

|  |
| --- |
| TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC KHOA CÔNG NGHÊ THÔNG TIN |

|  |
| --- |
| **KIỂM THỬ PHẦN MỀM 2**  **BÁO CÁO ĐỒ ÁN** |

**Họ và Tên :**

TRẦN THÁI TÚ

**Giáo viên** : Phan Gia Phước

Tp Hồ Chí Minh, 05/2025

MỤC LỤC

[CHƯƠNG 1: LÝ THUYẾT VỀ KIỂM THỬ PHẦN MỀM 4](#_Toc198243025)

[I. Giới thiệu về kiểm thử phần mềm 4](#_Toc198243026)

[**1. Kiểm thử phần mềm là gì?** 4](#_Toc198243027)

[**2. Vai trò** 4](#_Toc198243028)

[**3. Quy trình kiểm thử** 4](#_Toc198243029)

[II. Bảy nguyên lý kiểm thử 4](#_Toc198243030)

[**1. Kiểm thử đưa ra lỗi** 4](#_Toc198243031)

[**2. Kiểm thử toàn diện là không thể** 4](#_Toc198243032)

[**3. Kiểm thử càng sớm càng tốt** 4](#_Toc198243033)

[**4. Sự tập trung của lỗi** 4](#_Toc198243034)

[**5. Nghịch lí thuốc trừ sâu** 5](#_Toc198243035)

[**6.  Kiểm thử phụ thuộc vào ngữ cảnh** 5](#_Toc198243036)

[**7.  Không có lỗi - Sai lầm** 5](#_Toc198243037)

[III. Kỹ thuật kiểm thử hộp đen 5](#_Toc198243038)

[**1. Kiểm thử chức năng (Functional Testing).** 5](#_Toc198243039)

[**2. Kiểm thử giá trị biên (Boundary Value Testing)** 5](#_Toc198243040)

[**3. Kiểm thử phân vùng tương đương (Equivalence Partitioning)** 6](#_Toc198243041)

[**4. Kiểm thử bảng quyết định (Decision Table Testing)** 6](#_Toc198243042)

[IV. Kiểm thử tự động với selenium 6](#_Toc198243043)

[**1**. **Tóm lược về kiểm thử tự động với selenium.** 6](#_Toc198243044)

[V. Kiểm thử tự động với Postman. 7](#_Toc198243045)

[**1.** **Tóm lược về kiểm thử tự động với Postman.** 7](#_Toc198243046)

[Mục đích: 7](#_Toc198243047)

[Chương 2: THỰC HIỆN DỰ ÁN 8](#_Toc198243048)

[I. Question & Answer (SRS\_POL). 8](#_Toc198243049)

[**1. Description.** 8](#_Toc198243050)

[**2. Hình ảnh minh họa.** 8](#_Toc198243051)

[II. Viết Test Case cho SRS\_POL (ver1.0). 8](#_Toc198243052)

[**1. Test Design.** 8](#_Toc198243053)

[**2. Hình ảnh Một số Test Case minh họa trang web (SRS\_POL).** 9](#_Toc198243054)

[III. Viết Test Case cho SRS\_POL (ver1.1). 9](#_Toc198243055)

[**1. Test Design.** 9](#_Toc198243056)

[**2. Hình ảnh Một số Test Case và màn hình minh họa trang web (SRS\_POL).** 10](#_Toc198243057)

[IV. Code kiểm thử tự động Selenium. 12](#_Toc198243058)

[**1. Test Design.** 13](#_Toc198243059)

[**2. Hình ảnh minh họa code Selenium.** 13](#_Toc198243060)

[V. Code kiểm thử tự động Selenium (lần 2). 16](#_Toc198243061)

[**1. Test Design.** 16](#_Toc198243062)

[**2. Hình ảnh minh họa Code Selenium.** 17](#_Toc198243063)

[VI. Viết Defect List cho trang web con muỗi. 21](#_Toc198243064)

[**1. Test Design.** 21](#_Toc198243065)

[**2. Hình ảnh minh họa Defect List và Test Condition.** 22](#_Toc198243066)

[VII. Code kiểm thử tự động POSTMAN. 23](#_Toc198243067)

[**1. Test Design.** 23](#_Toc198243068)

[**2. Hình ảnh minh họa Code POSTMAN.** 23](#_Toc198243069)

[VIII. Code kiểm thử tự động Selenium (lần 3). 32](#_Toc198243070)

[**1. Test Design.** 32](#_Toc198243071)

[**2. Hình ảnh minh họa Code Selenium.** 33](#_Toc198243072)

[IX. Viết Test Case cho tựa game Đấu Trường Chân Lý (TFT). 34](#_Toc198243073)

[1. Test Design. 35](#_Toc198243074)

[2. Hình ảnh minh họa Test Case. 35](#_Toc198243075)

[X. Code kiểm thử tự động KatalonStudio. 37](#_Toc198243076)

[**1. Test Design.** 37](#_Toc198243077)

[**2. Hình ảnh minh họa code Katalon.** 38](#_Toc198243078)

[XI. Viết Test Case cho ứng dụng Quản Lý Thực Tập Viên. 39](#_Toc198243079)

[**1. Test Design.** 40](#_Toc198243080)

[2. **Hình ảnh viết Test Case và màn hình minh họa.** 40](#_Toc198243081)

[XII. Viết Test Case cho ứng dụng Quản Lý Thực Tập Viên (thêm chức năng) 41](#_Toc198243082)

[**1. Test Design.** 41](#_Toc198243083)

[**2. Hình ảnh viết Test Case và màn hình minh họa.** 42](#_Toc198243084)

[Chương 3: KẾT LUẬN 48](#_Toc198243085)

[I. Kết luận 48](#_Toc198243086)

[II. Hướng phát triển 48](#_Toc198243087)

[LỜI CẢM ƠN 49](#_Toc198243088)

[TÀI LIỆU THAM KHẢO 49](#_Toc198243089)

# CHƯƠNG 1: LÝ THUYẾT VỀ KIỂM THỬ PHẦN MỀM

## I. Giới thiệu về kiểm thử phần mềm

### **1. Kiểm thử phần mềm là gì?**

### **2. Vai trò**

### **3. Quy trình kiểm thử**

## II. Bảy nguyên lý kiểm thử

### **1. Kiểm thử đưa ra lỗi**

Kiểm thử chỉ ra sự hiện diện của lỗi trong phần mềm nhưng không thể chứng minh rằng phần mềm không có lỗi. Kiểm thử giúp làm giảm xác suất của các lỗi tiềm ẩn vẫn còn trong phần mềm nhưng ngay cả khi không tìm thấy thêm bất kỳ môt lỗi nào thì cũng không thể khẳng định phần mềm không còn lỗi. Nó cũng không phải là một bằng chứng về tính đúng đắn của phần mềm.

### **2. Kiểm thử toàn diện là không thể**

Kiểm thử toàn diện là hoàn toàn không khả thi. Đơn giản ta hiểu là không thể kiểm tra tất cả các chức năng với tất cả các kết hợp dữ liệu đầu vào hợp lệ và không hợp lệ trong quá trình kiểm thử thực tế. Thay vì kiểm thử tất cả, phân tích rủi ro và sắp xếp thứ tự ưu tiên được sử dụng để tập trung trong kiểm thử.

### **3. Kiểm thử càng sớm càng tốt**

Các hoạt động test nên bắt đầu càng sớm càng tốt trong quy trình phát triển phần mềm và nên tập trung vào mục tiêu được xác định.Thực hiện thiết kế và review test càng sớm thì lỗi càng được phát hiện sớm khi đó tốn ít công và thời gian để tìm kiếm và sửa chữa lỗi.

### 

### **4. Sự tập trung của lỗi**

Các chuyên gia testing đã chỉ ra rằng các lỗi sẽ nằm tập trung trong một số module nhất định nào đó chứ không phải nằm rãi đều trên tất cả các modules của sản phẩm nên khi phát hiện 1 bug thuộc module nào đó thì nên cần phải test kĩ hơn nữa để đảm bảo tìm ra được nhiều bug tiềm ẩn có thể nhất.

### **5. Nghịch lí thuốc trừ sâu**

Nếu các test được thực hiện lặp đi lặp lại nhiều lần thì không có lỗi mới nào có thể được tìm thấy. Hiệu quả của các trường hợp kiểm thử bắt đầu giảm xuống sau một số lần thực hiện. Để khắc phục nguyên lý thuốc trừ sâu này, các test case cần phải được thường xuyên rà soát và sửa đổi. Test mới và khác đi để có thể tìm ra nhiều lỗi tiềm ẩn hơn.

### **6.  Kiểm thử phụ thuộc vào ngữ cảnh**

Test được thực hiện khác nhau trong các bối cảnh khác nhau. Ngữ cảnh ở đây là bản chất của các ứng dụng mà ta sẽ áp dụng những phương thức, kỹ thuật, cũng như loại kiểm thử khác nhau.

### **7.  Không có lỗi - Sai lầm**

Tất cả các yêu cầu được chỉ định và sửa tất cả các lỗi được tìm thấy vẫn có thể tạo ra một hệ thống khó sử dụng, không đáp ứng được nhu cầu và mong đợi của người dùng, hoặc kém hơn so với các hệ thống cạnh tranh khác. Phần mềm được xây dựng không chỉ là phần mềm không có lỗi 99% mà còn phải đáp ứng nhu cầu kinh doanh nếu không nó sẽ trở thành một phần mềm không thể sử dụng được.

## III. Kỹ thuật kiểm thử hộp đen

### **1. Kiểm thử chức năng (Functional Testing).**

Kiểm thử chức năng là một loại kiểm thử hộp đen, trong đó người kiểm thử tập trung vào việc kiểm tra các chức năng của phần mềm xem có đáp ứng được các yêu cầu được xác định trước hay không.

### **2. Kiểm thử giá trị biên (Boundary Value Testing)**

* Kiểm tra các giá trị có xu hướng ẩn nấp gần ranh giới
* Có 2 loại:
  + - Tow-boundary: Giá trị lớn nhất và nhỏ nhất trong 1 phân vùng là giá trị biên
    - Three-boundary: Giá trị trước, tại, sau trong 1 phân vùng
* Test được thiết kế để cover các giá trị biên valid và invalid
* BVA được áp dụng ở tất cả các level test. Nó khá dễ dàng áp dụng để tìm ra lỗi.
* Đặc tả chi tiết sẽ giúp cho các xác định được giá trị biên này

### **3. Kiểm thử phân vùng tương đương (Equivalence Partitioning)**

* Chia các vùng dựa vào đầu vào, đầu ra,.. thành các khu vực mà kết quả mong đợi là giống nhau (tương đương) vì thế chúng được xử lý theo một cách giống nhau.
* Chỉ cần kiểm chứng 1 giá trị thì sẽ có thể khẳng định được vùng đó là valid hay invalid
* Testcase được thiết kế để bao phủ tất cả các TH valid và invalid. EP được áp dụng ở tất cả các level test

### **4. Kiểm thử bảng quyết định (Decision Table Testing)**

* Kết hợp các input, tình huống hoặc sự kiện
* Diễn đạt bằng các điều kiện đầu vào để họ có TRUE or FALSE

## IV. Kiểm thử tự động với selenium

### **1**. **Tóm lược về kiểm thử tự động với selenium.**

* Toàn bộ bộ công cụ Selenium bao gồm bốn thành phần:
  + Selenium IDE , một add-on của Firefox mà bạn chỉ có thể sử dụng trong việc tạo các trường hợp thử nghiệm tương đối đơn giản và các bộ phần mềm thử nghiệm.
  + Điều khiển từ xa Selenium , còn được gọi là Selenium 1 , là công cụ Selenium đầu tiên cho phép người dùng sử dụng các ngôn ngữ lập trình để tạo các bài kiểm tra phức tạp.
  + WebDriver , bước đột phá mới hơn cho phép các tập lệnh kiểm tra của bạn liên lạc trực tiếp với trình duyệt, do đó kiểm soát nó từ cấp hệ điều hành.
  + Selenium Grid cũng là một công cụ được sử dụng với Selenium RC để thực hiện các kiểm tra song song trên các trình duyệt và hệ điều hành khác nhau.
* Selenium RC và WebDriver đã được sáp nhập để tạo ra Selenium 2 .
* Selenium là thuận lợi hơn QTP về chi phí và tính linh hoạt . Nó cũng cho phép bạn chạy thử nghiệm song song , không giống như trong QTP, nơi bạn chỉ được phép chạy thử nghiệm theo tuần tự.

## V. Kiểm thử tự động với Postman.

### **Tóm lược về kiểm thử tự động với Postman.**

### Mục đích:

Tự động hóa kiểm thử API giúp phát hiện lỗi sớm, tiết kiệm thời gian và đảm bảo chất lượng phần mềm.

Các bước thực hiện:

1. **Tạo request**: Gửi các yêu cầu HTTP (GET, POST, PUT, DELETE...) tới API.
2. **Viết test case**: Sử dụng JavaScript trong tab **Tests** để kiểm tra phản hồi (status code, nội dung trả về...).
3. **Tạo collection**: Gom nhóm các request và test case để dễ quản lý.
4. **Chạy bằng Collection Runner**: Tự động thực thi nhiều request và test cùng lúc.
5. **Tích hợp với CI/CD**: Dùng công cụ **Newman** để tự động kiểm thử khi cập nhật mã nguồn.

Lợi ích:

* **Phát hiện lỗi sớm**
* **Tiết kiệm thời gian kiểm thử lặp**
* **Đảm bảo chất lượng API**
* **Hỗ trợ cộng tác nhóm tốt hơn**

Tính năng nổi bật của Postman:

* Giao diện trực quan, dễ sử dụng
* Hỗ trợ nhiều ngôn ngữ
* Có mock server, data store, tích hợp dễ dàng với các công cụ DevOps

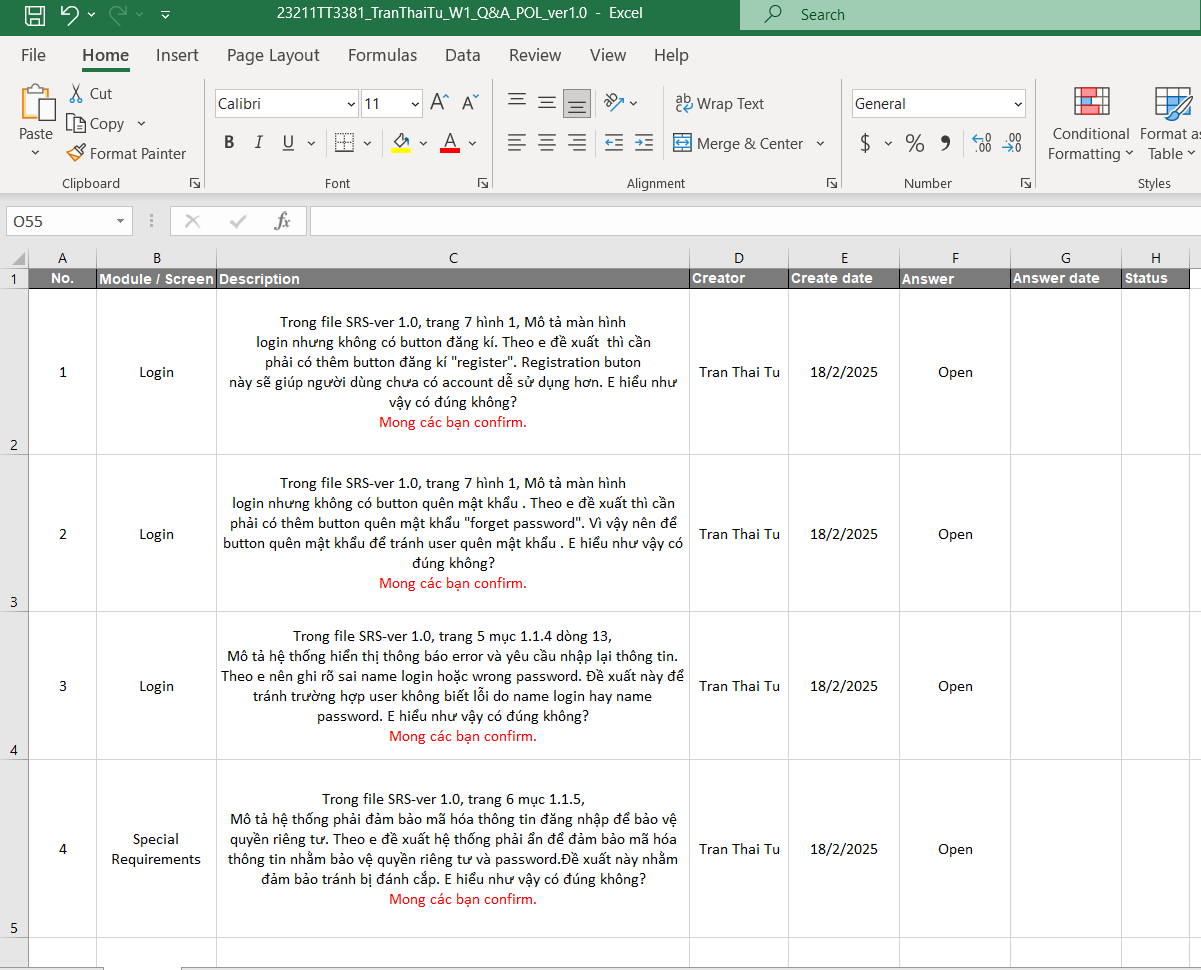
# Chương 2: THỰC HIỆN DỰ ÁN

## Question & Answer (SRS\_POL).

### **Description.**

- Câu hỏi phải có địa chỉ dẩn chứng rõ ràng, ngắn gọn, dễ hiểu, lịch sự, có thời gian đặt câu hỏi, phải có lời cảm ơn. Nếu có đề xuất thì ít nhất hãy đưa ra đề xuất cho khách hàng lựa chọn hoặc đưa ra mong muốn của họ tránh việc nhờ họ đưa ra ý kiến quá nhiều.

### **Hình ảnh minh họa.**

****Hình 1.1: hình ảnh mình họa Question & Answer.

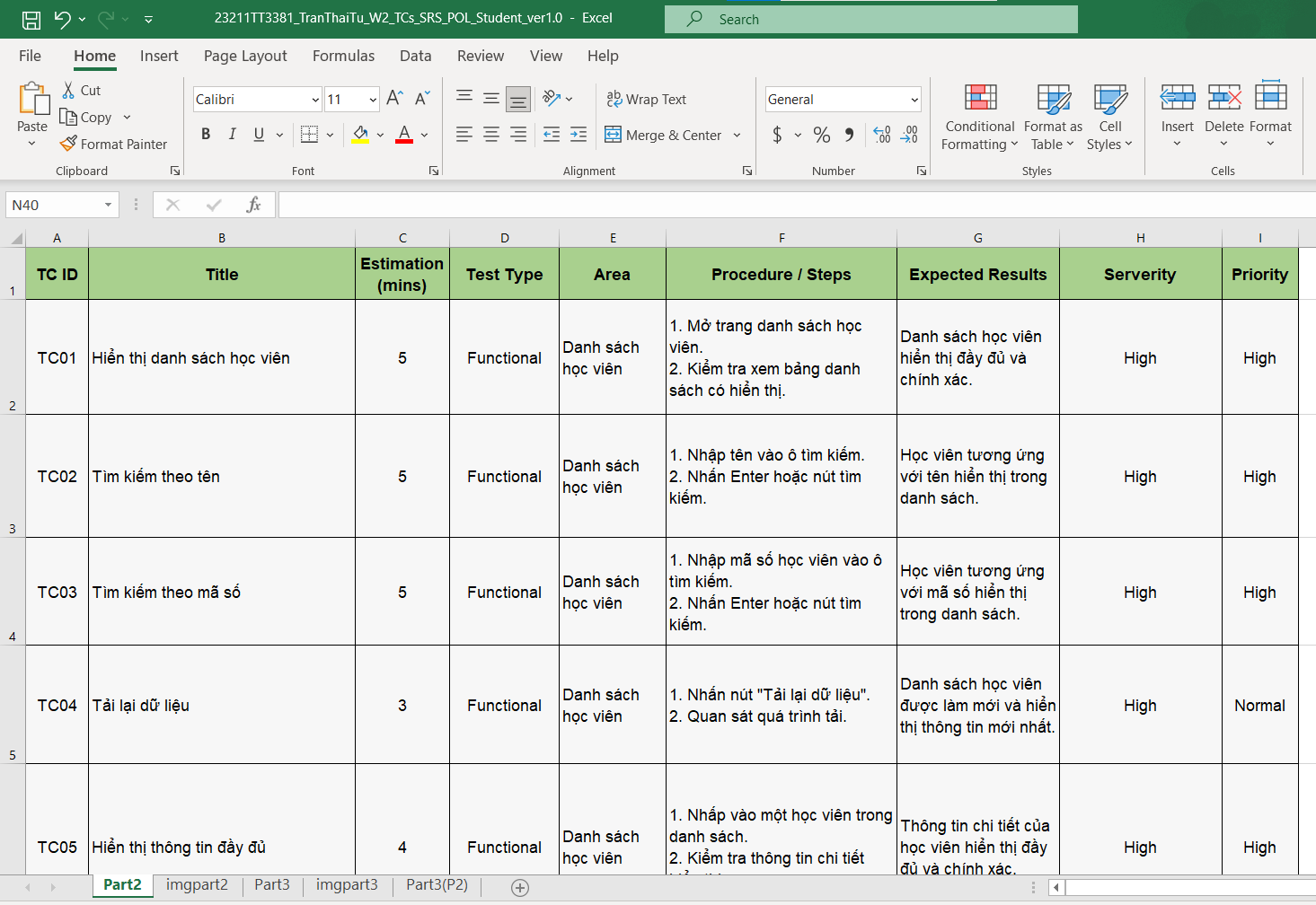
## Viết Test Case cho SRS\_POL (ver1.0).

### **Test Design.**

-Trang web cho phép người dùng dưới quyền Admin quản lý học viên bằng cách (thêm, xóa, sửa, tìm kiếm,…). Củng như là quản lý thêm Khóa học, Bài tập.

-Trang web cho phép người dùng dưới quyền Học sinh có thể xem thông tin của mình và có chức năng tìm kiếm. Củng như có những quyền của khóa học (hủy khóa học, đăng ký khóa học, tìm kiếm) sau khi đã có khóa học thì bài tập mới có thông tin và học sinh chỉ có quyền xem bài tập, làm bài tập và nộp bài tập.

### **Hình ảnh Một số Test Case minh họa trang web (SRS\_POL).**



Hình 2.1: Một số Test Case minh họa trang web (SRS\_POL).

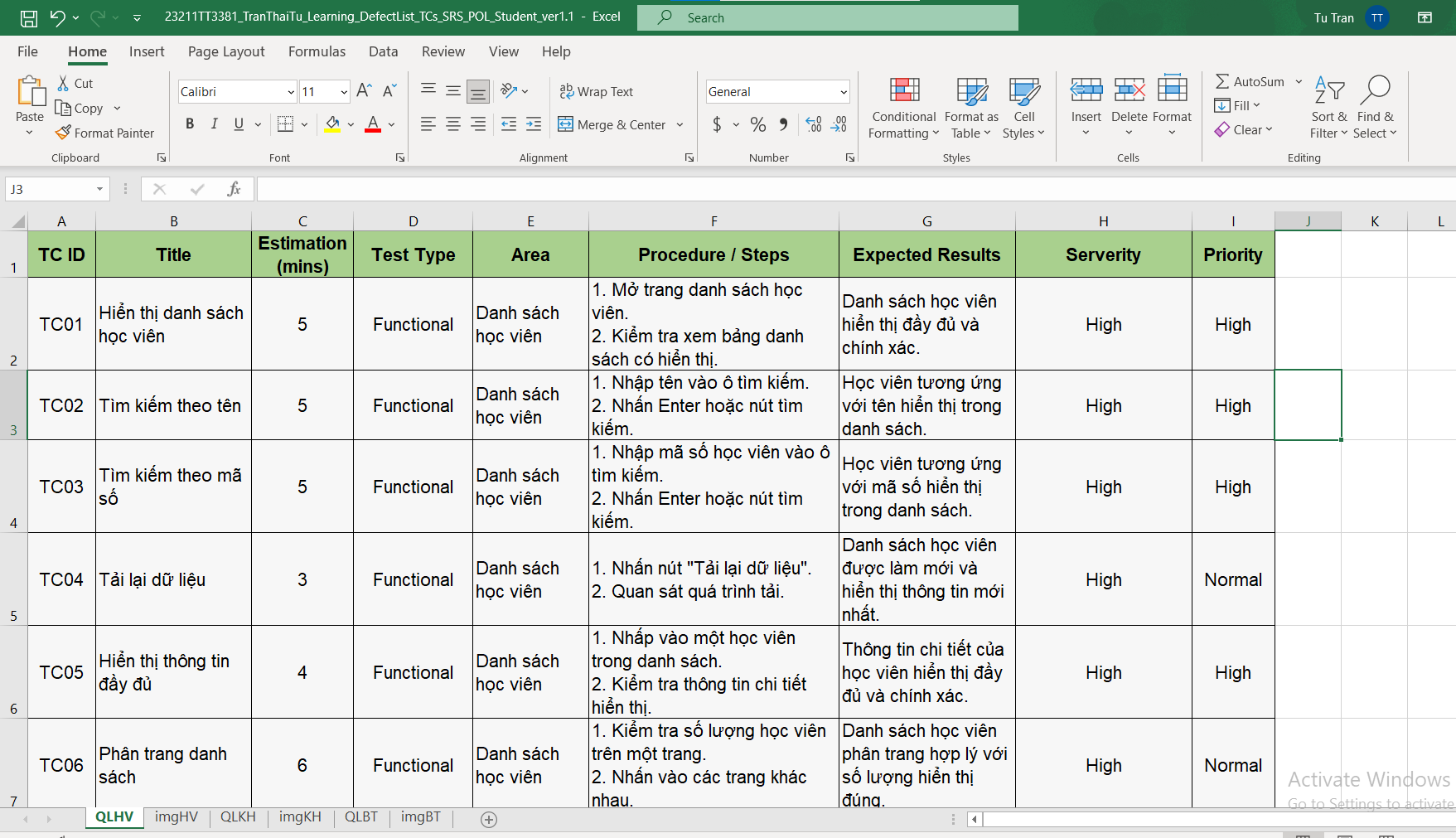
## Viết Test Case cho SRS\_POL (ver1.1).

### **Test Design.**

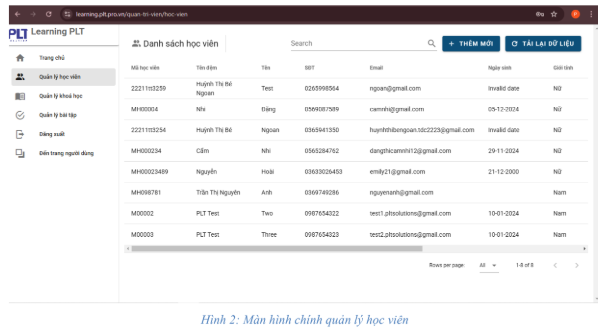
-Trang web cho phép người dùng dưới quyền Admin quản lý học viên bằng cách (thêm, xóa, sửa, tìm kiếm,…). Củng như là quản lý thêm Khóa học, Bài tập.

-Trang web cho phép người dùng dưới quyền Học sinh có thể xem thông tin của mình và có chức năng tìm kiếm. Củng như có những quyền của khóa học (hủy khóa học, đăng ký khóa học, tìm kiếm) sau khi đã có khóa học thì bài tập mới có thông tin và học sinh chỉ có quyền xem bài tập, làm bài tập và nộp bài tập.

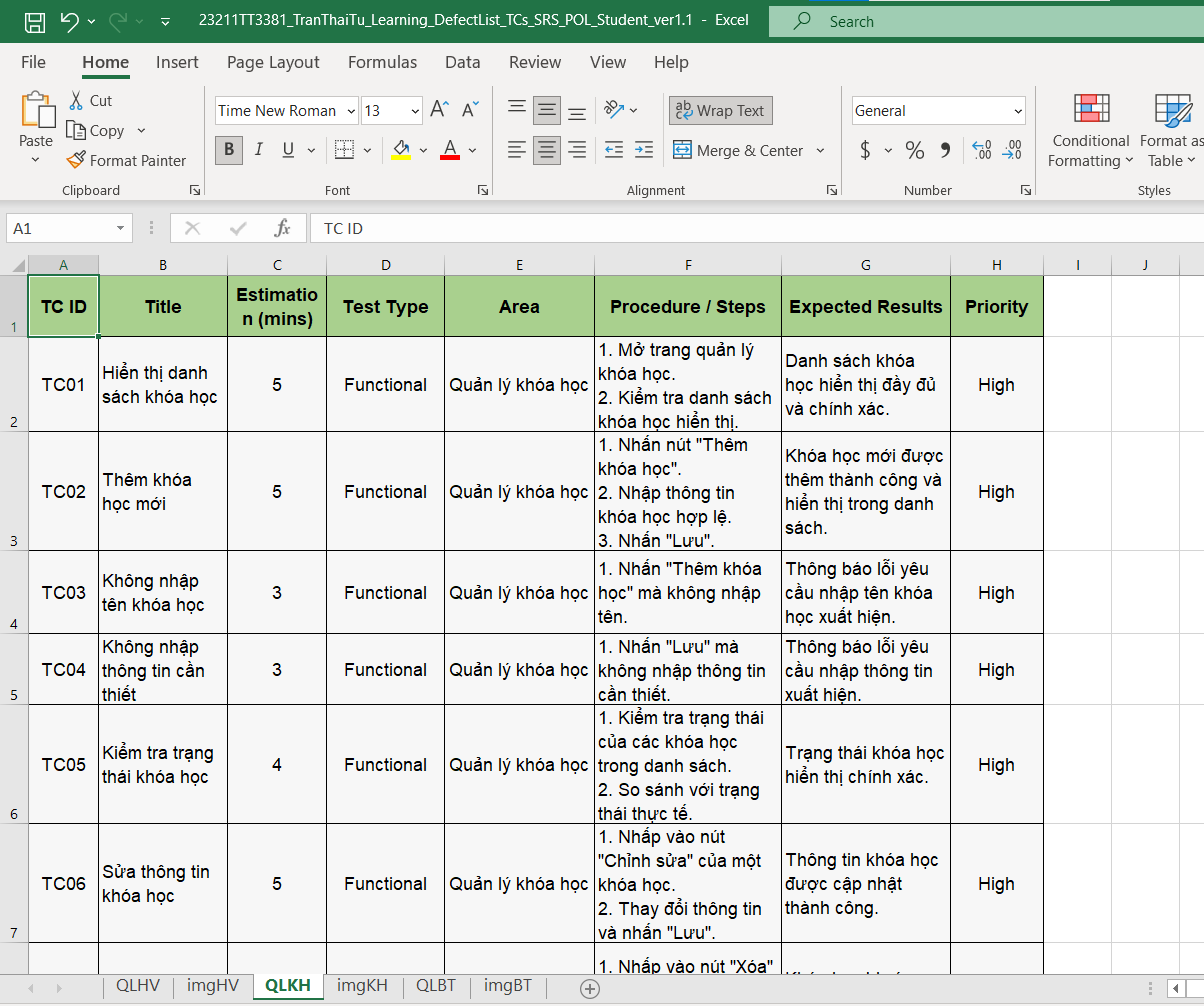
### **Hình ảnh Một số Test Case và màn hình minh họa trang web (SRS\_POL).**



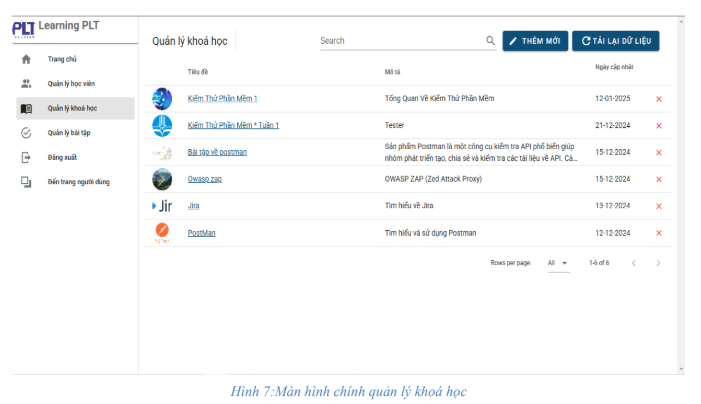
Hình 3.1: Một số Test Case minh họa chức năng Quản lý học viên.



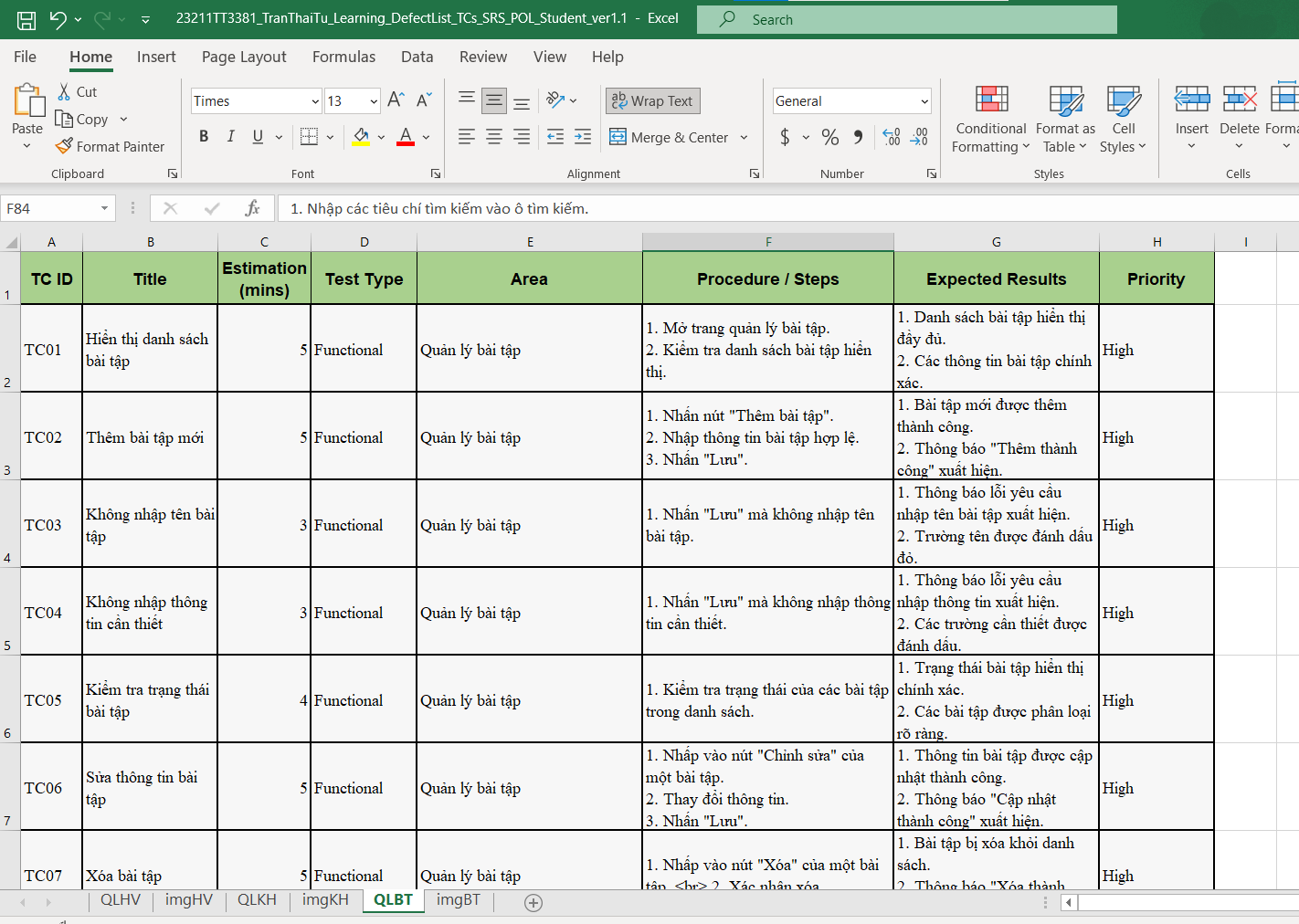
Hình 3.2: Màn hình minh họa chức năng Quản lý học viên.



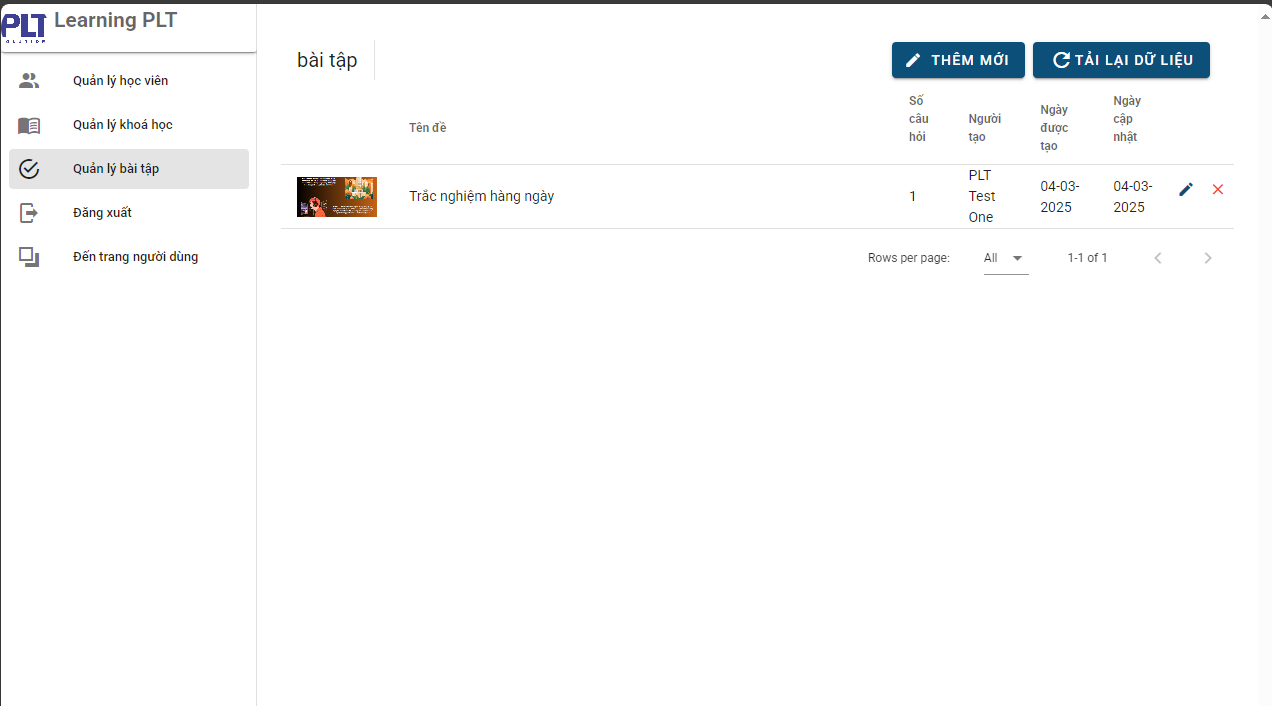
Hình 3.3: Một số Test Case minh họa chức năng Quản lý khóa học.



Hình 3.4: Màn hình minh họa chức năng Quản lý khóa học.



Hình 3.5: Một số Test Case minh họa chức năng Quản lý bài tập.



Hình 3.1: Màn hình minh họa chức năng Quản lý bài tập.

## Code kiểm thử tự động Selenium.

### **Test Design.**

- Selenium WebDriver: Thư viện mã nguồn mở giúp điều khiển trình duyệt tự động.

- Ngôn ngữ sử dụng: JavaScript,…

- Quy trình cơ bản:

Khởi tạo WebDriver (ví dụ: ChromeDriver).

Mở trang web cần kiểm thử.

Tìm kiếm và tương tác với các phần tử (nhập liệu, nhấn nút, chọn tùy chọn,...).

Kiểm tra kết quả (so sánh nội dung, URL, trạng thái,...).

Đóng trình duyệt sau khi hoàn tất.

Ưu điểm:

- Tự động hóa kiểm thử nhanh chóng.

- Lặp lại dễ dàng nhiều lần.

- Hỗ trợ nhiều trình duyệt.

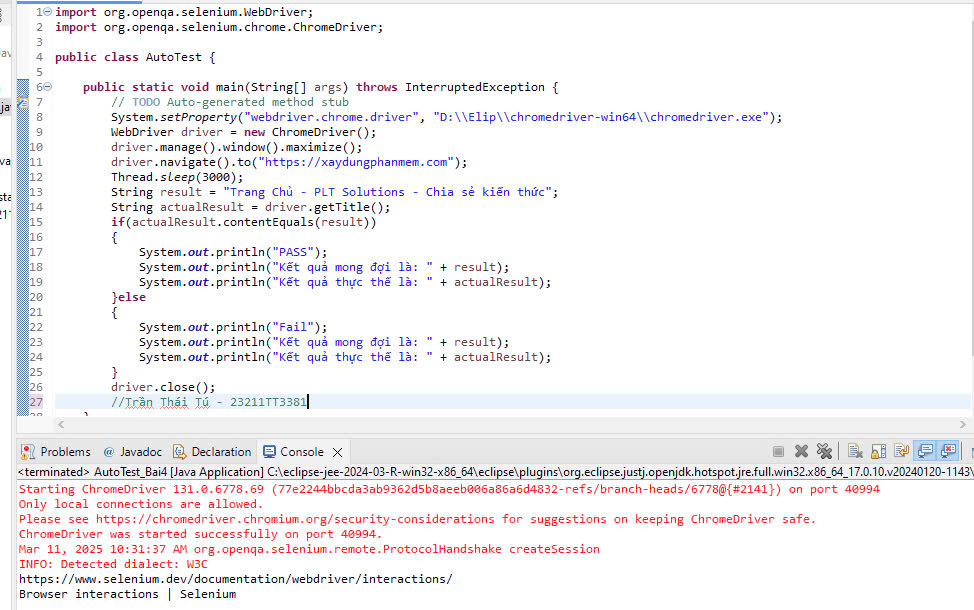
Nhược điểm:

- Cần bảo trì mã test khi giao diện thay đổi.

- Chạy chậm hơn kiểm thử ở mức code (unit test).

### **Hình ảnh minh họa code Selenium.**

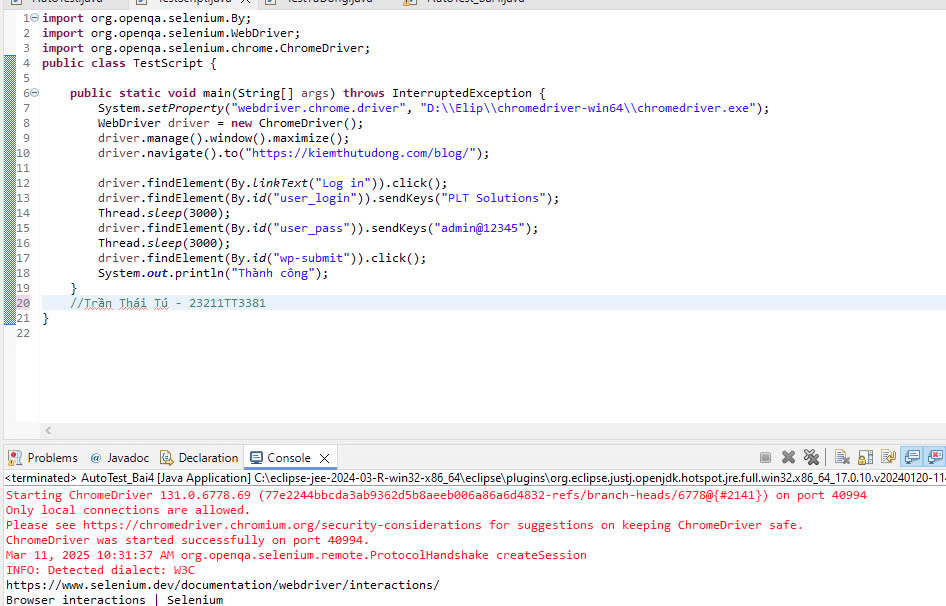
-Dòng 13: Code kiểm tra giao diện của trang.

-Dòng 16 -> 24: Những dòng code đưa ra kết quả của Test. 

Hình 4.1: Code minh họa kiểm tra kết quả thực tế có giống mong đợi.

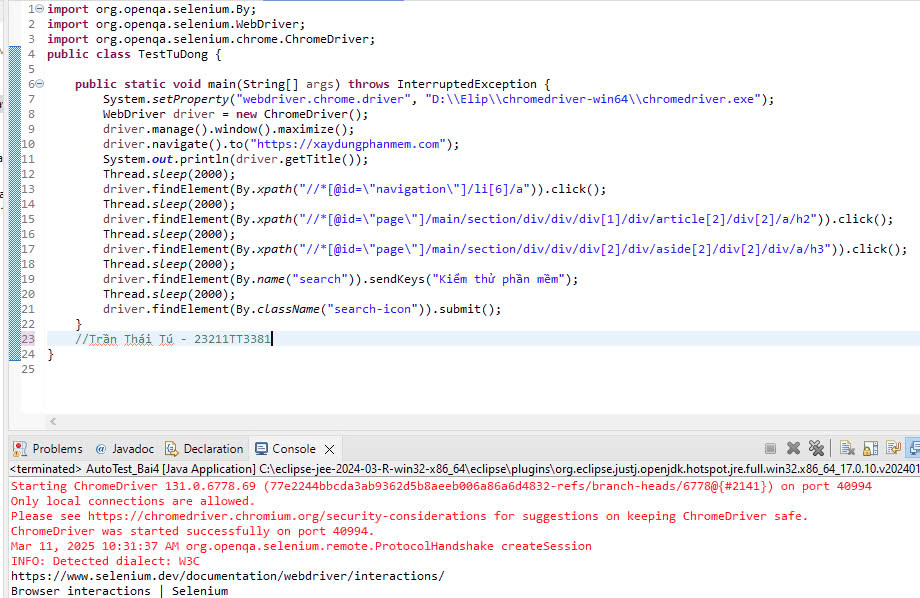
-Dòng 13 - 18: Code dùng để máy tính tự động click vào Textbox hoặc Button

-Dòng 14 – 16: Code dùng để máy tính tự động nhập nội dung vào Textbox

-Dòng 19: Giúp hiển thị kết quả kiểm thử đã thành công.

Hình 4.2: Code minh họa kiểm tra kết quả đăng nhập.

-Dòng 14-16-18-22: Code dùng để máy tính tự động click vào Textbox hoặc Button.

-Dòng 20: Code dùng để máy tính tự động nhập nội dung vào Textbox.

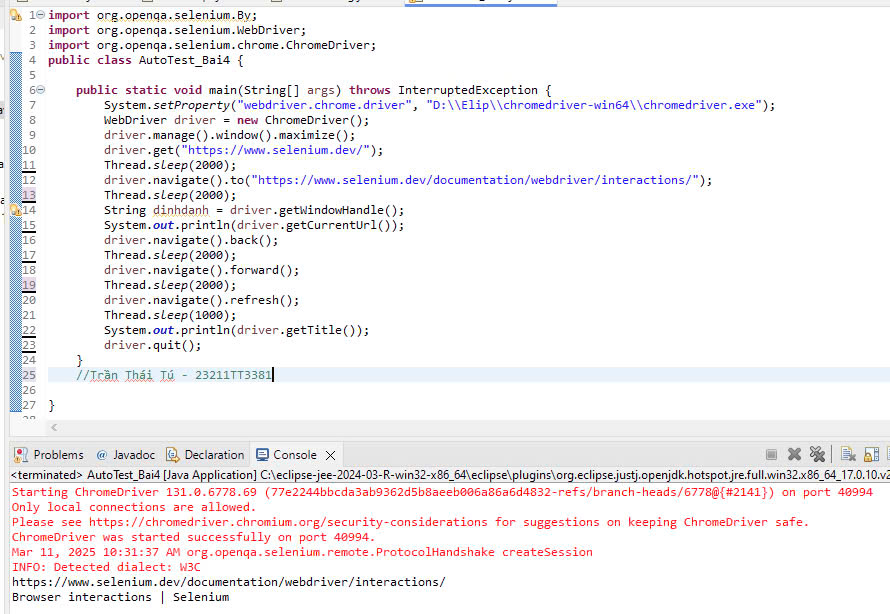
Hình 4.3: Code minh họa nhấn và nhập nội dung.

-Dòng 14-15: Lấy mã định danh của cửa sổ trình duyệt hiện tại.

-từ Dòng 18 -> 24: Quay lại, tiến tới và tải lại trang.

-Dòng 25: Lấy tiêu đề của trang web hiện tại.

-Dòng 27: Đóng trình duyệt sau khi hoàn thành thao tác



Hình 4.1: Code minh họa thực hiện các thao tác trên selenium.

## Code kiểm thử tự động Selenium (lần 2).

### **Test Design.**

- Selenium WebDriver: Thư viện mã nguồn mở giúp điều khiển trình duyệt tự động.

- Ngôn ngữ sử dụng: JavaScript,…

- Quy trình cơ bản:

Khởi tạo WebDriver (ví dụ: ChromeDriver).

Mở trang web cần kiểm thử.

Tìm kiếm và tương tác với các phần tử (nhập liệu, nhấn nút, chọn tùy chọn,...).

Kiểm tra kết quả (so sánh nội dung, URL, trạng thái,...).

Đóng trình duyệt sau khi hoàn tất.

Ưu điểm:

- Tự động hóa kiểm thử nhanh chóng.

- Lặp lại dễ dàng nhiều lần.

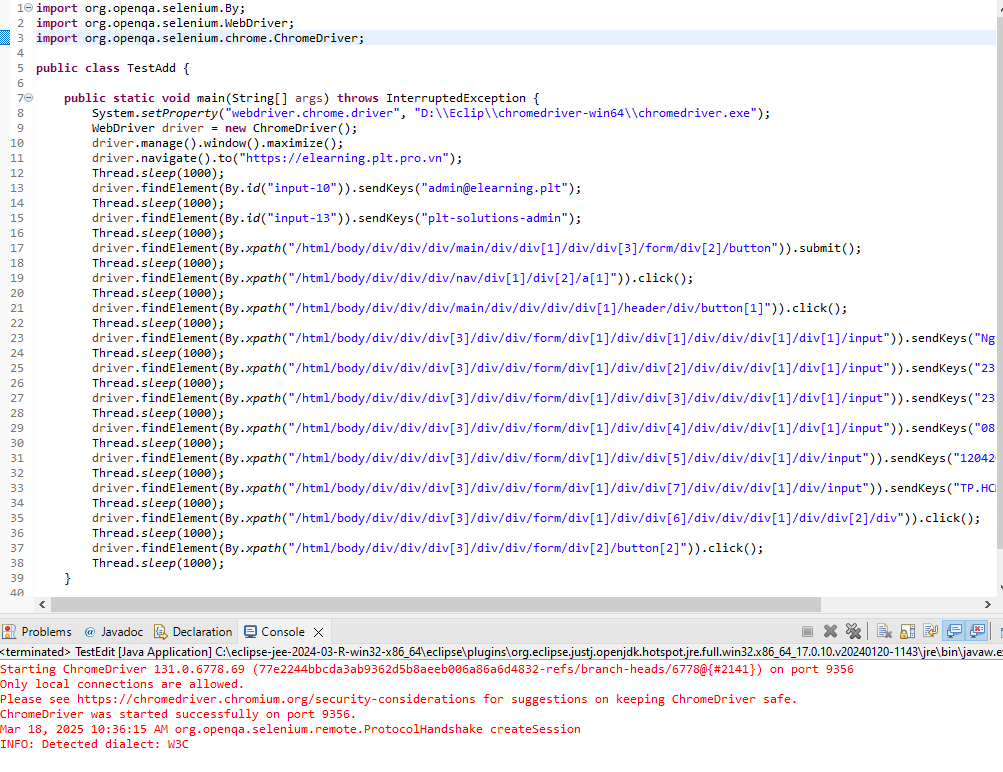
- Hỗ trợ nhiều trình duyệt.

Nhược điểm:

- Cần bảo trì mã test khi giao diện thay đổi.

- Chạy chậm hơn kiểm thử ở mức code (unit test).

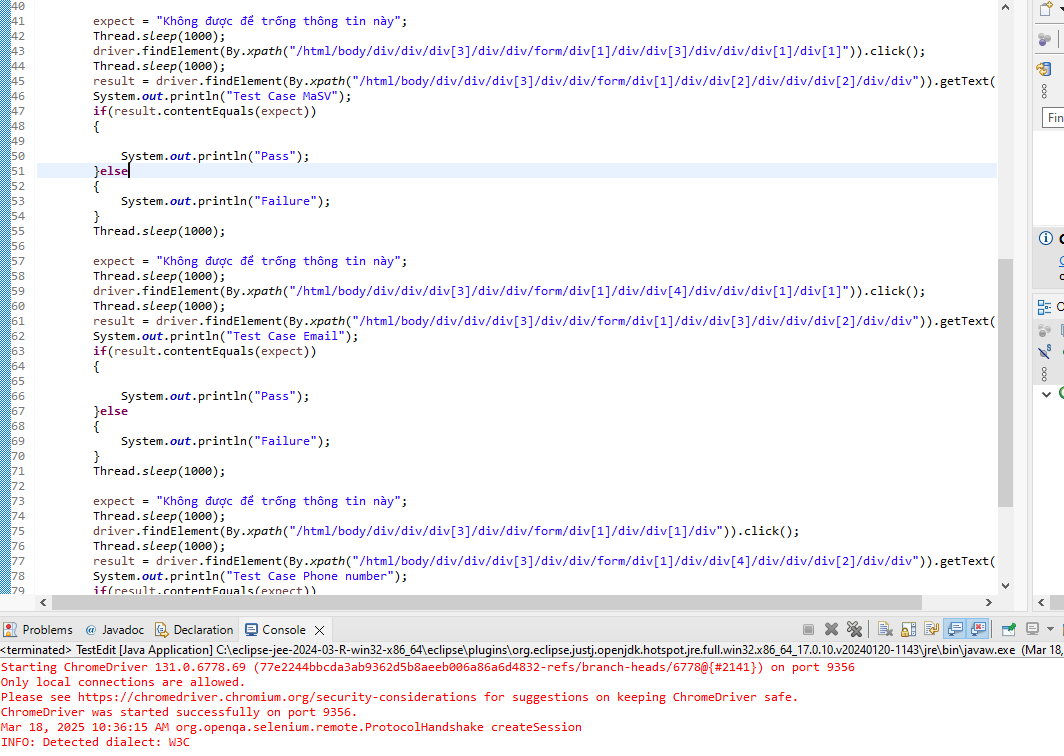
### **Hình ảnh minh họa Code Selenium.**



Hình 5.1: Hình minh họa Code (Thêm đúng).



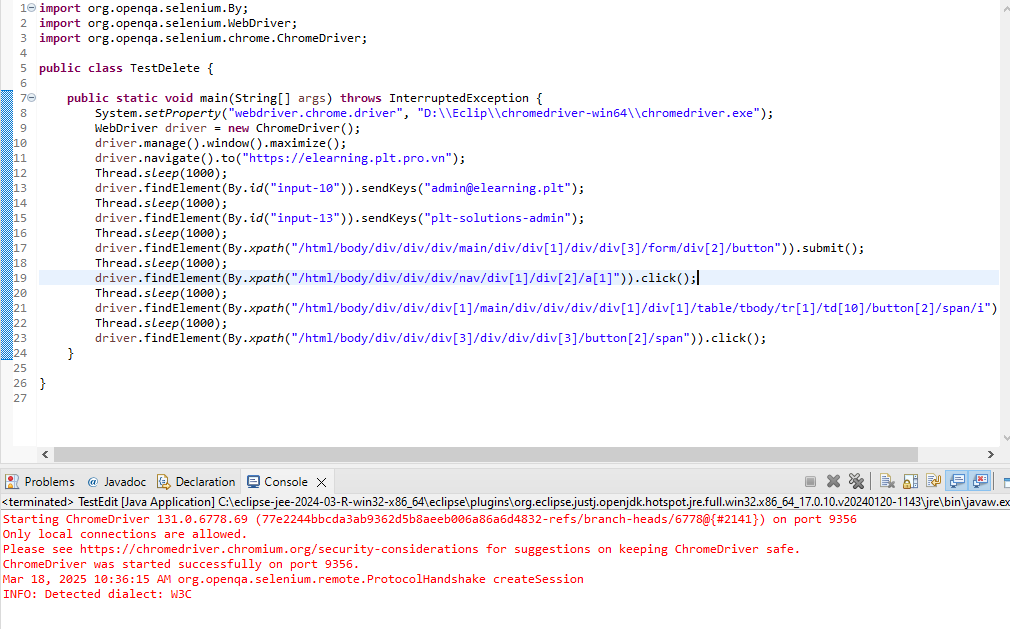
Hình 5.2.a: Hình minh họa Code (Thêm sai).



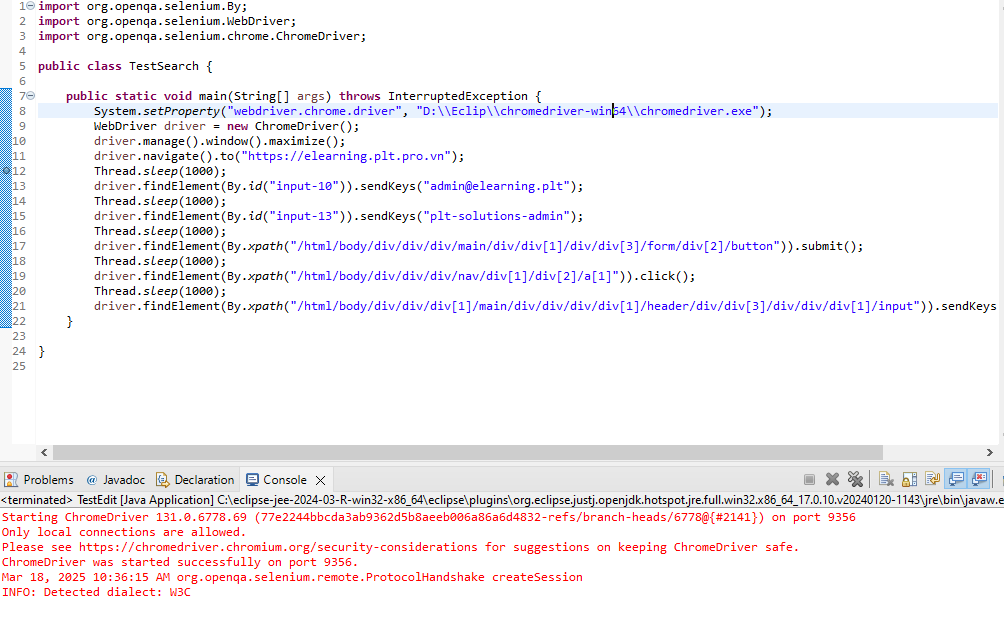
Hình 5.2.b: Hình minh họa Code (Thêm sai).



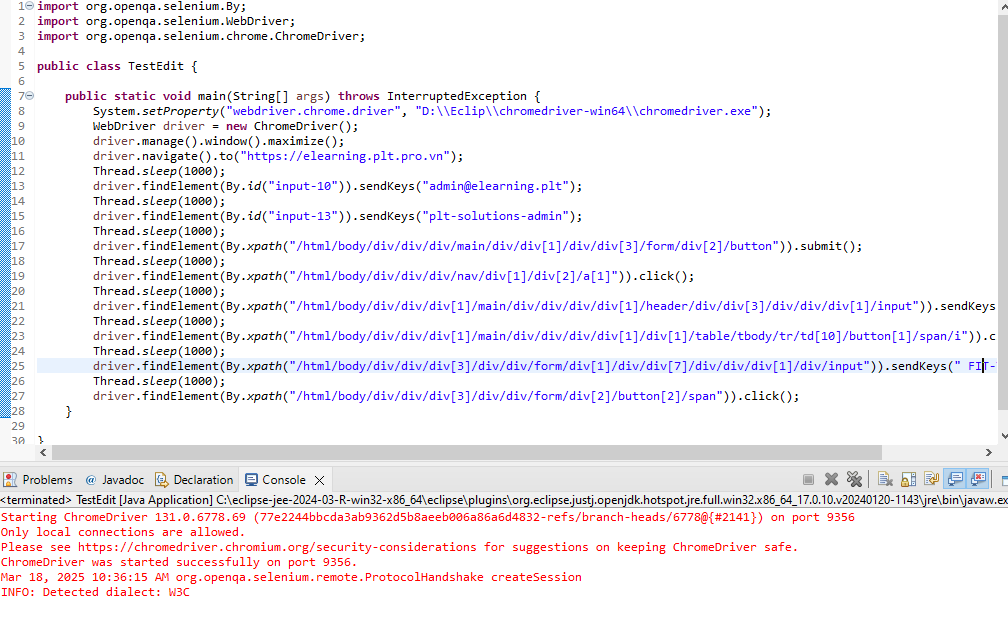
Hình 5.2.c: Hình minh họa Code (Thêm sai).



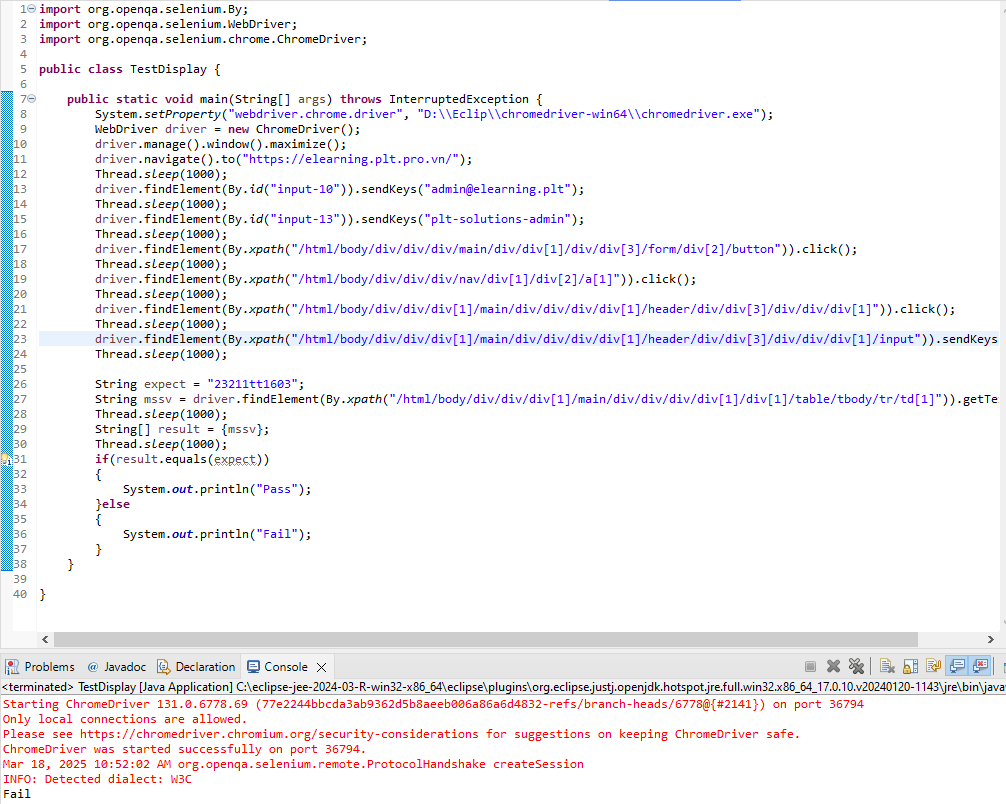
Hình 5.3: Hình minh họa Code (Xóa).



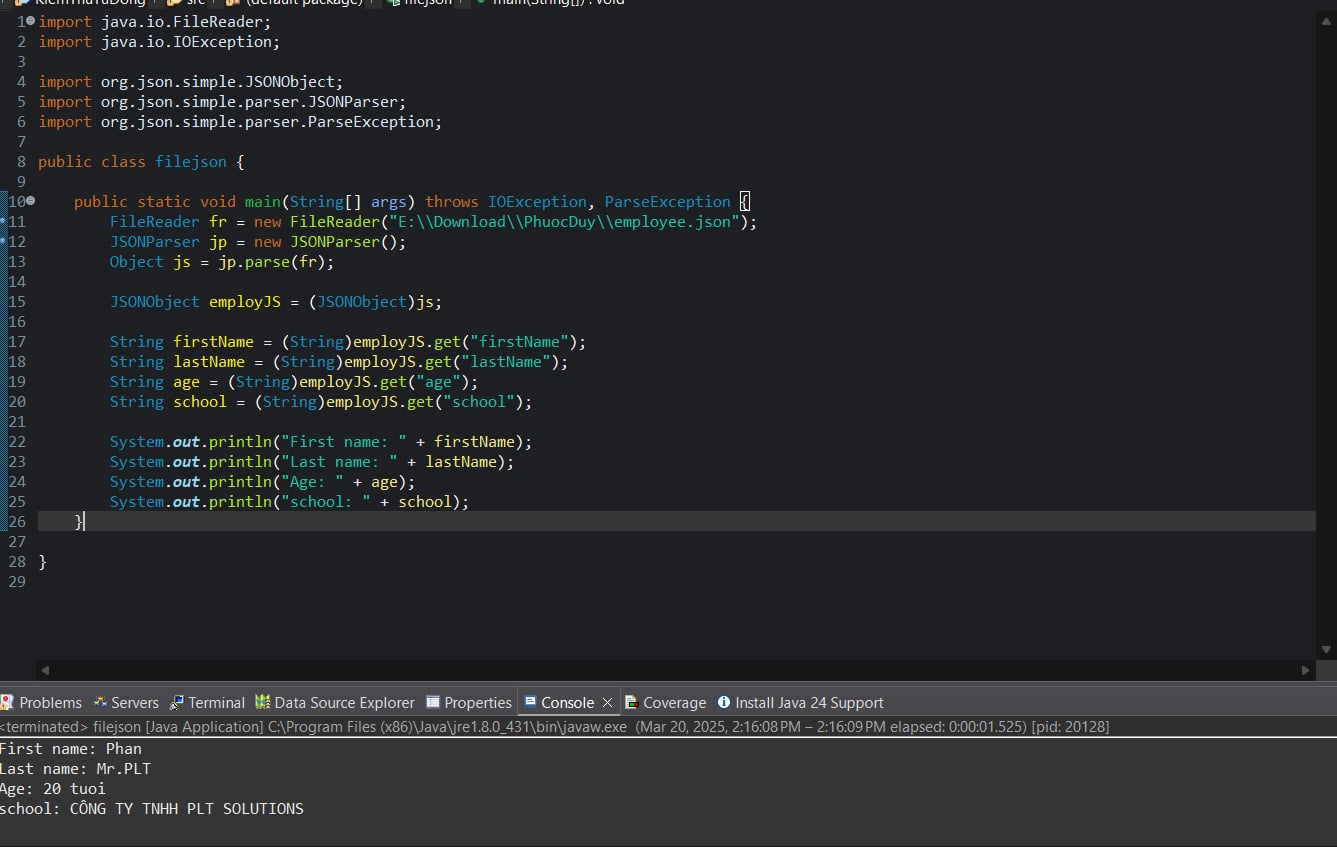
Hình 5.4: Hình minh họa Code (Tìm Kiếm).



Hình 5.5: Hình minh họa Code (Sửa).



Hình 5.6: Hình minh họa Code (Kiếm tra Display).



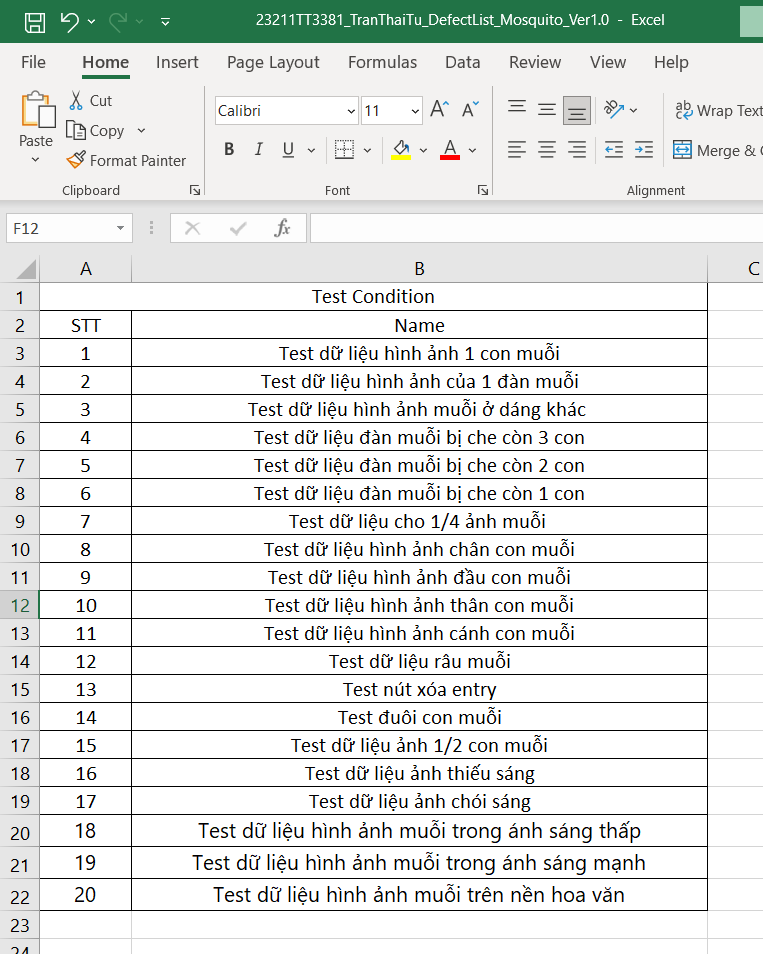
Hình 5.3: Hình minh họa Code (Lấy dữ liệu từ file JSON).

## Viết Defect List cho trang web con muỗi.

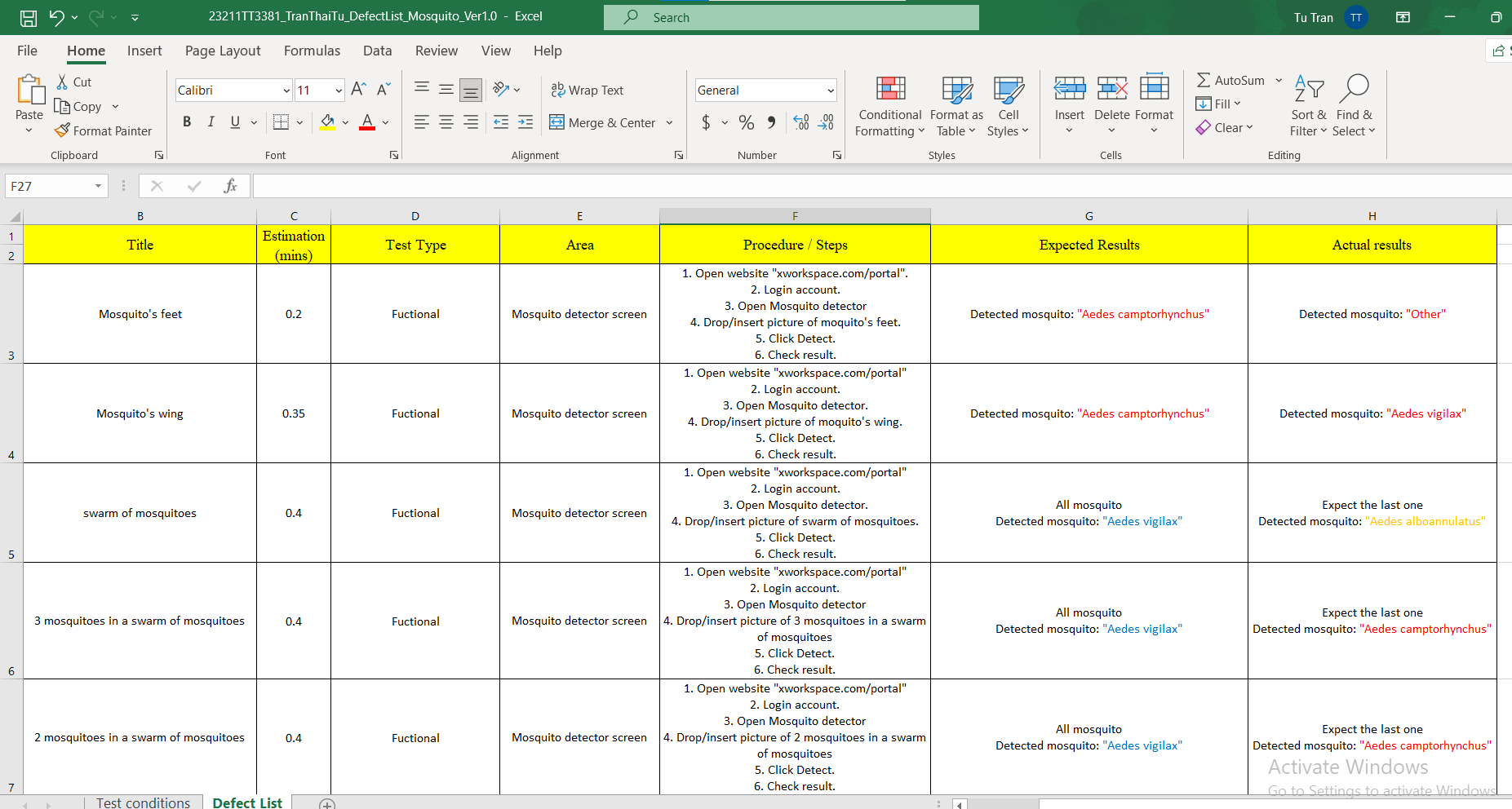
### **Test Design.**

-Trang web con muỗi là trang web cho người dùng truy cập và sử dụng miễn phí. Cho phép người dùng hình ảnh con muỗi chụp ảnh được và gửi lên cho AI của trang web con muỗi tìm kiếm và đưa ra kết quá nhanh nhất. Ngoài ra người dùng còn có thêm nhiều chức năng khác để có thể truy tìm loài muỗi mình cần tìm nhanh chống và chính xác hơn.

### **Hình ảnh minh họa Defect List và Test Condition.**



Hình 6.1: hình ảnh minh họa Test Design Mosquito.



Hình 6.2: hình ảnh minh họa Defect List Mosquito.

## Code kiểm thử tự động POSTMAN.

### **Test Design.**

Quy trình cơ bản:

-Tạo request API (GET, POST, PUT, DELETE,...).

-Gửi request đến server.

-Viết test script để kiểm tra phản hồi (status, dữ liệu, thời gian phản hồi,...).

-Chạy test tự động hoặc trong Collection Runner/CI tool.

Ưu điểm:

-Dễ sử dụng, không cần nhiều kiến thức lập trình.

-Hữu ích để kiểm thử API nhanh chóng.

-Có thể chạy hàng loạt với Collection Runner hoặc tích hợp CI/CD.

Nhược điểm:

-Không phù hợp với kiểm thử giao diện (UI).

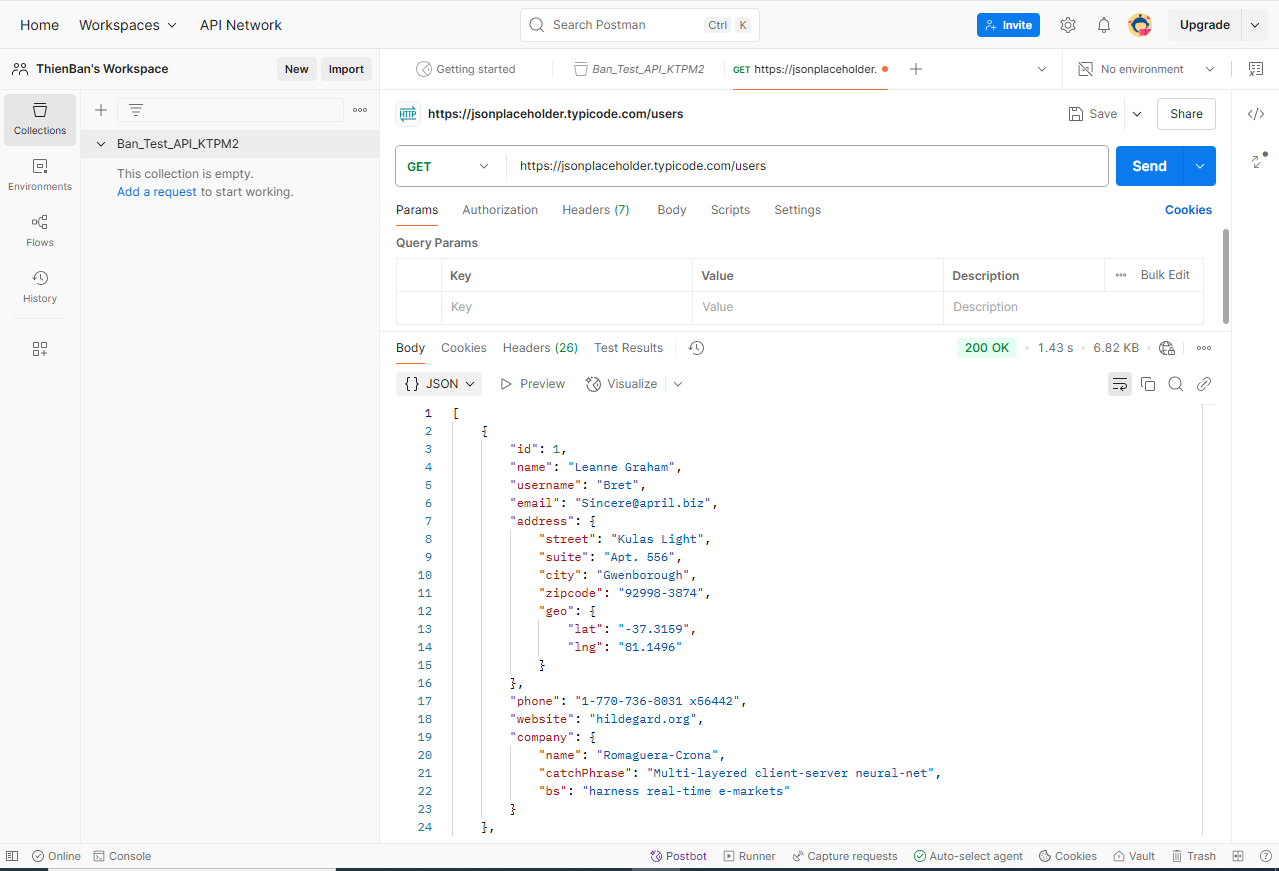
-Tính năng kiểm thử nâng cao còn hạn chế so với framework chuyên biệt.

### **Hình ảnh minh họa Code POSTMAN.**

Tính năng GET: Tính năng lấy và hiển thị Dữ liệu trong File JSON.

Vào GET -> Nhập link JSON -> nhấn nút Send.

Sau khi nhấn Send xem kết quả có hiện thị đúng không.

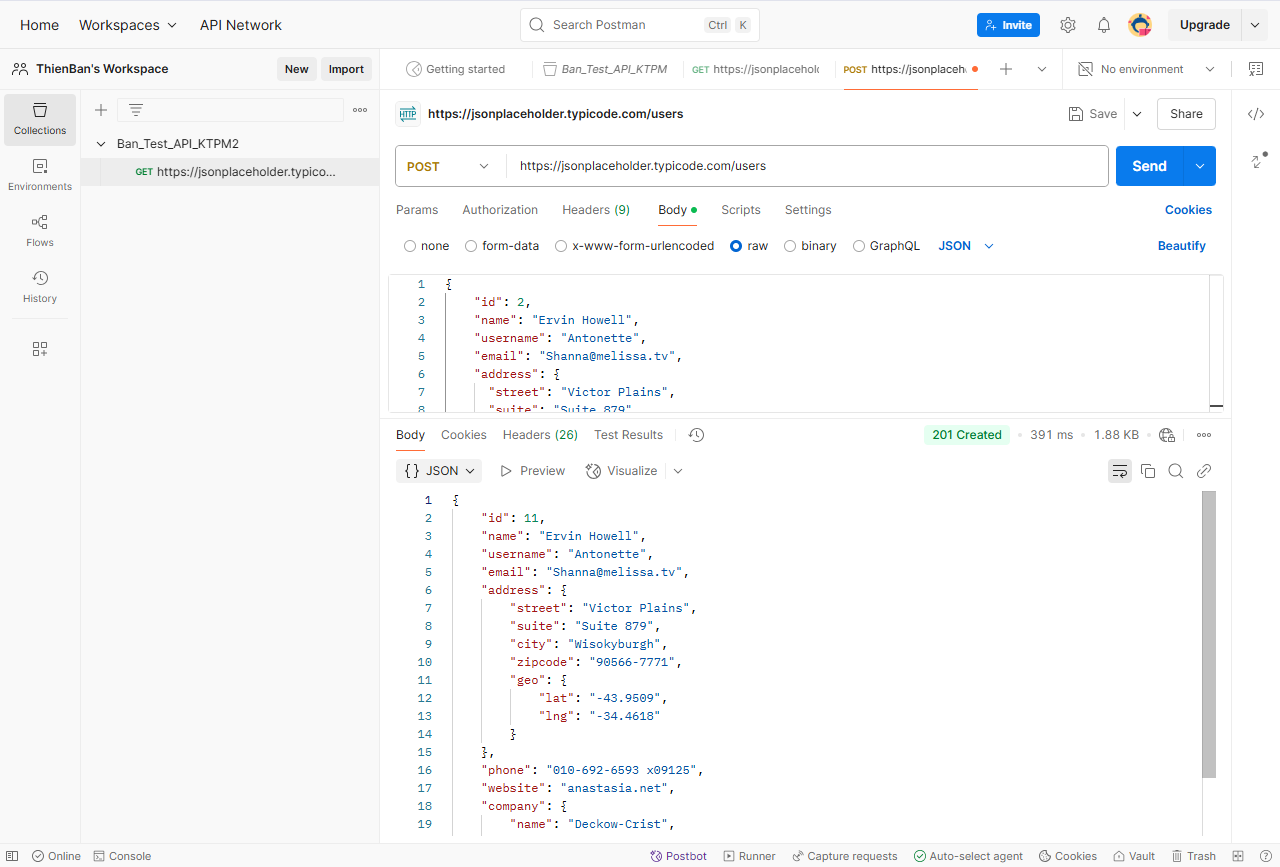


Hình 7.1: Code minh họa Test API (Get).

Tính Năng Post: test tính năng thêm dữ liệu ảo vào file JSON.

Vào POST -> nhập link JSON -> trên thanh mục lục chọn Body -> chọn raw -> nhập dữ liệu vào -> Nhấn Send

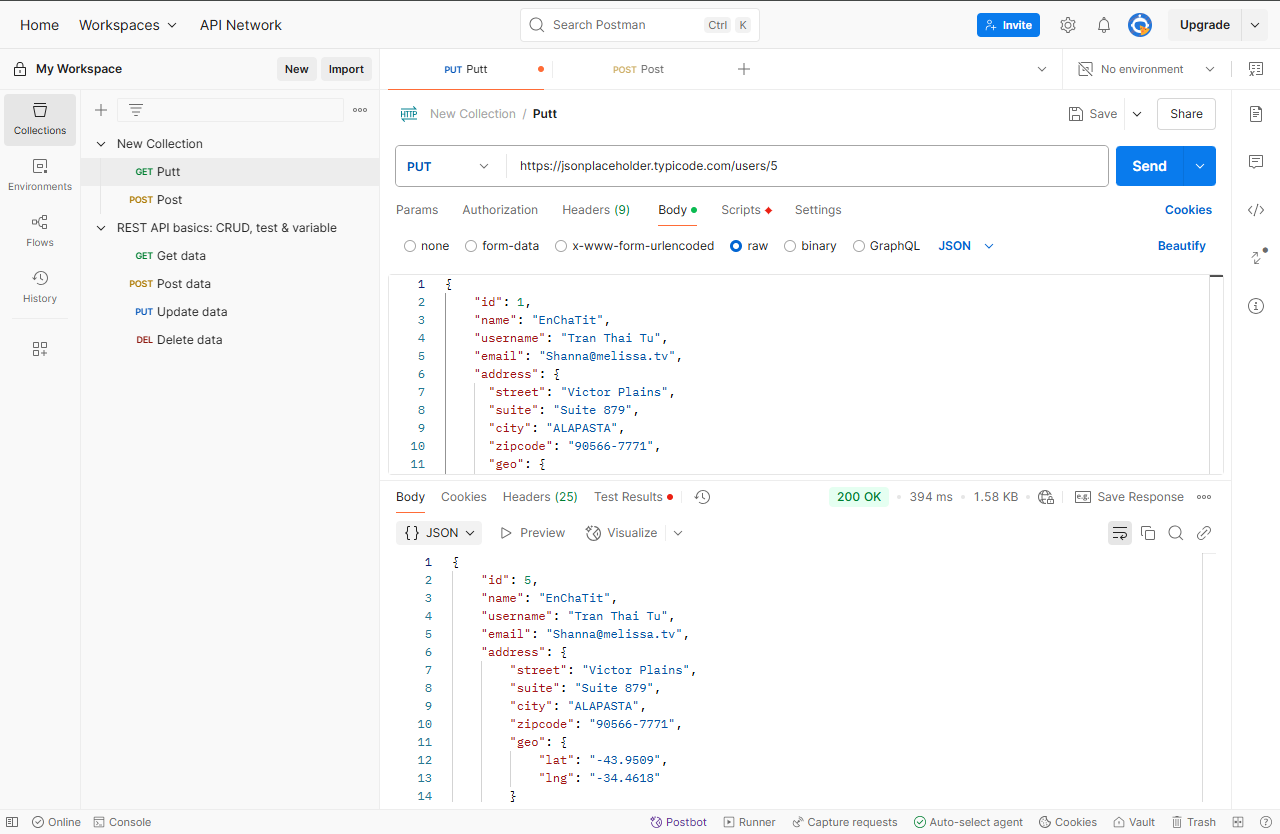
Dòng code 2 -> 21: code thêm dữ liệu ảo vào File JSON.



Hình 7.2: Code minh họa Test API (POST).

Tính năng PUT: test tính năng sửa dữ liệu ảo vào file JSON

Vào PUT -> nhập link JSON (chọn link dữ liệu cần sửa)-> trên thanh mục lục chọn Body -> chọn raw -> nhập dữ liệu vào -> Nhấn Send

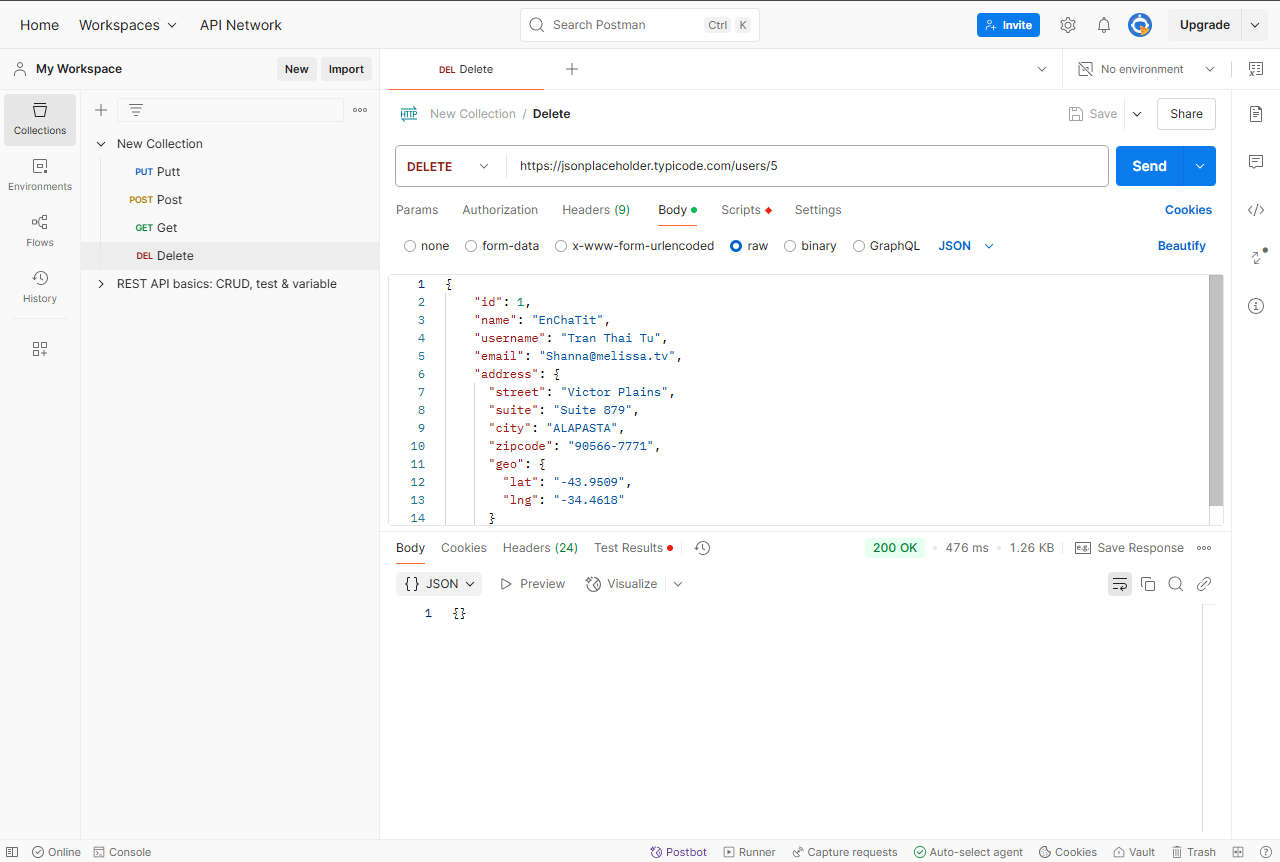
Dòng code 2 -> 13: Các dòng code Sửa dữ liệu ảo trong file JSON.

Hình 7.3: Code minh họa Test API (PUT).

Tính năng DELETE: test tính năng xóa dữ liệu ảo từ file JSON

Vào GET -> Nhập link JSON (chọn link dữ liệu cần xóa)-> nhấn nút Send.

Sau khi nhấn Send kiểm tra xem dữ liệu ảo có được xóa hay chưa và chạy thành công không.



Hình 7.4: Code minh họa Test API (DELETE).

Tính năng GET -> scripts: Code kiểm tra.

Vào GET -> Nhập link JSON -> trên thanh mục lục chọn Scripts -> dưới góc bên phải chọn Shippets -> chọn Status code: Code is 200 -> Send -> Test Results

Vào GET -> Nhập link JSON -> trên thanh mục lục chọn Scripts -> dưới góc bên phải chọn Shippets -> chọn Response time is less than 200ms -> Send -> Test Results

Vào GET -> Nhập link JSON -> trên thanh mục lục chọn Scripts -> nhập Code tìm kiếm dữ liệu cụ thể -> Send -> Test Results

Sau khi nhấn Send nếu kiểm tra đúng thì Hiện “PASSED” ngược lại hiện “FAILED”

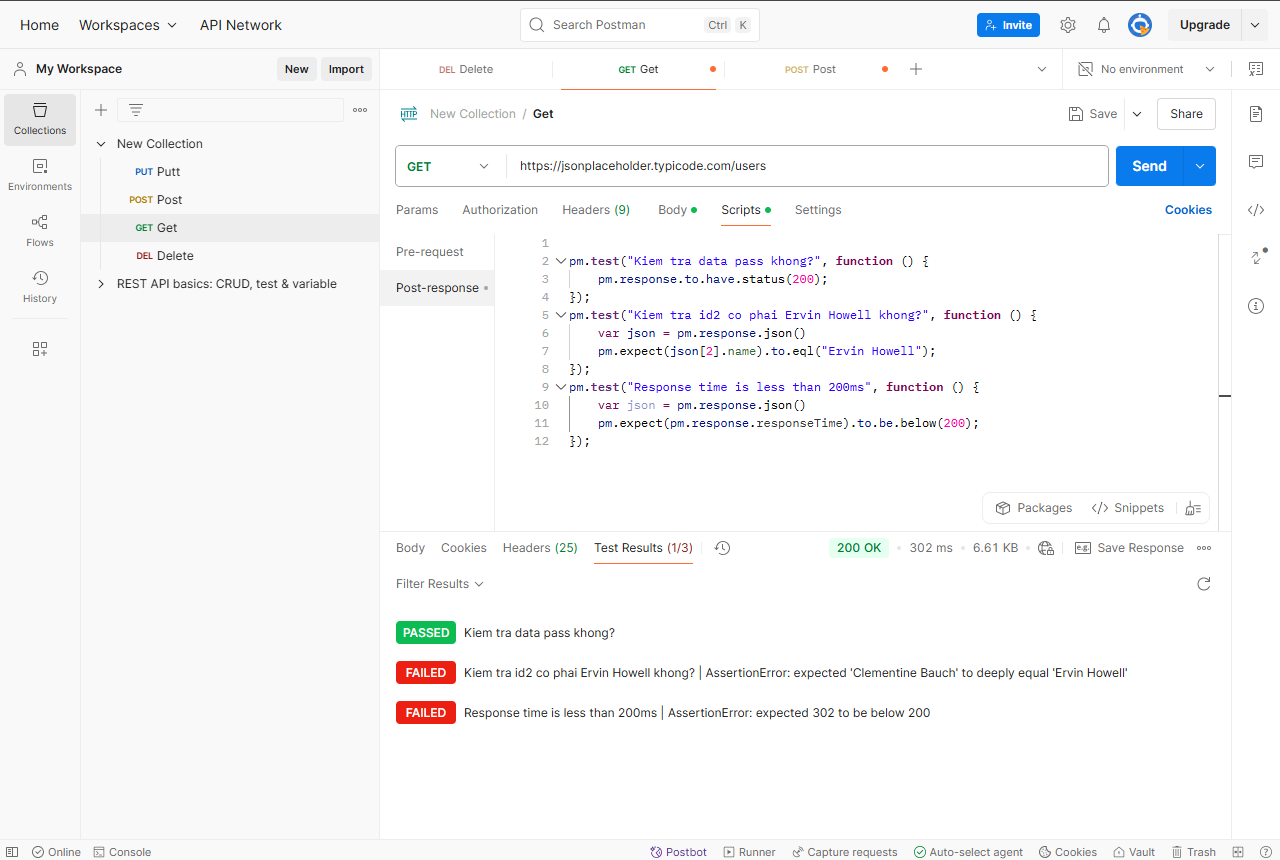
Dòng code 1: Dòng hiển thị kết quả

Dòng code 2: Kiểm tra xem kết quả có ra 200 không

Dòng code 4: Dòng hiển thị kết quả

Dòng code 5: Kiểm tra xem thời gian chạy có be hơn 500ms không

Dòng code 7: Dòng hiển thị kết quả

Dòng code 8: Tìm xem trong dữ liệu JSON tại ID3 có tên Clementine Bauch không

Hình 7.5: Code minh họa Test API (CODE - GET).

(Hình 7.6.a) Vào Environments -> Global: Thêm giá trị Variable chon cục bộ rồi Save.

(Hình 7.6.b) Vào POST -> Nhập link JSON ->trên thanh chọn Body -> chọn raw -> nhập code rồi nhấn Send.

Dòng code 2 -> 5: các dòng code cho nó chạy đọc và nhập dữ liệu từ File JSON.

(Hình 7.6.c) Tính năng POST -> scripts: Code kiểm tra.

Vào GET -> Nhập link JSON -> trên thanh mục lục chọn Scripts -> dưới góc bên phải chọn Shippets -> chọn Status code: Code is 201 -> Send -> Test Results

Vào GET -> Nhập link JSON -> trên thanh mục lục chọn Scripts -> dưới góc bên phải chọn Shippets -> chọn Response time is less than 3000ms -> Send -> Test Results

Sau khi nhấn Send nếu kiểm tra đúng thì Hiện “PASSED” ngược lại hiện “FAILED”

Dòng code 1: Dòng hiển thị kết quả

Dòng code 2: Kiểm tra xem kết quả có ra 201 không

Dòng code 4: Dòng hiển thị kết quả

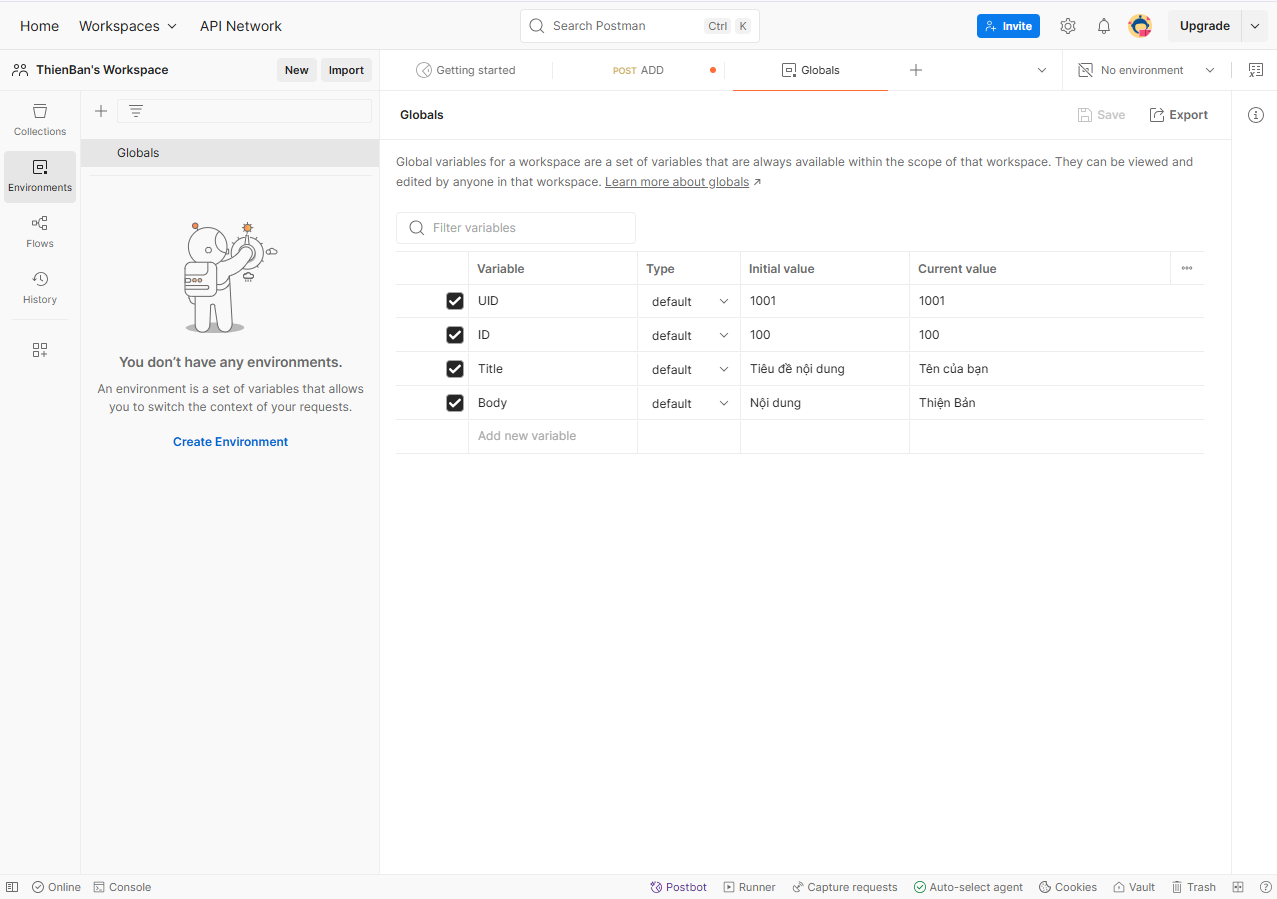
Dòng code 5: Kiểm tra xem thời gian chạy có be hơn 3000ms không

(Hình 7.6.d) Vào Mục tổng (Ban\_Test\_API\_KTPM2) chọn Run – Tác checkbox các phần không liên quan -> chọn Select File -> chọn File JSON hoặc File CVS -> chọn Run.

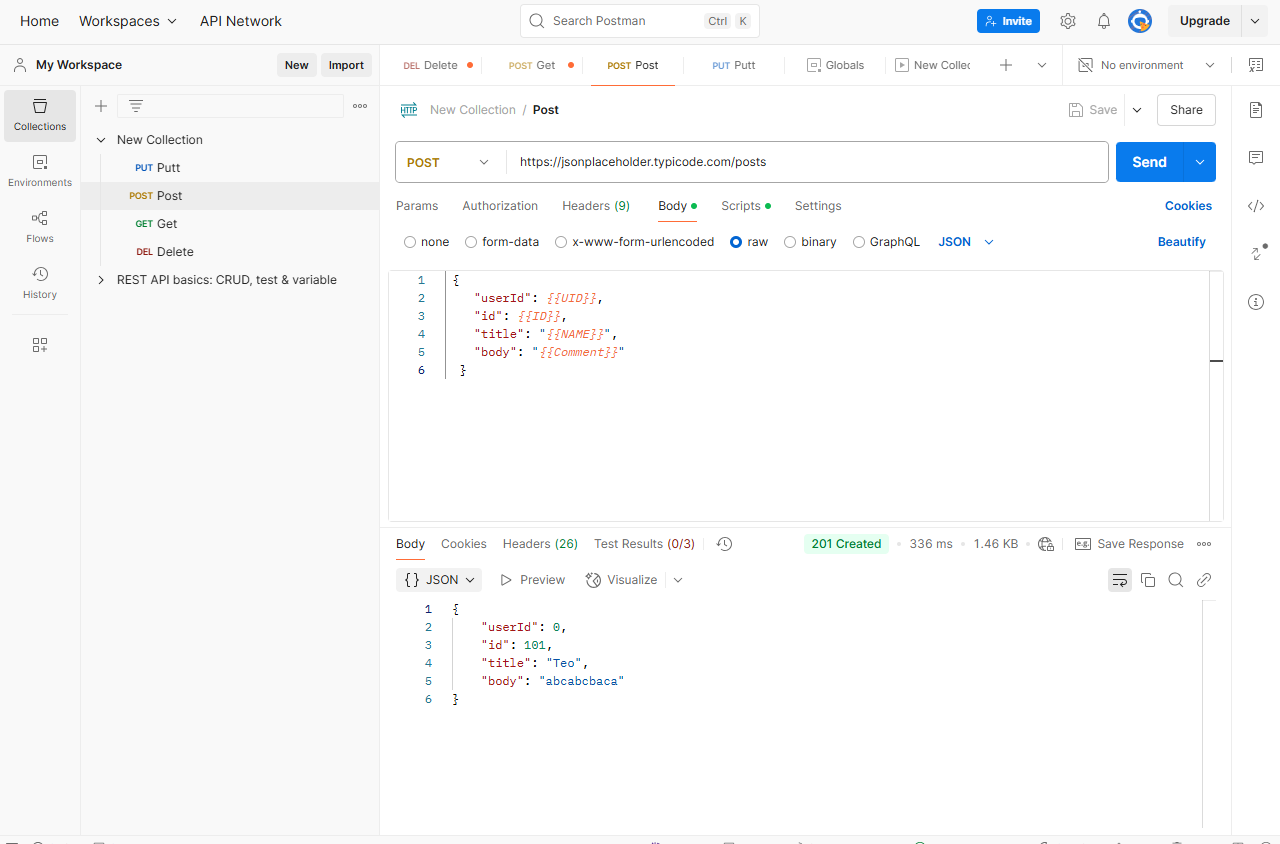
Nó sẽ hiển thị kết quả thêm vào sau khi đọc dữ liệu từ File JSON hoặc File CVS

(Hình 7.6.e) Sau khi chạy xong chọn bất kì kết quả nào xem kết quả thêm được sau khi đọc File JSON hoặc File CVS từ dòng 2 -> 5

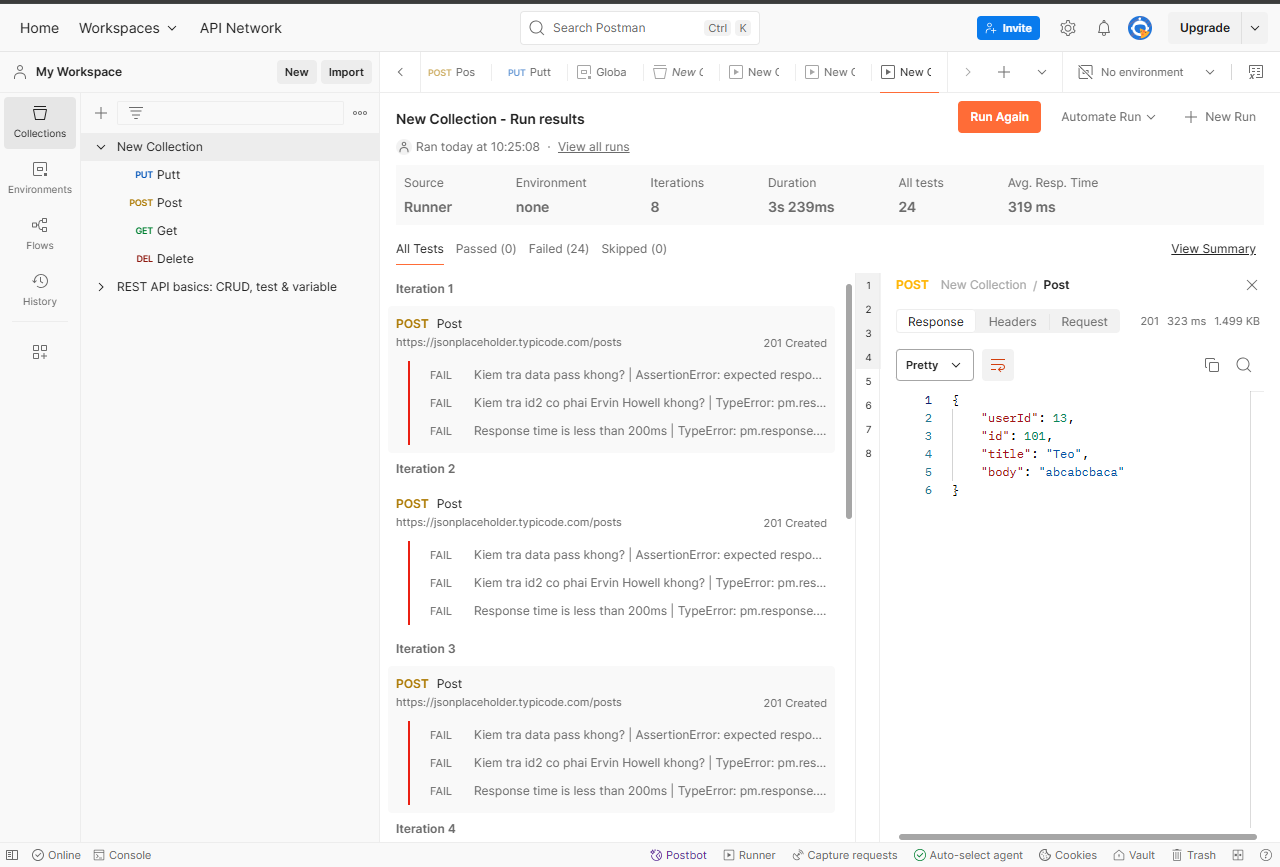
Kết quả “PASSED” hoặc “FAILED” nó sẽ kiểm tra bên phần Scripts của mục liên quan (ví dụ: vào POST -> chọn Scripts sau khi nhấn send chọn mục Test Results dưới kết quả).



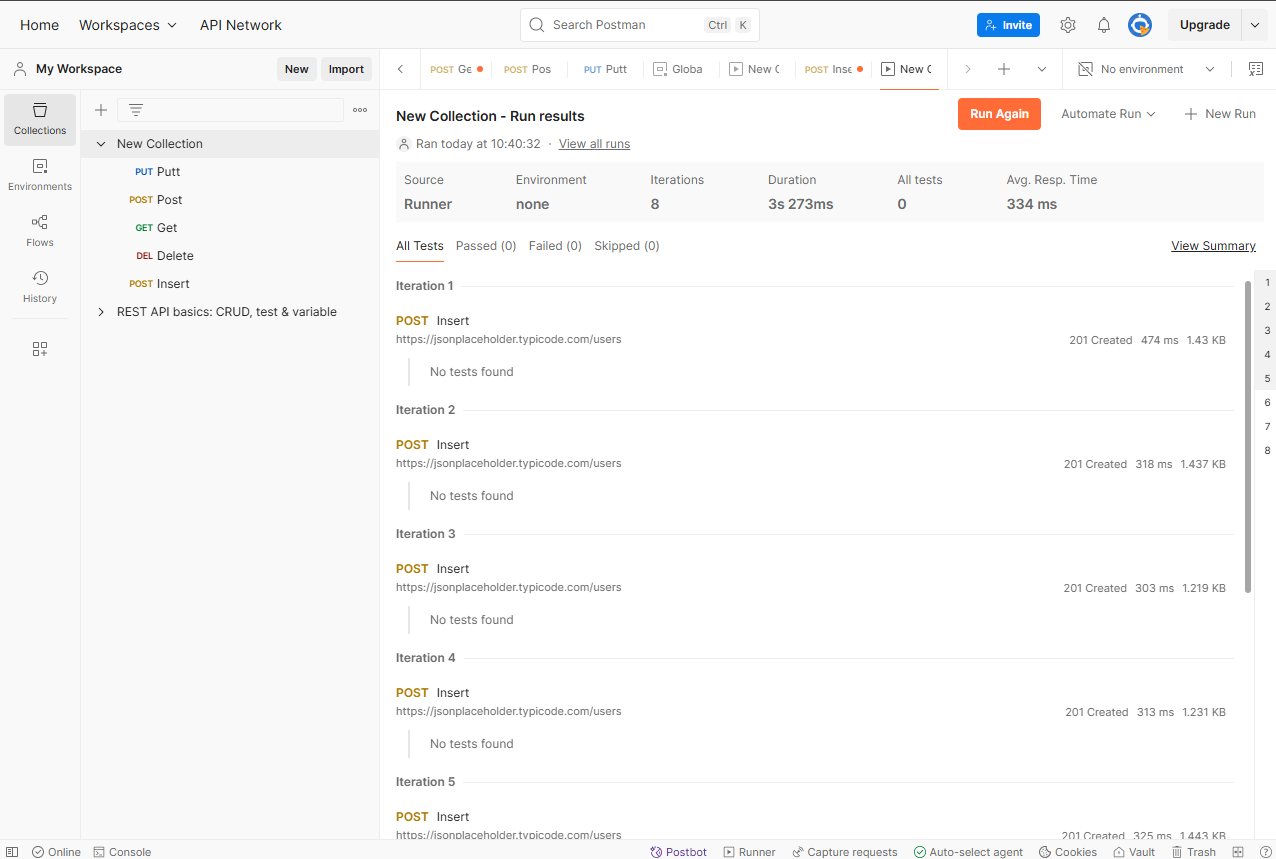
Hình 7.6.a: Minh họa bước làm.



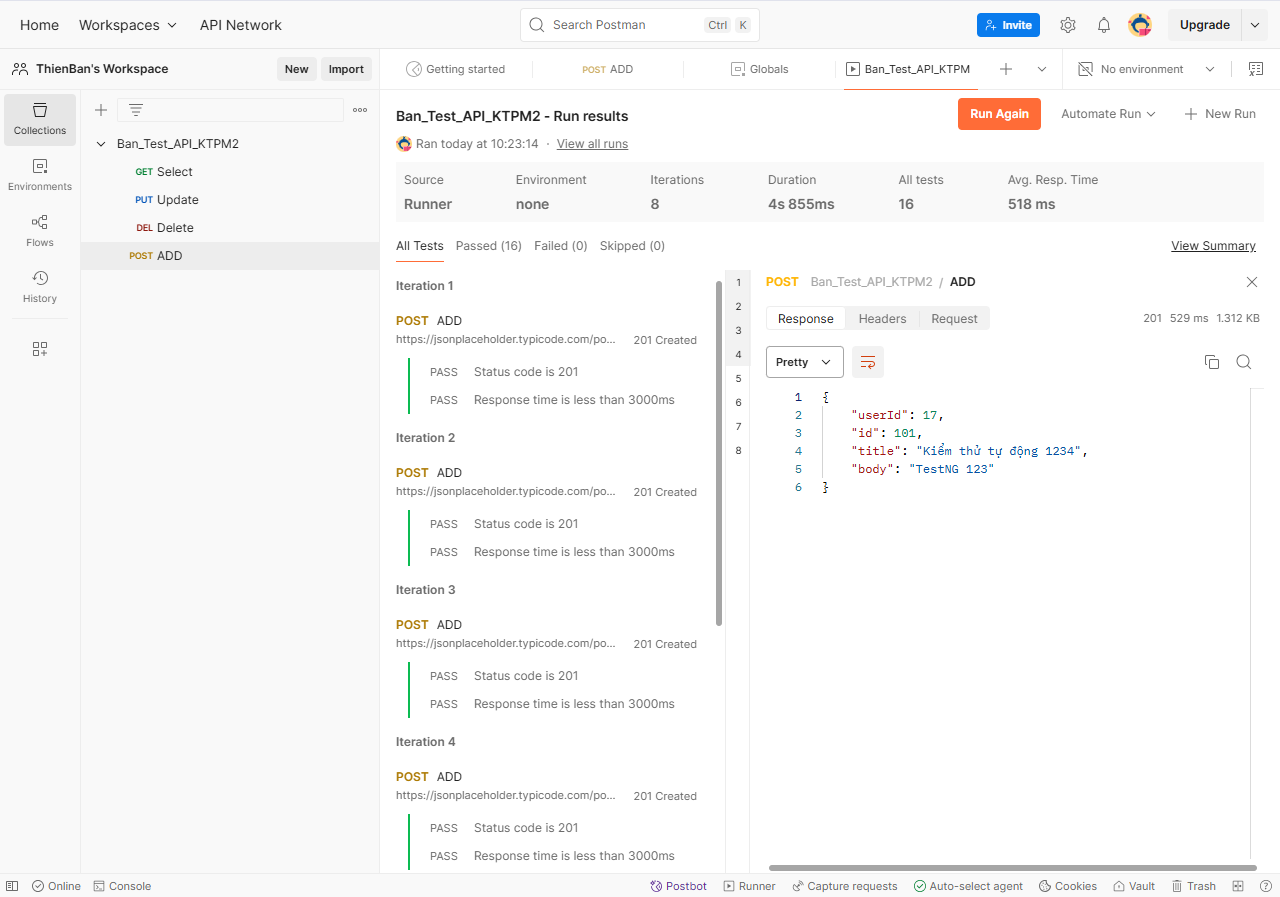
Hình 7.6.b: Minh họa bước làm.



Hình 7.6.c: Minh họa bước làm.



Hình 7.6.d: Minh họa bước làm.



Hình 7.6.e: Minh họa bước làm.

## Code kiểm thử tự động Selenium (lần 3).

### **Test Design.**

- Selenium WebDriver: Thư viện mã nguồn mở giúp điều khiển trình duyệt tự động.

- Ngôn ngữ sử dụng: JavaScript,…

- Quy trình cơ bản:

Khởi tạo WebDriver (ví dụ: ChromeDriver).

Mở trang web cần kiểm thử.

Tìm kiếm và tương tác với các phần tử (nhập liệu, nhấn nút, chọn tùy chọn,...).

Kiểm tra kết quả (so sánh nội dung, URL, trạng thái,...).

Đóng trình duyệt sau khi hoàn tất.

Ưu điểm:

- Tự động hóa kiểm thử nhanh chóng.

- Lặp lại dễ dàng nhiều lần.

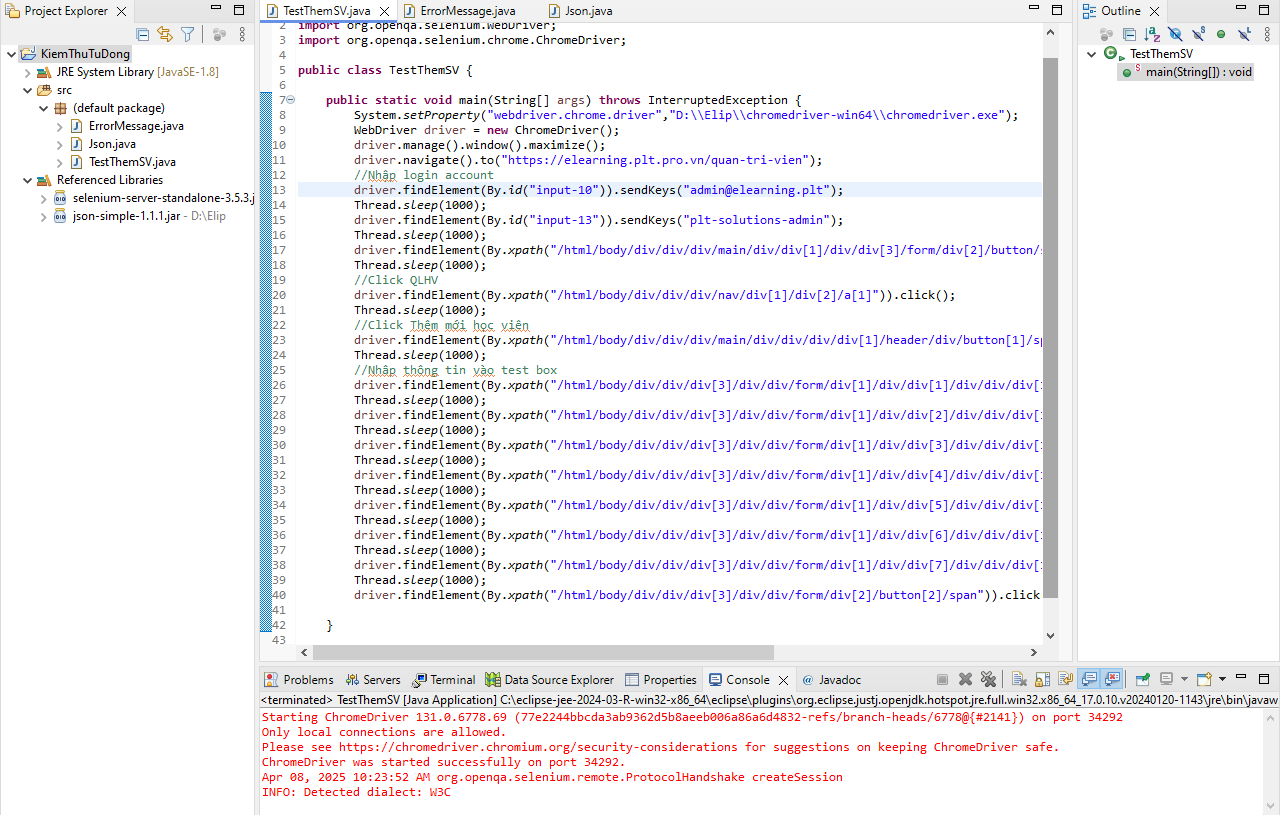
- Hỗ trợ nhiều trình duyệt.

Nhược điểm:

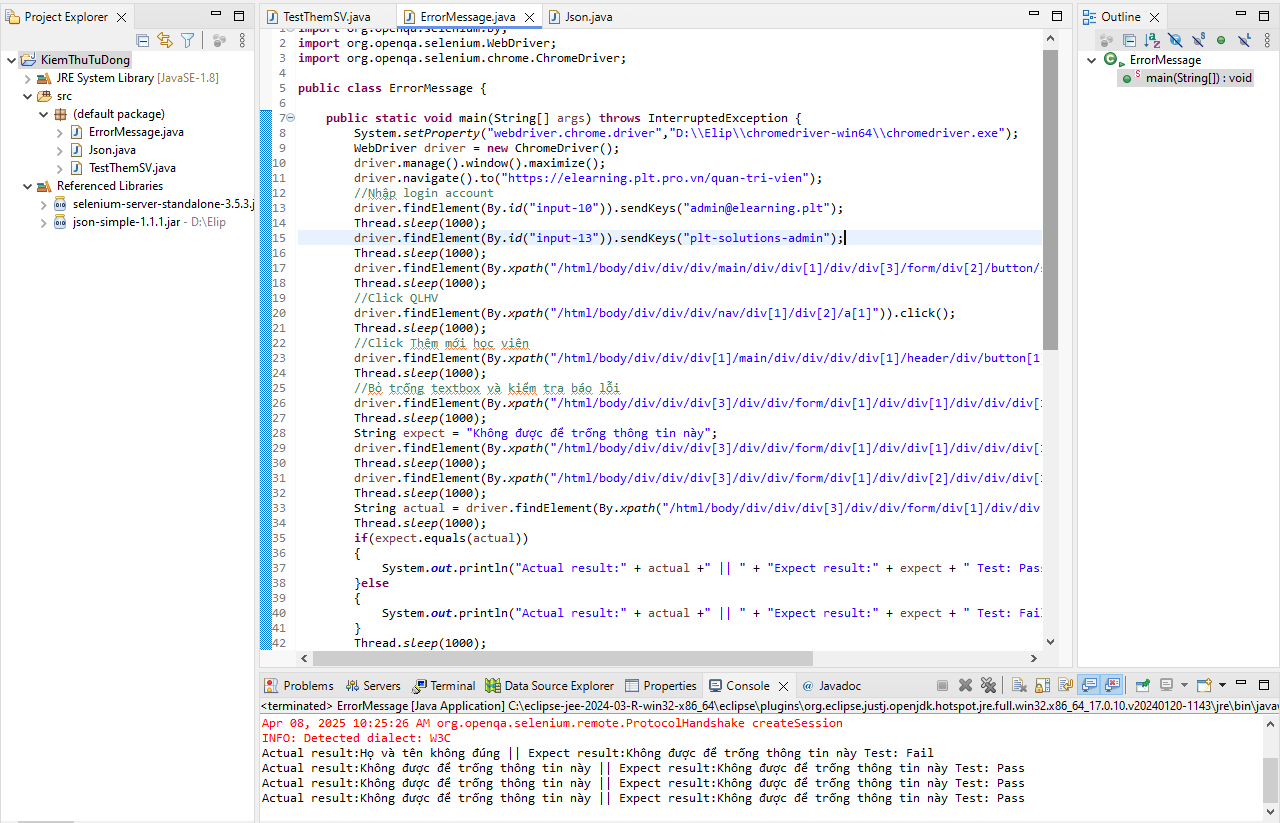
- Cần bảo trì mã test khi giao diện thay đổi.

- Chạy chậm hơn kiểm thử ở mức code (unit test).

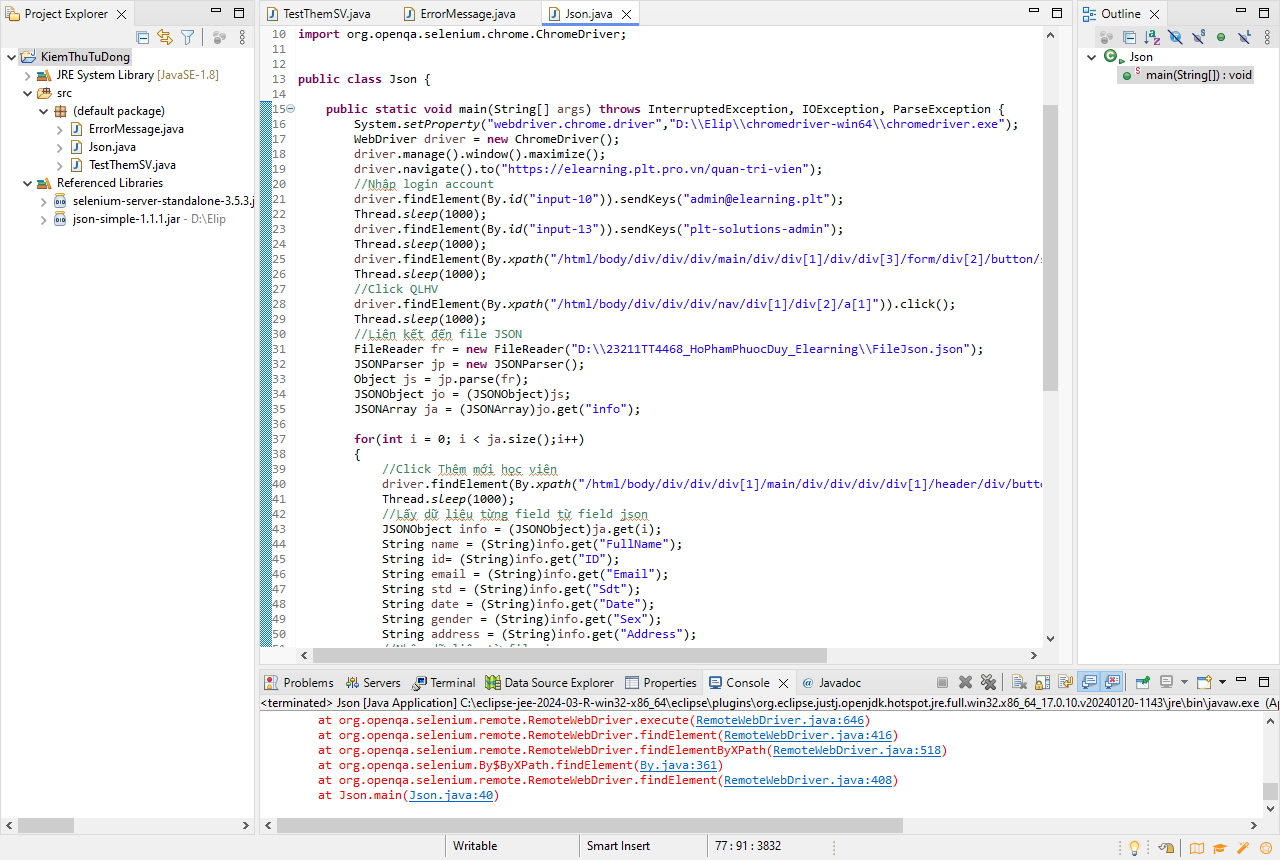
### **Hình ảnh minh họa Code Selenium.**

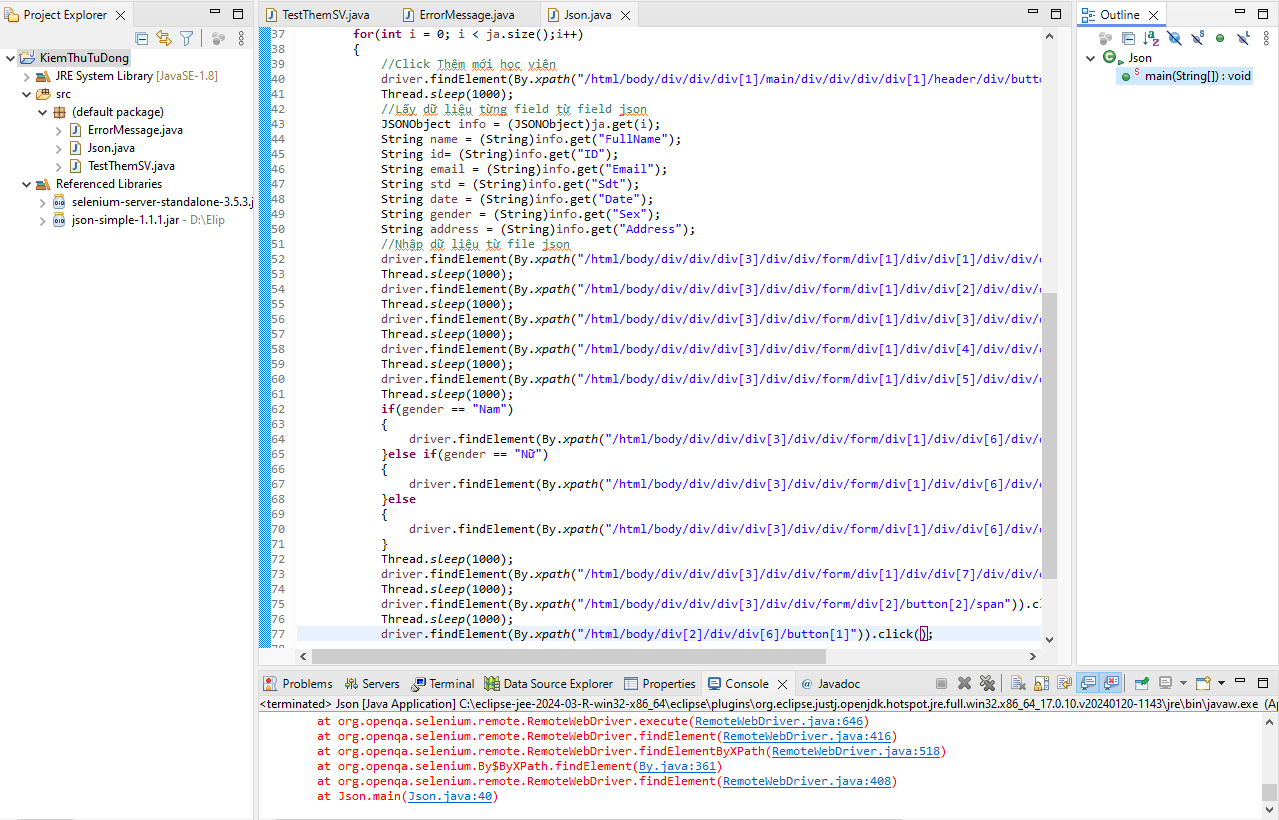


Hình 8.1: Mô phỏng code đăng nhập vào web Elearning và thực hiện chức năng thêm.



Hình 8.2: Mô phỏng code đăng nhập vào web Elearning và Test thông báo thêm sai.



Hình 8.3.a: Mô phỏng code đăng nhập vào web Elearning và Đọc và thêm sinh viên từ File JSON.

Hình 8.3.b: Mô phỏng code đăng nhập vào web Elearning và Đọc và thêm sinh viên từ File JSON.

## Viết Test Case cho tựa game Đấu Trường Chân Lý (TFT).

### Test Design.

-Cách chơi ngắn gọn:

-Chọn tướng từ cửa hàng theo vòng.

-Sắp xếp đội hình lên bàn cờ chiến đấu.

-Kết hợp tướng giống nhau để nâng cấp (3 tướng cùng loại → cấp 2).

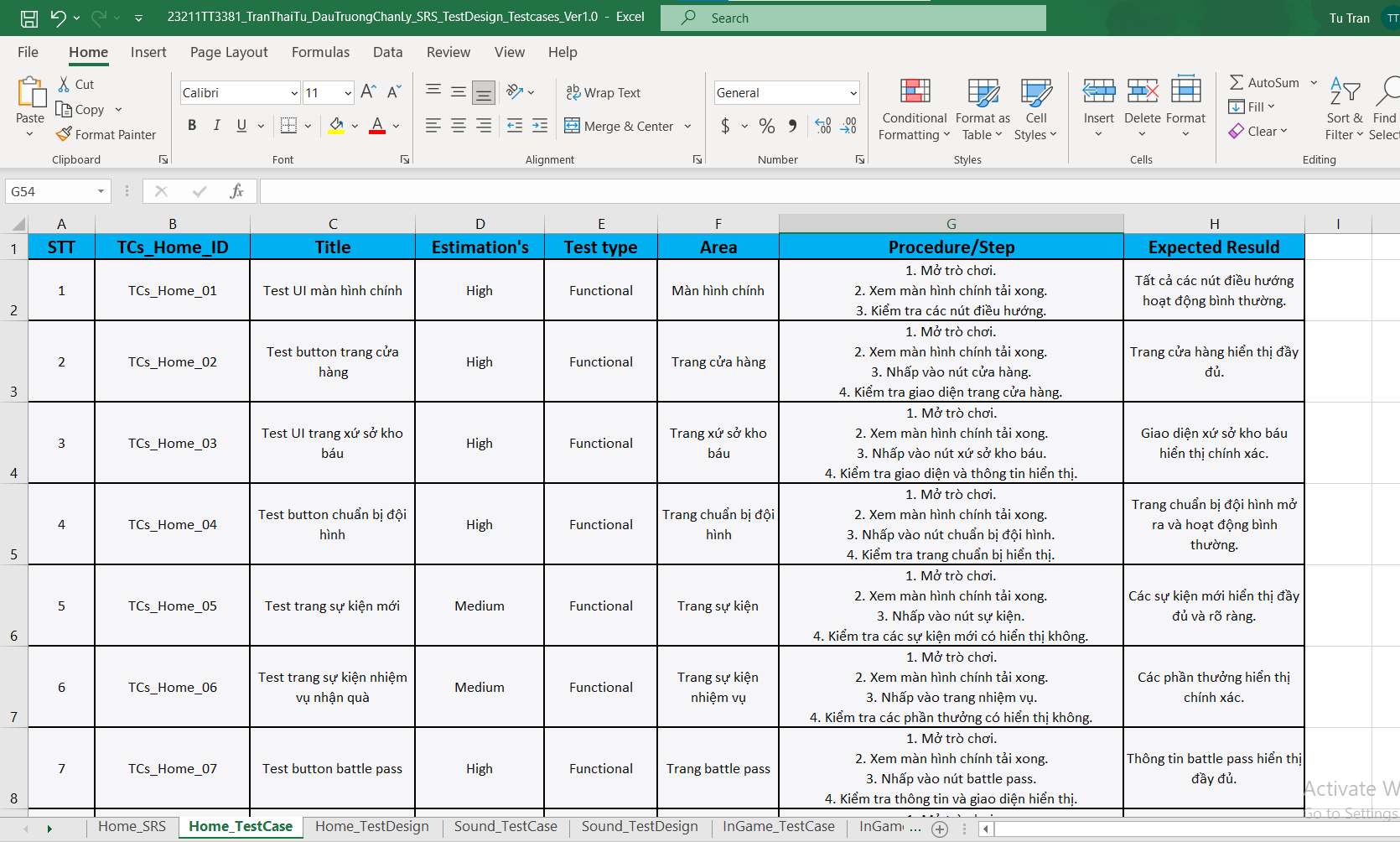
-Trang bị vật phẩm để tăng sức mạnh cho tướng.

-Chiến đấu tự động với người chơi khác qua các vòng.

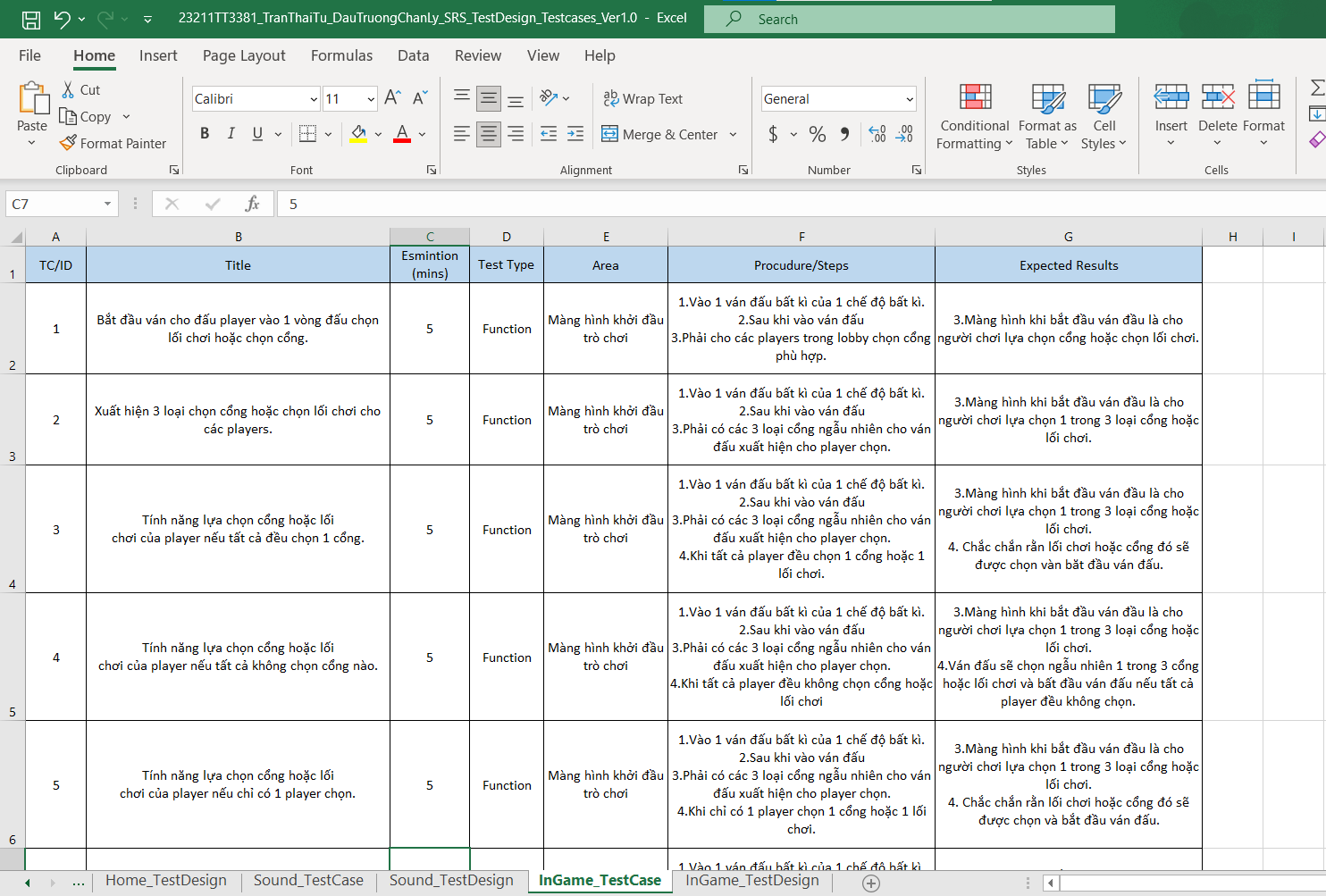
-Sống sót cuối cùng để chiến thắng.

=> Chiến thắng đòi hỏi chiến thuật, xoay đội hình hợp meta, và quản lý vàng, máu, nâng cấp hợp lý.

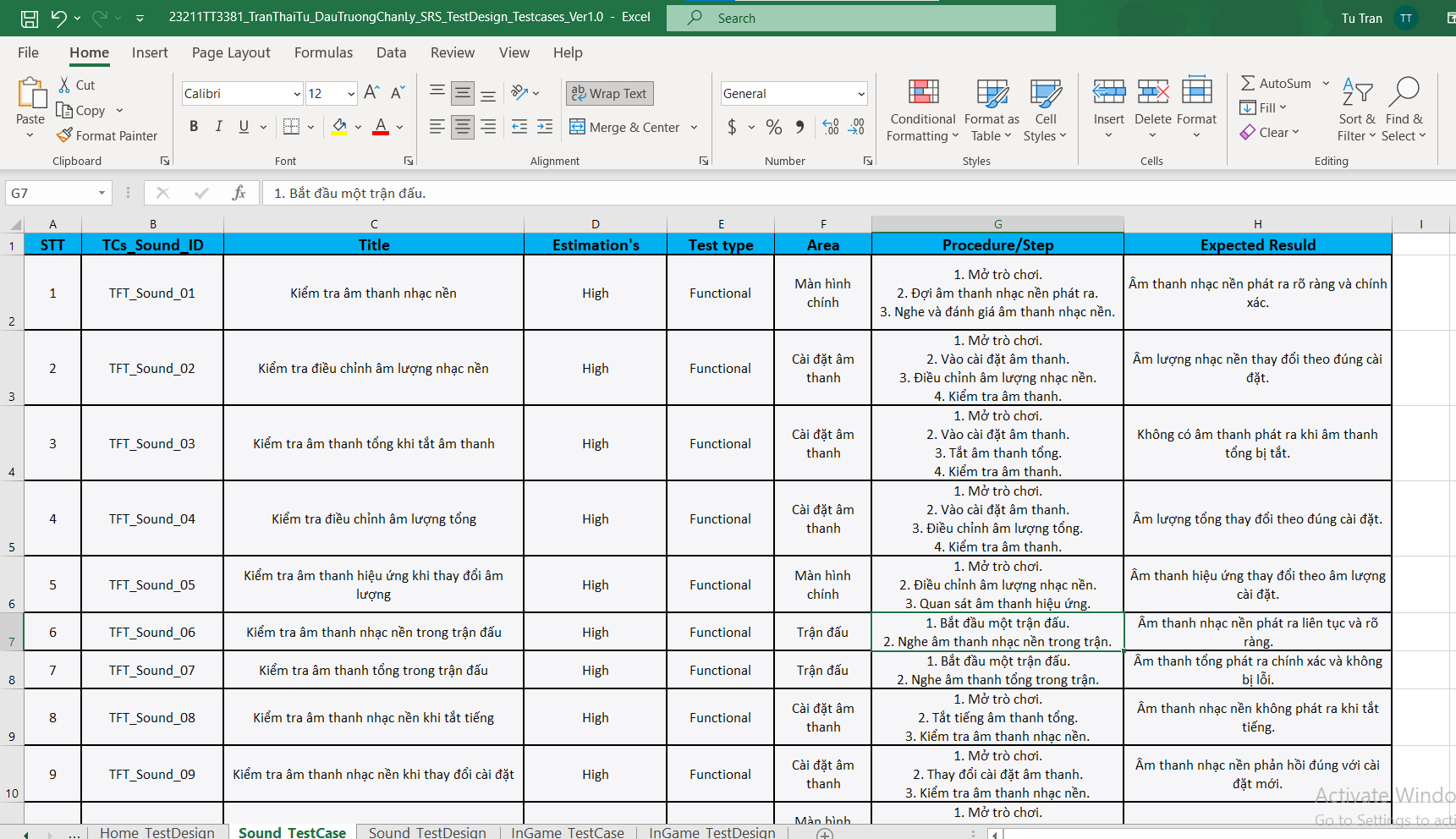
### Hình ảnh minh họa Test Case.



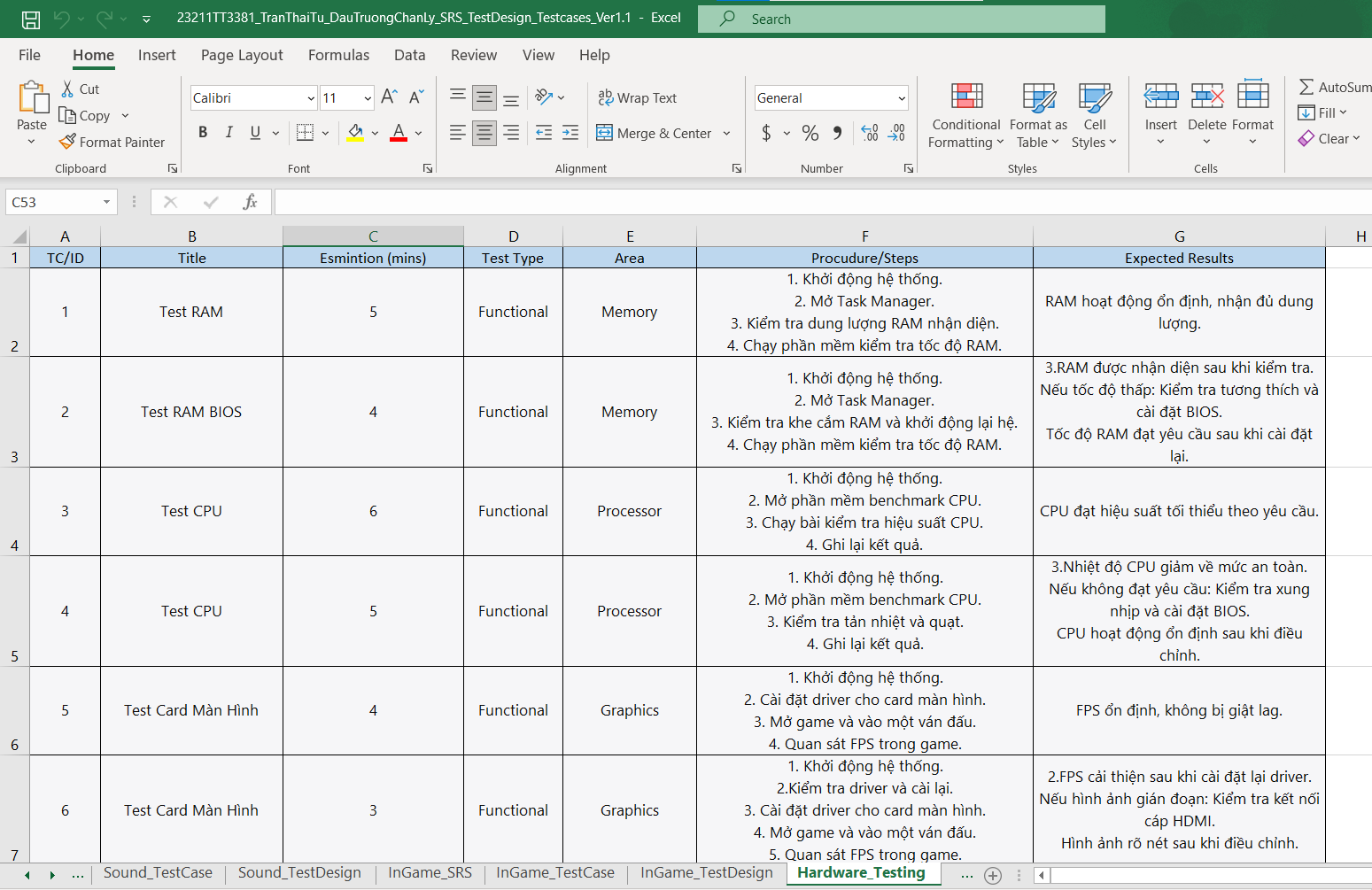
Hình 9.1: Minh họa Test Case đã viết để Test Home.



Hình 9.2: Minh họa Test Case đã viết để Test Game Play.



Hình 9.3: Minh họa Test Case đã viết để Test Sound.



Hình 9.4: Minh họa Test Case đã viết để Test các chức năng phần cứng.

## Code kiểm thử tự động KatalonStudio.

### **Test Design.**

-Katalon Studio: Công cụ kiểm thử tự động miễn phí, giao diện thân thiện, hỗ trợ cả web, mobile, API.

-Ngôn ngữ: Groovy (dựa trên Java), có thể dùng code hoặc ghi lại thao tác (record & playback).

-Cách dùng cơ bản:

+Tạo test case mới.

+Ghi thao tác hoặc viết script với lệnh kiểm thử.

+Thêm bước kiểm tra (assert/verify).

+Chạy test và xem kết quả (pass/fail).

+Báo cáo kết quả tự động.

Ưu điểm:

-Giao diện dễ dùng, không cần giỏi lập trình.

-Hỗ trợ đa nền tảng (web, mobile, API).

-Tích hợp CI/CD dễ dàng.

Nhược điểm:

-Tùy biến nâng cao cần kiến thức lập trình.

-Hiệu suất thấp khi chạy nhiều test cùng lúc.

### **Hình ảnh minh họa code Katalon.**



Hình 10.1: Minh họa Test cơ bản với record.

A screenshot of a computer

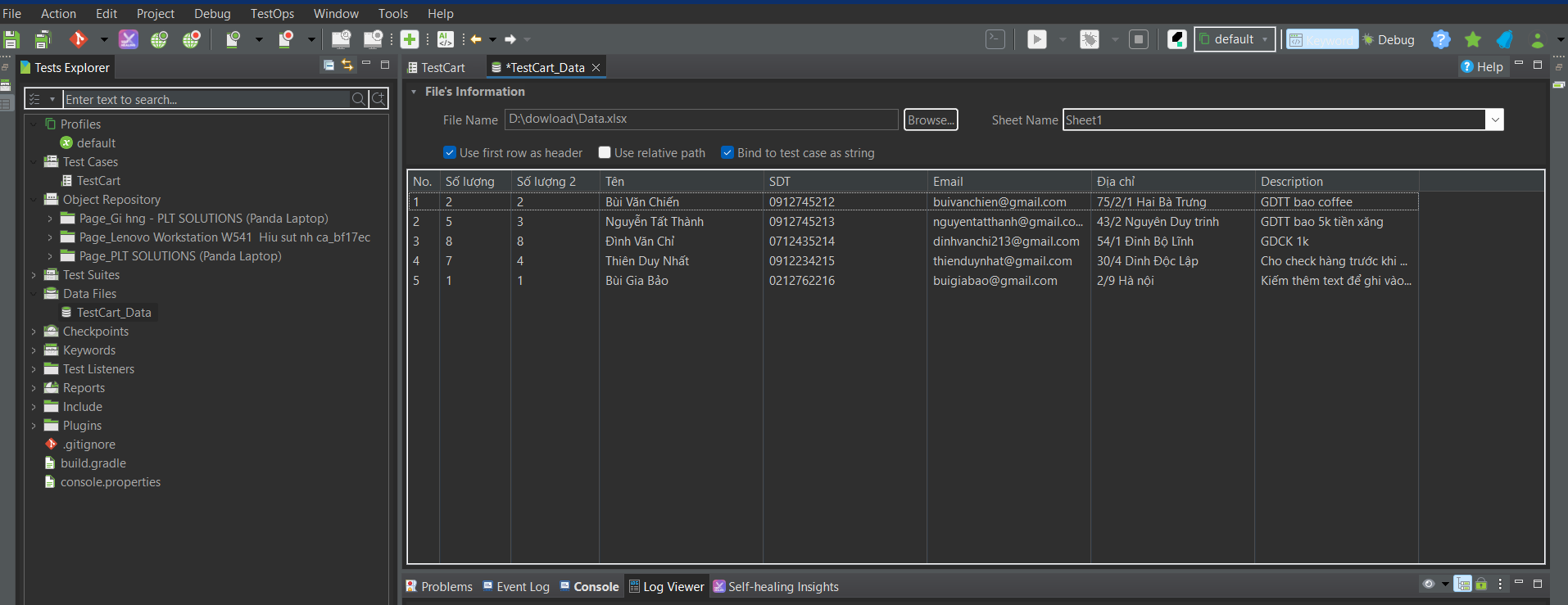
AI-generated content may be incorrect.

Hình 10.2.a: Minh họa Test Tạo biến .

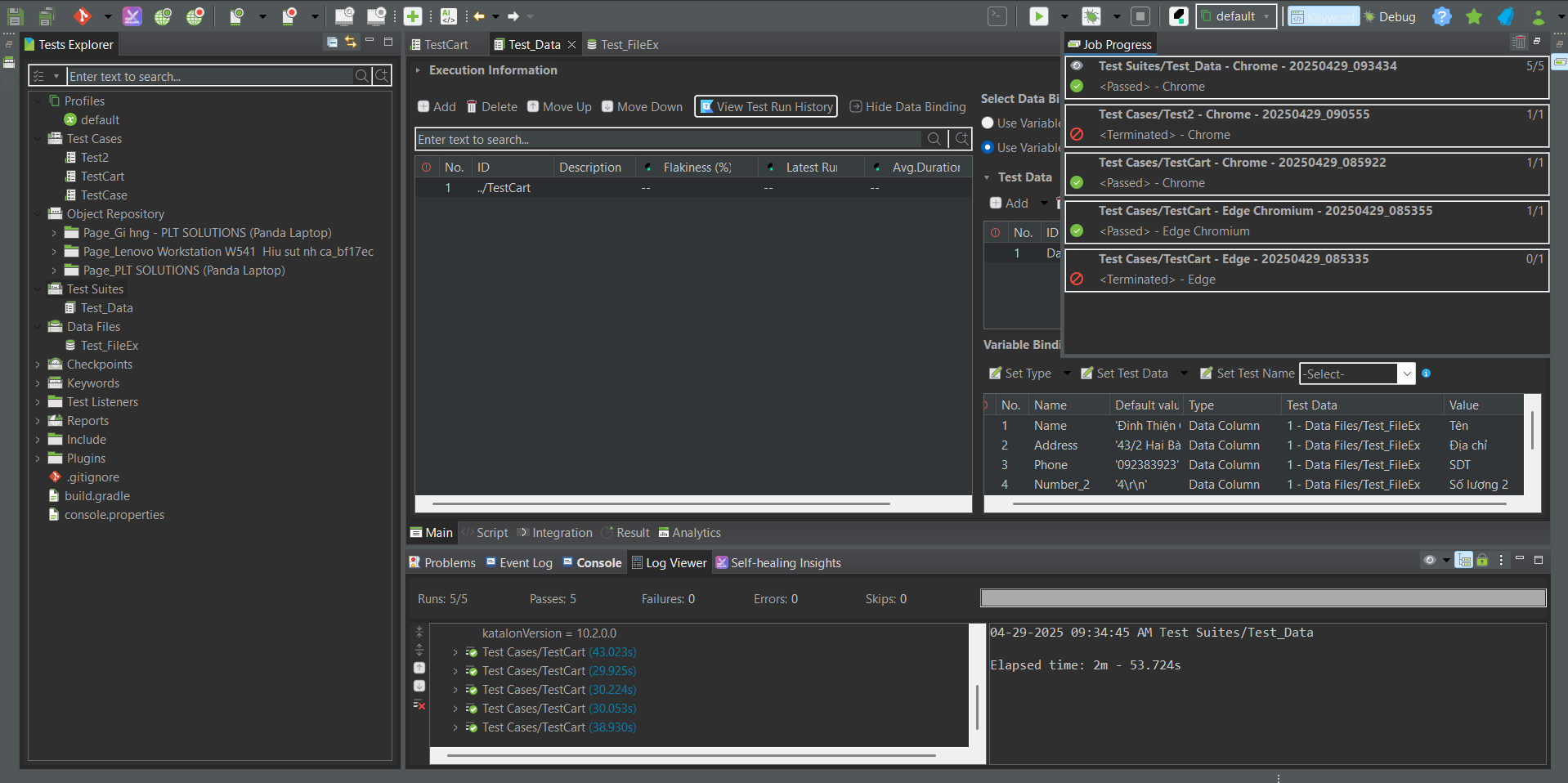
A computer screen shot of a program

AI-generated content may be incorrect.

Hình 10.2.b: Minh họa Test Tạo biến.



Hình 10.3.a: Minh họa Test Thêm File Data.



Hình 10.3.b: Minh họa Test Thêm File Data.

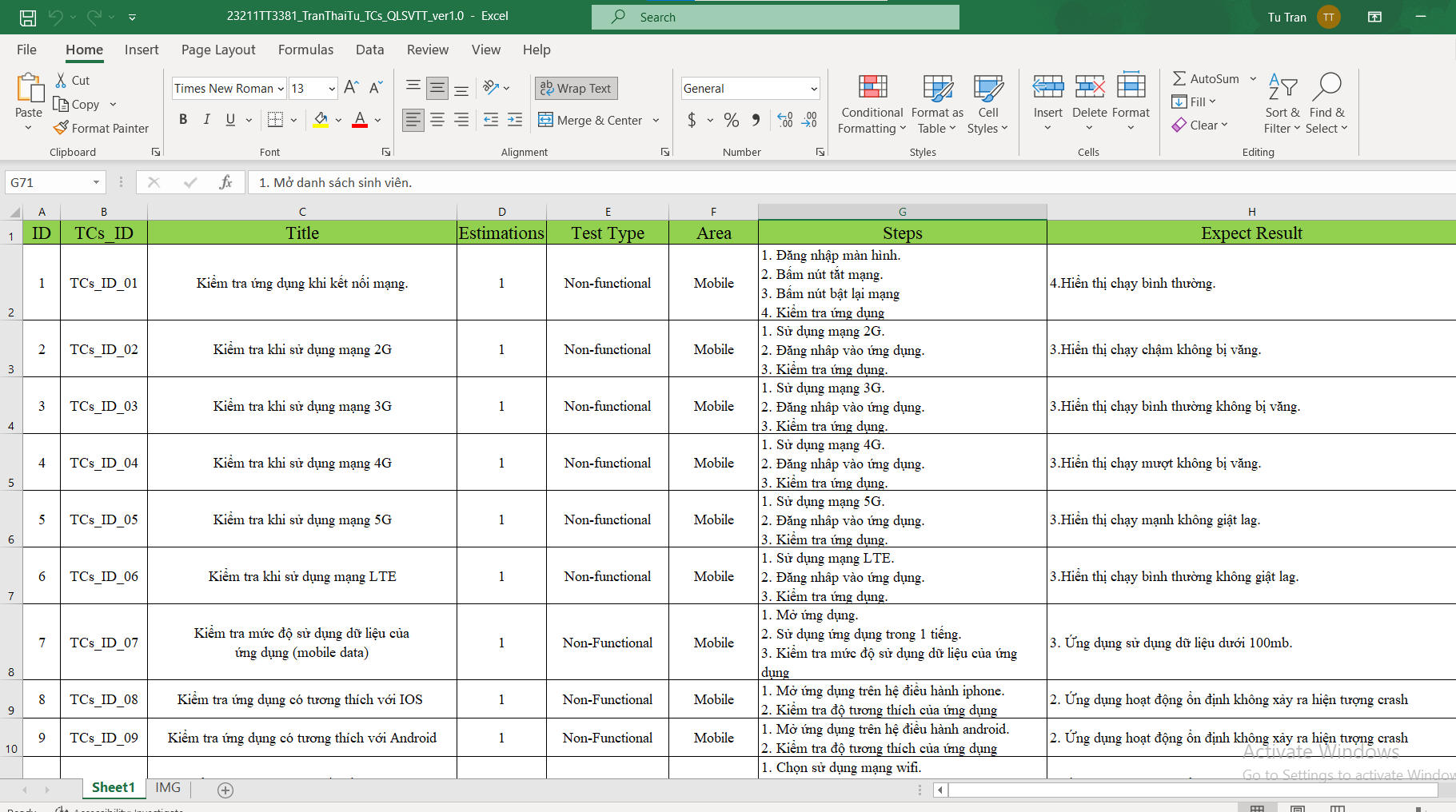
## Viết Test Case cho ứng dụng Quản Lý Thực Tập Viên.

### **Test Design.**

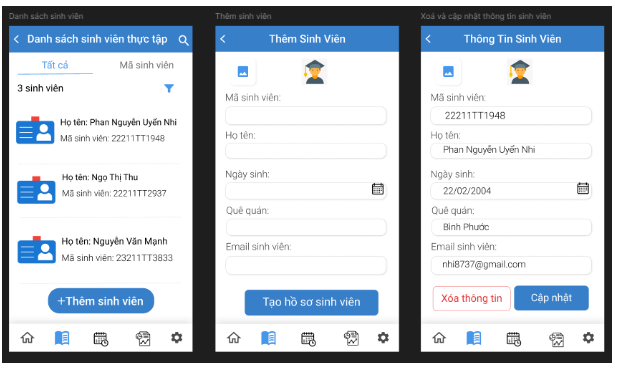
-Ứng dụng cho phép người dùng dưới quyền Admin quản lý thực tập sinh bằng cách (thêm, xóa, sửa, tìm kiếm,…).

-Ứng dụng cho phép người dùng dưới quyền thực tập sinh có thể xem thông tin của mình và có chức năng tìm kiếm. Củng như có những quyền cở bản của thực tập sinh trong ứng dụng cho phép (nhắn tin, điểm danh,…).

### **Hình ảnh viết Test Case và màn hình minh họa.**



Hình 11.1: Hình mình họa một số Test Case của ứng dụng Quản Lý Thực Tập Sinh.



Hình 11.2: Hình mình họa màn hình của ứng dụng Quản Lý Thực Tập Sinh.

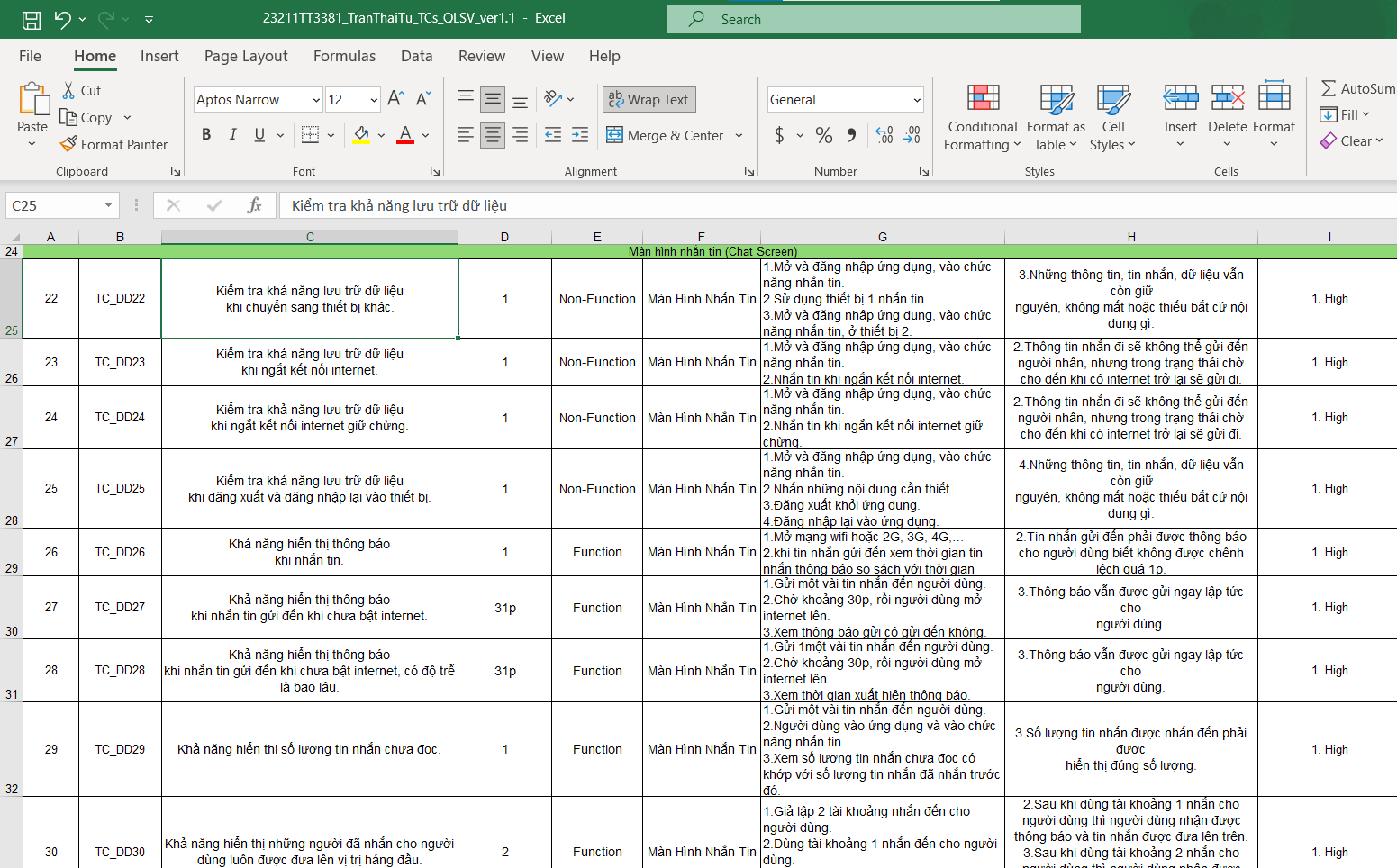
## Viết Test Case cho ứng dụng Quản Lý Thực Tập Viên (thêm chức năng)

### **Test Design.**

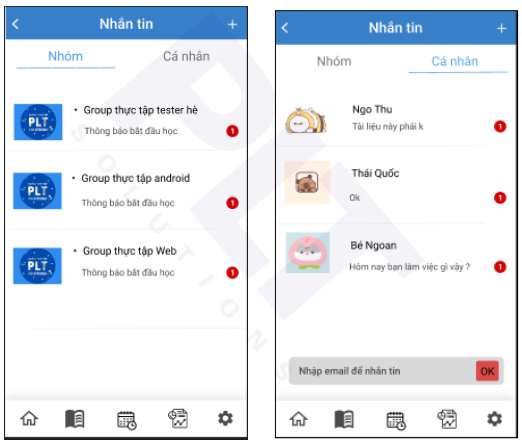
-Ứng dụng cho phép người dùng dưới quyền Admin quản lý thực tập sinh bằng cách (thêm, xóa, sửa, tìm kiếm,…).

-Ứng dụng cho phép người dùng dưới quyền thực tập sinh có thể xem thông tin của mình và có chức năng tìm kiếm. Củng như có những quyền cở bản của thực tập sinh trong ứng dụng cho phép (nhắn tin, điểm danh,…).

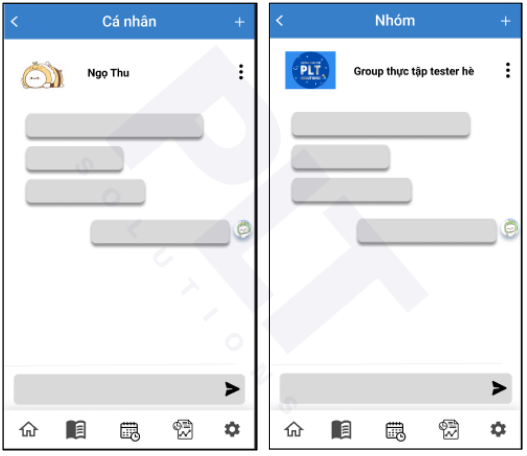
### **Hình ảnh viết Test Case và màn hình minh họa.**



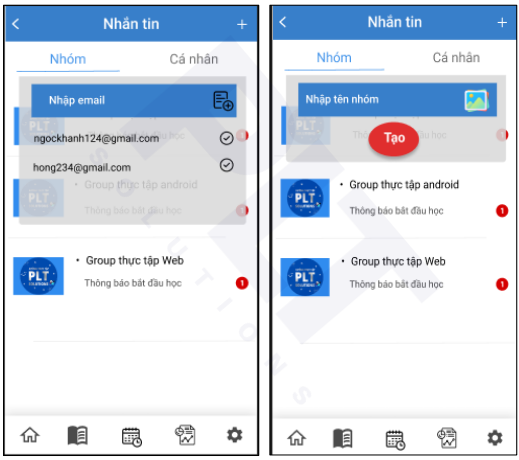
Hình 12.1: Minh họa một số Test Case của chức năng nhắn tin.



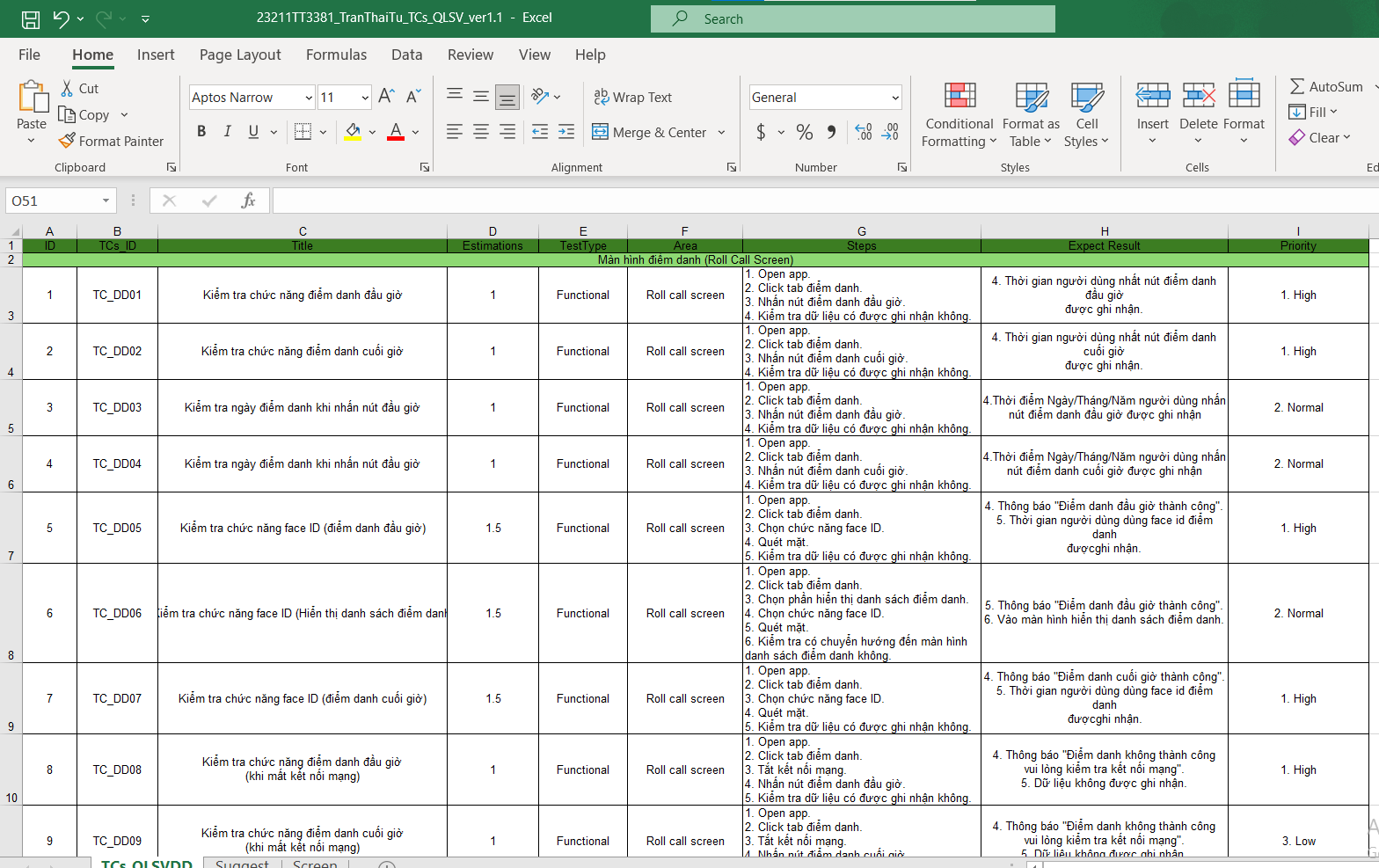
Hình 12.1.a: Minh họa chức năng của màn hình nhắn tin.



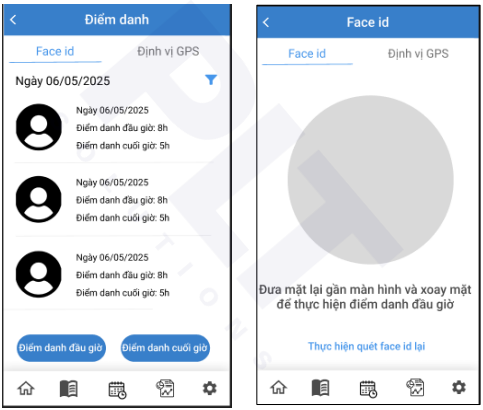
Hình 12.1.b: Minh họa chức năng của màn hình nhắn tin.



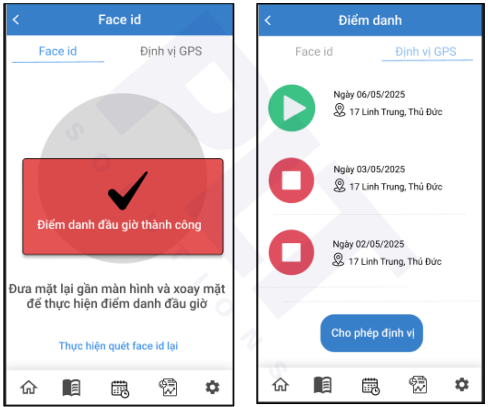
Hình 12.1.b: Minh họa chức năng của màn hình nhắn tin.



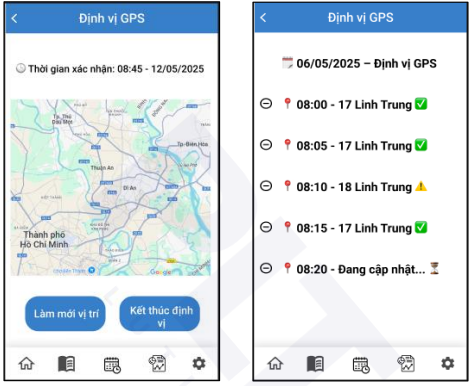
Hình 12.2: Minh họa một số Test Case của chức năng Điểm Danh và Định Vị GPS.



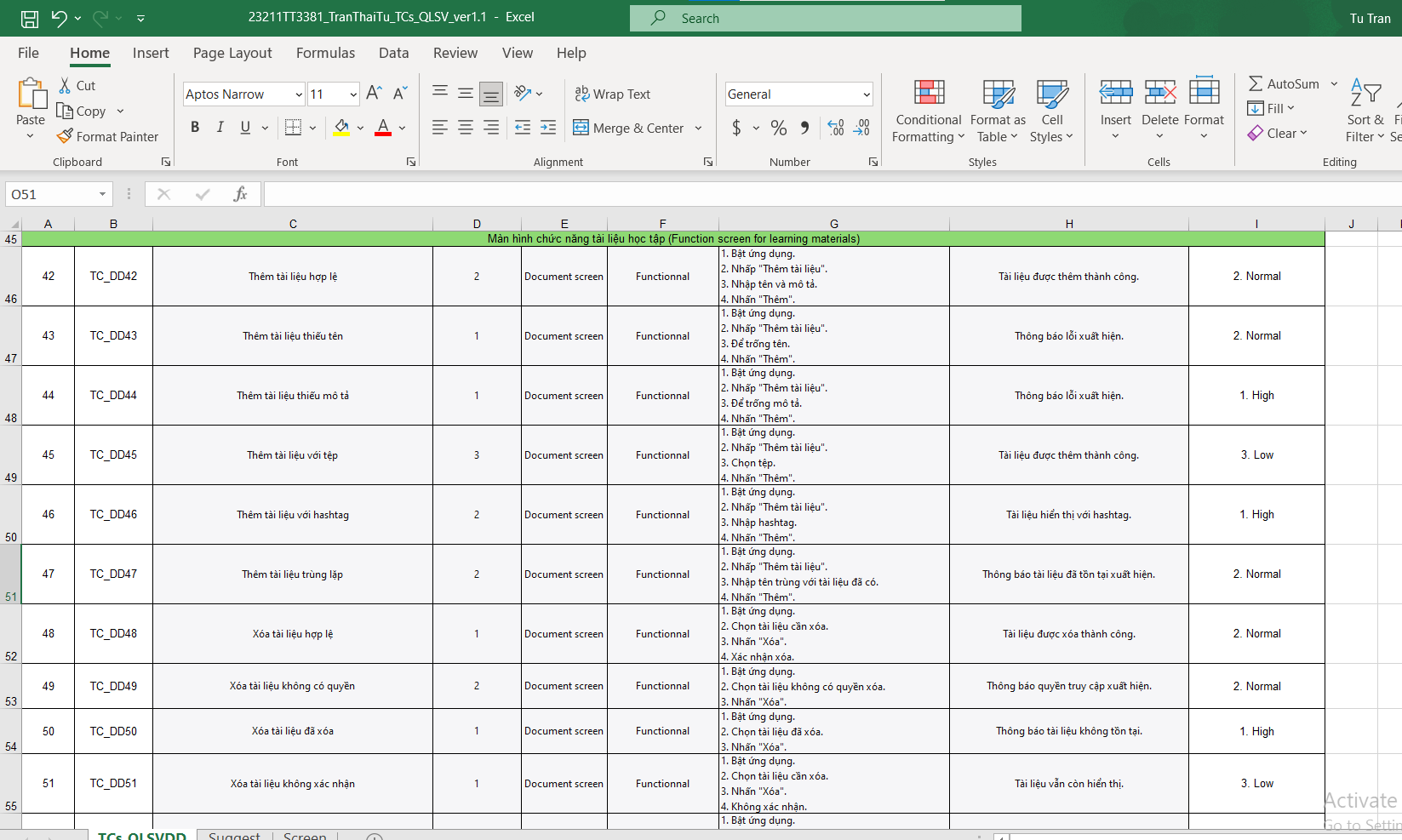
Hình 12.2.a: Minh họa chức năng của màn hình Điểm Danh và Định Vị GPS.



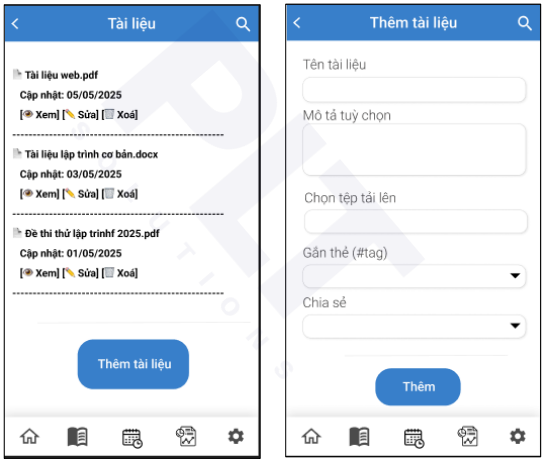
Hình 12.2.b: Minh họa chức năng của màn hình Điểm Danh và Định Vị GPS



Hình 12.2.c: Minh họa chức năng của màn hình Điểm Danh và Định Vị GPS



Hình 12.3: Minh họa một số Test Case của chức năng Tài Liệu Học Tập.



Hình 12.3.a: Minh họa chức năng của màn hình Tài Liệu Học Tập.



Hình 12.3.b: Minh họa chức năng của màn hình Tài Liệu Học Tập.

# Chương 3: KẾT LUẬN

## I. Kết luận

Từ kiểm thử phần mềm, mình rút ra rằng hiểu rõ 7 nguyên lý kiểm thử giúp mình kiểm thử có mục tiêu và hiệu quả hơn. Bên cạnh đó, việc áp dụng kiểm thử tự động bằng Selenium (cho giao diện) Postman (cho API), KatalonStudio (hỗ trợ cả web, mobile, API) không chỉ tiết kiệm thời gian mà còn giúp phát hiện lỗi sớm, đảm bảo phần mềm ổn định và chất lượng hơn trong quá trình phát triển.

## II. Hướng phát triển

Sau khi nắm vững các kiến thức nền tảng, mình có thể tiếp tục phát triển bằng cách:

* Tìm hiểu sâu hơn về CI/CD để tích hợp kiểm thử tự động vào quy trình DevOps (sử dụng Jenkins, GitHub Actions...).
* Nâng cao kỹ năng viết test case tự động, đặc biệt là sử dụng framework như TestNG (với Selenium) hoặc kết hợp Postman với Newman và dữ liệu đầu vào (data-driven testing).
* Học thêm về kiểm thử hiệu năng (Performance Testing) với các công cụ như JMeter hoặc k6.
* Học thêm về kiểm thử hiệu năng (Performance Testing) với các công cụ như Katalon Studio.
* Tìm hiểu về kiểm thử bảo mật (Security Testing) để nâng cao độ tin cậy cho ứng dụng.
* Tham gia các dự án thực tế hoặc open-source để rèn kỹ năng xử lý lỗi và viết test trong môi trường thật.

# LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến thầy Phan Gia Phước. Người đã tận tâm giảng dạy và truyền đạt kiến thức môn Kiểm Thử Phần Mềm trong suốt học kỳ vừa qua.

Nhờ thầy, em không chỉ hiểu rõ hơn về vai trò và tầm quan trọng của kiểm thử trong phát triển phần mềm, mà còn học được cách tư duy logic, cẩn thận và có hệ thống khi tiếp cận một vấn đề kỹ thuật. Những ví dụ thực tế và cách giảng dạy dễ hiểu, gần gũi của thầy đã giúp em cảm thấy môn học này không còn khô khan, mà ngược lại rất thực tế và thú vị.

Cảm ơn thầy vì sự nhiệt huyết, sự kiên nhẫn và cả những lời chia sẻ đầy tâm huyết trên lớp. Chúc thầy luôn mạnh khỏe, hạnh phúc và tiếp tục truyền cảm hứng cho nhiều thế hệ sinh viên phía trước.

# TÀI LIỆU THAM KHẢO

<https://www.google.com>

[https://www.youtube.com](https://www.google.com)

Software: LOL (TFT).