhất đứng trước

a) Ý tưởng:

- **Ý tưởng:** Đọc dữ liệu sinh viên từ file JSON, chuyển đổi dữ liệu JSON thành đối tượng sinh viên, sau đó sử dụng phương thức `getTopNLowestCPA(n)` trong lớp `StudentManager` để tạo ra bản sao của mảng sinh viên để không thay đổi mảng gốc, rồi sắp xếp giảm dần và lấy n sinh viên có CPA cao nhất

- **Mục đích:** Lấy n sinh viên có CPA cao nhất, trả về mssv trên 1 dòng qua tham số n được truyền từ file `main.js` và data qua tệp JSON

- **Đầu vào:** Tham số n (n sinh viên cpa cao nhất cần lấy, n là số nguyên ≥ 1)

- **Đầu ra:** Danh sách sinh viên có CPA cao nhất: Dựa trên giá trị n, hàm sẽ in ra thông tin của n sinh viên có CPA cao nhất. Mỗi sinh viên sẽ được hiển thị theo định dạng:

MSSV: <MSSV>, Name: <Tên>, CPA: <Điểm CPA>

b) Cấu trúc:

Phần lớp: Class Student

- **Mục đích:** Class `Student` đại diện cho một sinh viên với các thông tin cơ bản như mã số sinh viên (MSSV), tên sinh viên và điểm CPA.

- **Thuộc tính:**

+ `this.mssv`: Gán giá trị cho thuộc tính `mssv`.

+ `this.name`: Gán giá trị cho thuộc tính `name`.

+ `this.cpa`: Gán giá trị cho thuộc tính `cpa` và đảm bảo giá trị là kiểu `float` bằng cách sử dụng `parseFloat(cpa)`.

```
class Student {
  constructor(mssv, name, cpa) {
    this.mssv = mssv;
    this.name = name;
    this.cpa = parseFloat(cpa);
  }
}
```

Phần lớp: Class StudentManager

```
class StudentManager {
  constructor(students) {
    this.students = students;
  }
}
```

- **Mục đích:** Class `StudentManager` là một lớp quản lý danh sách các sinh viên. Lớp này có các phương thức để xử lý các thao tác như cập nhật điểm CPA của sinh viên, lưu dữ liệu vào tệp, và khôi phục danh sách sinh viên từ tệp JSON.

- **Thuộc tính:**

`students`: Là mảng chứa các đối tượng `Student`. Mảng này lưu trữ tất cả các sinh viên mà hệ thống đang quản lý.

Mô tả ngắn gọn cấu trúc:

- Class `Student`: Đại diện cho một sinh viên với ba thuộc tính: `mssv` (mã sinh viên), `name` (tên sinh viên), và `cpa` (điểm CPA, kiểu số thực).

- Class `StudentManager`: Lớp này quản lý danh sách các sinh viên và cung cấp các phương thức để xử lý các sinh viên này. Thuộc tính `students` là mảng chứa danh sách các đối tượng `Student`.

Các phương thức trong StudentManager:

- `quickSort(arr, low, high)`: Phương thức sắp xếp mảng `arr` trong khoảng từ `low` đến `high` bằng thuật toán Quick Sort.

- `partition(arr, low, high)`: Phương thức phân vùng mảng trong thuật toán Quick Sort. Phương thức này dùng phần tử cuối làm chốt và di chuyển các phần tử nhỏ hơn chốt sang bên trái và các phần tử lớn hơn sang bên phải.

- `getTopNHighestCPA(n)`: Phương thức này sắp xếp danh sách sinh viên theo CPA giảm dần (dùng thuật toán Quick Sort) và trả về n sinh viên có CPA cao nhất.

- `static fromJSON(jsonData)`: Phương thức tĩnh để chuyển đổi dữ liệu JSON thành danh sách đối tượng `Student` và khởi tạo một đối tượng `StudentManager`.

Các thao tác chính trong hàm findTop(n):

- Đọc dữ liệu sinh viên từ tệp JSON (`fs.readFileSync()`).

- Tạo đối tượng StudentManager từ dữ liệu JSON.
 - Gọi phương thức getTopNHighestCPA(n) để tìm n sinh viên có CPA cao nhất.
- In kết quả lên màn hình.

Độ phức tạp:

- Trung bình: $O(n \log n)$ – Khi pivot được chọn tốt và mảng được phân chia gần đều.
- Tệ nhất: $O(n^2)$ – Nếu pivot chọn không hợp lý, ví dụ, trong trường hợp mảng đã được sắp xếp sẵn hoặc đảo ngược.
- Tốt nhất: $O(n \log n)$ – Khi pivot chọn hợp lý và phân mảng đều.

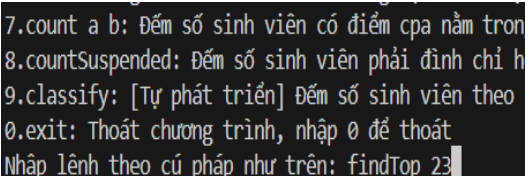
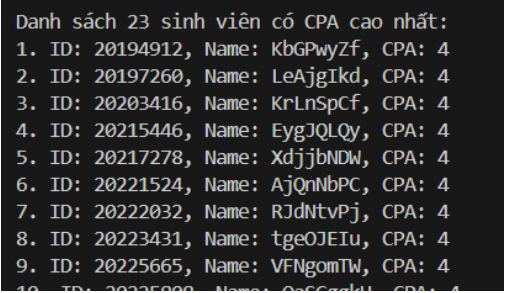
Các phép toán:

- Hoán đổi phần tử (Swap): Trong quá trình phân vùng, các phần tử sẽ được hoán đổi giữa các chỉ số i và j. Hoán đổi này là phép toán cơ bản trong Quick Sort.
- Phân vùng: Phương thức partition chia mảng thành các phần nhỏ hơn và tạo ra một chốt sao cho mọi phần tử bên trái là nhỏ hơn pivot và mọi phần tử bên phải là lớn hơn pivot.

Ý nghĩa code:

Quicksort (Sắp xếp)	getTopNHighestCPA (Lấy n sinh viên CPA cao nhất)
<p>Phương thức quickSort(arr, low, high): Đầu vào của phương thức này là mảng arr, và hai chỉ số low và high, xác định phạm vi con của mảng mà ta cần sắp xếp.</p> <p>Cấu trúc:</p> <ul style="list-style-type: none"> - Điều kiện dừng: Nếu $low \geq high$, thuật toán không làm gì cả, vì mảng đã được sắp xếp hoặc không còn phần tử nào để sắp xếp. - Phân vùng: Phương thức gọi this.partition(arr, low, high) để phân chia mảng tại chỉ số pi (pivot index), sao cho mọi phần tử trước pi có CPA lớn hơn pivot và mọi phần tử sau pi có CPA nhỏ hơn pivot. - đệ quy: Sau khi phân chia, thuật toán tiếp tục gọi đệ quy trên hai phần con: <p>this.quickSort(arr, low, pi - 1): Sắp xếp phần bên trái của pivot. this.quickSort(arr, pi + 1, high): Sắp xếp phần bên phải của pivot.</p> <p>Phương thức partition(arr, low, high): Ý nghĩa: Phương thức này thực hiện phân vùng mảng arr tại chỉ số pivot (phần tử tại vị trí high). Sau khi phân vùng, nó sẽ đảm bảo rằng mọi phần tử có CPA lớn hơn phần tử pivot sẽ nằm phía trước và phần tử có CPA nhỏ hơn sẽ nằm phía sau.</p> <p>Cấu trúc:</p> <ul style="list-style-type: none"> - Chọn pivot: - const pivot = arr[high].cpa;: Phần tử pivot được chọn là phần tử cuối cùng trong mảng con (arr[high]), và giá trị pivot là CPA của phần tử đó. - Khởi tạo chỉ số i: let i = low - 1;: i - - Lặp qua mảng con: for (let j = low; j < high; j++): Lặp qua tất cả các phần tử từ low đến high - 1 (tức là tất cả các phần tử trừ pivot). - So sánh CPA: if (arr[j].cpa > pivot): Nếu CPA của phần tử arr[j] lớn hơn pivot, thì Tăng chỉ số i lên (i++). Hoán đổi phần tử arr[i] và arr[j] để đảm bảo các phần tử có CPA lớn hơn pivot được đưa vào phía trước pivot. Hoán đổi pivot: Sau khi vòng lặp kết thúc, tất cả các phần tử có CPA lớn hơn pivot đã được di chuyển về phía trước. Giờ là lúc đặt pivot vào đúng vị trí của nó: [arr[i + 1], arr[high]] = [arr[high], arr[i + 1]]: Hoán đổi phần tử tại vị trí i + 1 (là vị trí thích hợp cho pivot) với phần tử tại vị trí high (pivot). - Trả về chỉ số của pivot: return i + 1;: Phương thức trả về chỉ số i + 1, tức là vị trí của pivot trong mảng đã phân vùng. Điều này giúp phương thức quickSort biết được nơi để chia mảng thành hai phần con tiếp theo. 	<p>□ Sắp xếp sinh viên theo CPA giảm dần: this.quickSort(this.students, 0, this.students.length - 1);</p> <p>• Lấy n sinh viên đầu tiên từ danh sách đã sắp xếp::: Mục đích: Sau khi sắp xếp, phương thức sẽ lấy n sinh viên có CPA cao nhất (n phần tử đầu tiên từ mảng đã sắp xếp).</p> <p>Cách thức: const result = []; Khởi tạo một mảng rỗng result để chứa kết quả. Vòng lặp for (let i = 0; i < n && i < this.students.length; i++): Vòng lặp này chạy từ i = 0 đến i = n-1 hoặc đến khi hết sinh viên trong mảng (nếu n lớn hơn số sinh viên có trong mảng). Mỗi sinh viên tại vị trí i trong mảng this.students sẽ được thêm vào mảng result với result.push(this.students[i]).</p> <p>Trả về danh sách n sinh viên có CPA cao nhất: Sau khi lấy xong n sinh viên đầu tiên trong danh sách đã sắp xếp, phương thức trả về mảng result chứa các sinh viên có CPA cao nhất.</p>
<p><i>Đây là một phương thức đệ quy thực hiện thuật toán Quick Sort để sắp xếp mảng arr trong khoảng từ low đến high. Quick Sort là một thuật toán sắp xếp chia để trị (Divide and Conquer).</i></p>	<p><i>Sau khi thực hiện phương thức quickSort, mảng this.students sẽ được sắp xếp từ sinh viên có CPA cao nhất đến sinh viên có CPA thấp nhất (theo thứ tự giảm dần). Sau đó trả kết quả về mảng result</i></p>

c) Nguyên lý hoạt động, vận hành:

<p>- Bước 1: Người dùng nhập lệnh findTop <n> ở dòng thông báo “Nhập lệnh theo cú pháp như trên”, nếu gõ sai chương trình hiện thông báo không hợp lệ và bắt người dùng nhập lại.</p> <p>Nếu n<1 hoặc n=0 chương trình cũng sẽ báo lỗi</p>	
<p>- Bước 2: Nếu gõ đúng cú pháp và n hợp lệ, chương trình sẽ chạy và hiện ra danh sách thông tin sinh viên câu lệnh có CPA cao nhất</p>	

4.2.5 Chức năng 5: Tìm n sinh viên có cpa thấp nhất, trả về mỗi mssv trên một dòng, mssv có cpa thấp nhất đứng trước

a) Ý tưởng:

- **Ý tưởng:** Tương tự như chức năng 4 nhưng ngược lại
- **Mục đích:** Lấy n sinh viên có CPA thấp nhất, trả về mssv trên 1 dòng qua tham số n được truyền từ file main.js và data qua tệp JSON
- **Đầu vào:** Tham số n (n sinh viên cpa thấp nhất cần lấy, n là số nguyên >=1)
- **Đầu ra:** Danh sách sinh viên có CPA thấp nhất: Dựa trên giá trị n, hàm sẽ in ra thông tin của n sinh viên có CPA cao nhất. Mỗi sinh viên sẽ được hiển thị theo định dạng:
MSSV: <MSSV>, Name: <Tên>, CPA: <Điểm CPA>

b) Cấu trúc:

Thuật toán Quick Sort.

Ý tưởng:

- Quick Sort là một thuật toán sắp xếp chia để trị (Divide and Conquer).
- Thuật toán này chọn một phần tử làm "chốt" (pivot) và chia mảng thành hai phần: Các phần tử nhỏ hơn hoặc bằng pivot. Các phần tử lớn hơn pivot.
- Hai phần này tiếp tục được sắp xếp đệ quy cho đến khi toàn bộ mảng được sắp xếp.

```

getBotNLowestCPA(n) {
  // Sắp xếp sinh viên theo CPA tăng dần bằng Quick Sort
  this.quickSort(this.students, 0, this.students.length - 1);

  // Lấy n phần tử đầu tiên
  const result = [];
  for (let i = 0; i < n && i < this.students.length; i++) {
    result.push(this.students[i]);
  }

  return result;
}

```

Hình 8: Phần source code file findBottom.js

Mô tả thuật toán:

* Hàm quickSort

Mục đích:

- Hàm quickSort thực hiện việc sắp xếp đệ quy.
- if (low < high): Kiểm tra nếu có ít nhất 2 phần tử trong đoạn cần sắp xếp.
- This.partition(arr, low, high): Chia mảng tại chỉ số pi (pivot index), các phần tử nhỏ hơn pivot nằm bên trái, lớn hơn nằm bên phải.
- This.quickSort(arr, low, pi - 1): Gọi đệ quy để sắp xếp đoạn bên trái pivot.
- This.quickSort(arr, pi + 1, high): Gọi đệ quy để sắp xếp đoạn bên phải pivot.

* Hàm partition

Mục đích: Phân chia mảng thành hai phần dựa trên pivot.

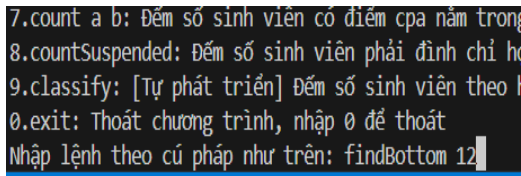
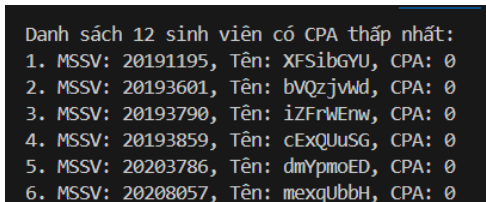
- Const pivot = arr[high].cpa: Chọn phần tử cuối cùng làm pivot.
- Let i = low - 1: i là vị trí cuối của phần tử nhỏ hơn pivot.
- For (let j = low; j < high; j++): Duyệt qua tất cả các phần tử từ low đến high - 1.
- if (arr[j].cpa < pivot): Kiểm tra nếu phần tử nhỏ hơn pivot:
- i++: Tăng vị trí của phần tử nhỏ hơn.
- [arr[i], arr[j]] = [arr[j], arr[i]]: Hoán đổi phần tử nhỏ hơn với phần tử tại vị trí i.
- [arr[i + 1], arr[high]] = [arr[high], arr[i + 1]]: Đặt pivot vào vị trí chính xác.
- return i + 1: Trả về chỉ số của pivot.

* Hàm getTopNLowestCPA

Mục đích: Lấy n sinh viên có CPA thấp nhất từ danh sách đã sắp xếp.

- this.quickSort(this.students, 0, this.students.length - 1): Sắp xếp toàn bộ danh sách sinh viên theo CPA tăng dần.
- const result = []: Tạo mảng rỗng để lưu kết quả.
- for (let i = 0; i < n && i < this.students.length; i++): Lặp qua n phần tử đầu tiên hoặc toàn bộ danh sách nếu danh sách ít hơn n.
- result.push(this.students[i]): Thêm sinh viên vào mảng kết quả.
- return result: Trả về danh sách sinh viên có CPA thấp nhất.

c) Nguyên lý hoạt động, vận hành:

<p>- Bước 1: Người dùng nhập lệnh findBottom <n> ở dòng thông báo “Nhập lệnh theo cú pháp như trên”, nếu gõ sai chương trình hiện thông báo không hợp lệ và bắt người dùng nhập lại.</p> <p>Nếu n<1 hoặc n=0 chương trình cũng sẽ báo lỗi</p>	 <pre>7.count a b: Đếm số sinh viên có điểm cpa nằm trong 8.countSuspended: Đếm số sinh viên phải đình chỉ học 9.classify: [Tự phát triển] Đếm số sinh viên theo 0.exit: Thoát chương trình, nhập 0 để thoát Nhập lệnh theo cú pháp như trên: findBottom 12</pre>
<p>- Bước 2: Nếu gõ đúng cú pháp và n hợp lệ, chương trình sẽ chạy và hiện ra danh sách thông tin sinh viên câu lệnh có CPA thấp nhất</p>	 <pre>Danh sách 12 sinh viên có CPA thấp nhất: 1. MSSV: 20191195, Tên: XFSibGYU, CPA: 0 2. MSSV: 20193601, Tên: bVQzjvWd, CPA: 0 3. MSSV: 20193790, Tên: iZFrWEnw, CPA: 0 4. MSSV: 20193859, Tên: cExQUuSG, CPA: 0 5. MSSV: 20203786, Tên: dmYpmoED, CPA: 0 6. MSSV: 20208057, Tên: mexqUbbH, CPA: 0</pre>

4.2.6 Chức năng 6: tìm các sinh viên đang bị cảnh cáo, kèm mức cảnh cáo, 1, 2, 3. Việc tính cảnh cáo chỉ dựa trên điểm cpa hiện có, và sử dụng luật của ĐHBK Hà Nội, mức 3 $cpa \leq 0.5$, mức 2: $0.5 < cpa \leq 1.0$, mức 1: $1.0 < cpa \leq 1.5$

a) Ý tưởng:

- **Ý tưởng:** Sử dụng Quicksort sắp xếp và tìm kiếm, lọc thủ công

- Quicksort là thuật toán sắp xếp chia để trị (Divide and Conquer). Ý tưởng chính là chọn một phần tử làm pivot (điểm trụ), sau đó phân chia mảng thành hai phần:

+ Các phần tử lớn hơn pivot nằm bên trái (trong trường hợp sắp xếp giảm dần).

+ Các phần tử nhỏ hơn hoặc bằng pivot nằm bên phải.

+ Sau đó, thực hiện đệ quy sắp xếp trên hai phần này.

- Duyệt qua danh sách sinh viên bằng vòng lặp để kiểm tra điều kiện mức cảnh cáo của từng sinh viên. Chỉ các sinh viên có mức cảnh cáo lớn hơn 0 mới được đưa vào danh sách cảnh cáo.

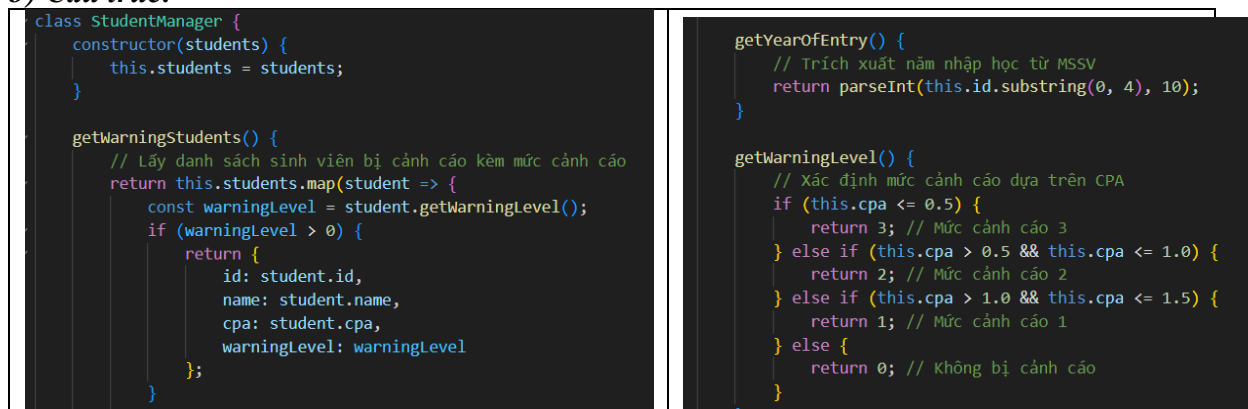
- **Mục đích:** tìm các sinh viên đang bị cảnh cáo, kèm mức cảnh cáo, 1, 2, 3.

- **Đầu vào:** Người dùng chỉ cần nhập lệnh mà không cần truyền tham số

- **Đầu ra:** Danh sách sinh viên đang bị cảnh cáo. Mỗi sinh viên sẽ được hiển thị theo định dạng:

MSSV: <MSSV>, Name: <Tên>, CPA: <Điểm CPA>, Mức cảnh cáo: <mức>

b) Cấu trúc:



```
class StudentManager {
  constructor(students) {
    this.students = students;
  }

  getWarningStudents() {
    // Lấy danh sách sinh viên bị cảnh cáo kèm mức cảnh cáo
    return this.students.map(student => {
      const warningLevel = student.getWarningLevel();
      if (warningLevel > 0) {
        return {
          id: student.id,
          name: student.name,
          cpa: student.cpa,
          warningLevel: warningLevel
        };
      }
    });
  }
}

class Student {
  constructor(mssv, name, cpa) {
    this.mssv = mssv;
    this.name = name;
    this.cpa = cpa;
  }

  getYearOfEntry() {
    // Trích xuất năm nhập học từ MSSV
    return parseInt(this.id.substring(0, 4), 10);
  }

  getWarningLevel() {
    // Xác định mức cảnh cáo dựa trên CPA
    if (this.cpa <= 0.5) {
      return 3; // Mức cảnh cáo 3
    } else if (this.cpa > 0.5 && this.cpa <= 1.0) {
      return 2; // Mức cảnh cáo 2
    } else if (this.cpa > 1.0 && this.cpa <= 1.5) {
      return 1; // Mức cảnh cáo 1
    } else {
      return 0; // Không bị cảnh cáo
    }
  }
}
```

Hình 9: Phần source code file findWarning.js

Lớp Student

Lớp Student đại diện cho một sinh viên, với các thuộc tính cơ bản như mã số sinh viên (MSSV), tên và điểm CPA. Lớp này cũng cung cấp các phương thức để xử lý các thông tin liên quan đến sinh viên.

- **constructor(mssv, name, cpa):**
 - **mssv:** Mã số sinh viên (MSSV).
 - **name:** Tên sinh viên.
 - **cpa:** Điểm CPA của sinh viên, được chuyển đổi thành kiểu số thực (float) để đảm bảo tính chính xác khi xử lý.
- **getYearOfEntry():**
 - Phương thức này trích xuất năm nhập học từ mã số sinh viên. Mã số sinh viên (mssv) thường có định dạng năm học và một số tự động sau đó (ví dụ: 20190001), do đó, phương thức này lấy 4 ký tự đầu tiên của mssv và chuyển đổi chúng thành số nguyên để đại diện cho năm nhập học.
 - Ví dụ, nếu mssv là 20190001, phương thức này sẽ trả về 2019.
- **getWarningLevel():**

Phương thức này xác định mức cảnh cáo của sinh viên dựa trên giá trị CPA. Các mức cảnh cáo được phân loại như sau:

 - **Mức 3:** Nếu $cpa \leq 0.5$.
 - **Mức 2:** Nếu $0.5 < cpa \leq 1.0$.
 - **Mức 1:** Nếu $1.0 < cpa \leq 1.5$.

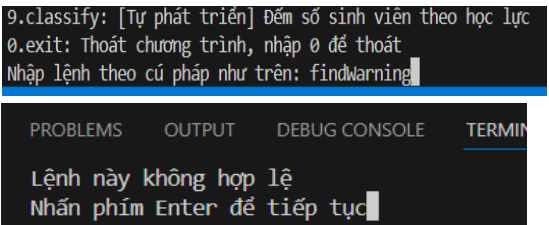
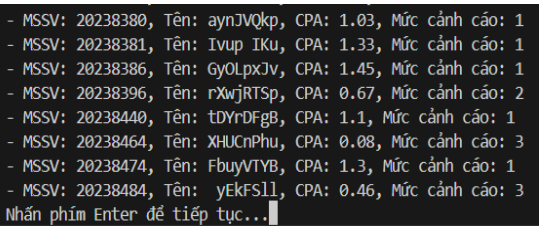
- **Không bị cảnh cáo:** Nếu $cpa > 1.5$.
- Phương thức trả về một giá trị mức cảnh cáo, hoặc 0 nếu sinh viên không bị cảnh cáo.

Lớp StudentManager

Lớp StudentManager đại diện cho một bộ quản lý sinh viên, có nhiệm vụ tổ chức và thao tác trên một danh sách các đối tượng Student. Lớp này chủ yếu chịu trách nhiệm về các chức năng quản lý và lọc danh sách sinh viên.

- **Hàm getWarningStudents:**
 - Duyệt qua danh sách students, kiểm tra mức cảnh cáo của từng sinh viên thông qua hàm getWarningLevel.
 - Chỉ thêm vào mảng warningStudents các sinh viên có mức cảnh cáo lớn hơn 0.
 - Sử dụng thuật toán quickSort để sắp xếp danh sách theo mức cảnh cáo giảm dần.
- **Hàm quickSort:**
 - Triển khai thuật toán Quicksort để sắp xếp danh sách warningStudents dựa trên trường warningLevel.
 - Sắp xếp danh sách arr từ vị trí low đến high.
 - Đề quy chia nhỏ danh sách và sắp xếp từng phần.
- **Hàm partition:**
 - Xác định vị trí của **pivot** trong danh sách và phân chia các phần tử lớn hơn sang trái, nhỏ hơn sang phải.
 - pivot: Chọn mức cảnh cáo của phần tử cuối cùng làm chốt.
 - i: Đánh dấu vị trí phần tử lớn hơn pivot.
 - Duyệt mảng: So sánh mức cảnh cáo từng phần tử với pivot:
 - Nếu lớn hơn pivot, hoán đổi phần tử đó với phần tử tại vị trí i.
 - Hoán đổi pivot: Đưa pivot về đúng vị trí giữa phần tử lớn hơn và nhỏ hơn.
 - Trả về: Chỉ số mới của pivot sau khi sắp xếp.
- **Hàm fromJSON:**
 - Chuyển đổi dữ liệu JSON thành danh sách đối tượng Student.

c) Nguyên lý hoạt động, vận hành:

<p>- Bước 1: Người dùng nhập lệnh findWarning, dòng thông báo “Nhập lệnh theo cú pháp như trên”, nếu gõ sai chương trình hiện thông báo không hợp lệ và bắt người dùng nhập lại.</p>	
<p>- Bước 2: Nếu gõ đúng cú pháp, chương trình sẽ chạy và hiện ra danh sách thông tin sinh viên cảnh cáo, mỗi dòng đều có thông tin MSSV, Tên, CPA và mức cảnh cáo</p> <p><i>Người dùng có thể ấn Enter để trở lại màn hình menu</i></p>	

4.2.7 Chức năng 7: Đếm số sinh viên có điểm cpa là nằm trong đoạn [a b] ($a \leq cpa \leq b$)

a) Ý tưởng:

- **Ý tưởng:** Đọc dữ liệu từ tệp JSON chứa thông tin sinh viên.
- Lọc sinh viên thỏa mãn điều kiện $\min CPA \leq CPA \leq \max CPA$.
- Trả về danh sách sinh viên phù hợp và hiển thị số lượng.
- **Mục đích:** Đếm số sinh viên có điểm cpa là nằm trong đoạn [a b] ($a \leq cpa \leq b$).
- **Đầu vào:** a, b (với a, b là số thực được nhập từ bàn phím) từ file main.js

- **Đầu ra:** Số lượng sinh viên có điểm CPA nằm trong khoảng trên
Số lượng sinh viên có CPA từ <a> đến là:

b) Cấu trúc:

Các phép toán ứng với thuật toán

- Phép so sánh: $cpa \geq minCPA \ \&\& \ cpa \leq maxCPA$ (kiểm tra điều kiện CPA).
- Phép gán: $filtered[index] = data[i]$ (thêm sinh viên vào danh sách).
- Phép chuyển đổi kiểu dữ liệu: $parseFloat$ (chuyển CPA từ chuỗi sang số thực).
- Phép duyệt: for (lặp qua toàn bộ danh sách sinh viên).

Giải nghĩa 1 số code:

$if (cpa \geq minCPA \ \&\& \ cpa \leq maxCPA)$: Kiểm tra CPA có nằm trong khoảng $[minCPA, maxCPA]$.

$filtered[index] = data[i]$: Thêm sinh viên thỏa mãn vào mảng $filtered$.

$index++$: Tăng chỉ số để thêm phần tử tiếp theo.

$return filtered$: Trả về danh sách sinh viên thỏa mãn.

Độ phức tạp thời gian:

- $O(n)$ với n là số lượng sinh viên trong danh sách.
- Lý do: Chỉ duyệt qua danh sách một lần.

Ưu điểm:

- Dễ hiểu và triển khai đơn giản.
- Dữ liệu lọc được trả về dưới dạng mảng, dễ xử lý tiếp.
- Phù hợp cho các tập dữ liệu nhỏ hoặc vừa.

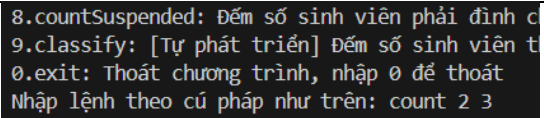
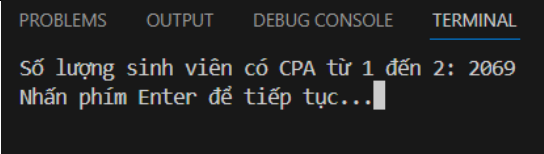
Nhược điểm:

- Khi số lượng sinh viên lớn, việc duyệt từng phần tử tốn thời gian.
- Dữ liệu cần được đọc từ file JSON trước khi xử lý, gây phụ thuộc vào file.

```
function filterByCPA(data, minCPA, maxCPA) {  
  const filtered = [];  
  let index = 0; // Cho index = 0 để bắt đầu đếm số lượng  
  for (let i = 0; i < data.length; i++) {  
    const cpa = parseFloat(data[i].cpa);  
    // Điều kiện của CPA  
    if (cpa >= minCPA && cpa <= maxCPA) {  
      filtered[index] = data[i]; // Thêm sinh viên vào mảng nếu mà CPA đó hợp lệ  
      index++;  
    }  
  }  
  return filtered;  
}
```

Hình 10: Một phần source code file count.js

c) Nguyên lý hoạt động, vận hành:

- Bước 1: Người dùng nhập lệnh <code>count <a> </code> dòng thông báo “Nhập lệnh theo cú pháp như trên”, nếu gõ sai chương trình hiện thông báo không hợp lệ và bắt người dùng nhập lại.	
- Bước 2: Nếu gõ đúng cú pháp, chương trình sẽ chạy và hiện ra số lượng sinh viên CPA trong khoảng trên <i>Người dùng có thể ấn Enter để trở lại màn hình menu</i>	

4.2.8 Chức năng 8: Tính số lượng sinh viên phải đình chỉ học. Điều kiện đình chỉ học là mức cảnh cáo dựa trên cpa và thời gian học tối đa cho phép (> 5 năm bị đình chỉ) (được tính dựa trên năm vào trường (dựa trên mssv) và thời điểm hiện tại mm/yyyy)

a) Ý tưởng:

- **Ý tưởng:** Đọc dữ liệu từ một tệp JSON (dataStudent.json) chứa danh sách sinh viên. Duyệt qua danh sách sinh viên, kiểm tra từng tiêu chí để xác định xem sinh viên có bị đình chỉ không. Tính tổng số lượng sinh viên bị đình chỉ và hiển thị kết quả.

- **Mục đích:** Tính số lượng sinh viên đình chỉ học theo điều kiện đề bài

- **Đầu vào:** Không có tham số đầu vào mà người dùng cần phải nhập ở file main.js

- **Đầu ra:** 3 dòng kết quả

Số lượng sinh viên bị đình chỉ do CPA ≤ 1.5 :

Số lượng sinh viên bị đình chỉ do thời gian học > 5 năm:

Tổng số sinh viên bị đình chỉ:

b) Cấu trúc:

- Khai báo Class (lớp) Student và StudentManager:

```
class Student {
  constructor(mssv, name, cpa) {
    this.mssv = mssv; // Sửa lại để đúng với tham số đầu vào
    this.name = name;
    this.cpa = parseFloat(cpa); // Đảm bảo CPA là kiểu số
  }
}
```

Hình 11: Một phần source code file countSuspended.js

Mô tả ngắn gọn cấu trúc code:

- Khởi tạo biến để bắt đầu đếm và năm hiện tại
 - lowCPA: Đếm số sinh viên có CPA ≤ 1.5 .
 - longDuration: Đếm số sinh viên học quá 5 năm.
- Sử dụng vòng lặp for để duyệt qua toàn bộ sinh viên trong danh sách
- Kiểm tra điều kiện của CPA xem có ≤ 1.5 hay là không
 - parseFloat: Chuyển đổi CPA từ chuỗi sang số thực.
 - Tăng lowCPA nếu CPA ≤ 1.5 .

Kiểm tra điều kiện thời gian học tối đa

- Tách năm nhập học từ MSSV (4 ký tự đầu).
- Tính thời gian học (currentYear - yearOfEntry)
- Tăng longDuration nếu vượt quá 5 năm.

Sau đó tính tổng số sinh viên bị đình chỉ theo 2 điều kiện trên

Ý nghĩa từng câu lệnh:

- countSuspended: Hàm thực hiện việc đếm số lượng sinh viên bị đình chỉ học tập.
- const filePath = './dataStudent.json';: Xác định đường dẫn tệp JSON chứa dữ liệu sinh viên.
- let students;: Khai báo biến students để lưu danh sách sinh viên sau khi đọc từ tệp JSON.
- const Data = fs.readFileSync(filePath, 'utf-8');: Đọc nội dung của tệp JSON dưới dạng chuỗi UTF-8.
- students = JSON.parse(Data);: Chuyển đổi chuỗi JSON sang mảng đối tượng sinh viên.
- let lowCPA = 0;: Tạo biến đếm lowCPA để lưu số sinh viên có CPA ≤ 1.5 .
- let longDuration = 0;: Tạo biến đếm longDuration để lưu số sinh viên có thời gian học > 5 năm.
- const currentYear = 2024;: Xác định năm hiện tại để tính thời gian học của sinh viên.

- for (let i = 0; i < students.length; i++) { ... }: Duyệt qua từng sinh viên trong danh sách.
- const student = students[i];: Lấy thông tin của sinh viên tại chỉ mục i.
- const cpa = parseFloat(student.cpa);: Chuyển CPA từ chuỗi sang số thực để dễ dàng so sánh.
- if (cpa <= 1.5) { lowCPA++; } : Kiểm tra nếu $CPA \leq 1.5$, tăng biến lowCPA lên 1.
- const mssv = student.mssv.toString();: Chuyển MSSV của sinh viên thành chuỗi để xử lý ký tự.
- const yearOfEntry = parseInt(mssv.substring(0, 4), 10);: Tách và chuyển đổi 4 ký tự đầu của MSSV thành năm nhập học.
- const studyDuration = currentYear - yearOfEntry;: Tính thời gian học của sinh viên bằng cách lấy năm hiện tại trừ đi năm nhập học.
- if (studyDuration > 5) { longDuration++; } : Kiểm tra nếu thời gian học > 5 năm, tăng biến longDuration lên 1.
- const totalSuspended = lowCPA + longDuration;: Tính tổng số sinh viên bị đình chỉ học tập.
- console.clear();: Xóa sạch màn hình console trước khi hiển thị kết quả.
- console.log(...);: Hiển thị thông tin chi tiết số lượng sinh viên bị đình chỉ vì từng lý do và tổng số.
- prompt("Nhấn phím Enter để tiếp tục...");: Chờ người dùng nhấn phím Enter để kết thúc chương trình.
- return;: Kết thúc hàm và thoát chương trình.

Các phép toán sử dụng

Phép toán số học: Tính thời gian học ($studyDuration = currentYear - yearOfEntry$).

Phép so sánh:

- Kiểm tra CPA ($cpa \leq 1.5$).
- Kiểm tra thời gian học ($studyDuration > 5$).

Phép chuyển đổi:

- parseFloat: Chuyển đổi chuỗi CPA sang số thực.
- parseInt: Tách năm nhập học từ MSSV.

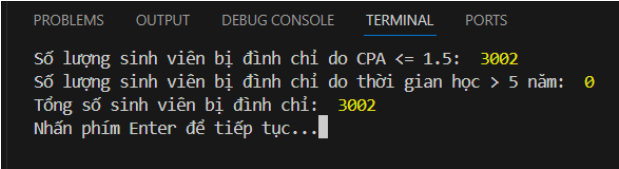
Độ phức tạp, ưu - nhược điểm, đánh giá

- Độ phức tạp:
 - + Thời gian: $O(n)$ (n là số lượng sinh viên, vì duyệt qua từng sinh viên).
 - + Không gian: $O(1)$ (không sử dụng bộ nhớ bổ sung đáng kể ngoài biến đếm).
- Ưu điểm:
 - + Đơn giản, dễ hiểu.
 - + Hoạt động hiệu quả với dữ liệu có kích thước vừa và nhỏ.
- Nhược điểm:
 - + Phụ thuộc vào định dạng chính xác của dữ liệu JSON và MSSV.
 - + Không xử lý các trường hợp dữ liệu thiếu hoặc lỗi logic (như MSSV không chuẩn).

c) Nguyên lý hoạt động, vận hành:

- **Bước 1:** Người dùng nhập lệnh countSuspended dòng thông báo “Nhập lệnh theo cú pháp như trên”, nếu gõ sai chương

```
8.countSuspended: Đếm số sinh viên phải đình chỉ học kèm điều kiện
9.classify: [Tự phát triển] Đếm số sinh viên theo học lực
0.exit: Thoát chương trình, nhập 0 để thoát
Nhập lệnh theo cú pháp như trên: countSuspended
```

trình hiện thông báo không hợp lệ và bắt người dùng nhập lại.	
- Bước 2: Nếu gõ đúng cú pháp, chương trình sẽ chạy và hiện ra 3 dòng: - Số lượng sinh viên bị đình chỉ do CPA ≤ 1.5 - Số lượng sinh viên bị đình chỉ do thời gian học ≥ 5 năm - Tổng số sinh viên bị đình chỉ <i>Người dùng có thể ấn Enter để trở lại màn hình menu</i>	

4.2.8 Chức năng 9: Tự phát triển thêm – Phân loại học lực theo điểm

a) Ý tưởng:

- Đọc dữ liệu sinh viên từ tệp JSON (dataStudent.json).
- Phân loại học lực dựa trên CPA của từng sinh viên theo các mức: "Xuất sắc", "Giỏi", "Khá", "Trung bình", và "Yếu".
- Tính toán số lượng sinh viên thuộc từng mức học lực.
- In kết quả ra màn hình.

b) Cấu trúc:

- Lớp Student

Phương thức `getAcademicPerformance()`:

Xác định học lực của sinh viên dựa trên CPA:

- $CPA \geq 3.6$: "Xuất sắc".
- $3.2 \leq CPA < 3.6$: "Giỏi".
- $2.5 \leq CPA < 3.2$: "Khá".
- $2.0 \leq CPA < 2.5$: "Trung bình".
- $CPA < 2.0$: "Yếu".

- Lớp Student

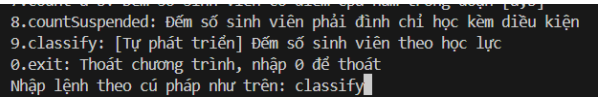
Quản lý danh sách sinh viên thông qua mảng `students`.

Phương thức `getAcademicPerformanceCounts()`:

Duyệt qua danh sách sinh viên để tính số lượng thuộc từng mức học lực.

Phương thức tính `fromJSON(jsonData)`:

c) Nguyên lý hoạt động, vận hành:

- Bước 1: Người dùng nhập lệnh <code>classify</code> dòng thông báo “Nhập lệnh theo cú pháp như trên”, nếu gõ sai chương trình hiện thông báo không hợp lệ và bắt người dùng nhập lại.	
---	--

- **Bước 2:** Nếu gõ đúng cú pháp, chương trình sẽ chạy và hiện ra các dòng số lượng sinh viên theo mức Xuất sắc, Giỏi, Khá, Trung bình, Yếu
Người dùng có thể ấn Enter để trở lại màn hình menu

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Số lượng sinh viên theo học lực:
Xuất sắc: 800
Giỏi: 822
Khá: 1391
Trung bình: 963
Yếu: 4023
Nhấn phím Enter để tiếp tục...

Hình 12: Một phần source code file classify.js

PHẦN V. TRIỂN KHAI CÀI ĐẶT VÀ ĐÓNG GÓI

- Tổ chức chương trình:

- + Dự án gồm các file main.js (file chính), các file chức năng và file readme.txt (thông tin thành viên và dự án)
- + Ở mỗi file chức năng, nhóm đều sử dụng các lớp (**Class**) gồm: Lớp Student (Lớp này sẽ lưu trữ thông tin của từng sinh viên, bao gồm MSSV, họ tên, điểm CPA và mức cảnh cáo), Lớp StudentManager: Lớp này sẽ quản lý danh sách sinh viên, cung cấp các phương thức để tìm kiếm, thay đổi điểm CPA, tính toán mức cảnh cáo, và thực hiện các thao tác như tìm kiếm sinh viên có CPA cao nhất, thấp nhất, và các sinh viên bị đình chỉ học.
- + Mỗi lớp, tùy theo file chức năng sẽ chứa các **phương thức, các hàm** để thực hiện nhiệm vụ xử lý chức năng theo yêu cầu bài toán. Ví dụ, file findTop.js sử dụng phương thức như getTopNLowestCPA(n), chức các hàm như .slice(); .sort(a,b); slice(0,n) để tạo bản sao cho mảng Student gốc, sắp xếp theo thứ tự giảm dần và lấy mảng từ 0 đến vị trí n để lấy n sinh viên có CPA cao nhất
- + Các thành viên trong nhóm mỗi người viết code ở file riêng lẻ khác nhau, mỗi file code xử lý 1 chức năng theo ứng với tên file của nó, sau đó nhóm trưởng kết hợp với các file thành viên trong nhóm để xây dựng bộ chương trình hoàn chỉnh.

- Đóng gói:

- + Toàn bộ file main.js, file chức năng, thư viện, package.json, file dữ liệu sinh viên,... đều được đóng gói thành tệp zip: Group 7 – Ma lop 155470.zip
- + Bài báo cáo: Group 7 – Ma lop 155470.pdf.

PHẦN VI. KẾT QUẢ THỰC NGHIỆM

6.1 Bộ dữ liệu:

- Dữ liệu được lưu trữ tại file dataStudent.json, nằm cùng với thư mục chứa file bài làm của nhóm



Hình 13: Ảnh data file dataStudent.json

6.2 Kết quả thực nghiệm:

- Chức năng 1: Hiển thị danh sách sinh viên

```
MSSV: 20238477, họ và tên: QSRcwJXN
MSSV: 20238484, họ và tên: yEkFSll
MSSV: 20238496, họ và tên: hIpycguQ
Nhấn phím Enter để tiếp tục...
```

- Chức năng 2: Tìm kiếm sinh viên 20190506

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
MSSV: 20190506 - Tên: yDeUazIn - CPA: 3.20 - Mức cảnh cáo: 0
Nhấn phím Enter để tiếp tục...
```

- Chức năng 3: Thay đổi CPA sinh viên 20239944 từ 0.96 thành 3.2

```
[Running] node "c:\Users\DELL\Downloads\Student management\Function 3 SV.js"
Đã cập nhật CPA của sinh viên ID: 20239944 (WtQZnWxYfYFvVdxS) từ 0.96 thành 3.2
Dữ liệu đã được lưu vào file: dataStudent.json
```

- Chức năng 4: Tìm 10 sinh viên có CPA cao nhất

```
Danh sách 10 sinh viên có CPA cao nhất:
1. MSSV: 20194912, Tên: KbGPwyZf, CPA: 4
2. MSSV: 20197260, Tên: LeAjgIkd, CPA: 4
3. MSSV: 20203416, Tên: KrLnSpCf, CPA: 4
4. MSSV: 20215446, Tên: EygJQLQy, CPA: 4
5. MSSV: 20217278, Tên: XdjjbNDW, CPA: 4
6. MSSV: 20221524, Tên: AjQnNbPC, CPA: 4
7. MSSV: 20222032, Tên: RJdNtvPj, CPA: 4
8. MSSV: 20223431, Tên: tgeOJEIu, CPA: 4
9. MSSV: 20225665, Tên: VFNgomTW, CPA: 4
10. MSSV: 20225808, Tên: OaSGggh, CPA: 4
Nhấn phím Enter để tiếp tục...
```

- Chức năng 5: Tìm 10 sinh viên có CPA thấp nhất

```
Danh sách 10 sinh viên có CPA thấp nhất:
1. MSSV: 20191195, Tên: XFSibGYU, CPA: 0
2. MSSV: 20193601, Tên: bvQZjvwD, CPA: 0
3. MSSV: 20193790, Tên: iZFwWEnw, CPA: 0
4. MSSV: 20193859, Tên: cExQUuSG, CPA: 0
5. MSSV: 20203786, Tên: dmYpmoED, CPA: 0
6. MSSV: 20208057, Tên: mexQubbH, CPA: 0
7. MSSV: 20211221, Tên: rpKCTv X, CPA: 0
8. MSSV: 20211278, Tên: UBHffvG1, CPA: 0
9. MSSV: 20211753, Tên: ZcJvI gS, CPA: 0
10. MSSV: 20221707, Tên: RpIOLymt, CPA: 0
Nhấn phím Enter để tiếp tục...
```

- Chức năng 6: Hiển thị danh sách sinh viên bị cảnh cáo kèm mức cảnh cáo

```
- MSSV: 20238396, Tên: PxmJRTsp, CPA: 0.67, Mức cảnh cáo: 2
- MSSV: 20238440, Tên: tDYrDFgb, CPA: 1.1, Mức cảnh cáo: 1
- MSSV: 20238464, Tên: XHUCnPhu, CPA: 0.08, Mức cảnh cáo: 3
- MSSV: 20238474, Tên: FbuyVTyB, CPA: 1.3, Mức cảnh cáo: 1
- MSSV: 20238484, Tên: yEkFSll, CPA: 0.46, Mức cảnh cáo: 3
Nhấn phím Enter để tiếp tục...
```

- Chức năng 7: Đếm số sinh viên có CPA từ 2 đến 3

```
Số lượng sinh viên có CPA từ 2 đến 3: 1983
Nhấn phím Enter để tiếp tục...
```

- Chức năng 8: Tính số lượng sinh viên bị đình chỉ học

```
Số lượng sinh viên bị đình chỉ do CPA <= 1.5: 3002
Số lượng sinh viên bị đình chỉ do thời gian học > 5 năm: 0
Tổng số sinh viên bị đình chỉ: 3002
Nhấn phím Enter để tiếp tục...
```

- Chức năng 9: Tự phát triển: Đếm số sinh viên theo học lực

```
Số lượng sinh viên theo học lực:
Xuất sắc: 800
Giỏi: 822
Khá: 1391
Trung bình: 963
Yếu: 4023
Nhấn phím Enter để tiếp tục...
```


PHẦN VII. KẾT LUẬN

7.1 Đánh giá mức độ hoàn thành:

Từ kết quả thực nghiệm, chúng em thấy rằng:

- **Tính hoạt động và tính ổn định:** Ứng dụng hoạt động ổn định các chức năng một cách bình thường, đặc biệt khi xử lý dữ liệu với số lượng sinh viên lớn (trên 100 sinh viên), không gặp phải sự cố hoặc lỗi khi thực hiện các phép tính. Hệ thống có khả năng xử lý và trả về kết quả nhanh chóng.

- **Tính chính xác:** Kết quả thực nghiệm cho thấy các chức năng xử lý chính xác yêu cầu đầu vào và đầu ra. Cụ thể, chức năng đếm số sinh viên trong khoảng CPA và chức năng đếm sinh viên bị đình chỉ học hoặc vượt quá thời gian học đều trả về kết quả chính xác, phù hợp với dữ liệu đã chuẩn bị.

- **Tính mở rộng:** Các chức năng của ứng dụng có thể dễ dàng mở rộng và điều chỉnh để xử lý các yêu cầu khác trong tương lai, như thêm các chỉ số học tập mới hoặc cải thiện thuật toán tính toán.

Về ưu, nhược điểm:

Ưu điểm	Nhược điểm
<ul style="list-style-type: none">- Project của nhóm đã hoàn thành đúng các yêu cầu đề bài, chạy ổn định, không gặp lỗi và hoạt động với lượng data tương đối lớn.- Các thành viên trong nhóm đã biết ứng dụng kiến thức môn học, để hoàn thành dự án.- Project cũng sử dụng các class, async, các phương thức để thao tác dữ liệu. Bằng cách này, chương trình không chỉ đáp ứng các yêu cầu bài toán mà còn đảm bảo tính mở rộng và dễ bảo trì. Việc áp dụng kỹ thuật async giúp tối ưu hóa hiệu suất khi làm việc với các thao tác IO hoặc các tác vụ xử lý dữ liệu lớn.- Phân tách các chức năng rõ ràng giữa các lớp (Lớp Student và lớp StudentManager).- Hỗ trợ việc xử lý dữ liệu sinh viên từ tệp JSON, thông báo lệnh khi tệp lỗi, người dùng nhập sai,... và giao diện menu để điều hướng, cho phép người dùng vào/ ra dễ dàng.	<ul style="list-style-type: none">- Code cần phải rút ngắn và tối giản hơn- Cần cải tiến thêm tính linh hoạt, chẳng hạn như cho phép người dùng nhập đường dẫn tệp hoặc lựa chọn mức cảnh cáo để lọc sinh viên.- Dù có sử dụng async cho các thao tác trên dữ liệu sinh viên, nhưng việc đọc tệp JSON thông qua fs.readFileSync là một thao tác đồng bộ, điều này có thể gây gián đoạn và ảnh hưởng đến hiệu suất của chương trình, đặc biệt khi có nhiều sinh viên và tệp dữ liệu lớn.- Mặc dù chương trình đã sử dụng async cho việc đọc tệp nhưng các thao tác như tìm kiếm, thay đổi CPA, tính toán mức cảnh cáo... không sử dụng async khi cần thiết. Điều này có thể khiến chương trình thiếu tối ưu khi xử lý các tác vụ tính toán phức tạp.

Những khó khăn khi làm việc:

- Lượng kiến thức môn học lớn, nhóm loay hoay trong việc lựa chọn kiến thức.
- Khó khăn khi sử dụng các phương thức, sử dụng các hàm nâng cao
- Lỗi sử dụng các thư viện phù hợp với mô đun ES được sử dụng trong toàn bộ chương trình
- Lỗi đồng bộ các tệp với nhau trong giai đoạn ghép các file vào project chung, lệch đường dẫn,...

Bài học rút ra:

- Lập kế hoạch chi tiết: Xây dựng kế hoạch cụ thể với các mốc thời gian rõ ràng, phân chia công việc hợp lý và theo dõi tiến độ hàng tuần và sử dụng công cụ quản lý dự án, theo dõi tiến độ và đảm bảo mọi người đều nắm được tình hình dự án.

- Thực hiện đánh giá tiến độ định kỳ: Đặt các mốc tiến độ và yêu cầu nhóm báo cáo tình trạng công việc theo các giai đoạn.

- Cần tìm hiểu kỹ về các hàm, kỹ thuật, phương thức nâng cao để đồng bộ kỹ thuật được sử dụng trong file bài làm giữa các thành viên trong nhóm

7.2 Kiến nghị, giải pháp:

a) Kết luận:

- Bài toán quản lý kết quả học tập của sinh viên ĐHBK Hà Nội trong kỳ 20232 yêu cầu xây dựng một hệ thống có khả năng quản lý, tra cứu, sửa đổi thông tin sinh viên và tính toán các thông tin liên quan đến điểm CPA và mức cảnh cáo của sinh viên. Các yêu cầu đã được phân chia thành nhiều lệnh, từ việc liệt kê danh sách sinh viên, tìm kiếm sinh viên theo mã số, thay đổi điểm CPA, đến việc tính toán các chỉ số như cảnh cáo và xác định số lượng sinh viên bị đình chỉ học.
- Kết quả thực hiện bài toán này cho thấy việc áp dụng các lớp (classes) trong việc tổ chức dữ liệu và các thao tác liên quan đến sinh viên là một cách hiệu quả và có cấu trúc rõ ràng. Việc sử dụng các phương thức như tìm kiếm sinh viên, thay đổi điểm CPA, và tính toán cảnh cáo, cũng như các thao tác liên quan đến dữ liệu sinh viên trong một kỳ học cụ thể, giúp đơn giản hóa và quản lý tốt hơn thông tin trong hệ thống.

b) Kiến nghị, giải pháp:

- Cải thiện khả năng mở rộng của hệ thống: Hệ thống hiện tại chỉ xử lý dữ liệu cho một kỳ học cụ thể. Tuy nhiên, để hệ thống có thể áp dụng cho nhiều kỳ học hoặc quản lý thông tin của nhiều sinh viên hơn, cần mở rộng khả năng lưu trữ và xử lý dữ liệu.
- Tối ưu hóa hiệu suất với các thao tác lớn: Một số thao tác như tìm kiếm, lọc sinh viên theo CPA hoặc cảnh cáo có thể trở nên chậm khi có một lượng dữ liệu lớn. Cần phải tối ưu hóa các thuật toán tìm kiếm, sắp xếp dữ liệu để giảm thiểu thời gian thực thi. Ví dụ, sử dụng các cấu trúc dữ liệu như bảng băm hoặc danh sách liên kết để tìm kiếm nhanh hơn thay vì sử dụng các vòng lặp tuyến tính.
- Giao diện người dùng thân thiện: Sau này bài toán được triển khai trong thực tế, cần có một giao diện người dùng (UI) dễ sử dụng để người quản lý hoặc giảng viên có thể dễ dàng thao tác trên dữ liệu mà không cần phải can thiệp vào mã nguồn.
- Hệ thống này có thể mở rộng thêm với các chức năng quản lý sinh viên khác, đồng thời dễ dàng tích hợp vào các ứng dụng hoặc hệ thống lớn hơn của trường Đại học Bách Khoa Hà Nội trong tương lai.

PHẦN VIII. DANH MỤC TÀI LIỆU THAM KHẢO

- Tài liệu trên Microsoft Teams
- Tài liệu W3School: <https://www.w3schools.com/js/default.asp>
- Tài liệu https://www.w3schools.com/js/js_html_dom_document.asp
- Tài liệu <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- <https://github.com/Ren0503/javascript-algorithms-and-data-structure>