BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

ĐỒ ÁN MÔN HỌC

THU THẬP VÀ PHÂN TÍCH DỮ LIỆU ĐIỂM THI THPT QUỐC GIA SỬ DỤNG CÔNG CỤ SELENIUM VÀ SQLITE

Ngành: **KHOA HỌC DỮ LIỆU**

Môn học: **MÃ NGUỒN MỞ**

Giảng viên hướng dẫn : ThS. Lê Nhật Tùng

Sinh viên thực hiện :

2386400052-Nguyễn Đức Vinh

2386400966-Phan Xuân Dương

2386400026-Nguyễn Đăng Khoa

Lớp: 23DKHA1

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TPHCM, Ngày….. Tháng…...Năm 2024

Giáo viên hướng dẫn

(Ký tên, đóng dấu)

DANH MỤC CÁC KÝ HIỆU,HÌNH ẢNH, BẢNG BIỂU, TỪ VIẾT TẮT VÀ TỪ KHÓA..... 6
1. Danh mục từ viết tắt..... 6
2. Danh mục ký hiệu..... 7
3. Danh mục từ khóa..... 7
4. Danh mục hình ảnh..... 8
5. Danh mục bảng biểu..... 9
CHƯƠNG 1: TỔNG QUAN..... 10
1.1. Giới thiệu đề tài..... 10
1.2. Nhiệm vụ của đề tài..... 10
1.2.1. Tính cấp thiết của đề tài..... 11
1.2.2. Ý nghĩa khoa học và thực tiễn của đề tài..... 12
1.3. Mục tiêu..... 12
1.3.2. Mục tiêu cụ thể..... 13
1.4. Đối tượng và phạm vi..... 14
1.4.1. Đối tượng..... 14
1.4.2. Phạm vi..... 14
1.5. Phương pháp nghiên cứu..... 15
1.5.1. Phương pháp nghiên cứu tài liệu..... 15
1.5.2. Phương pháp phân tích trang web (Web Analysis)..... 16
1.5.3. Phương pháp thực nghiệm..... 16
1.5.4. Phương pháp kết hợp công cụ (Selenium)..... 16
1.5.5. Phương pháp đánh giá và phân tích dữ liệu..... 17
1.6. Những đóng góp nghiên cứu của đề tài..... 17
1.6.1. Trong lĩnh vực học thuật..... 17
1.6.2. Trong thực tiễn..... 18
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT..... 19
2.1 Tổng quan về kỹ thuật cào dữ liệu (Web Scraping)..... 19
2.1.1 Định nghĩa và Quy trình vận hành tự động..... 19
2.1.2 Phân loại và Công nghệ hỗ trợ trích xuất..... 20
2.2 Ngôn ngữ lập trình Python..... 22
2.2.1 Python là gì?..... 22
2.2.2 Đặc điểm cốt lõi của Python..... 22
2.2.3 Cài đặt và chuẩn bị môi trường..... 23
2.3 Thư viện tự động hóa Selenium..... 23

2.3.1 Kiến trúc và Cơ chế WebDriver.....23
2.3.2 Chuẩn bị môi trường và cài đặt.....25
2.3.3 Ưu điểm và Nhược điểm của Selenium.....26
2.3.4 Tối ưu hóa Selenium trong thu thập dữ liệu.....27
2.4 Hệ quản trị cơ sở dữ liệu SQLite.....28
2.4.1 Đặc điểm của Cơ sở dữ liệu nhúng (Embedded Database).....28
2.4.2 So sánh giải pháp lưu trữ: SQLite vs MongoDB.....29
CHƯƠNG 3: PHƯƠNG PHÁP THỰC NGHIỆM.....31
3.1 Sơ đồ mô tả quá trình thu thập dữ liệu.....31
3.2 Phương pháp thu thập dữ liệu.....32
3.2.1 Một số thư viện cần dùng.....32
3.2.2 .khởi tạo môi trường đa luồng.....32
3.2.3 Thu thập dữ liệu.....33
3.3 lưu vào SQLITE.....36
3.4 Phương pháp phân tích dữ liệu.....38
3.4.1 Thông tin tổng quan.....38
3.4.2 Các biểu đồ trực quan.....38
CHƯƠNG 4: KẾT LUẬN VÀ KIẾN NGHỊ.....40
TÀI LIỆU THAM KHẢO.....41
PHỤ LỤC 1.....43
PHỤ LỤC 2.....51

DANH MỤC CÁC KÝ HIỆU,HÌNH ẢNH, BẢNG BIỂU, TỪ VIẾT TẮT VÀ TỪ KHÓA

1. Danh mục từ viết tắt

Từ viết tắt	Ý nghĩa
HTML	HyperText Markup Language – Ngôn ngữ đánh dấu siêu văn bản
DOM	Document Object Model – Mô hình đối tượng tài liệu
HTTP	HyperText Transfer Protocol – Giao thức truyền tải siêu văn bản
HTTPS	HyperText Transfer Protocol Secure – Giao thức HTTP bảo mật
API	Application Programming Interface – Giao diện lập trình ứng dụng
RDBMS	Relational Database Management System – Hệ quản trị CSDL quan hệ
SQL	Structured Query Language – Ngôn ngữ truy vấn có cấu trúc
GUI	Graphical User Interface – Giao diện đồ họa người dùng
SPA	Single Page Application – Ứng dụng một trang
RAM	Random Access Memory – Bộ nhớ truy cập ngẫu nhiên
CPU	Central Processing Unit – Bộ xử lý trung tâm
IDE	Integrated Development Environment – Môi trường phát triển tích hợp

2. Danh mục ký hiệu

Ký hiệu	Diễn giải
GET	Phương thức HTTP dùng để yêu cầu dữ liệu từ máy chủ
POST	Phương thức HTTP dùng để gửi dữ liệu lên máy chủ
XPath	Ngôn ngữ truy vấn dùng để định vị phần tử trong tài liệu XML/HTML
CSS Selector	Bộ chọn CSS dùng để truy xuất phần tử HTML
.db	Phần mở rộng tệp cơ sở dữ liệu SQLite
.csv	Định dạng tệp dữ liệu dạng bảng (Comma-Separated Values)
.json	Định dạng dữ liệu JavaScript Object Notation

3. Danh mục từ khóa

Từ khóa	Mô tả
Web Scraping	Kỹ thuật tự động thu thập dữ liệu từ website
Static Scraping	Cào dữ liệu từ website tĩnh
Dynamic Scraping	Cào dữ liệu từ website động có JavaScript
Selenium	Thư viện tự động hóa trình duyệt

WebDriver	Thành phần điều khiển trình duyệt trong Selenium
Headless Browser	Trình duyệt chạy không giao diện
Explicit Wait	Cơ chế chờ có điều kiện trong Selenium
Python	Ngôn ngữ lập trình sử dụng trong đồ án
SQLite	Hệ quản trị cơ sở dữ liệu nhúng
ACID	Tập hợp các thuộc tính đảm bảo tính toàn vẹn dữ liệu
B-Tree	Cấu trúc dữ liệu lưu trữ trong SQLite
Anti-bot	Cơ chế phát hiện và ngăn chặn bot của website

4. Danh mục hình ảnh

Hình 2.1 Quá trình Web Scraping

Hình 2.3 Selenium

Hình 3.1 Sơ đồ mô tả quá trình thu thập dữ liệu

Hình 3.2 Hình ảnh minh họa thư viện

Hình 3.3 Hàm khởi tạo môi trường đa luồng

Hình 3.4 Hàm truy cập và thu thập dữ liệu

Hình 3.5 Xử lý nếu có popup

Hình 3.6 Tìm năm điểm và lấy tất cả các dòng

Hình 3.7 Gom dữ liệu và check dữ liệu

Hình 3.8 Định hướng generator

Hình 3.9 Giải phóng tài nguyên

Hình 3.10 Hàm kết nối với SQL

Hình 3.11 Tạo hàm khởi tạo database (tạo bảng, cấu hình SQLite)

Hình 3.12 : Biểu đồ số lượng thí sinh không thi môn

Hình 3.13 Biểu đồ phân bố điểm toán THPTQG 2025

Hình 3.14 Biểu đồ top tính có điểm trung bình môn toán cao nhất

5. Danh mục bảng biểu

Bảng 2.1 So sánh SQLite và MongoDB

CHƯƠNG 1: TỔNG QUAN

1.1. Giới thiệu đề tài

Trong những năm gần đây, dữ liệu điểm thi THPT quốc gia và điểm chuẩn tuyển sinh của các trường đại học ngày càng được công bố rộng rãi trên các trang thông tin điện tử. Tuy nhiên, các dữ liệu này thường tồn tại ở dạng rời rạc, phân tán trên nhiều website khác nhau và chưa được tổ chức một cách hệ thống, gây khó khăn cho việc tổng hợp, phân tích và khai thác thông tin phục vụ cho học sinh, phụ huynh cũng như các nhà nghiên cứu giáo dục.

Bên cạnh đó, với sự phát triển mạnh mẽ của khoa học dữ liệu và công nghệ thông tin, việc tự động hóa quá trình thu thập dữ liệu từ các website và lưu trữ vào cơ sở dữ liệu đã trở thành một xu hướng tất yếu. Công cụ Selenium cho phép mô phỏng hành vi người dùng trên trình duyệt, từ đó hỗ trợ việc thu thập dữ liệu web một cách linh hoạt và hiệu quả. Đồng thời, SQLite là một hệ quản trị cơ sở dữ liệu nhẹ, dễ triển khai, phù hợp cho các ứng dụng vừa và nhỏ trong việc lưu trữ và xử lý dữ liệu.

Xuất phát từ thực tế trên, đề tài "Thu thập và phân tích dữ liệu điểm thi THPT và điểm chuẩn các trường đại học sử dụng công cụ Selenium và SQLite" được thực hiện nhằm xây dựng một quy trình tự động thu thập, lưu trữ và phân tích dữ liệu, góp phần hỗ trợ người dùng trong việc tra cứu thông tin, so sánh điểm số và đưa ra các nhận định khách quan về tình hình tuyển sinh đại học.

1.2. Nhiệm vụ của đề tài

Đề tài được thực hiện nhằm xây dựng một hệ thống thu thập, lưu trữ và phân tích dữ liệu điểm thi THPT và điểm chuẩn của các trường đại học

một cách tự động và hiệu quả. Thông qua việc ứng dụng công cụ Selenium và hệ quản trị cơ sở dữ liệu SQLite, đề tài hướng đến việc giải quyết các hạn chế trong việc khai thác dữ liệu tuyển sinh hiện nay, đồng thời hỗ trợ người dùng trong việc tra cứu và phân tích thông tin phục vụ cho quá trình định hướng và lựa chọn ngành học, trường học.

Để đạt được mục tiêu trên, đề tài tập trung thực hiện các nhiệm vụ sau:

- Khảo sát và phân tích nguồn dữ liệu điểm thi THPT và điểm chuẩn các trường đại học.

- Xây dựng chương trình thu thập dữ liệu tự động từ các website bằng công cụ Selenium.

- Thiết kế và triển khai cơ sở dữ liệu SQLite để lưu trữ dữ liệu thu thập được.

- Thực hiện các truy vấn và phân tích dữ liệu nhằm rút ra các thông tin thống kê cần thiết.

- Đánh giá kết quả thu được và đề xuất hướng phát triển cho hệ thống trong tương lai.

1.2.1. Tính cấp thiết của đề tài

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ trong lĩnh vực giáo dục, nhu cầu khai thác và phân tích dữ liệu ngày càng trở nên quan trọng. Hàng năm, số lượng thí sinh tham gia kỳ thi THPT quốc gia rất lớn,

kéo theo một khối lượng dữ liệu điểm thi và điểm chuẩn tuyển sinh đồ sộ. Tuy nhiên, các dữ liệu này hiện nay chủ yếu được công bố dưới dạng bảng biểu trên các website, chưa được chuẩn hóa và tích hợp để phục vụ cho việc phân tích chuyên sâu.

Việc thu thập dữ liệu thủ công không chỉ tốn nhiều thời gian, công sức mà còn dễ xảy ra sai sót. Do đó, việc ứng dụng công cụ tự động hóa như Selenium để thu thập dữ liệu và sử dụng SQLite để lưu trữ, quản lý dữ liệu là hết sức cần thiết. Đề tài đáp ứng nhu cầu thực tiễn trong việc xây dựng một hệ thống thu thập và phân tích dữ liệu điểm thi, góp phần nâng cao hiệu quả khai thác thông tin tuyển sinh.

1.2.2. Ý nghĩa khoa học và thực tiễn của đề tài

Về ý nghĩa khoa học, đề tài góp phần làm rõ quy trình thu thập, lưu trữ và phân tích dữ liệu web trong lĩnh vực giáo dục. Việc kết hợp giữa Selenium và SQLite giúp minh họa một mô hình ứng dụng công nghệ thông tin vào xử lý dữ liệu thực tế, qua đó hỗ trợ cho việc nghiên cứu, học tập và giảng dạy các môn học liên quan đến khoa học dữ liệu, lập trình và cơ sở dữ liệu.

Về ý nghĩa thực tiễn, kết quả của đề tài giúp tạo ra một hệ thống có khả năng tự động thu thập và quản lý dữ liệu điểm thi và điểm chuẩn một cách khoa học, chính xác và dễ sử dụng. Hệ thống này hỗ trợ học sinh, phụ huynh và nhà trường trong việc tra cứu, so sánh và phân tích dữ liệu, từ đó đưa ra các quyết định phù hợp trong công tác tư vấn và định hướng tuyển sinh đại học.

1.3. Mục tiêu

Đề tài được thực hiện nhằm xây dựng một hệ thống thu thập và phân tích dữ liệu điểm thi THPT và điểm chuẩn các trường đại học dựa trên công nghệ tự động hóa và cơ sở dữ liệu. Thông qua việc ứng dụng công cụ Selenium và SQLite, đề tài hướng tới việc hỗ trợ người dùng trong việc khai thác thông tin tuyển sinh một cách nhanh chóng, chính xác và hiệu quả.

1.3.1. Mục tiêu tổng quát

Xây dựng một hệ thống tự động thu thập, lưu trữ và phân tích dữ liệu điểm thi THPT và điểm chuẩn của các trường đại học, góp phần hỗ trợ công tác tra cứu, so sánh và phân tích thông tin tuyển sinh phục vụ cho việc định hướng và lựa chọn ngành học, trường học.

1.3.2. Mục tiêu cụ thể

Để đạt được mục tiêu tổng quát, đề tài tập trung thực hiện các mục tiêu cụ thể sau:

- Nghiên cứu nguyên lý hoạt động và cách sử dụng công cụ Selenium trong việc thu thập dữ liệu từ các website.

- Xây dựng chương trình tự động thu thập dữ liệu điểm thi THPT và điểm chuẩn tuyển sinh của các trường đại học.

- Thiết kế và xây dựng cơ sở dữ liệu bằng SQLite để lưu trữ và quản lý dữ liệu thu thập được.

- Thực hiện các truy vấn và phân tích dữ liệu nhằm thống kê, so sánh và rút ra các thông tin cần thiết.

- Đánh giá hiệu quả của hệ thống và đề xuất các hướng cải tiến, mở

rộng trong tương lai.

1.4. Đối tượng và phạm vi

Đề tài tập trung nghiên cứu việc thu thập và phân tích dữ liệu điểm thi THPT và điểm chuẩn tuyển sinh của các trường đại học tại Việt Nam thông qua các công cụ công nghệ thông tin, nhằm phục vụ cho công tác phân tích và khai thác thông tin tuyển sinh.

1.4.1. Đối tượng

Đối tượng nghiên cứu của đề tài bao gồm:

- Dữ liệu điểm thi THPT quốc gia của thí sinh.
- Dữ liệu điểm chuẩn tuyển sinh của các trường đại học.
- Các công cụ và kỹ thuật phục vụ cho việc thu thập và phân tích dữ liệu, bao gồm Selenium và hệ quản trị cơ sở dữ liệu SQLite.

1.4.2. Phạm vi

- Về nội dung: Đề tài tập trung vào việc thu thập, lưu trữ và phân tích dữ liệu điểm thi THPT và điểm chuẩn của một số trường đại học tiêu biểu, không đi sâu vào toàn bộ các phương thức tuyển sinh khác.
- Về không gian: Dữ liệu được thu thập từ các website công bố thông tin điểm thi và điểm chuẩn của các trường đại học tại Việt Nam.

- Về thời gian: Dữ liệu được thu thập năm 2025, phục vụ cho mục đích nghiên cứu và minh họa cho đề tài.

1.5. Phương pháp nghiên cứu

Để đảm bảo đề tài được triển khai một cách khoa học, logic và đạt được các mục tiêu đề ra, nhóm nghiên cứu đã lựa chọn và kết hợp nhiều phương pháp nghiên cứu khác nhau. Việc sử dụng đa dạng các phương pháp giúp tiếp cận vấn đề từ nhiều góc độ, đồng thời nâng cao độ chính xác và độ tin cậy của kết quả nghiên cứu. Các phương pháp nghiên cứu được áp dụng xuyên suốt trong quá trình khảo sát, xây dựng hệ thống, thu thập dữ liệu và phân tích kết quả.

1.5.1. Phương pháp nghiên cứu tài liệu

Phương pháp nghiên cứu tài liệu được sử dụng nhằm thu thập, tổng hợp và phân tích các tài liệu liên quan đến nội dung đề tài. Các tài liệu bao gồm văn bản, quy chế của Bộ Giáo dục và Đào tạo về kỳ thi THPT quốc gia; các công trình nghiên cứu, bài báo khoa học liên quan đến thu thập và phân tích dữ liệu; tài liệu hướng dẫn sử dụng các công cụ và công nghệ như Selenium, SQLite và ngôn ngữ SQL.

Thông qua việc nghiên cứu tài liệu, nhóm nghiên cứu xây dựng được cơ sở lý thuyết vững chắc, nắm bắt được các phương pháp tiếp cận đã được áp dụng trước đó, từ đó định hướng đúng đắn cho việc triển khai đề tài và hạn chế các sai sót trong quá trình thực hiện.

1.5.2. Phương pháp phân tích trang web (Web Analysis)

Phương pháp phân tích trang web được sử dụng để khảo sát và đánh giá cấu trúc, cách tổ chức nội dung cũng như cơ chế hiển thị dữ liệu trên các website công bố điểm thi THPT và điểm chuẩn tuyển sinh của các trường đại học. Nhóm nghiên cứu tiến hành phân tích mã nguồn HTML, các thẻ dữ liệu, bảng biểu và các thành phần động trên trang web.

Kết quả của quá trình phân tích giúp xác định chính xác các vị trí chứa dữ liệu cần thu thập, từ đó xây dựng thuật toán thu thập dữ liệu phù hợp, đảm bảo dữ liệu được trích xuất đầy đủ, chính xác và hạn chế tối đa các lỗi phát sinh trong quá trình tự động hóa.

1.5.3. Phương pháp thực nghiệm

Phương pháp thực nghiệm được áp dụng trong suốt quá trình xây dựng và hoàn thiện hệ thống. Chương trình thu thập dữ liệu được triển khai thử nghiệm trên một số website thực tế, với nhiều lần chạy và các điều kiện khác nhau. Thông qua quá trình thực nghiệm, nhóm nghiên cứu tiến hành theo dõi, đánh giá hiệu năng, tốc độ thu thập dữ liệu và độ ổn định của hệ thống.

Các lỗi phát sinh trong quá trình chạy thử được ghi nhận và điều chỉnh kịp thời, từ đó nâng cao tính hoàn thiện và khả năng ứng dụng thực tế của hệ thống.

1.5.4. Phương pháp kết hợp công cụ (Selenium)

Trong đề tài, Selenium đóng vai trò là công cụ chính để tự động hóa quá

trình thu thập dữ liệu từ các website. Công cụ này cho phép mô phỏng các thao tác của người dùng như truy cập website, nhập dữ liệu tìm kiếm, chuyển trang và trích xuất thông tin. Việc kết hợp Selenium với ngôn ngữ lập trình giúp xử lý hiệu quả các website có nội dung động, nơi dữ liệu chỉ được tải sau khi người dùng tương tác.

Phương pháp này giúp tiết kiệm thời gian, công sức so với phương pháp thu thập thủ công, đồng thời nâng cao độ chính xác và khả năng mở rộng của hệ thống thu thập dữ liệu.

1.5.5. Phương pháp đánh giá và phân tích dữ liệu

Sau khi dữ liệu được thu thập và lưu trữ trong cơ sở dữ liệu SQLite, nhóm nghiên cứu tiến hành xử lý và phân tích dữ liệu thông qua các truy vấn SQL. Các thao tác phân tích bao gồm thống kê, tổng hợp, so sánh và phân tích xu hướng điểm thi và điểm chuẩn theo từng năm, từng trường hoặc từng ngành học.

Kết quả phân tích được trình bày dưới dạng bảng số liệu, biểu đồ và các nhận xét đánh giá, giúp làm rõ giá trị của dữ liệu và hỗ trợ người dùng trong việc khai thác thông tin một cách hiệu quả.

1.6. Những đóng góp nghiên cứu của đề tài

Thông qua quá trình nghiên cứu và triển khai, đề tài đã đạt được một số đóng góp nhất định cả về mặt học thuật lẫn thực tiễn.

1.6.1. Trong lĩnh vực học thuật

Về mặt học thuật, đề tài góp phần làm rõ và hệ thống hóa quy trình thu thập,

lưu trữ và phân tích dữ liệu web trong lĩnh vực giáo dục. Việc áp dụng Selenium và SQLite trong đề tài là minh chứng cho khả năng kết hợp giữa lý thuyết và thực hành, giúp sinh viên và người nghiên cứu hiểu rõ hơn về các kỹ thuật thu thập dữ liệu tự động và quản lý cơ sở dữ liệu trong các bài toán thực tế.

1.6.2. Trong thực tiễn

Về mặt thực tiễn, đề tài xây dựng được một hệ thống có khả năng tự động thu thập và phân tích dữ liệu điểm thi THPT và điểm chuẩn các trường đại học một cách nhanh chóng, chính xác và có tính ứng dụng cao. Kết quả nghiên cứu có thể được sử dụng làm công cụ hỗ trợ cho học sinh, phụ huynh và các đơn vị tư vấn tuyển sinh trong việc tra cứu, so sánh và đánh giá thông tin, góp phần nâng cao hiệu quả trong công tác định hướng và tuyển sinh đại học.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về kỹ thuật cào dữ liệu (Web Scraping)

2.1.1 Định nghĩa và Quy trình vận hành tự động

Định nghĩa: Web Scraping là một kỹ thuật khai thác dữ liệu tự động từ các trang web công khai. Bản chất của công nghệ này là việc sử dụng các kịch bản lập trình (Scripts) để mô phỏng lại toàn bộ hành vi của một người dùng thực thụ khi truy cập Internet. Thay vì hiển thị dữ liệu lên màn hình cho con người đọc, các "Scrapers" sẽ truy cập thẳng vào mã nguồn HTML, bóc tách các trường thông tin cần thiết và chuyển đổi chúng từ dạng phi cấu trúc (Unstructured Data) sang dạng có cấu trúc (Structured Data) như CSV, JSON hoặc SQL.

Quy trình vận hành :

Hình 2.1 Quá trình Web Scraping

Một quy trình cào dữ liệu tiêu chuẩn không chỉ đơn thuần là tải nội dung mà là một chuỗi các thao tác kỹ thuật hạ tầng mạng:

1. Giao thức kết nối (HTTP/HTTPS): Scraper khởi tạo các yêu cầu GET hoặc POST. Để vượt qua các bộ lọc chống bot (Anti-bot), chương trình cần cấu hình Request Headers tinh vi bao gồm User-Agent (định danh trình duyệt) và Cookies (duy trì phiên làm việc).

2. Tải nội dung và DOM: Máy chủ trả về mã HTML. Hệ thống sẽ dựng lại cây DOM (Document Object Model) để hiểu được thứ tự các thẻ dữ liệu.

3. Bóc tách bằng Selector: Sử dụng XPath hoặc CSS Selector để định vị chính xác dữ liệu giữa hàng vạn dòng code. XPath cho phép điều hướng đa chiều (cha-con, anh em), giúp xử lý các cấu trúc trang web phức tạp nhất.

4. Xử lý hậu kỳ (Post-processing): Dữ liệu thô thường chứa các ký tự nhiễu như khoảng trắng, mã script ẩn. Scraper thực hiện các thuật toán làm sạch, định dạng lại kiểu dữ liệu (Số, Ngày tháng) về chuẩn ISO để đảm bảo tính nhất quán trước khi lưu trữ.

2.1.2 Phân loại và Công nghệ hỗ trợ trích xuất

Tùy thuộc vào kiến trúc của website mục tiêu (Tĩnh hay Động), kỹ thuật Scraping được chia thành hai nhánh công nghệ riêng biệt:

Static Scraping (Cào dữ liệu tĩnh):

1. Cơ chế: Sử dụng các thư viện như BeautifulSoup hoặc lxml. Phương pháp này chỉ tải mã nguồn HTML gốc mà máy chủ gửi về lần đầu tiên.

2. Ưu điểm: Tốc độ thực thi cực nhanh, chiếm dụng ít băng thông và RAM. Phù hợp cho các trang tin tức hoặc blog đơn giản.

3. Hạn chế: Hoàn toàn không thể lấy được dữ liệu được sinh ra bởi các đoạn mã JavaScript (như giá vàng, chứng khoán cập nhật liên tục) sau khi trang đã load.

Dynamic Scraping (Cào dữ liệu động):

1. Cơ chế: Sử dụng các trình duyệt ảo (Headless Browsers) như Selenium, Playwright. Scraper sẽ khởi chạy một nhân trình duyệt thực, đợi cho các script của trang web thực thi xong rồi mới lấy kết quả cuối cùng từ cây DOM.

2. Ưu điểm: Vượt qua được các rào cản của website hiện đại (SPA), xử lý được các sự kiện cuộn trang (Infinite Scroll), click chuột để mở rộng dữ liệu.

3. Hạn chế: Tiêu tốn tài nguyên hệ thống rất lớn.

2.2 Ngôn ngữ lập trình Python

2.2.1 Python là gì?

Định nghĩa: Python là một ngôn ngữ lập trình bậc cao, thông dịch, hướng đối tượng và đa mục đích. Được tạo ra bởi Guido van Rossum và phát hành lần đầu vào năm 1991, Python được thiết kế với triết lý ưu tiên sự rõ ràng, dễ đọc của mã nguồn. Nhờ cú pháp gần gũi với ngôn ngữ tự nhiên, Python đã trở thành lựa chọn hàng đầu cho các dự án tự động hóa và khai thác dữ liệu.

2.2.2 Đặc điểm cốt lõi của Python

- Dễ học và dễ sử dụng: Cú pháp của Python rất tinh gọn, cho phép lập trình viên viết ít dòng code hơn so với Java hay C++ nhưng vẫn đạt được cùng một kết quả.

- Hệ sinh thái thư viện khổng lồ: Python sở hữu kho thư viện PyPI (Python Package Index) cực kỳ phong phú, đặc biệt là các thư viện hỗ trợ cào dữ liệu (Selenium, Scrapy) và xử lý dữ liệu (Pandas, Numpy).

- Khả năng tương thích: Python hoạt động mượt mà trên nhiều hệ điều hành (Windows, Linux, macOS) và dễ dàng kết nối với các hệ quản trị cơ sở dữ liệu như SQLite.

2.2.3 Cài đặt và chuẩn bị môi trường

Để xây dựng hệ thống thu thập dữ liệu, quy trình cài đặt bao gồm các bước:

Cài đặt trình thông dịch Python: Tải phiên bản Python mới nhất (thường là 3.x) từ trang chủ python.org.

Quản lý thư viện với PIP: Sử dụng công cụ pip (Python Package Installer) để cài đặt các thư viện bổ trợ:

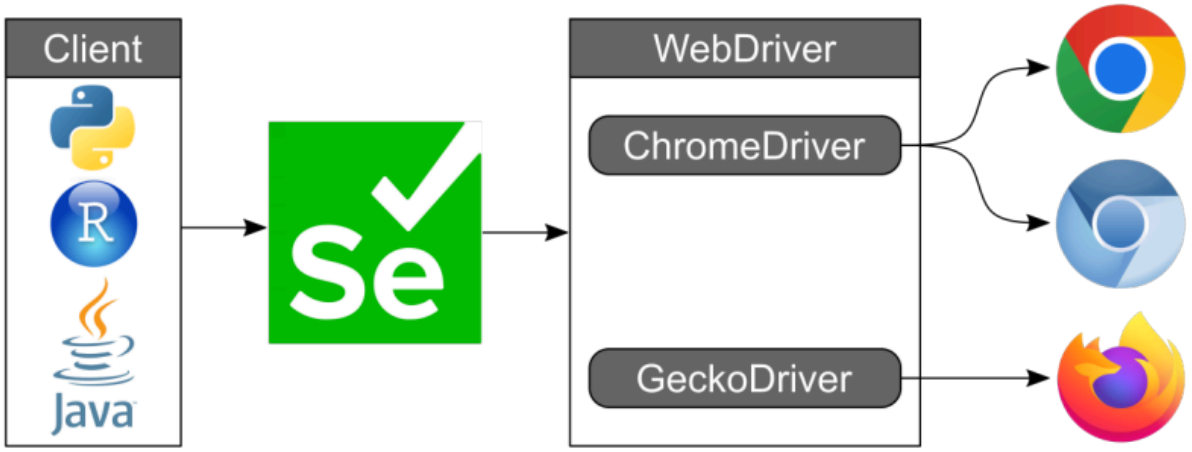
1. pip install selenium: Thư viện tương tác trình duyệt.

2. Thư viện sqlite3 thường có sẵn trong gói cài đặt chuẩn của Python.

3. Môi trường lập trình (IDE): Các công cụ phổ biến bao gồm Visual Studio Code, PyCharm hoặc Jupyter Notebook, giúp việc viết và gỡ lỗi (debug) mã nguồn trở nên thuận tiện.

2.3 Thư viện tự động hóa Selenium

2.3.1 Kiến trúc và Cơ chế WebDriver



Hình 2.3 Selenium

Định nghĩa: Selenium WebDriver không chỉ là một thư viện hỗ trợ mã nguồn, mà còn là một khung làm việc (framework) mạnh mẽ cho phép điều khiển trình duyệt web một cách tự động. Điểm khác biệt cốt lõi của WebDriver so với các công cụ cào dữ liệu tĩnh là khả năng can thiệp trực tiếp vào nhân trình duyệt thông qua các trình điều khiển (drivers) cấp thấp, cho phép mô phỏng chính xác mọi thao tác của người dùng như click, cuộn trang, di chuột và xử lý các sự kiện JavaScript phức tạp.

Cơ chế vận hành: Hệ thống hoạt động dựa trên mô hình giao tiếp Client-Server chặt chẽ, đảm bảo tính tách biệt giữa mã kịch bản và môi trường thực thi:

2.3.1.a Tầng Client (Language Bindings)

Tại tầng này, lập trình viên sử dụng ngôn ngữ Python để xây dựng các kịch bản điều khiển. Selenium hỗ trợ một thư viện Client Binding

chuyên biệt cho Python, giúp chuyển đổi các phương thức hướng đối tượng (như driver.find_element() hay element.click()) thành các lệnh có cấu trúc. Các lệnh này được chuẩn hóa để có thể truyền tải qua môi trường mạng mà không bị sai lệch về logic.

2.3.1.b Tầng Trung gian (Browser Drivers)

Đây là thành phần cầu nối cực kỳ quan trọng, bao gồm các tệp thực thi như ChromeDriver (cho Google Chrome) hoặc GeckoDriver (cho Firefox). Tầng này đóng vai trò là một máy chủ cục bộ (Local Server), tiếp nhận các lệnh từ kịch bản Python qua giao thức HTTP. Driver có nhiệm vụ biên dịch các lệnh này thành ngôn ngữ nội bộ của trình duyệt (Native API). Điều này giải thích tại sao cùng một mã Python có thể điều khiển nhiều loại trình duyệt khác nhau miễn là có Driver tương ứng.

2.3.1.c Tầng Thực thi (Browsers)

Tầng này bao gồm các ứng dụng trình duyệt thực tế được cài đặt trên hệ điều hành. Khi nhận được lệnh từ Driver, trình duyệt sẽ khởi tạo một phiên làm việc (Session). Tại đây, mọi thành phần trên trang web (nút bấm, ô nhập liệu, bảng dữ liệu) đều được ánh xạ thành các nút trong cây DOM. WebDriver sẽ tác động trực tiếp lên các nút này, thực thi các kịch bản JavaScript cần thiết để lấy dữ liệu cuối cùng sau khi trang web đã hoàn tất các tiến trình động.

2.3.2 Chuẩn bị môi trường và cài đặt

Để Selenium vận hành ổn định trong môi trường Python, quy trình chuẩn bị cần thực hiện đồng bộ qua ba thành phần chính:

- **Cài đặt thư viện Python:** Sử dụng công cụ quản lý gói **pip**

để tải về gói Selenium Binding bằng lệnh **pip install selenium**. Thư viện này đóng vai trò là cầu nối cho phép Python gọi các hàm điều khiển trình duyệt.

- **Thiết lập Browser Driver:** Người dùng phải tải về Driver có phiên bản tương thích hoàn toàn với phiên bản trình duyệt đang cài đặt trên máy (ví dụ: Chrome v115 thì phải dùng ChromeDriver v115). Driver này sau đó phải được cấu hình trong biến môi trường `PATH` để hệ thống có thể triệu gọi từ bất kỳ đâu.
- **Cấu hình trình duyệt (Options):** Thiết lập các thông số khởi tạo như kích thước cửa sổ, cổng Proxy, hoặc đường dẫn lưu dữ liệu tạm (Profile) để giúp Scraper hoạt động ổn định và tránh bị các website chặn truy cập.

2.3.3 Ưu điểm và Nhược điểm của Selenium

Ưu điểm:

- **Xử lý dữ liệu động tuyệt vời:** Khác với các thư viện tĩnh, Selenium có thể đợi JavaScript thực thi xong, cuộn trang để tải thêm dữ liệu (Lazy Loading), giúp thu thập dữ liệu từ các trang web hiện đại (SPA).
- **Mô phỏng hành vi người dùng:** Có khả năng xử lý các tương tác phức tạp như đăng nhập, giải mã một số loại CAPTCHA đơn giản, tương tác với các thẻ iframe hoặc cửa sổ pop-up.
- **Đa trình duyệt và đa nền tảng:** Hỗ trợ hầu hết các trình duyệt phổ biến và chạy tốt trên Windows, Linux, macOS.

Nhược điểm:

- **Tốc độ thực thi chậm:** Do phải khởi chạy toàn bộ nhân trình duyệt, tốc độ tải trang và trích xuất dữ liệu của Selenium chậm hơn đáng kể so với phương pháp sử dụng Request trực tiếp.
- **Chiếm dụng tài nguyên lớn:** Mỗi cửa sổ trình duyệt mở ra chiếm từ vài trăm MB đến cả GB RAM. Khi chạy đa luồng (Multi-threading), Selenium đòi hỏi phần cứng cực kỳ mạnh mẽ.
- **Độ ổn định phụ thuộc vào UI:** Nếu website thay đổi cấu trúc giao diện hoặc ID của phần tử, kịch bản Selenium dễ bị lỗi và cần phải bảo trì, cập nhật lại thường xuyên.

2.3.4 Tối ưu hóa Selenium trong thu thập dữ liệu

Do đặc thù phải khởi chạy một trình duyệt hoàn chỉnh, Selenium tiêu tốn tài nguyên hệ thống (CPU, RAM) lớn hơn nhiều lần so với các phương pháp cào tĩnh. Để tối ưu hóa cho bài toán cào dữ liệu quy mô lớn, các kỹ thuật sau là bắt buộc:

- Chế độ Headless Mode (Trình duyệt ẩn): Kỹ thuật này cho phép trình duyệt vận hành mà không cần vẽ giao diện đồ họa (GUI) ra màn hình. Khi chạy ở chế độ này, trình duyệt loại bỏ các bước render hình ảnh phức tạp, giúp giảm tải tới 40% dung lượng RAM tiêu thụ và tăng tốc độ xử lý dữ liệu. Đây là cấu hình tiêu chuẩn khi triển khai công cụ cào dữ liệu trên các máy chủ không có giao diện (Server Linux).

- Chiến lược Đợi thông minh (Explicit Waits): Một lỗi phổ biến khiến chương trình cào dữ liệu bị dừng đột ngột là cố gắng truy cập phần tử khi nó chưa kịp tải xong từ máy chủ. Kỹ thuật WebDriverWait kết hợp với các điều kiện chờ (Expected Conditions) cho phép chương trình dừng lại trong một khoảng thời gian động, chỉ tiếp tục thực thi khi dữ liệu đã hiện diện hoàn toàn. Điều này giúp hệ thống hoạt động ổn định ngay cả trong điều kiện mạng không ổn định.

- Quản lý tài nguyên và Chặn tải nội dung phụ: Scraper được cấu hình thông qua lớp Options để ngăn chặn trình duyệt tải các tệp tin không phục vụ cho việc lấy dữ liệu như: hình ảnh, tệp tin CSS định dạng, các video quảng cáo hoặc các đoạn mã theo dõi (tracking scripts). Việc chặn các tài nguyên này giúp giảm băng thông mạng và rút ngắn thời gian tải trang một cách đáng kể.

2.4 Hệ quản trị cơ sở dữ liệu SQLite

2.4.1 Đặc điểm của Cơ sở dữ liệu nhúng (Embedded Database)

Định nghĩa: SQLite là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mang tính cách mạng với triết lý thiết kế "Serverless" (Không máy chủ). Khác với MySQL hay SQL Server cần một quy trình cài đặt và vận hành phức tạp, SQLite là một thư viện phần mềm được tích hợp thẳng vào mã nguồn ứng dụng. Toàn bộ cơ sở dữ liệu chỉ nằm trong một tệp tin duy nhất trên đĩa cứng.

Các đặc tính kỹ thuật vượt trội:

1. Cấu trúc lưu trữ B-Tree: SQLite tổ chức dữ liệu theo thuật toán cây B-Tree cân bằng. Khi Scraper lưu hàng chục ngàn bản ghi, cấu trúc này cho phép tìm kiếm và truy xuất dữ liệu với tốc độ cực nhanh (log n), đảm bảo hiệu năng tối ưu cho ứng dụng.

2. Tiêu chuẩn ACID nghiêm ngặt: SQLite đảm bảo tính Nguyên tố (Atomicity), Nhất quán (Consistency), Cô lập (Isolation) và Bền vững (Durability). Điều này cực kỳ quan trọng trong Web Scraping: nếu máy tính bị tắt đột ngột khi đang lưu dữ liệu cào được, SQLite sẽ đảm bảo dữ liệu không bị hỏng (Corruption) và tự động khôi phục về trạng thái an toàn nhất.

3. Cấu hình bằng không (Zero-Configuration): Không cần thiết lập cổng mạng, không cần quản trị viên. Đặc điểm này giúp đồ án có tính di động tuyệt đối: bạn chỉ cần copy file .db sang máy tính khác là hệ thống hoạt động ngay lập tức.

2.4.2 So sánh giải pháp lưu trữ: SQLite vs MongoDB

Để tối ưu hóa việc lưu trữ dữ liệu cào được, việc so sánh giữa mô hình SQL (SQLite) và NoSQL (MongoDB) là cần thiết:

Tiêu chí	SQLite (SQL)	MongoDB (NoSQL)
Mô hình	Bảng và Cột (Schema cố định)	Tài liệu JSON (Linh hoạt)
Triển khai	File-based (Gọn nhẹ)	Server-based (Phức tạp)
Sử dụng	Đồ án, dữ liệu cấu trúc ổn định	Big Data, dữ liệu đa dạng

Bảng 2.1 So sánh SQLite và MongoDB

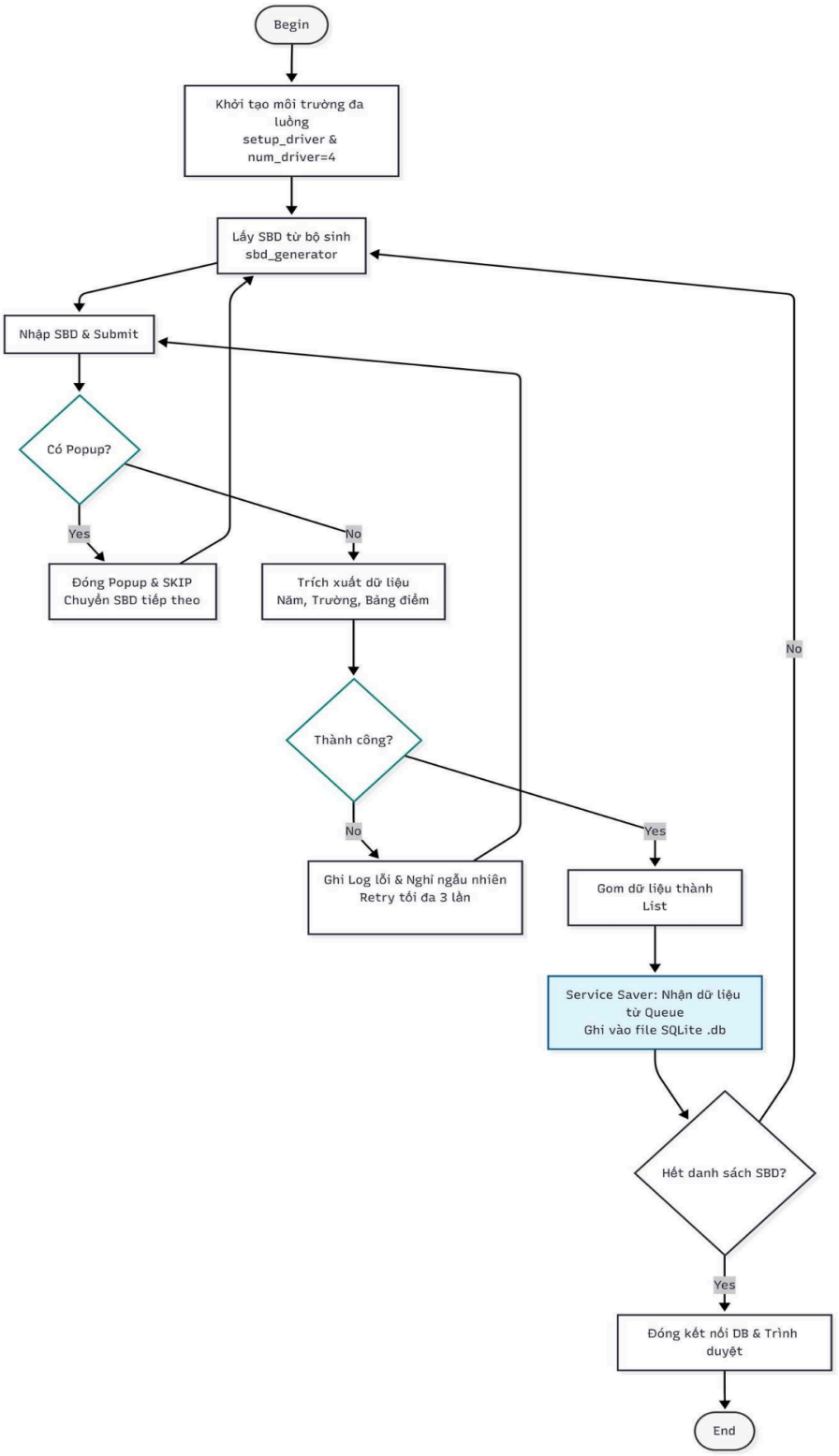
Mô hình dữ liệu: SQLite yêu cầu một lược đồ (Schema) cố định với các bảng và cột rõ ràng. Điều này rất phù hợp khi ta đã xác định rõ các trường cần cào (ví dụ: Tên sản phẩm, Giá, Mô tả). MongoDB cho phép lưu dữ liệu linh hoạt (Schema-less), phù hợp khi cấu trúc website thay đổi liên tục.

Triển khai thực tế: MongoDB yêu cầu cài đặt máy chủ, tốn tài nguyên chạy ngầm. SQLite nhẹ hơn, tích hợp sẵn trong Python (sqlite3), không tốn RAM để duy trì dịch vụ.

Kết luận lựa chọn: Trong phạm vi đồ án và các bài toán thu thập thông tin có cấu trúc ổn định, SQLite là sự lựa chọn tối ưu nhất. Nó vừa đảm bảo tốc độ truy vấn SQL mạnh mẽ, vừa dễ dàng đóng gói để nộp báo cáo mà không yêu cầu người chấm phải cài đặt thêm hạ tầng cơ sở dữ liệu.

CHƯƠNG 3: PHƯƠNG PHÁP THỰC NGHIỆM

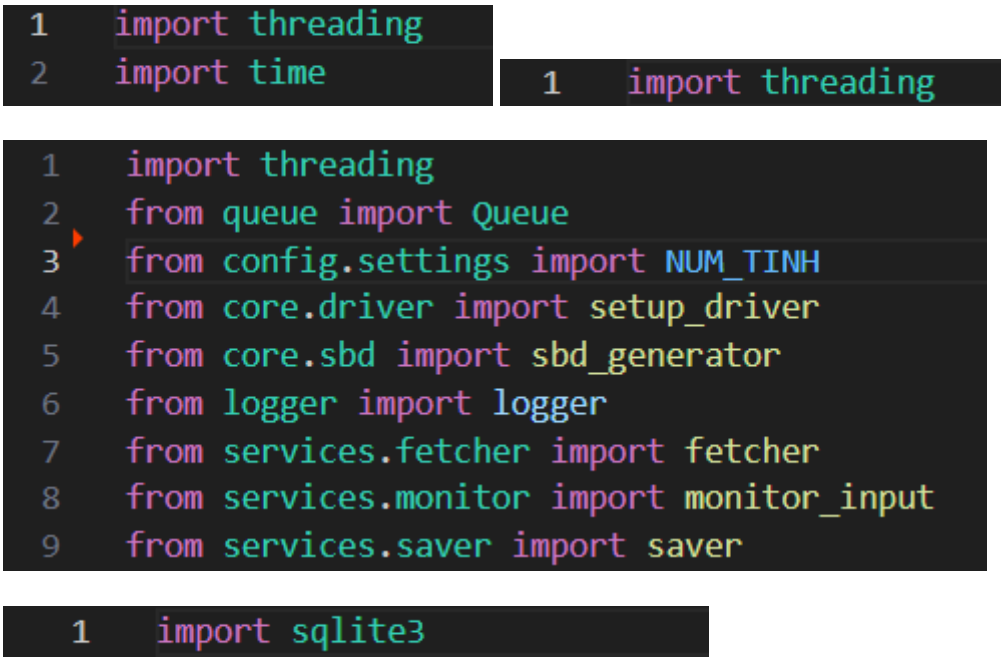
3.1 Sơ đồ mô tả quá trình thu thập dữ liệu



Hình 3.1 Sơ đồ mô tả quá trình thu thập dữ liệu

3.2 Phương pháp thu thập dữ liệu

3.2.1 Một số thư viện cần dùng



Hình 3.2 Hình ảnh minh họa thư viện

-**Sqlite3**: Thư viện làm việc với cơ sở dữ liệu SQLite (lưu dữ liệu vào file .db).

- **Threading:** Tạo và quản lý đa luồng để chạy nhiều tác vụ song song.
- **Queue** (from queue import Queue): Hàng đợi an toàn cho đa luồng, dùng để truyền dữ liệu giữa các thread.
- **Setup_driver** (from core.driver): Hàm khởi tạo trình duyệt (thường là Selenium WebDriver).
- **Logger** (from logger): Ghi log (thông tin, lỗi, trạng thái chương trình).
- **Fetcher** (from services.fetcher): Hàm lấy dữ liệu (crawl / request từ web).
- **Time:** Xử lý delay, sleep, đo thời gian.

3.2.2 .khởi tạo môi trường đa luồng

```python
from orchestrator.system import orchestrator_system

if __name__ == "__main__":
    orchestrator_system(num_driver=4)
```

Hình 3.3 Hàm khởi tạo môi trường đa luồng

Điểm khởi động toàn bộ hệ thống

Gọi hàm điều phối orchestrator system

num_driver =4 → chạy 4 trình duyệt / 4 luồng crawl song song

3.2.3 Thu thập dữ liệu

```python
def fetch_data(thread_id, driver, sbd, retry=3):
    wait = WebDriverWait(driver, 5)
    for attempt in range(1, retry + 1):
        try:
            # Input & submit
            search_input = wait.until(
                EC.presence_of_element_located((By.CSS_SELECTOR, "input.input-search"))
            )
            search_button = driver.find_element(By.CSS_SELECTOR, "button.btn-submit")

            search_input.clear()
            search_input.send_keys(sbd)
            search_button.submit()
```

Hình 3.4 Hàm truy cập và thu thập dữ liệu

- Tạo vòng lặp chờ tối đa 5s cho mỗi element
- Thử lại tối đa retry lần nếu lỗi
- Tìm ô nhập SBD ,Điền SBD, đưa vào from

Xử lý popup

```
# ---- HANDLE POPUP (nếu có) ----
try:
    popup = WebDriverWait(driver, 2).until(
        EC.element_to_be_clickable(
            (By.CSS_SELECTOR, "img.close__popupMessage")
        )
    )
    popup.click()
    logger.warning(f"[SKIP] {sbd}")
    return None, Feedback.SKIP
except TimeoutException:
    pass  # không có popup → tiếp tục
```

Hình 3.5 Xử lý nếu có popup

Nếu popup xuất hiện:

- Đóng popup
- Bỏ SBD này (SKIP)

Nếu không có popup → tiếp tục lấy dữ liệu

```
42          # ---- FETCH DATA ----
43          year = wait.until(
44              EC.presence_of_element_located((By.ID, "year"))
45          ).get_attribute("year")
46
47          edu = wait.until(
48              EC.presence_of_element_located((By.CSS_SELECTOR, "p.edu-institution"))
49          ).text
50
51          rows = wait.until(
52              EC.presence_of_element_located((By.TAG_NAME, "tbody"))
53          ).find_elements(By.TAG_NAME, "tr")
```

Hình 3.6 Tìm năm điểm và lấy tất cả các dòng

Tìm và lấy

- Năm thi
- Trường học

Lấy tất cả các dòng trong bảng điểm

```
            data = [
                [year, edu, sbd]
                + [td.text for td in row.find_elements(By.XPATH, "./*")]
                for row in rows
            ]

            logger.info(f"[OK] {sbd}")
            return data, Feedback.NEXT

        except Exception as e:
            logger.error(f"[Thread-{thread_id}] Retry {attempt} {sbd} - {e}")
            time.sleep(uniform(0.5, 1))

    return None, Feedback.STOP
```

Hình 3.7 Gom dữ liệu và check dữ liệu

Gom dữ liệu thành list

Cào thành công điền [OK] → chuyển sang SBD tiếp theo

Nếu có lỗi:

- Log lỗi
- Nghỉ ngẫu nhiên
- Thử lại

Nếu sau retry lần vẫn lỗi → báo dừng

```
def fetcher(thread_id, driver, gen_sbd, result_queue):
    print(f" Thread {thread_id} start")
    driver.get(URL)
    try:
        current_sbd = next(gen_sbd)
        skip_count = 0
        while not stop_event.is_set():
            data, status = fetch_data(thread_id, driver, current_sbd)
            if status is Feedback.NEXT:
                skip_count = 0
            if status is Feedback.SKIP:
                skip_count += 1
                status = Feedback.NEXT
            if skip_count >= 3:
                status = Feedback.SKIP
            current_sbd = gen_sbd.send(status)
            if data:
                result_queue.put(data)
```

Hình 3.8 định hướng generator

Mỗi thread xử lý nhiều SBD liên tiếp

Gửi kết quả về queue

Mở trang

Lấy SBD đầu tiên

Đếm số lần skip liên tiếp

Skip 3 SBD liên tiếp → báo generator đổi hướng

Generator quyết định SBD kế tiếp dựa trên status

Đưa dữ liệu cho thread ghi file / DB

```
89      except StopIteration:
90          logger.info(f"[Fetcher-{thread_id}] DONE")
91      finally:
92          result_queue.put(None)
93          driver.quit()
94
```

Hình 3.9 Giải phóng tài nguyên

- Khi hết SBD
- Kết thúc thread

3.3 lưu vào SQLITE

```
1  import sqlite3
2  DB_PATH = "results.db"
3  def get_connection():
4      conn = sqlite3.connect(DB_PATH, check_same_thread=False)
5      return conn
```

Hình 3.10 Hàm kết nối với SQL

Tạo kết nối tới database results.db

check_same_thread=False:

Cho phép nhiều thread cùng dùng chung DB

Rất hay dùng khi crawl dữ liệu bằng Selenium + đa luồng

Trả về đối tượng conn để:

- tạo cursor

- execute SQL

- commit / close

```python
from db.connection import get_connection
def init_db():
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute("PRAGMA journal_mode=WAL;")
    cursor.execute("PRAGMA synchronous=NORMAL;")

    cursor.execute("""
        CREATE TABLE IF NOT EXISTS results (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            year INTEGER,edu TEXT,sbd TEXT,subject TEXT,score REAL,UNIQUE(sbd, subject) )""")

    conn.commit()
    conn.close()
```

Hình 3.11 Tạo hàm khởi tạo database (tạo bảng, cấu hình SQLite)

Journal_mode=WAL :Bật Write-Ahead Logging

- Ghi dữ liệu nhanh hơn
- Giúp nhiều thread ghi cùng lúc

Synchronous=NORMAL

- Cân bằng giữa tốc độ và an toàn dữ liệu
- Phù hợp cho crawl dữ liệu

Tạo bảng results bảo gồm các thông tin như: id năm ,cụm thi ,sbd , môn, điểm

- mỗi số báo danh chỉ lưu điểm cho 1 môn

commit(): lưu dữ liệu xuống file DB

close(): giải phóng tài nguyên

3.4 Phương pháp phân tích dữ liệu

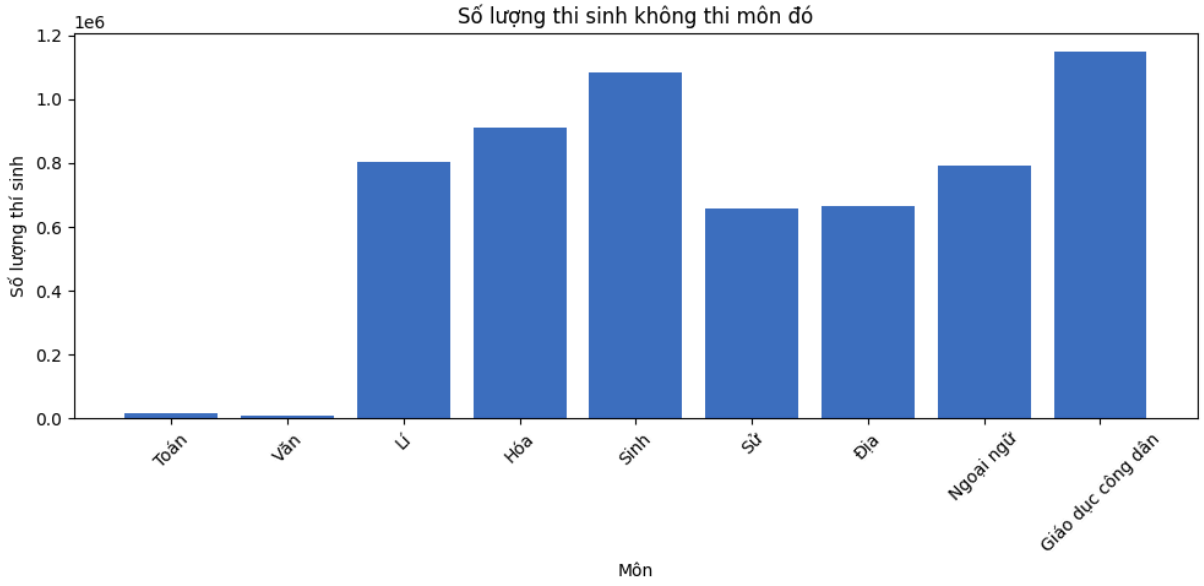
3.4.1 Thông tin tổng quan

Số lượng thí sinh: 1,153,228 dòng.

Các môn thi: Toán, Văn, Lí, Hóa, Sinh, Sử, Địa, GDCD, Ngoại ngữ.

Đặc điểm: Đây là bảng điểm tổng hợp, mỗi thí sinh thường chỉ có điểm ở một số môn nhất định (theo tổ hợp xét tuyển), do đó số lượng giá trị thiếu (Null) ở từng cột phản ánh số lượng thí sinh không thi môn đó.

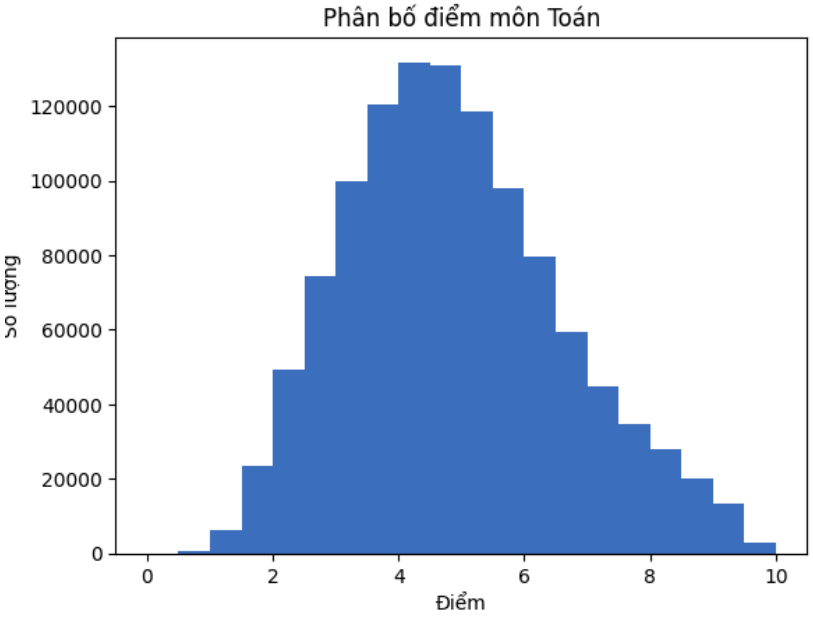
3.4.2 Các biểu đồ trực quan



Hình 3.12 Biểu đồ số lượng thí sinh không thi môn

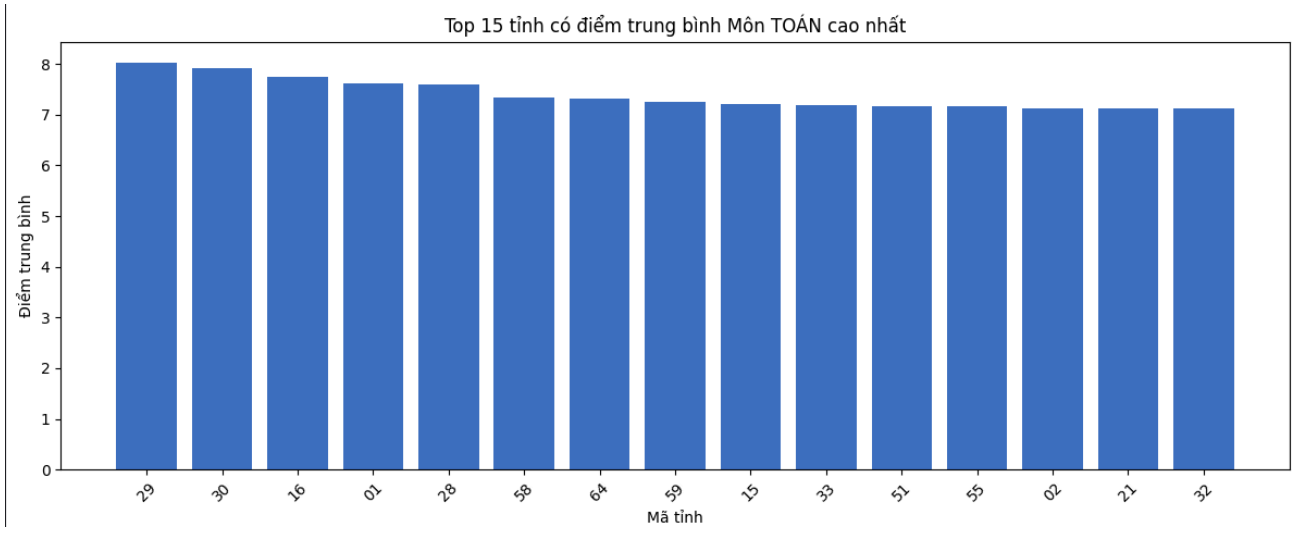
Biểu đồ 1 (Missing Values): Cho biết mức độ phổ biến của các môn. Ví dụ:

Môn Văn và môn Sử có lượng thí sinh tham gia đông nhất trong bộ dữ liệu này.



Hình 3.13 Biểu đồ phân bố điểm toán THPTQG 2025

Biểu đồ 2 (Math Distribution): Phân phối điểm môn Toán.Có thể thấy điểm tập trung nhiều nhất ở khoảng quanh mức 4-5.



Hình 3.14 Biểu đồ top tỉnh có điểm trung bình môn toán cao nhất

Biểu đồ 3 : qua biểu phân cụm thành các tỉnh dựa trên số báo danh (2 số đầu) ta thấy 3 tỉnh có điểm trung bình toán lớn nhất là 29 Hải Phòng , 30 Hà Nội và 16 Phú thọ

CHƯƠNG 4: KẾT LUẬN VÀ KIẾN NGHỊ

Qua quá trình nghiên cứu và thực nghiệm, thư viện mã nguồn mở Selenium WebDriver đã khẳng định được vai trò là một công cụ mạnh mẽ và linh hoạt trong việc thu thập dữ liệu từ các nền tảng web có tính tương tác cao. Khác với các phương pháp cào tĩnh, Selenium cho phép mô phỏng chính xác hành vi của người dùng, tự động tương tác với các ô nhập liệu và nút bấm trên các cổng tra cứu điểm thi THPT Quốc gia năm 2025. Đề tài đã triển khai thành công quy trình từ bước khởi tạo trình duyệt, tự động điền số báo danh, đến việc trích xuất điểm số các môn thi từ cây cấu trúc DOM và lưu trữ đồng bộ vào cơ sở dữ liệu SQLite. Kết quả thực hiện cho thấy Selenium xử lý hiệu quả các kịch bản đòi hỏi thực thi và phản hồi động, đảm bảo độ chính xác cao ngay cả với cấu trúc môn thi mới của kỳ thi năm 2025. Tuy nhiên, công cụ này vẫn tồn tại hạn chế về tốc độ và tiêu tốn nhiều tài nguyên hệ thống do phải khởi chạy toàn bộ nhân trình duyệt. Để tối ưu hóa trong tương lai, cần kết hợp Selenium với các thư viện parser tốc độ cao như lxml và áp dụng kỹ thuật chặn tài nguyên không thiết yếu để tăng hiệu năng xử lý. Việc tích hợp các giải pháp vượt rào cản kỹ thuật như Proxy hay xử lý CAPTCHA tự động cũng là hướng phát triển cần thiết để hệ thống vận hành ổn định và bền bỉ hơn trên quy mô lớn.

TÀI LIỆU THAM KHẢO

[1] “Selenium Python Bindings,” Read the Docs. Accessed: Nov. 27, 2025. [Online]. Available: https://selenium-python.readthedocs.io/

[2] “Selenium Documentation,” Selenium.dev. Accessed: Nov. 28, 2025. [Online]. Available: https://www.selenium.dev/documentation/

[3] “sqlite3 — DB-API 2.0 interface for SQLite databases,” Python Docs. Accessed: Dec. 02, 2025. [Online]. Available: https://docs.python.org/3/library/sqlite3.html

[4] “Selenium Python Tutorial,” GeeksforGeeks. Accessed: Dec. 05, 2025. [Online]. Available: https://www.geeksforgeeks.org/selenium-python-tutorial/

[5] “SQLite Tutorial - SQL Syntax and Features,” SQLite Tutorial. Accessed: Dec. 07, 2025. [Online]. Available: https://www.sqlitetutorial.net/

[6] “Modern Web Automation with Python and Selenium,” Real Python. Accessed: Dec. 10, 2025. [Online]. Available: https://realpython.com/modern-web-automation-with-python-and-selenium/

[7] “How To Use the sqlite3 Module in Python 3,” DigitalOcean. Accessed: Dec. 12, 2025. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-use-the-sqlite3-module-in-python-3

[8] “Selenium with Python,” Guru99. Accessed: Dec. 14, 2025. [Online]. Available: https://www.guru99.com/selenium-python-tutorial.html

[9] “Introduction to SQLite Database,” W3Schools. Accessed: Dec. 15, 2025. [Online]. Available: https://www.w3schools.com/sql/sql_intro.asp

[10] “WebDriver Manager for Python,” GitHub. Accessed: Dec. 18, 2025. [Online]. Available: https://github.com/SergeyPirogov/webdriver_manager

[11] “SQLite là gì? Tổng quan về hệ quản trị cơ sở dữ liệu SQLite,” Vietnix. Accessed: Nov. 29, 2025. [Online]. Available: https://vietnix.vn/sqlite-la-gi/

[12] TopDev, “Selenium là gì? Tất tần tật về Selenium Testing Tool,” TopDev. Accessed: Nov. 30, 2025. [Online]. Available: https://topdev.vn/blog/lam-quen-tom-tat-co-ban-ve-selenium/

[13] “Tổng quan về Selenium và vai trò của các thành phần,” Tech Cybozu. Accessed: Dec. 01, 2025. [Online]. Available: https://tech.cybozu.vn/tong-quan-ve-selenium-va-vai-tro-cua-cac-thanh-phan-74a12/

[14] “Python và cơ sở dữ liệu SQLite,” Tự Học ICT. Accessed: Dec. 20, 2025. [Online]. Available: https://tuhocict.com/python-va-co-so-du-lieu-sqlite/

[15] “Web Scraping with Selenium and Python,” DataCamp. Accessed: Dec. 22, 2025. [Online]. Available: https://www.datacamp.com/tutorial/selenium-python-tutorial-for-beginners

[16] “Database Programming in Python with SQLite,” Real Python. Accessed: Dec. 24, 2025. [Online]. Available: https://realpython.com/python-sqlite-sqlalchemy/

[17] “Selenium Browser Automation Project,” GitHub. Accessed: Dec. 25, 2025. [Online]. Available: https://github.com/SeleniumHQ/selenium

[18] “Architecture of SQLite,” SQLite.org. Accessed: Dec. 26, 2025. [Online]. Available: https://www.sqlite.org/arch.html

[19] “Newest ‘selenium’ Questions,” Stack Overflow. Accessed: Dec. 27, 2025. [Online]. Available: https://stackoverflow.com/questions/tagged/selenium

[20] “XPath in Selenium: A Complete Guide,” BrowserStack. Accessed: Dec. 28, 2025. [Online]. Available: https://www.browserstack.com/guide/xpath-in-selenium

PHỤ LỤC 1

báo cáo đc chia thành nhiều files khác nhau :

file drive.py (dùng để 1 số cài đặt chop drive)

src > core > driver.py > ...

```python
from selenium import webdriver
from selenium.webdriver.chrome.options import Options


def setup_driver():
    opts = Options()
    opts.add_argument("--headless=new")
    opts.add_argument("--disable-gpu")
    opts.add_argument("--no-sandbox")
    opts.add_argument("--incognito")  # profile mới
    opts.add_argument("--disk-cache-size=0")  # tắt cache
    opts.add_argument("--disable-dev-shm-usage")
    opts.add_argument("--disable-extensions")
    opts.add_argument("--disable-notifications")
    opts.add_argument("--disable-background-networking")
    opts.add_argument("--blink-settings=imagesEnabled=false")
    return webdriver.Chrome(options=opts)

```

file enum(dùng để báo cho các luồng biết giúp code rõ ràng và an toàn hơn)

src > core > enums.py > ...

```python
from enum import Enum, auto

class Feedback(Enum):
    STOP = auto()
    SKIP = auto()
    NEXT = auto()
```

file sbg (file tạo ra cấu trúc số báo danh)

src > core > sbd.py > ...

```python
from core.enums import Feedback


def format_sbd(tinh, cum, so):
    return f"{tinh:02d}{cum:02d}{so:04d}"


def sbd_generator(start_tinh, end_tinh):
    for tinh in range(start_tinh, end_tinh + 1):
        skip_tinh = False
        for cum in range(20):
            for so in range(10000):
                feedback = yield format_sbd(tinh, cum, so)
                if feedback is Feedback.STOP:
                    return
                elif feedback is Feedback.SKIP:
                    skip_tinh = True
                    break
            if skip_tinh:
                break
```

file connection (file dùng để kết nối đến SQLite)

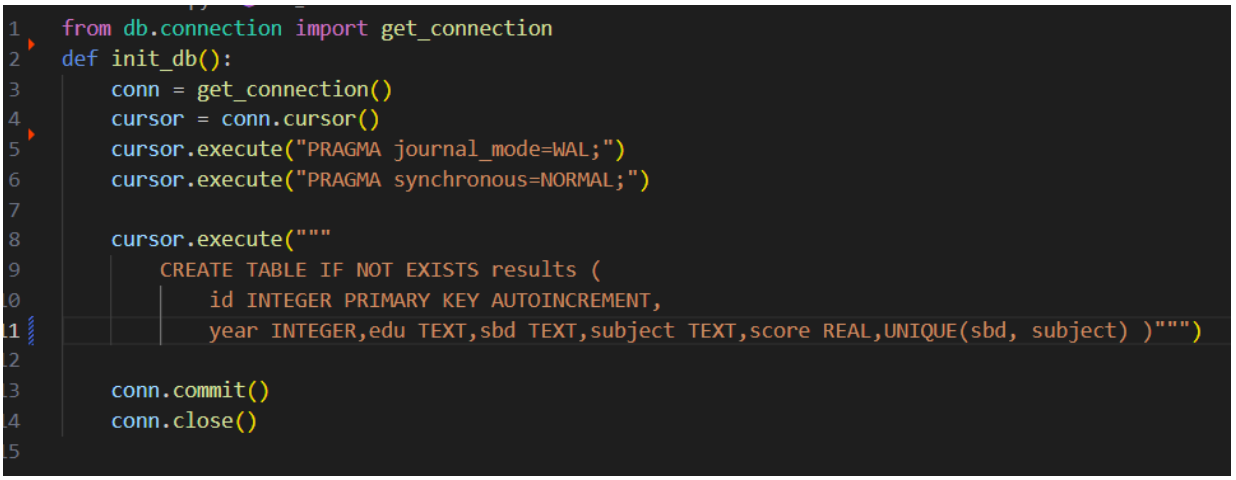
src > db > connection.py > ...

```python
import sqlite3
DB_PATH = "results.db"
def get_connection():
    conn = sqlite3.connect(DB_PATH, check_same_thread=False)
    return conn
```

file repository (file dùng để thêm từng dòng data vào sqlite)

```python
def insert_results(cursor, rows):
    cursor.executemany(
        """
        INSERT OR IGNORE INTO results
        (year, edu, sbd, subject, score)
        VALUES (?, ?, ?, ?, ?)
    """,
        rows,
    )
```

file scheme (dùng để tạo bảng và kết nối đến SQlite)



```python
from db.connection import get_connection
def init_db():
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute("PRAGMA journal_mode=WAL;")
    cursor.execute("PRAGMA synchronous=NORMAL;")

    cursor.execute("""
        CREATE TABLE IF NOT EXISTS results (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            year INTEGER,edu TEXT,sbd TEXT,subject TEXT,score REAL,UNIQUE(sbd, subject) )""")

    conn.commit()
    conn.close()
```

file fetcher(file chính dùng thu thập data từ trang wed selenium)

```python
import time
from random import uniform

from selenium.common.exceptions import TimeoutException
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait

from config.settings import URL
from core.enums import Feedback
from logger import logger
from utils.stop import stop_event


def fetch_data(thread_id, driver, sbd, retry=3):
    wait = WebDriverWait(driver, 5)
    for attempt in range(1, retry + 1):
        try:
            # Input & submit
            search_input = wait.until(
                EC.presence_of_element_located((By.CSS_SELECTOR, "input.input-search"))
            )
            search_button = driver.find_element(By.CSS_SELECTOR, "button.btn-submit")

            search_input.clear()
            search_input.send_keys(sbd)
            search_button.submit()

            # ---- HANDLE POPUP (nếu có) ----
            try:
                popup = WebDriverWait(driver, 2).until(
                    EC.element_to_be_clickable(
                        (By.CSS_SELECTOR, "img.close__popupMessage")
                    )
                )
                popup.click()
                logger.warning(f"[SKIP] {sbd}")
                return None, Feedback.SKIP
            except TimeoutException:
                pass  # không có popup → tiếp tục

            # ---- FETCH DATA ----
            year = wait.until(
                EC.presence_of_element_located((By.ID, "year"))
            ).get_attribute("year")

            edu = wait.until(
                EC.presence_of_element_located((By.CSS_SELECTOR, "p.edu-institution"))
            ).text

            rows = wait.until(
                EC.presence_of_element_located((By.TAG_NAME, "tbody"))
            ).find_elements(By.TAG_NAME, "tr")

            data = [
                [year, edu, sbd]
                + [td.text for td in row.find_elements(By.XPATH, "./*")]
                for row in rows
            ]

            logger.info(f"[OK] {sbd}")
            return data, Feedback.NEXT

        except Exception as e:
            logger.error(f"[Thread-{thread_id}] Retry {attempt} {sbd} - {e}")
            time.sleep(uniform(0.5, 1))

    return None, Feedback.STOP


def fetcher(thread_id, driver, gen_sbd, result_queue):
```

```python
def fetcher(thread_id, driver, gen_sbd, result_queue):
    print(f" Thread {thread_id} start")
    driver.get(URL)
    try:
        current_sbd = next(gen_sbd)
        skip_count = 0
        while not stop_event.is_set():
            data, status = fetch_data(thread_id, driver, current_sbd)
            if status is Feedback.NEXT:
                skip_count = 0
            if status is Feedback.SKIP:
                skip_count += 1
                status = Feedback.NEXT
            if skip_count >= 3:
                status = Feedback.SKIP
            current_sbd = gen_sbd.send(status)
            if data:
                result_queue.put(data)
    except StopIteration:
        logger.info(f"[Fetcher-{thread_id}] DONE")
    finally:
        result_queue.put(None)
        driver.quit()

```

file monitor (tạo giao diện dừng an toàn mà vẫn bảo toàn dữ liệu)

```python
from utils.stop import stop_event


def monitor_input():
    helper_text = """
HỆ THỐNG GIÁM SÁT LỆNH NGƯỜI DÙNG
---------------------------------
Y       : Dừng toàn bộ hệ thống
N       : Tiếp tục hệ thống
--help  : Hiển thị hướng dẫn này
---------------------------------
"""
    print("Monitor input đang chạy. Nhập '--help' để nhận trợ giúp.")

    while not stop_event.is_set():
        try:
            print("Bạn có muốn dựng toàn bộ hệ thống ?")
            user_input = input("Nhập lệnh [Y/N]: ").strip().upper()

            if user_input == "Y":
                print("Lệnh dừng hệ thống được kích hoạt!")
                stop_event.set()
                break
            elif user_input == "N":
                print("Hệ thống tiếp tục hoạt động.")
            elif user_input == "--HELP":
                print(helper_text)
            else:
                print("Lệnh không hợp lệ. Nhập '--help' để được hướng dẫn.")

        except Exception:
            print("\nDừng hệ thống ngay lập tức.")
            stop_event.set()
            break
```

file saver

src > services > saver.py > saver

```python
from db.connection import get_connection
from db.repository import insert_results
from db.schema import init_db
from logger import logger

BATCH_SIZE = 500
VALID_SIZE = 5


def saver(result_queue, num_fetcher):
    finished = 0
    buffer = []

    init_db()
    conn = get_connection()
    cursor = conn.cursor()

    def flush():
        if not buffer:
            return
        try:
            insert_results(cursor, buffer)
            conn.commit()
            logger.info(f"Saved batch: {len(buffer)}")
            buffer.clear()
        except Exception as e:
            logger.error(f"[DB ERROR] {e}")

    while finished < num_fetcher:
        items = result_queue.get()

        if items is None:
            finished += 1
            continue

        valid_items = [item for item in items if len(item) == VALID_SIZE]
        if not valid_items:
            logger.warning("[DATA WARNING] - No valid item")
            continue

        if len(valid_items) < len(items):
            sbd = valid_items[0][2]
            logger.warning(
                f"[DATA WARNING] {sbd} - Invalid item size: {len(items)}, expected {VALID_SIZE}"
            )
        buffer.extend(valid_items)

        if len(buffer) >= BATCH_SIZE:
            flush()

    flush()
    conn.close()
    logger.info("SQLite closed")
```

file logger (tạp **custom logger thread-safe**, có màu, lọc level, giups ghi file ổn định)

src > logger.py > Logger

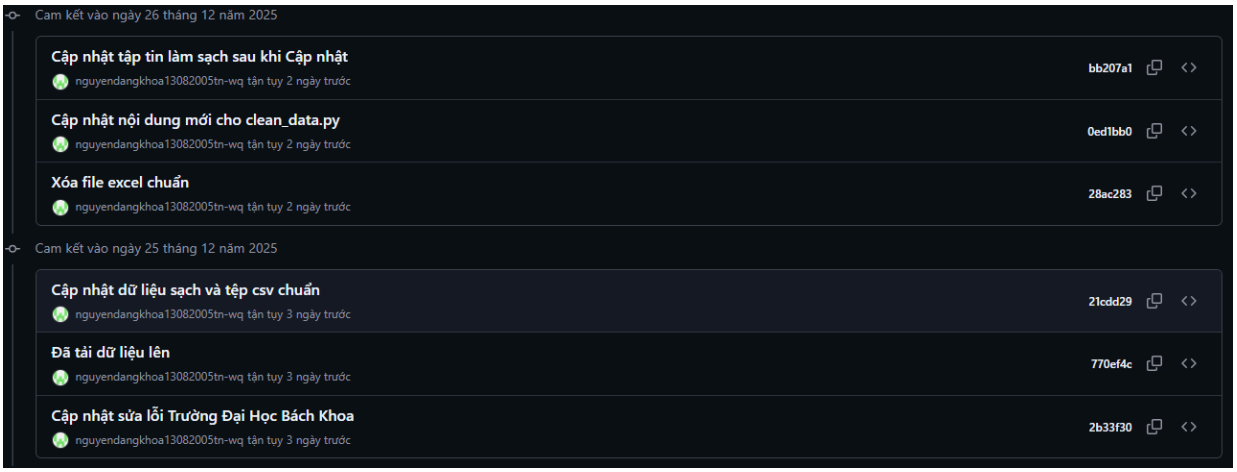
```python
import threading
import time
# Mã màu ANSI
class Color:
    RESET = "\033[0m"
    RED = "\033[91m"
    GREEN = "\033[92m"
    YELLOW = "\033[93m"
    BLUE = "\033[94m"
    MAGENTA = "\033[95m"
    CYAN = "\033[96m"
    WHITE = "\033[97m"
class Logger:
    LEVELS = {
        "DEBUG": {"priority": 10, "color": Color.CYAN},
        "INFO": {"priority": 20, "color": Color.GREEN},
        "WARNING": {"priority": 30, "color": Color.YELLOW},
        "ERROR": {"priority": 40, "color": Color.RED},
    }

    _lock = threading.Lock()  # thread-safe

    def __init__(self, filename="app.log", level="INFO", use_color=True):
        self.filename = filename
        self.level = level
        self.use_color = use_color

    def _log(self, level, msg):
        if Logger.LEVELS[level]["priority"] >= Logger.LEVELS[self.level]["priority"]:
            timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
            color = Logger.LEVELS[level]["color"] if self.use_color else ""
            reset = Color.RESET if self.use_color else ""
            log_line = f"{color}[{level}] {timestamp} | {msg}{reset}\n"
            with Logger._lock:
                with open(self.filename, "a", encoding="utf-8") as f:
                    f.write(log_line)
```

```python
    def debug(self, msg):
        self._log("DEBUG", msg)

    def info(self, msg):
        self._log("INFO", msg)

    def warning(self, msg):
        self._log("WARNING", msg)

    def error(self, msg):
        self._log("ERROR", msg)
# Singleton
logger = Logger()
```

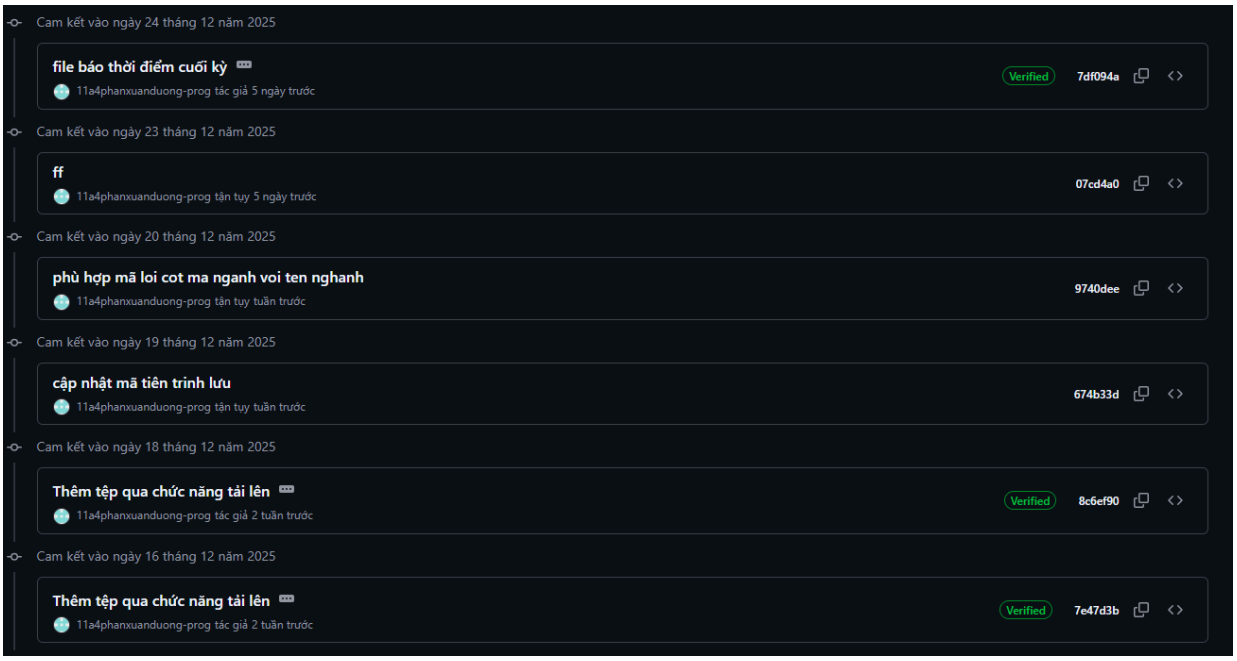
PHỤ LỤC 2

Lịch sử git commit (gồm 2 nhánh phụ, 1 nhánh chính)

Nhánh phụ 1(Nguyễn Đăng Khoa)



Nhánh phụ 2 (Phan Xuân Dương)



Nhánh chính (Nguyễn Đức Vinh MAIN)