|  |
| --- |
| Trường đại học sư phạm Thành Phố Hồ Chí Minh  C:\Users\hidadsek\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CNTT.PNG |
|  |
| Kiểm thử tự động với Katalon Studio |
|  |
|  |
| Đồ án tiểu luận khoa công nghệ thông tin |

**Giáo viên hướng dẫn: NCS.ThS. Trần Sơn Hải**

**Sinh viên:Nguyễn Đông Thức K39.104.199**

|  |
| --- |
|  |

Mục lục

[1 Tổng quan về kiểm thử 3](#_Toc23011581)

[1.1 Kiểm thử phần mềm là gì? 3](#_Toc23011582)

[1.2 Các cấp kiểm thử 4](#_Toc23011583)

[1.2.1 Unit Test - Kiểm tra mức đơn vị 4](#_Toc23011584)

[1.2.2 Integration Test - Kiểm tra tích hợp 4](#_Toc23011585)

[1.2.3 System Test - Kiểm tra mức hệ thống 5](#_Toc23011586)

[1.2.4 Acceptance test - Kiểm tra chấp nhận 5](#_Toc23011587)

[1.3 Các loại kiểm thử thông dụng 5](#_Toc23011588)

[1.3.1 Kiểm thử hộp đen( Black box testing) 5](#_Toc23011589)

[1.3.2 Kiểm thử hộp trắng( White box testing) 5](#_Toc23011590)

[1.3.3 Kiểm thử chức năng ( Functional testing) 5](#_Toc23011591)

[1.3.4 Kiểm thử phi chức năng( Non Functional testing) 6](#_Toc23011592)

[1.3.5 Test cấu hình (Shakeout testing) 6](#_Toc23011593)

[1.3.6 Smoke testing 6](#_Toc23011594)

[1.3.7 Adhoc testing 6](#_Toc23011595)

[1.3.8 Monkey testing 6](#_Toc23011596)

[1.3.9 Kiểm thử hiệu suất (Performance testing) 6](#_Toc23011597)

[1.3.10 Kiểm thử hồi quy (Regression testing) 6](#_Toc23011598)

[1.3.11 Re-test 7](#_Toc23011599)

[1.4 Tổng quan về kiểm thử web 7](#_Toc23011600)

[2 Kiểm thử tự động 7](#_Toc23011601)

[2.1 Kiểm thử thủ công là gì? 7](#_Toc23011602)

[2.2 Kiểm thử tự đông là gì? 8](#_Toc23011603)

[2.3 So sánh giữa kiểm thử thủ công và tự động 9](#_Toc23011604)

[3 Kiểm thử tự động với Katalon Studio 9](#_Toc23011605)

[3.1 Giới thiệu về Katalon Studio 9](#_Toc23011606)

[3.2 Cài đặt Katalon 10](#_Toc23011607)

[3.3 Tổng quan các tính năng của Katalon 11](#_Toc23011608)

[3.3.1 Tạo project 11](#_Toc23011609)

[3.3.2 Tạo test case 12](#_Toc23011610)

[3.3.3 Tạo test data 21](#_Toc23011611)

[4 Áp dụng Katalo kiểm thử ứng dụng web Our Better World 27](#_Toc23011612)

[4.1 Test Approach 27](#_Toc23011613)

[4.2 Hình ảnh demo 27](#_Toc23011614)

[5 Kinh nghiệm và bài học 30](#_Toc23011615)

**Lời mở đầu**

Ngày nay, công nghệ thông tin là một phần quan trọng không thể thiếu trong đời sống, đặc biệt là các ngành nghề cần tương tác cao như quảng cáo, mua bán… một trong những ví dụ rõ nét nhất là việc sử dụng rộng rãi các trang web mạng. Các trang web mạng cung cấp hầu hết tất cả dịch vụ cần có từ đọc báo, viết blog, chat tán gẫu, giải trí chơi game đến quảng cáo, mua bán qua mạng. Việc sử dụng rông rãi các trang web cùng với lượng data khổng lồ yêu cầu các công ty hay kiểm thử viên (tester) bảo đảm tính năng cũng như sự ổn định cho các trang web thường xuyên. Công việc trên thường rất nhiều và nhàm chán cho những người kiểm thử cũng như admin từ đó dẫn đến nhu cầu cần phải tự đông hóa các quá trình kiểm thử.

Sau quá trình học tập, thực tập cũng như nghiên cứu về phần kiểm thử cùng với sự hỗ trợ từ giáo viên hướng dẫn và mọi người, em đã hoàn thành xong đồ án “Kiểm thử tự động trang web Our Better World với Katalon”.

Đề tài bao gồm:

* Chương 1: Tổng quan về kiểm thử
* Chương 2: Kiểm thử tự động
* Chương 3: Kiểm thử tự động với Katalon Studio
* Chương 4: Áp dụng Katalo kiểm thử ứng dụng web Our Better World
* Chương 5: Kinh nghiệm và bài học
* Kết Luận
* Tài Liệu tham khảo

Quá trình làm đồ án còn khá thiếu sót do kinh nghiệm thực tế của em còn hạn chế, do đó em mong nhận được sự góp ý từ thầy cô và các bạn để đồ án này hoàn thiện hơn.

Em xin chân thành cảm ơn sự giúp đỡ của thầy cô.

# Tổng quan về kiểm thử

## Kiểm thử phần mềm là gì?

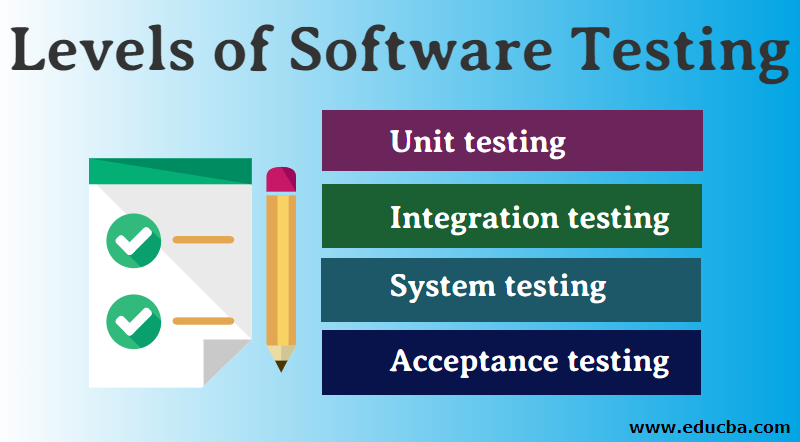
Kiểm thử hay còn gọi là testing, là quá trình đánh giá một hệ thống hay các thành phần của nó với mục đích tìm xem liệu hệ thống có đáp ứng các yêu cầu được đã được chỉ định hay không. Nói một cách đơn giản, kiểm thử được thực hiện trên một hệ thống để xác định bất kỳ lỗ hổng, các lỗi hoặc các yêu cầu đang bị thiếu hay trái ngược với các yêu cầu thực tế đã được đề ra.

Theo tiêu chuẩn ANSI / IEEE 1059, kiểm thử có thể được định nghĩa là quá trình phân tích các thành phần của phần mềm để phát hiện sự khác biệt giữa những điều kiện của phần mềm đang tồn tại thực tế và những điều kiện được yêu cầu (đó là defects/ errors/ bugs) và từ đó có thể đánh giá được chất lượng của chất lượng của phần mềm.

Người kiểm thử có thể là lập trình viên (Developer), kiểm thử viên (Tester), quản lý dự án (Project Manager) hay người dùng (End-user).



## Các cấp kiểm thử



### Unit Test - Kiểm tra mức đơn vị

Unit Test là hoạt động kiểm thử nhỏ nhất. Kiểm thử thực hiện trên các hàm hay thành phần riêng lẻ. Ví dụ: Các hàm, lớp, thủ tục, phương thức.

Yêu cầu: cần hiểu biết về thiết kế chương trình và code.

Unit Test được thực hiện bởi Developer.

Mục tiêu của việc kiểm tra unit test là cô lập từng thành phần của chương trình và chứng minh các bộ phận riêng lẻ chính xác về các yêu cầu chức năng

### Integration Test - Kiểm tra tích hợp

Integration Test nhằm phát hiện lỗi giao tiếp xảy ra giữa các thành phần cũng như lỗi của bản thân từng thành phần (nếu có).

Trong Unit Test, lập trình viên cố gắng phát hiện lỗi liên quan đến chức năng và cấu trúc nội tại của Unit. Có một số phép kiểm tra đơn giản trên giao tiếp giữa Unit với các thành phần liên quan khác, tuy nhiên mọi giao tiếp liên quan đến Unit thật sự được kiểm tra đầy đủ khi các Unit tích hợp với nhau trong khi thực hiện Integration Test.

Integration Test được thực hiện bởi Developer hay Tester.

Integration Test có 2 mục tiêu chính:

* Phát hiện lỗi giao tiếp xảy ra giữa các Unit.
* Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ (subsystem) và cuối cùng là nguyên hệ thống hoàn chỉnh (system) chuẩn bị cho kiểm tra ở mức hệ thống (System Test).

### System Test - Kiểm tra mức hệ thống

System Test là một mức của tiến trình kiểm thử phần mềm khi các module và tích hợp các module đã được test.

Điểm khác nhau then chốt giữa Integration Test và System Test là System Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau. Thông thường ta phải thực hiện Unit Test và Integration Test để bảo đảm mọi Unit và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện System Test.

System Test được thực hiện bởi Tester trong dự án hoặc Tester độc lập (Independent Tester).

Mục tiêu của kiểm thử hệ thống là để đánh giá phần mềm có tuân thủ theo các yêu cầu đã đưa ra không.

### Acceptance test - Kiểm tra chấp nhận

Acceptance test là một cấp độ trong tiến trình kiểm thử phần mềm nhằm kiểm thử hệ thống về khả năng chấp nhận được.

Gồm 2 loại kiểm thử là:

* Alpha Test, người dùng kiểm thử phần mềm ngay tại nơi phát triển phần mềm, lập trình viên sẽ ghi nhận các lỗi hoặc phản hồi, và lên kế hoạch sửa chữa.
* Beta Test, phần mềm sẽ được gửi tới cho người dùng để kiểm thử ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi ngược lại cho lập trình viên để sửa chữa

Kiểm thử chấp nhận được end-user thực hiện hoặc ủy quyền cho một team Independent Tester.

Mục tiêu của kiểm thử này là để đánh giá sự tuân thủ của hệ thống với các yêu cầu nghiệp vụ và thẩm định xem đã có thể chấp nhận để bàn giao chưa.

## Các loại kiểm thử thông dụng

### Kiểm thử hộp đen( Black box testing)

Black box testing là 1 phương pháp kiểm thử mà tester sẽ chỉ xem xét đến đầu vào và đầu ra của chương trình mà không quan tâm code bên trong được viết ra sao. Tester thực hiện kiểm thử dựa hoàn toàn vào đặc tả yêu cầu . Mục đích của kiểm thử hộp đen là tìm ra các lỗi ở giao diện , chức năng của phần mềm. Các trường hợp kiểm thử sẽ được xây dựng xung quanh đó.

### Kiểm thử hộp trắng( White box testing)

White box testing là phương pháp kiểm thử mà cấu trúc thuật toán của chương trình được đưa vào xem xét. Các trường hợp kiểm thử được thiết kế dựa vào cấu trúc mã hoặc cách làm việc của chương trình. Người kiểm thử truy cập vào mã nguồn của chương trình để kiểm tra nó.

### Kiểm thử chức năng ( Functional testing)

Functional testing là một loại kiểm thử hộp đen (black box) và các trường hợp kiểm thử của nó được dựa trên đặc tả của ứng dụng phần mềm/thành phần đang test. Các chức năng được test bằng cách nhập vào các giá trị nhập và kiểm tra kết quả đầu ra, và ít quan tâm đến cấu trúc bên trong của ứng dụng (không giống như kiểm thử hộp trắng - white-box testing). Có thể hiểu 1 cách đơn giản, kiểm thử chức năng là xác nhận tất cả các chức năng của hệ thống. Nó đánh giá ứng dụng và xác nhận liệu ứng dụng có đang hoạt động theo yêu cầu hay không.

### Kiểm thử phi chức năng( Non Functional testing)

Non Functional testing tập trung vào các khía cạnh phi chức năng của ứng dụng. Non Functional Testing thường bao gồm:

* Kiểm thử chịu tải
* Kiểm thử bảo mật
* Kiểm tra tính tương thích trên từng môi trường,...

### Test cấu hình (Shakeout testing)

Shakeout testing cơ bản là kiểu kiểm thử về khả năng của hệ thống mạng, kết nối dữ liệu và sự tương tác của các module. Thông thường thì kiểu test này là do nhóm quản lý cấu hình chuẩn bị thiết lập các môi trường test thực sự. Họ cũng kiểm tra xem liệu các thành phần chính của phần mềm có hoạt động bất thường không. Kiểu kiểm thử này thực hiện trước khi tiến hành thực hiện trong môi trường test.

### Smoke testing

Smoke Testing là 1 quá trình để kiểm tra sự ổn định hay tính năng quan trọng của 1 bản build . Nó là 1 bài test hồi quy nhỏ đơn giản và nhanh của các chức năng chính, cho thấy sản phẩm đã sẵn sàng cho việc test hay chưa.

### Adhoc testing

Thuật ngữ Adhoc testing là phương pháp kiểm thử dạng Black box test mà không theo tài liệu yêu cầu (requirement), kế hoạch test (test plan), kịch bản kiểm thử mà thường dựa vào kinh nghiệm và logic của tester. Mục đích loại test này sẽ nắm nhiều các vấn đề về UX cũng như tăng tính tương tác của hệ thống và người sử dụng.

### Monkey testing

Monkey testing được định nghĩa rất ngắn gọn: là một phương pháp kiểm thử với đầu vào và thao tác ngẫu nhiên, không theo testcase hay một chiến lược test nào. Cách Test này khác với Adhoc test là bạn sẽ không dựa vào logic mà thường sẽ nhập các data hay các thao tác mang tính ngẫu nhiên cao. Mục đích của loại test này sẽ đảm bảo hệ thống có thể hoạt đông hoặc trả về các response phù hợp khi nhân được các data hay các thao tác lạ. VD: bạn đang mua hàng online nhưng lại tắt browser khi đang trong quá trình loading và thanh toán món hàng.

### Kiểm thử hiệu suất (Performance testing)

Trong loại kiểm thử này, ứng dụng được test dựa vào sức nặng như sự phức tạp của giá trị, độ dài của đầu vào, độ dài của các câu truy vấn…Loại kiểm thử này kiểm tra bớt phần tải (stress/load) của ứng dụng có thể được chắc chắn hơn.

### Kiểm thử hồi quy (Regression testing)

Regression testing là kiểm thử lại 1 chức năng đã được làm xong, đã được test xong rồi, đã hết lỗi nhưng do có sự sửa đổi hay thêm 1 chức năng khác mà lại có ảnh hưởng đến nó.

### Re-test

Re-test là thực hiện test để đóng bug/ defect / lỗi sau khi developer đã sửa xong.

## Tổng quan về kiểm thử web

Kiểm thử website là tên gọi được đặt cho một quá trình kiểm thử phần mềm tập trung vào việc kiểm tra các ứng dụng web. Ứng dụng web cần được kiểm tra hoàn toàn trước khi đi vào hoạt động, điều này có thể giúp giải quyết các vấn đề trong ứng dụng web trước khi tiếp xúc với người dùng như các vấn đề về chức năng, bảo mật, các vấn đề dịch vụ web, các vấn đề tích hợp và khả năng xử lý lưu lượng truy cập, trong quá trình kiểm thử website, cần cố gắng phát hiện ra lỗi có thể xảy ra trong hệ thống nhằm giải quyết kịp thời.

Trong quá trình kiểm thử web, chúng ta thường sẽ:

* Kiểm thử liên kết (link): đảm bảo các link vẫn hoạt động bình thường.
* Kiểm thử web form: đảm bảo save và load data.
* Kiểm thử cookie: đảm bảo các session và data trong cookie.
* Kiểm thử HTML và CSS: đảm bảo cú pháp, màu sắc..
* Kiểm tra nội dung website.
* Kiểm thử sự tương thích với các trình duyệt (browser), hệ điều hành, trên điện thoại.
* Kiểm tra cơ sở dữ liệu.
* Kiểm thử hiệu năng website.
* Kiểm thử bảo mật website.

# Kiểm thử tự động

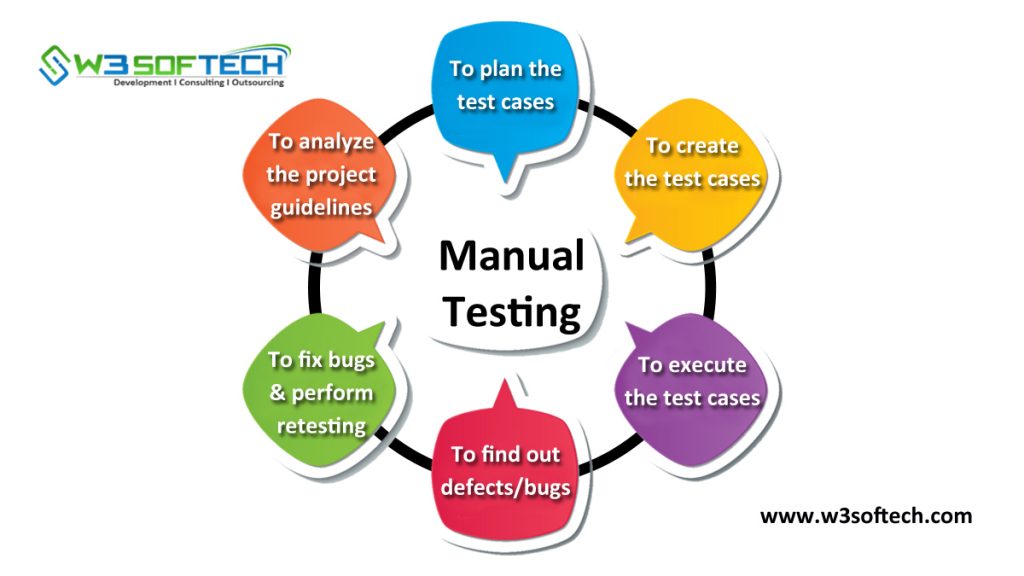
Kiểm thử thường được chia làm 2 loại dựa trên cách thực hiện kiểm thử : Kiểm thử thủ công (Manual Testing) và kiểm thử tự động (Automation testing).

## Kiểm thử thủ công là gì?

Kiểm thử thủ công là kiểm thử viên làm mọi công việc hoàn toàn bằng tay, từ viết test case đến thực hiện kiểm thử, mọi thao tác như nhập điều kiện đầu vào, thực hiện một số sự kiện khác như click nút và quan sát kết quả thực tế, sau đó so sánh kết quả thực tế với kết quả mong muốn trong test case, điền kết quả test…

Kiểm thử thủ công là cần thiết trong quá trình phát triển phần mềm do phần mềm vẫn chưa ổn định và luôn trong tình trạng có thể thay đổi bất kỳ lúc nào.

Hiện nay, phần lớn các tổ chức, các công ty phần mềm, hoặc các nhóm làm phần mềm đều thực hiện kiểm thử thủ công là chủ yếu.



## Kiểm thử tự đông là gì?

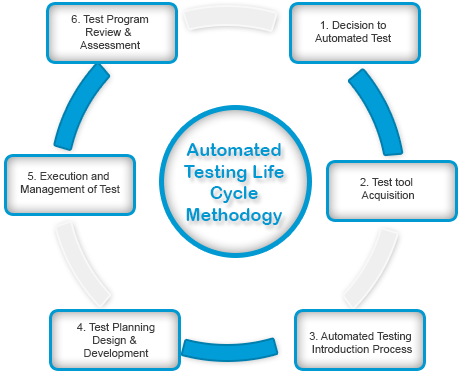
Kiểm thử tự động: Kiểm thử phần mềm tự động là thực hiện kiểm thử phần mềm bằng một chương trình đặc biệt hay còn gọi là Automation Testing Tool với rất ít hoặc không có sự tương tác của con người

Kiểm thử tự động thường được áp dụng sau khi chương trình đã tương đối hoàn thiện và ổn định.

Kiểm thử tự động đóng một vai trò quan trọng góp phần nâng cao năng suất kiểm thử, giảm thiểu lỗi cũng như sự nhàm chán với việc kiểm thử bằng tay trong một thời gian dài hoặc lặp đi lặp lại.

Một số phần mềm kiểm thử tự động nổi tiếng hiện nay như:

* Quick Test Profressional - (HP)
* Selenium
* Test Architect - (LogiGear)
* Ranorex
* Visual Studio CodedUI Testing
* TestComplete (SmartBear)
* SOAPUI - Web Services Testing (SmartBear)



## So sánh giữa kiểm thử thủ công và tự động

|  |  |  |  |
| --- | --- | --- | --- |
| **Kiểm thử** | **Điểm mạnh** | **Điểm yếu** | **Kết luận** |
| Thủ công | - Cho phép tester thực hiện kiểm thử khám phá.  - Thích hợp kiểm thử trong trường hợp các test case chỉ phải thực hiện một số ít lần.  - Giảm được chi phí ngắn hạn | - Tốn thời gian. Đối với mỗi lần release, người kiểm thử vẫn phải thực hiện lại quá trình regression với rất nhiều test case.  - Dễ mắc sai xót khi phải chạy quá nhiều test case. | - Thích hợp kiểm tra sản phẩm đang trong quá trình phát triển (develop). |
| Tự động | - Giảm thiểu các sai xót do con người gây ra.  - Giảm thời gian khi cần chạy những test case cần nhiều dữ liệu input.  - Phù hợp khi cần chạy nhiều test case trong thời gian dài.  - Giảm chi phí đầu tư dài hạn. | - Tốn kém hơn kiểm thử tự động, chi phí đầu tư ban đầu lớn  - Không thích hợp để làm khi code hay requirement của chương trình liên tục thay đổi | - Thích hợp kiểm thử sản phẩm đã ổn định.  - Thích hợp khi cần chạy performance testing hay load testing. |

# Kiểm thử tự động với Katalon Studio

## Giới thiệu về Katalon Studio

Katalon Studio là phần mềm kiểm thử tự động miễn phí được thiết kế bởi đội ngũ lập trình Katalon LLC. Phần mềm được phát triển dựa trên nền tảng của Selenium và Appium với giao diện riêng phù hợp cho việc kiểm thử Web, Mobile và API.

Ngôn ngữ lập trình chính của Katalon Studio là Apache Groovy, Java và Javascript.

Katalon studio tập trung vào việc hướng người dùng sử dụng keyword-driven testing.

Keyword- driven testing là một kỹ thuật lập trình mà ở đó mỗi keyword test case sẽ tương ứng với một action mà người dùng muốn thực hiện thay vì là function mà hệ thống thực hiện.

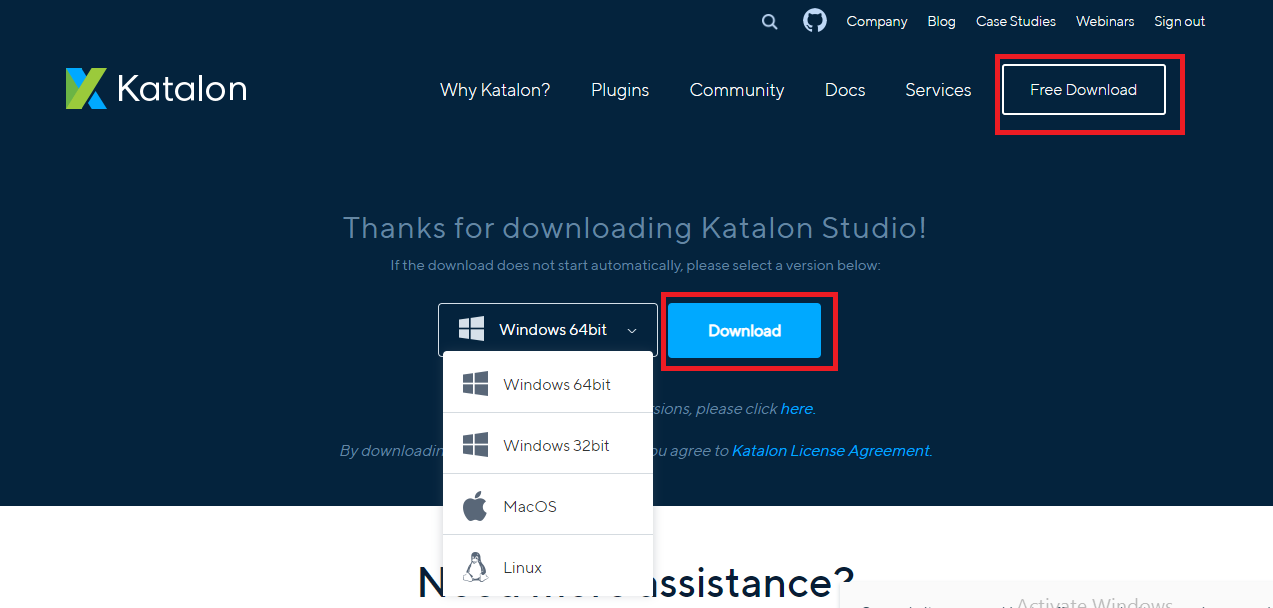
Keyword- driven testing tách riêng phần function code và data cho phép người dùng linh động trong việc thay đổi data cũng như cho phép việc một người hoàn toàn mới không biết gì về code có thể sử dụng lại những test case cũ do người khác viết.

Ví dụ, người dùng muốn đăng nhập vào hệ thống với keyword là Login. Người dung sẽ nhập input Username và Password vào keyword. Keyword sẽ chạy và map thông tin username, password và nhấn phím Login. Kết quả người dùng nhận được sẽ là đăng nhập thành công hay thất bại. Người dùng sẽ không cần quan tâm đến việc phải xử lý các bước mapping bên trong mà chỉ cần tập trung vào input và output.

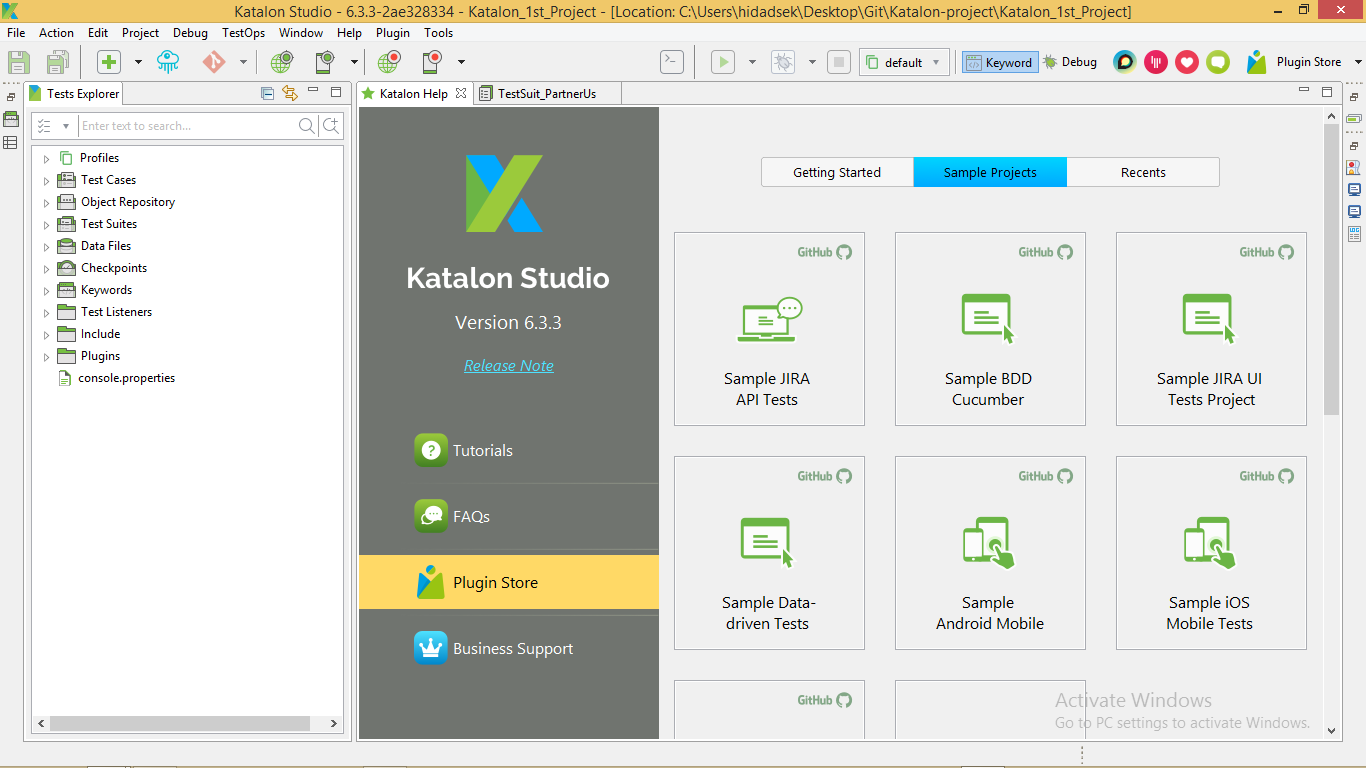
Ngoài ra, Katalon còn hỗ trợ data-driven testing (chạy 1 test case nhiều lần với các data khác nhau) bằng cách cho phép sử dụng các data với định dạng khác nhau như Excel, Database hay tạo riêng table để nhập data vào.

## Cài đặt Katalon

* Step 1: Download Katalon ở trong trang: <https://www.katalon.com/download/>



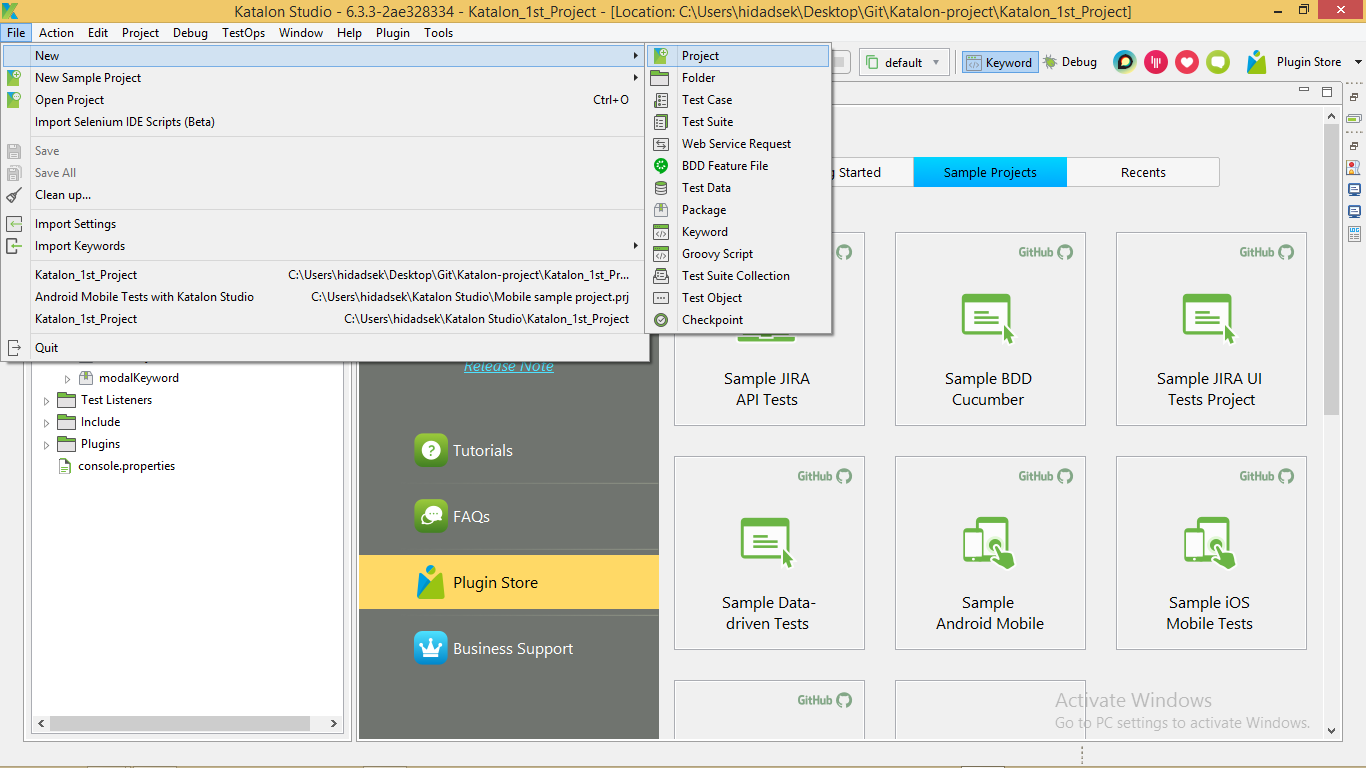
* Step 2: Sau khi download, extract file đã được download.
* Step 3: Vì là bản Portable nên mọi người có thể sử dụng ngay sau khi extract.



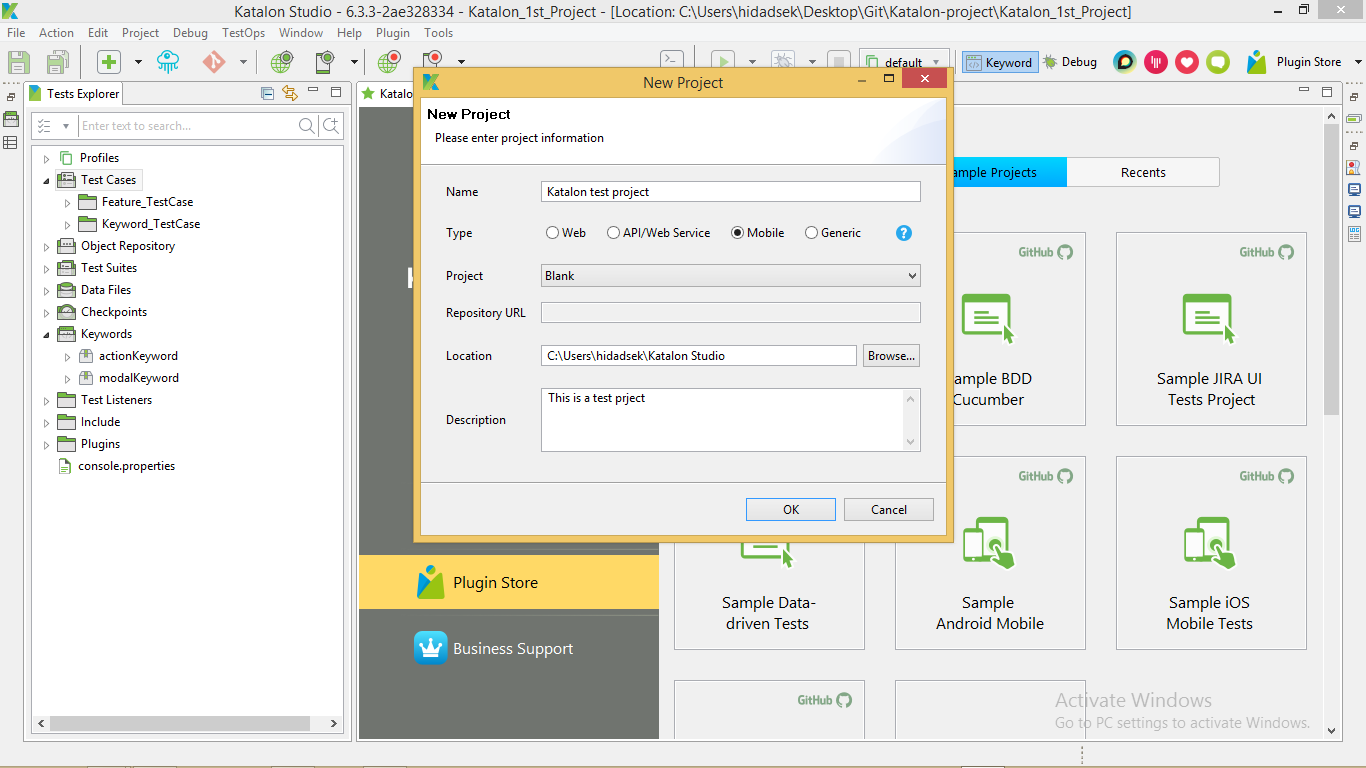
## Tổng quan các tính năng của Katalon

### Tạo project

* Step 1: Chọn File 🡪 New 🡪 Project



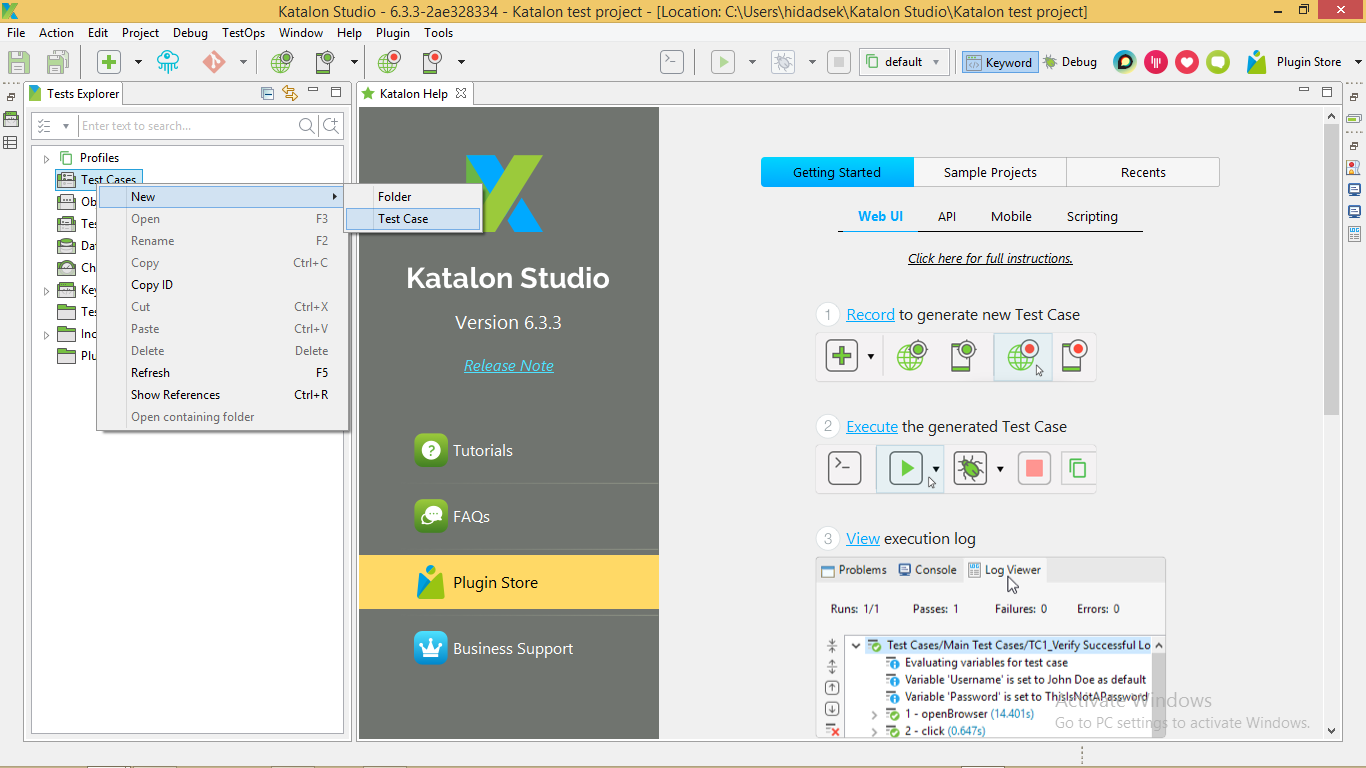
* Step 2: Nhập tên project, chọn type project, điền thông tin Description và nhấn OK



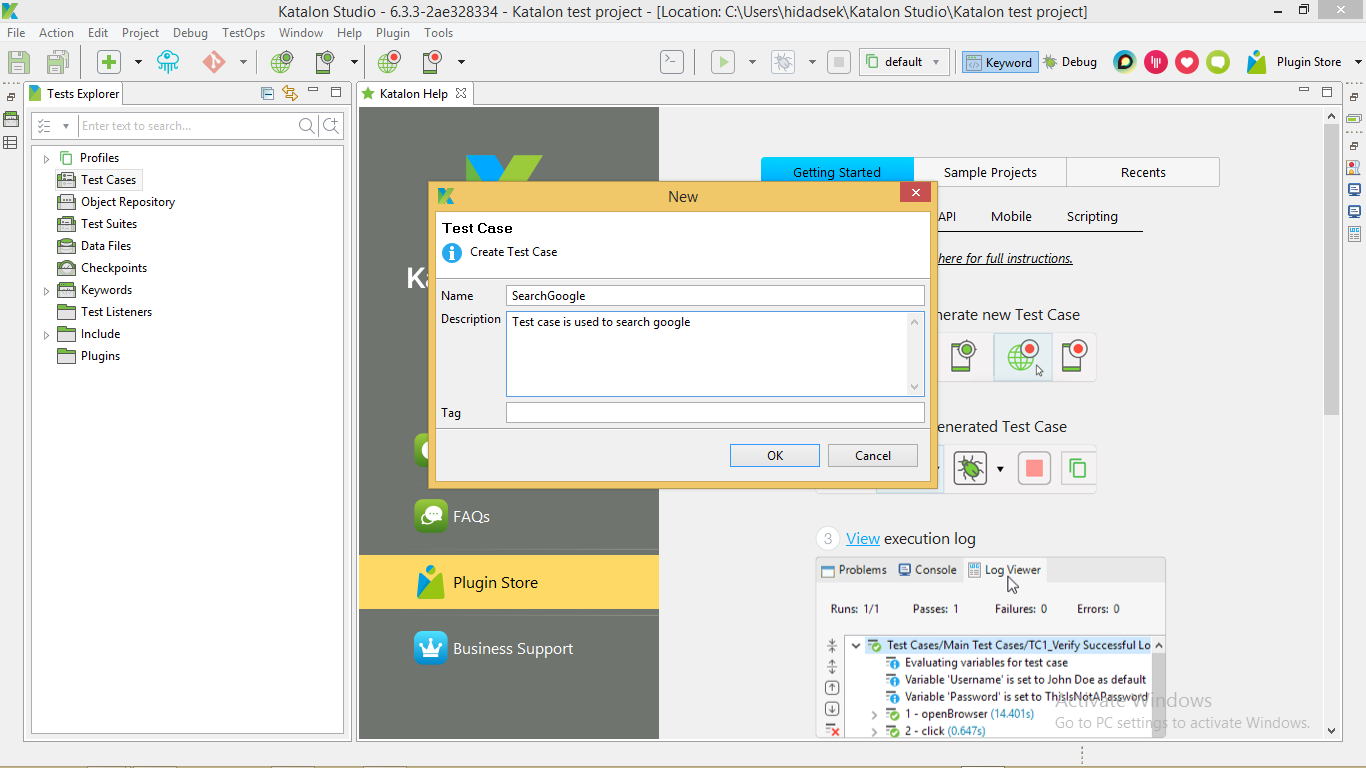
* Step 3: Sau khi tạo project xong, Katalon Studio sẽ tự load project mới lên. Bạn có thể bắt đầu tạo và chạy test case trên project.

### Tạo test case

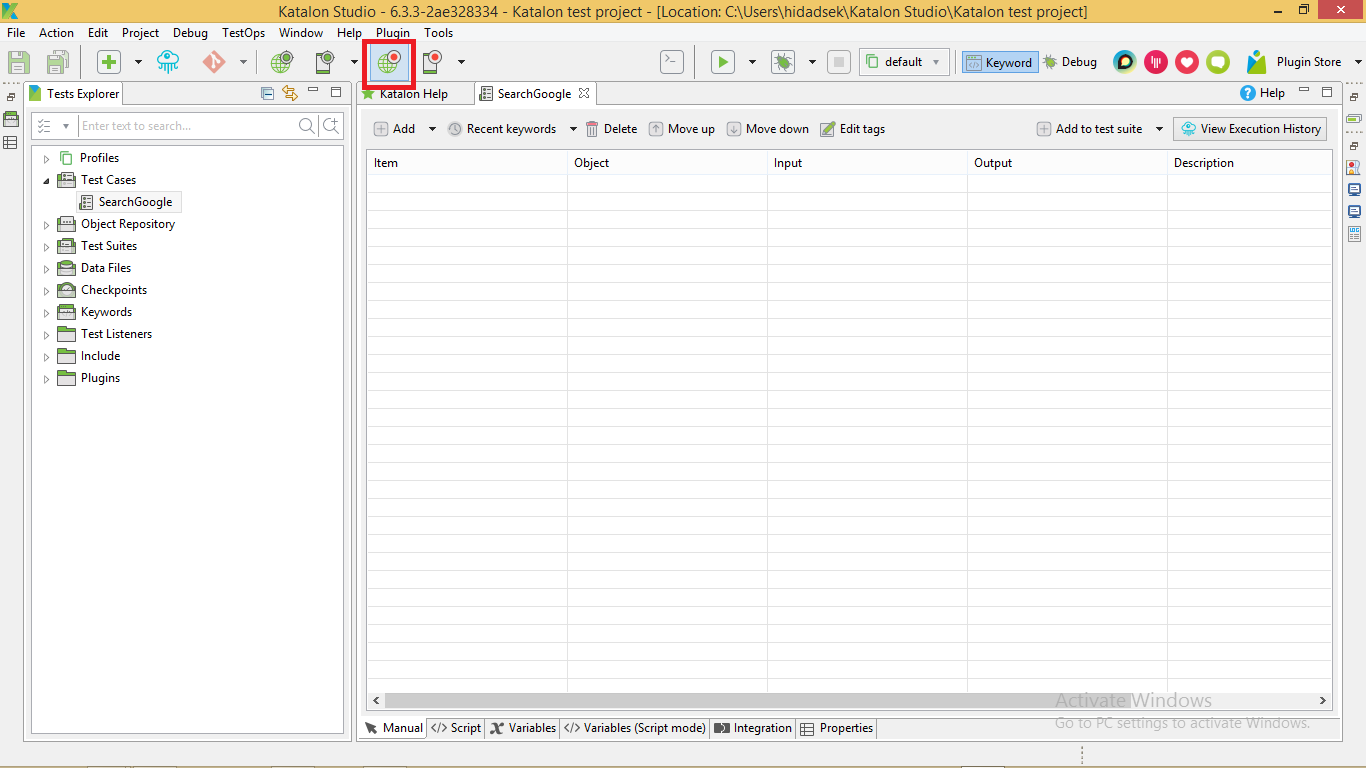
* Step 1: Sau khi vào project, click chuột phải Test Case 🡪 New 🡪 Test Case



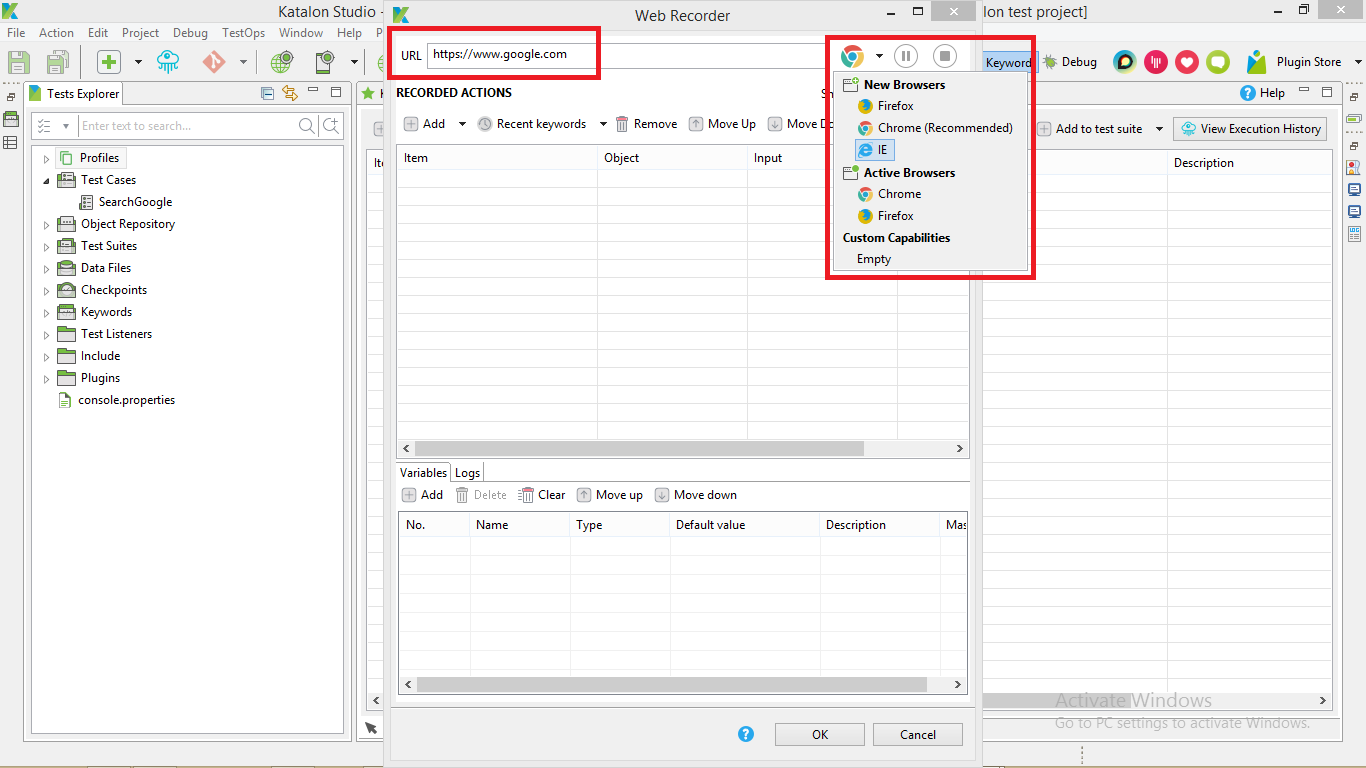
* Step 2: Nhập name và description của test case sau đó nhấn OK



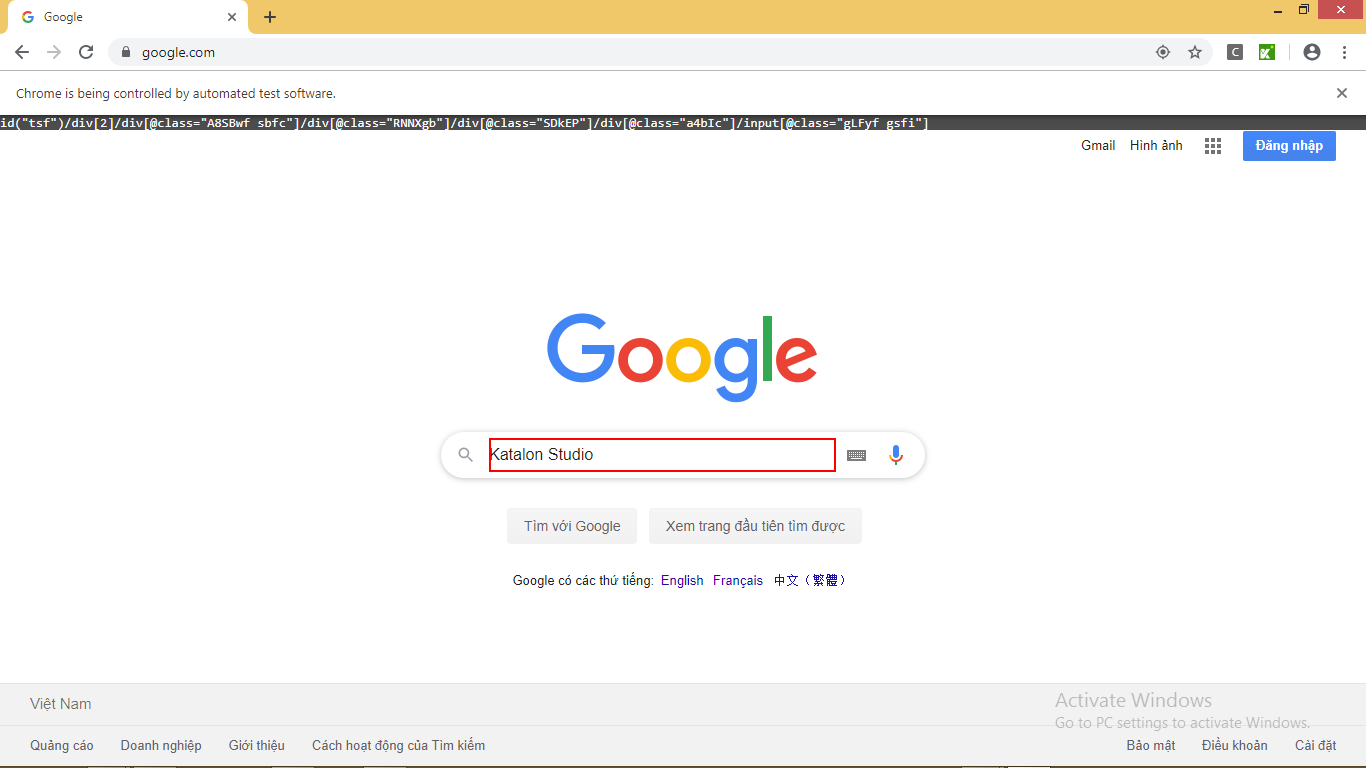
* Step 3: Khi test case được tạo, bạn có thể tạo các step trong test case. Có 2 cách để tạo step
* Step 4.1.1: Bạn có thể tạo bằng cách record test case.



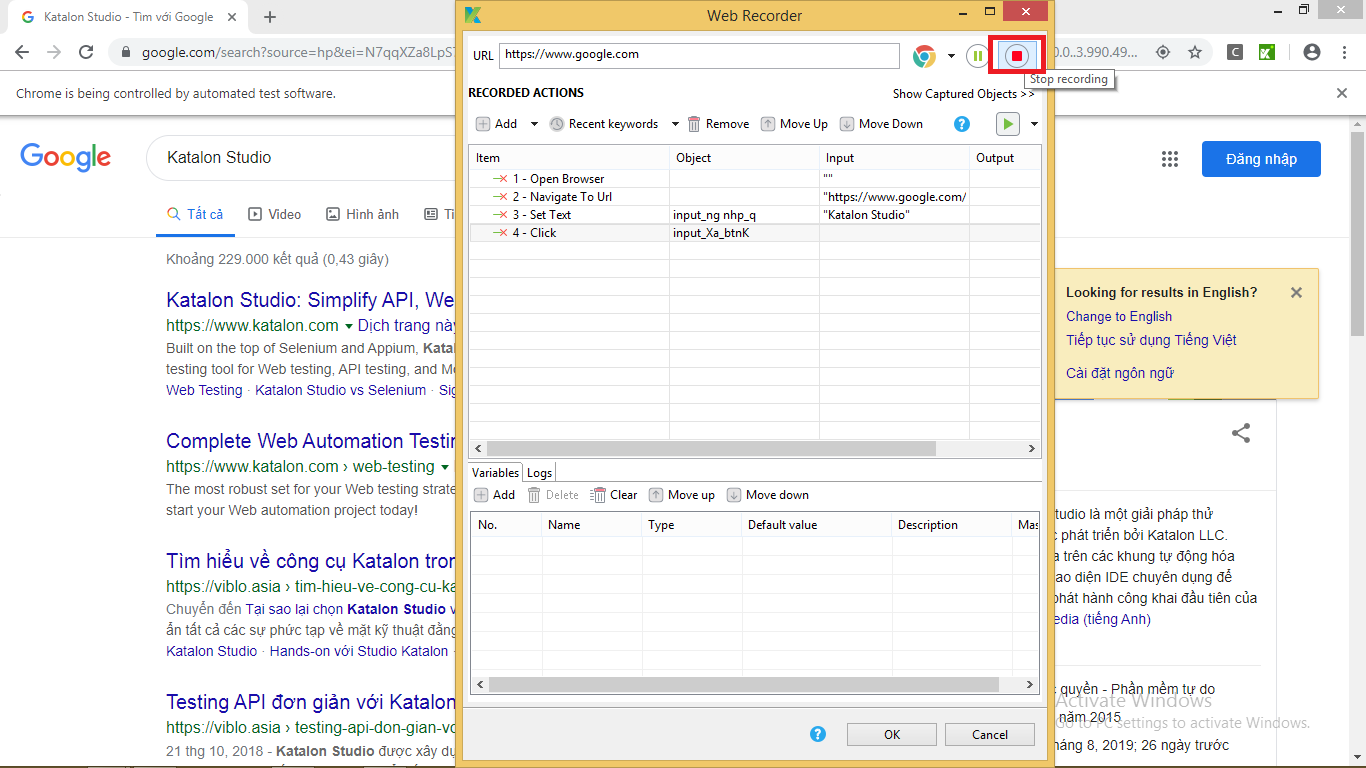
* Step 4.1.2: Nhập URL, chọn browser và nhấn Record web (browser icon)



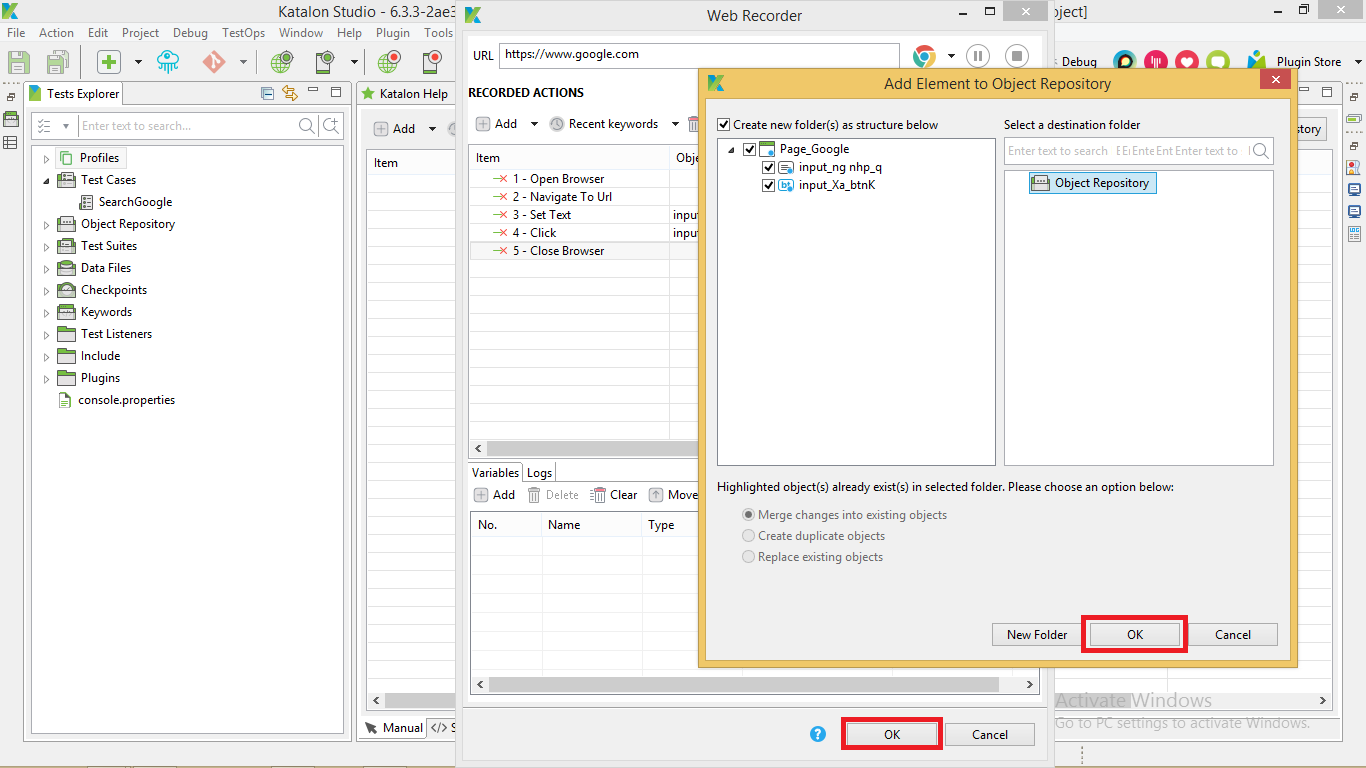
* Step 4.1.3: Sau khi browser mở và chuyển đến URL đã nhập, bạn bắt đầu chạy test case như đang chạy manual.



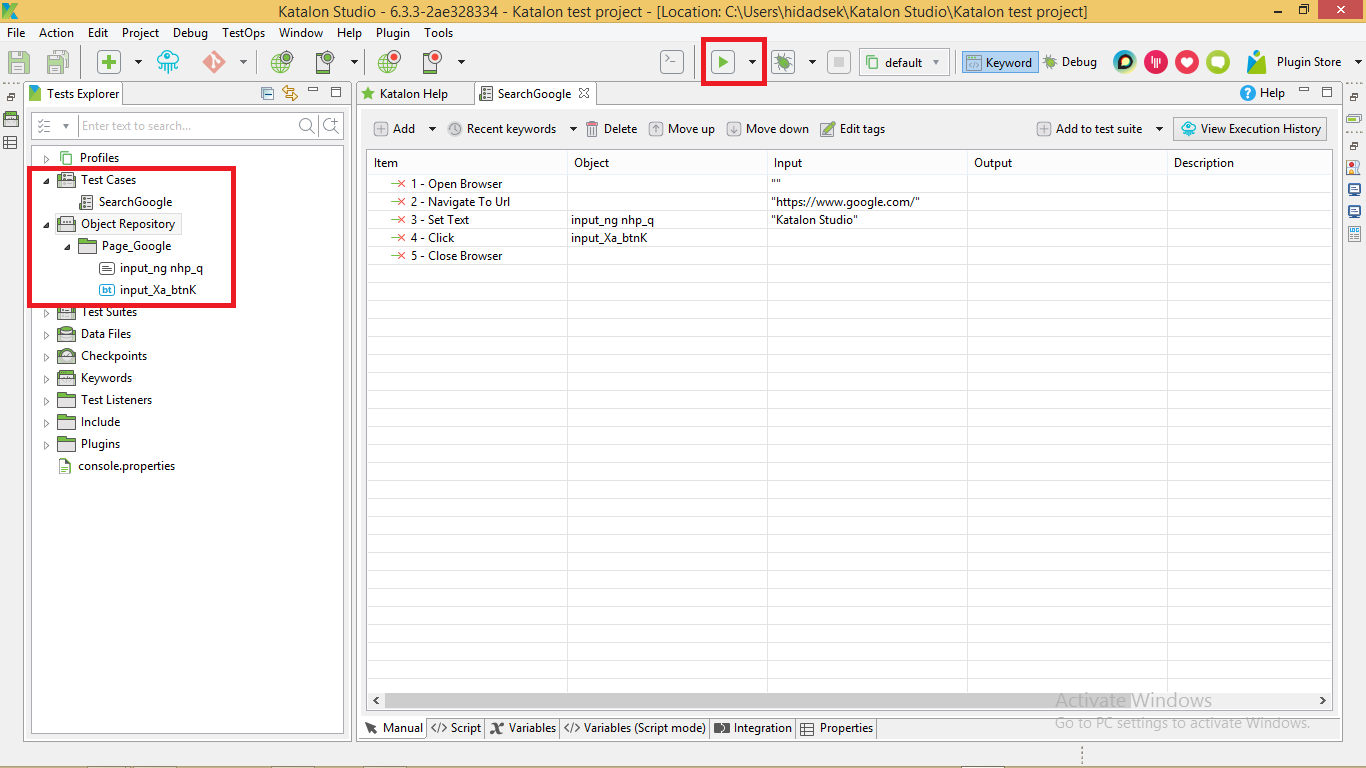
* Step 4.1.4: Sau khi chạy xong test case, bạn nhấn stop recording.



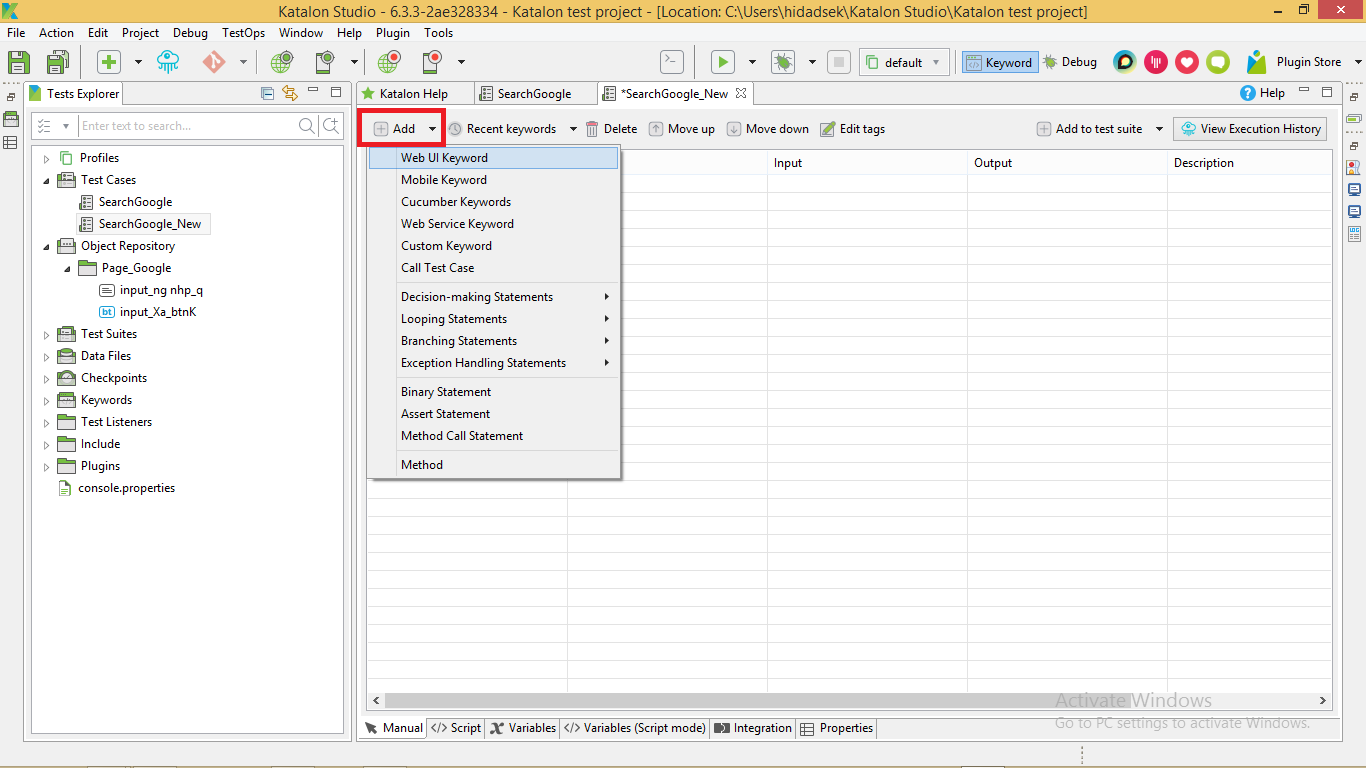
* Step 4.1.5: Katalon sẽ generate ra các step và các biến tương ứng với test case mà bạn chạy. Bạn có thể thay đổi các step và các tên biến tùy thích, sau đó ấn Ok.



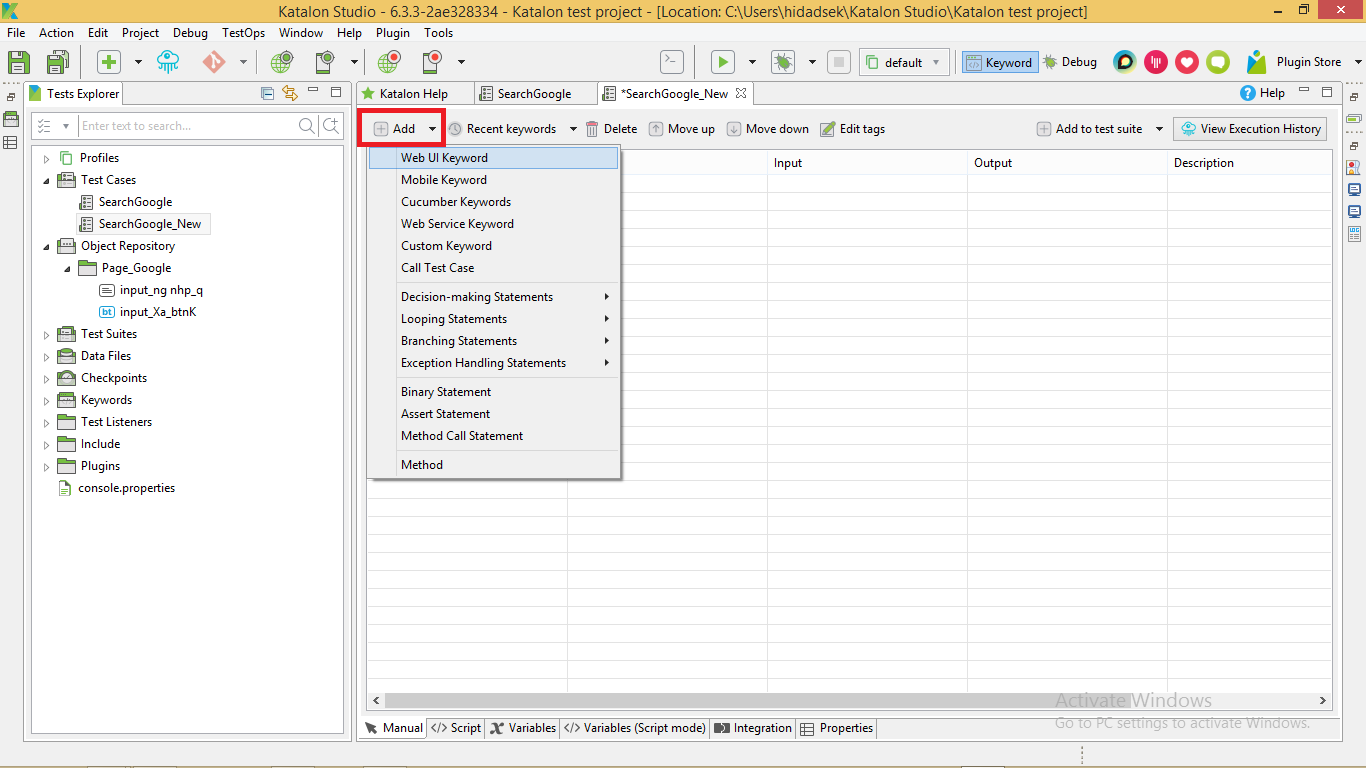
* Step 4.1.6: Việc record test đã hoàn tất. Khi cần chạy test bạn chỉ cần nhấn play.



* Step 4.2.1: Bạn có thể tạo bằng cách viết test case step by step
* Step 4.2.2: Click Add 🡪 Keyword bạn muốn dùng



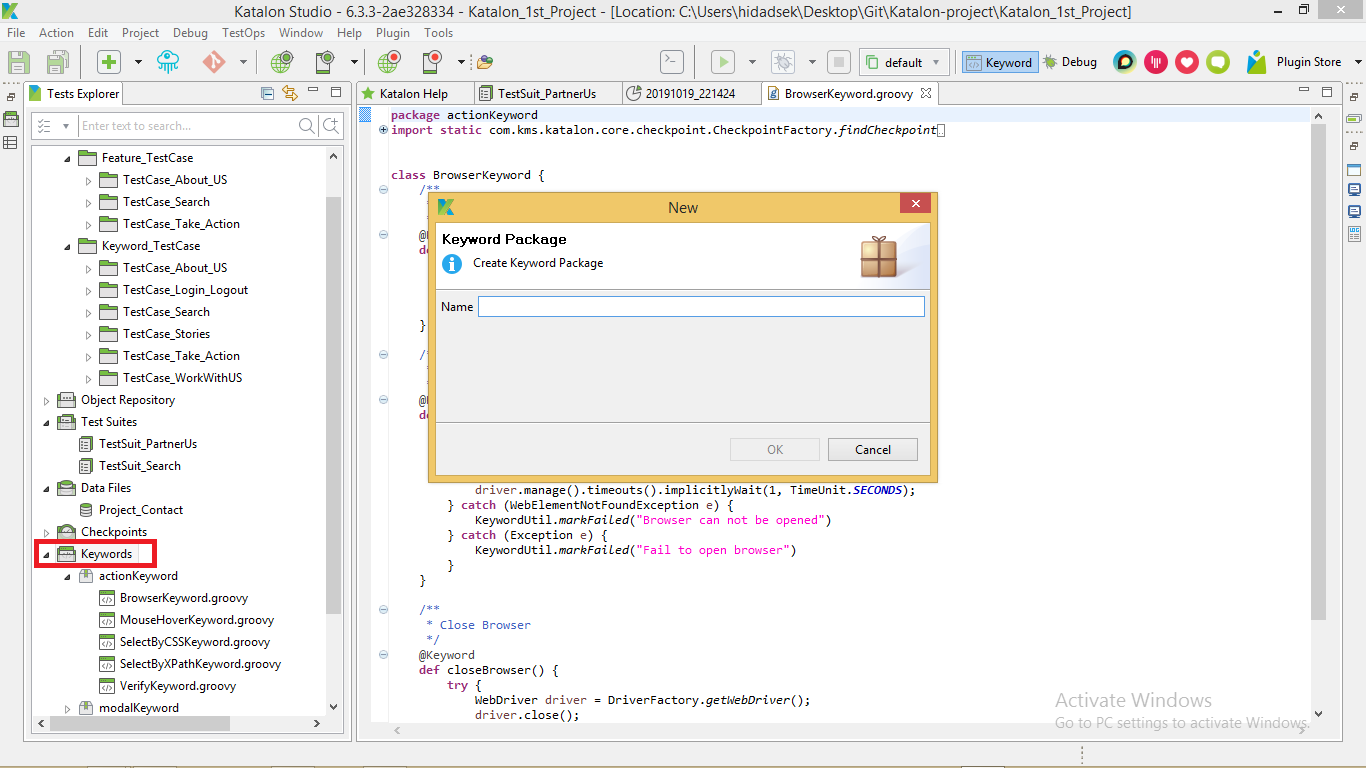
* Step 4.2.3: Chọn hàm bạn cần dùng và nhập input



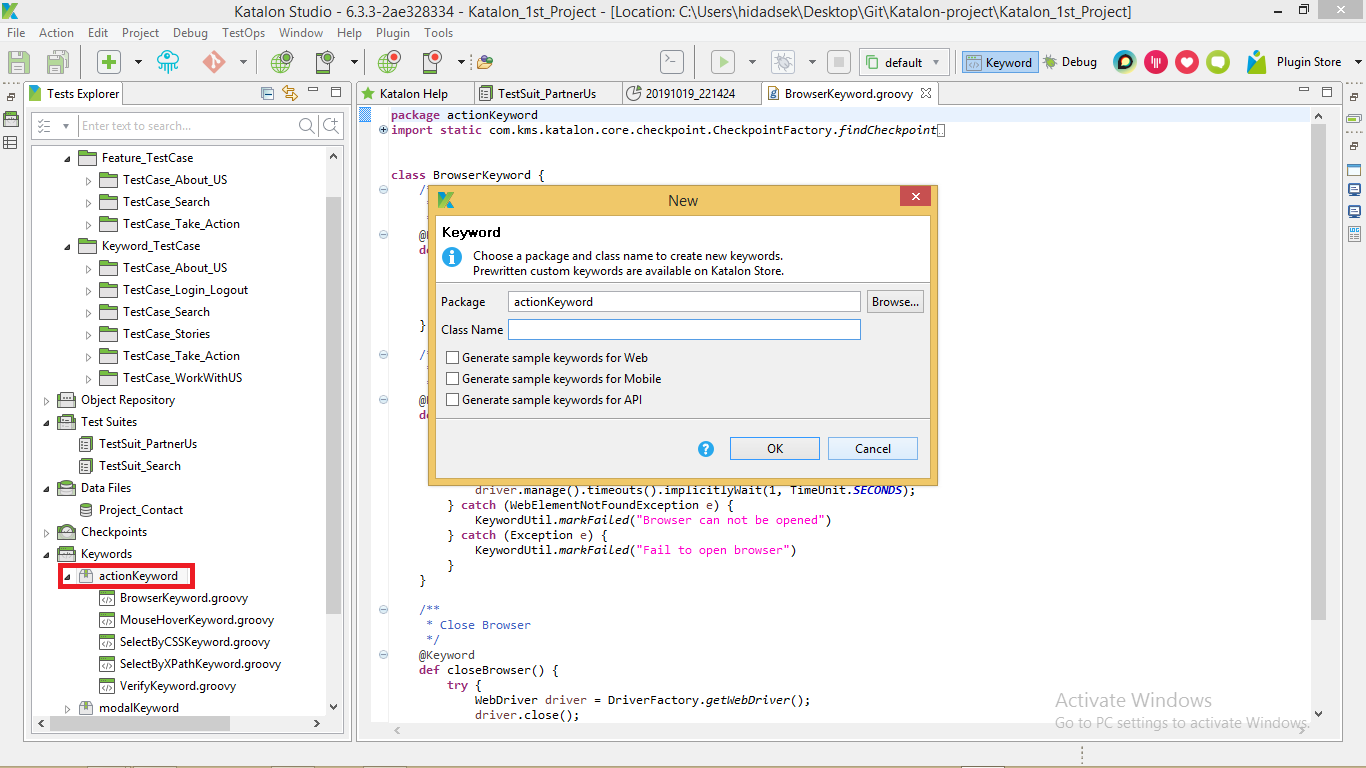
* Step 4.2.4: Việc tạo test case đã hoàn thành. Khi cần chạy test bạn chỉ cần nhấn play

Ngoài ra mọi người còn cố thể viết riêng các keyword để chạy test case mà không cần phải dùng keyword của Katalon. Lợi ích của việc viết riêng các keyword sẽ làm giảm bớt lượng data mà Katalon phải lưu trong Object Repository cũng như có thể customize theo ý muốn của bản thân để chạy keyword hiệu quả. Các step để tạo custom keyword:

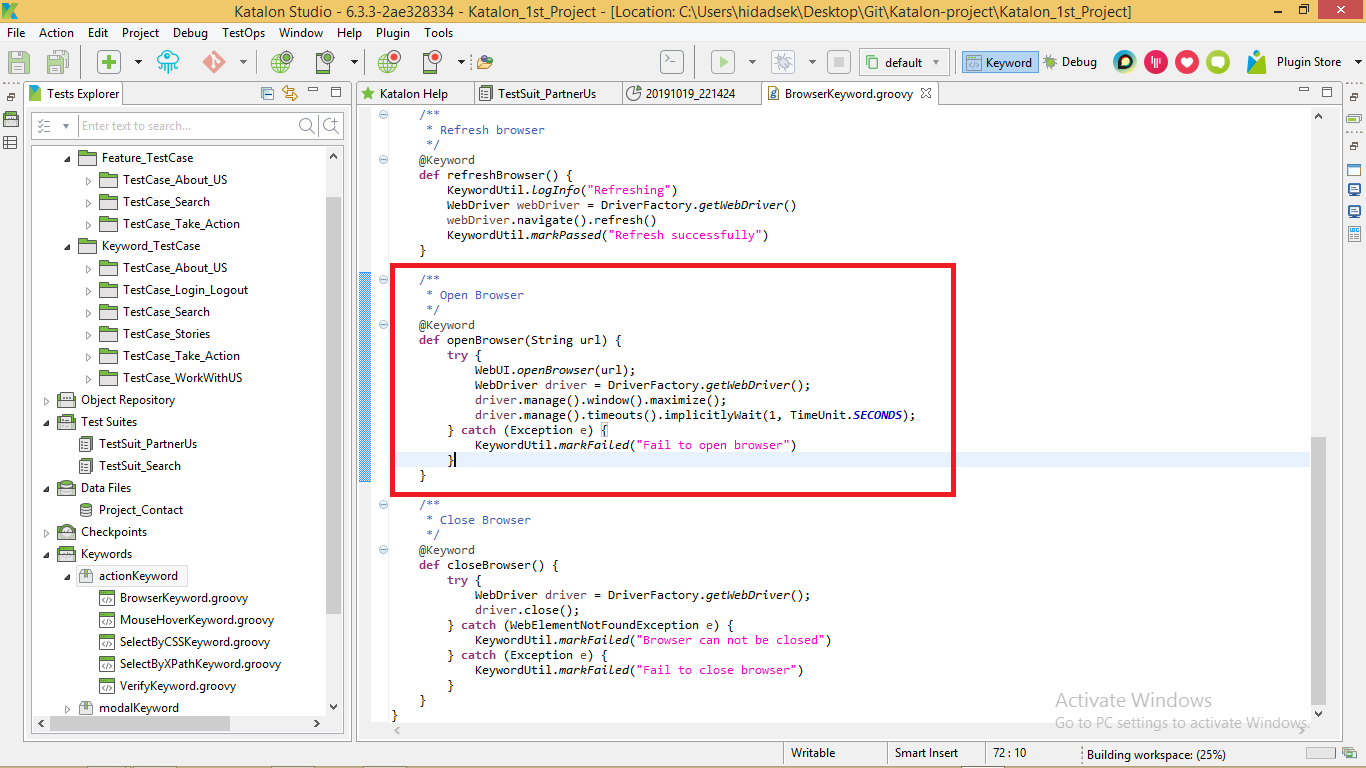
* Step 1: Nhấn chuột phải thư mục Keyword, chọn New 🡪 Package 🡪 Nhập tên và tạo package (nếu đã có package từ trước thì bỏ qua step 1)



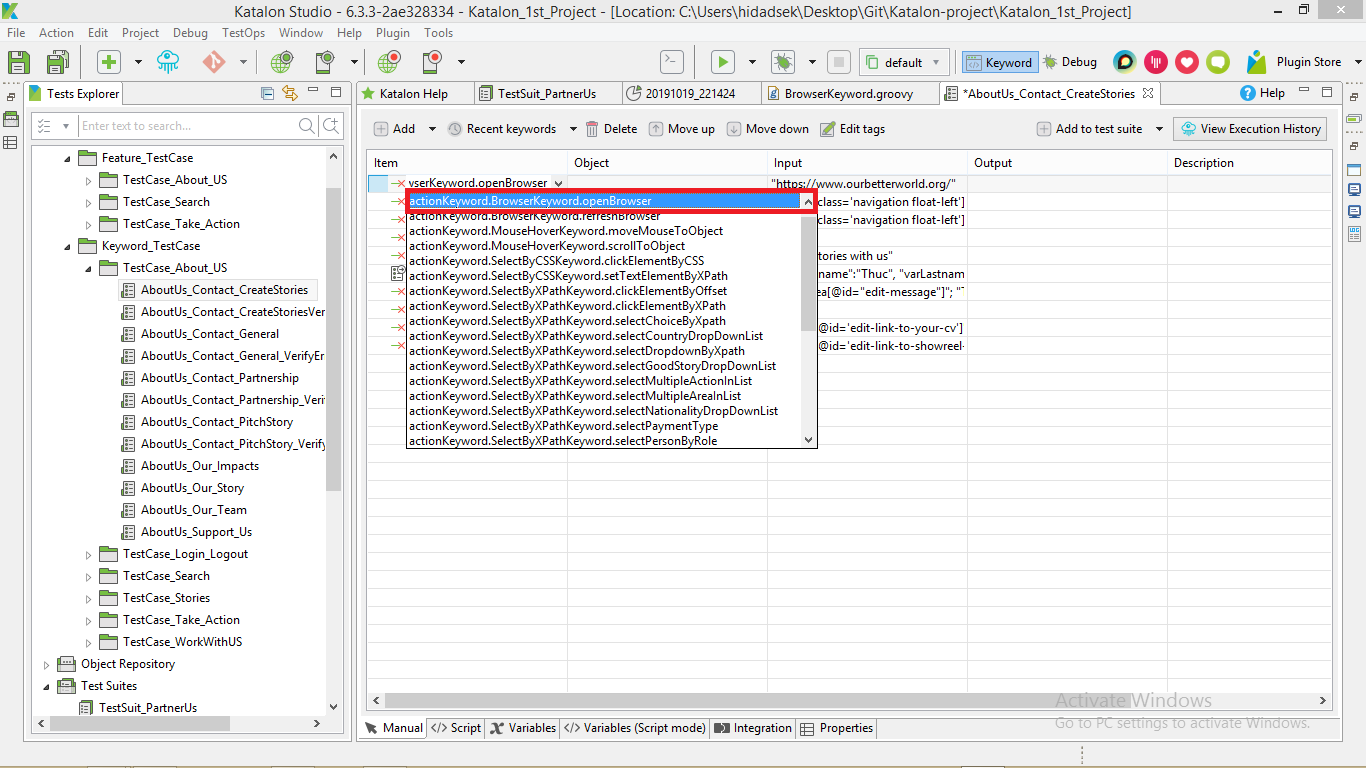
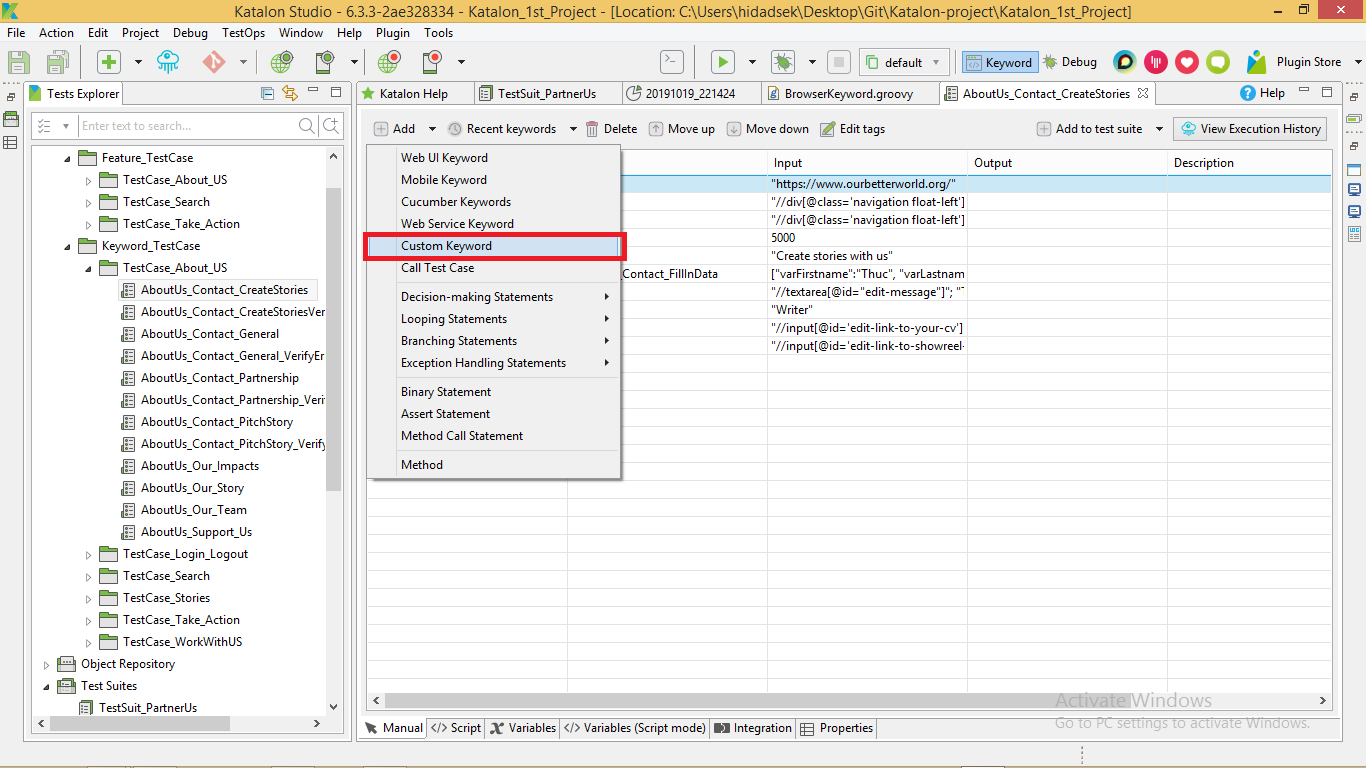
* Step 2: Nhấn chuột phải package vừa tạo, chọn New 🡪 Keyword
* Step 3: Nhập tên và tạo keyword. Katalon có cung cấp các sample keywords cho từng loại test. Các bạn có thể chọn để làm quen cách viết.



* Step 4: Sau khi keyword test được tạo, các bạn có thể vào keyword và viết các hàm mình muốn. Lưu ý các bạn phải thêm @Keyword vào trước các hàm để Katalon có thể hiểu đây là keyword mà các bạn muốn dùng để test



* Step 5: Để sử dụng các hàm vừa, các bạn vào test case chọn Add 🡪 Custom Keyword 🡪 Sau đó chọn keword mà mình đã tạo.

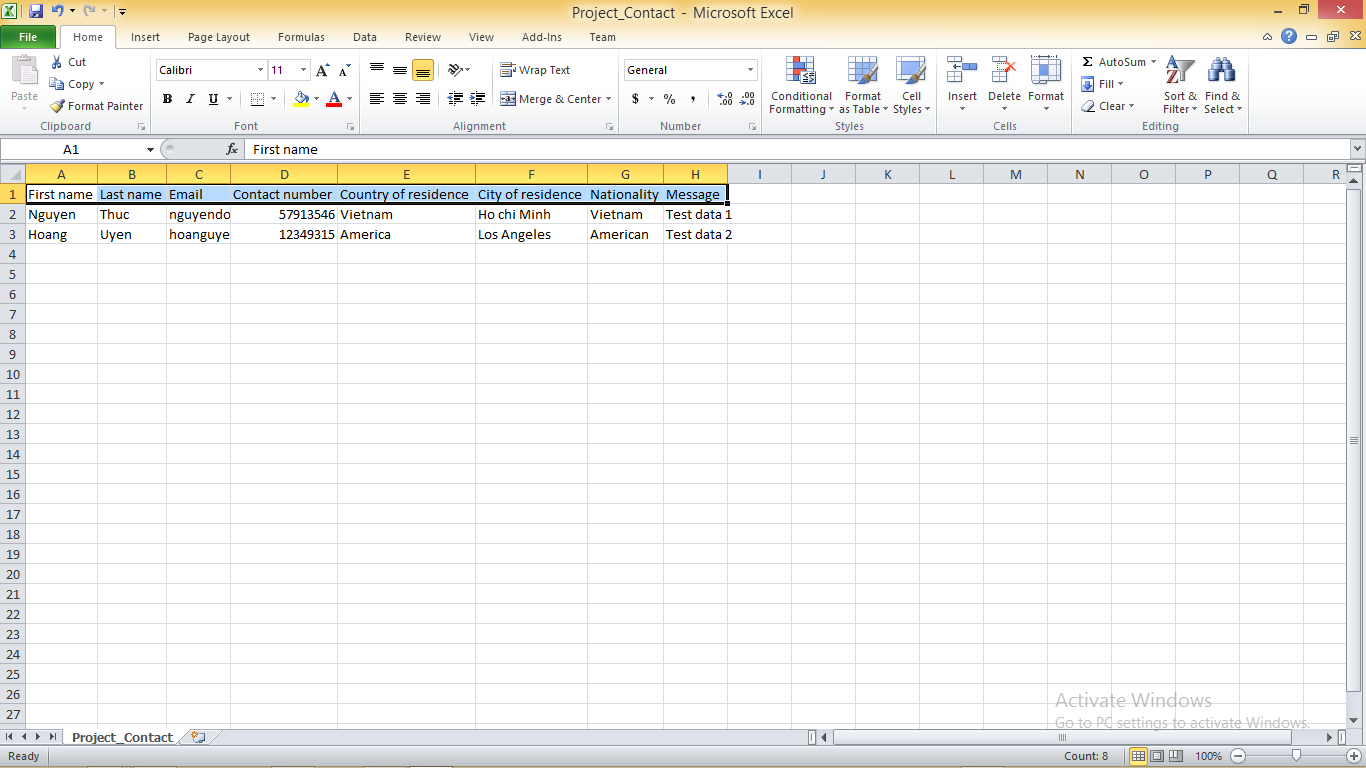


### Tạo test data

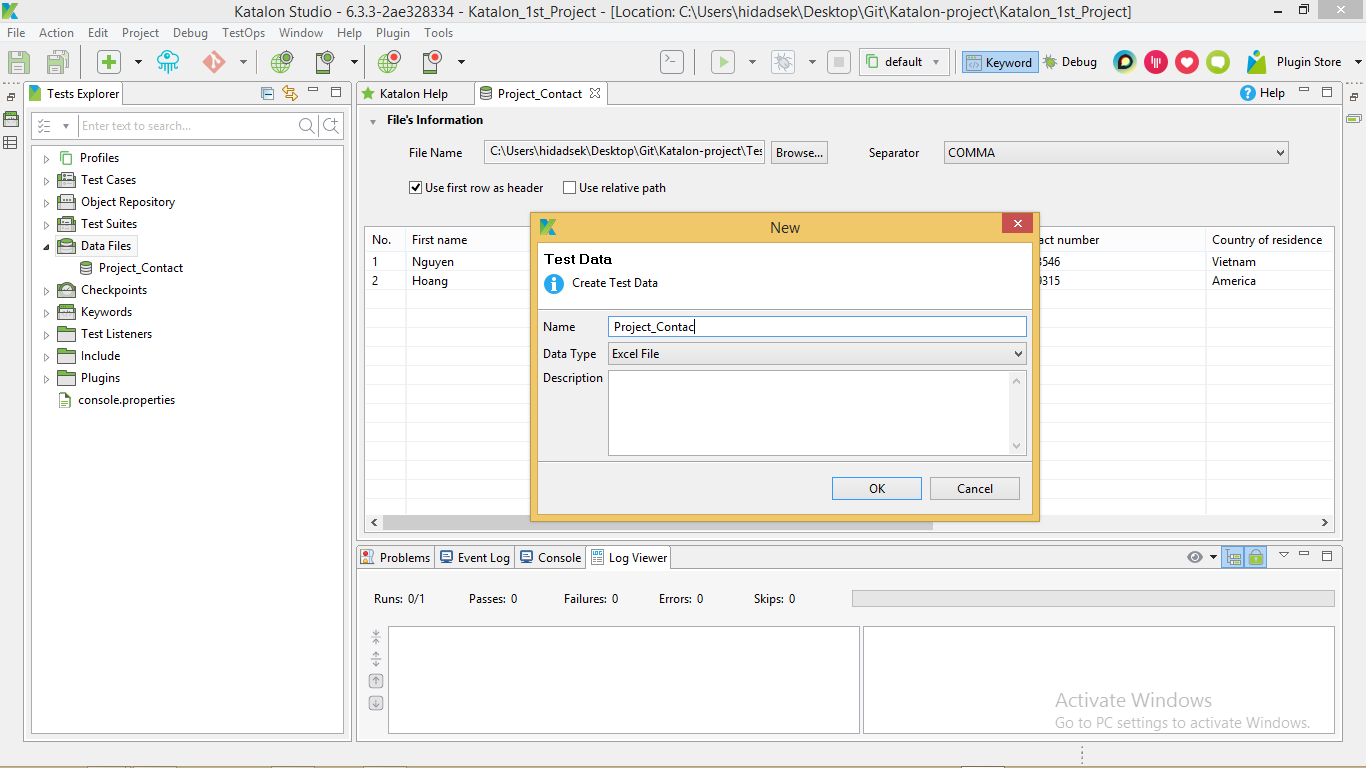
Katalon Studio hỗ trợ chạy 4 cách để tạo test data:

#### Tạo data bằng file CSV

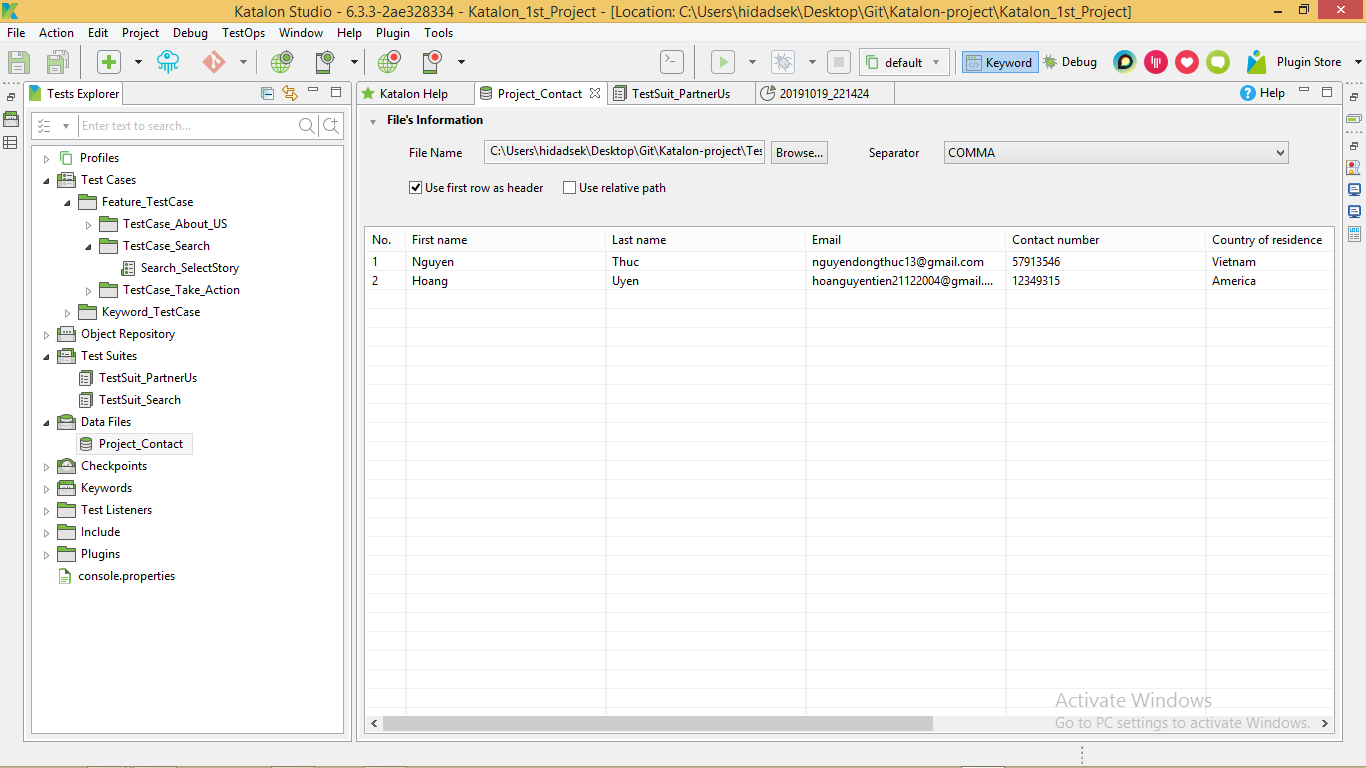
* Step 1: Chuẩn bị file CSV. Dòng thứ nhất là tên biến, dòng thứ 2 là data



* Step 2: Chọn File > New > Test Data. Chọn Data Type là CSV File. Nhấn OK.

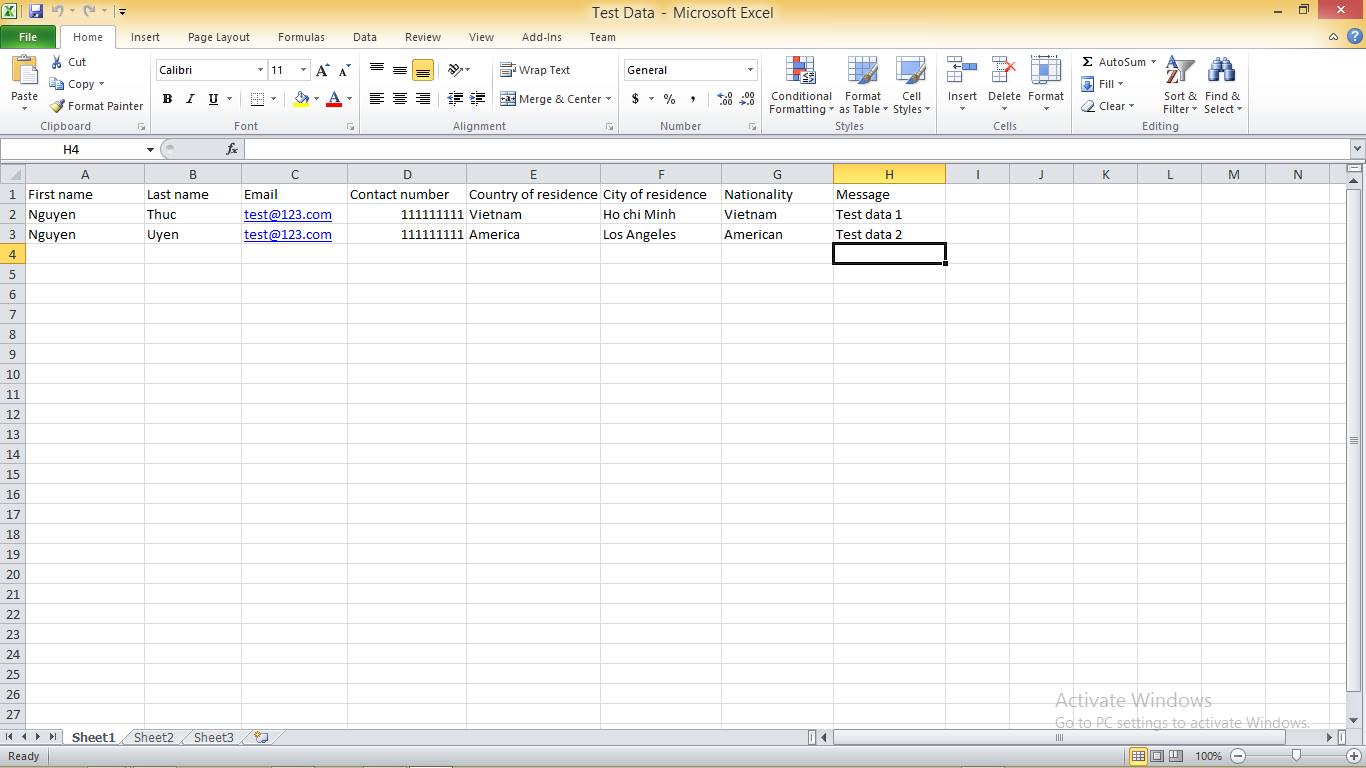


* Step 3: Nhấn Browse và chọn file

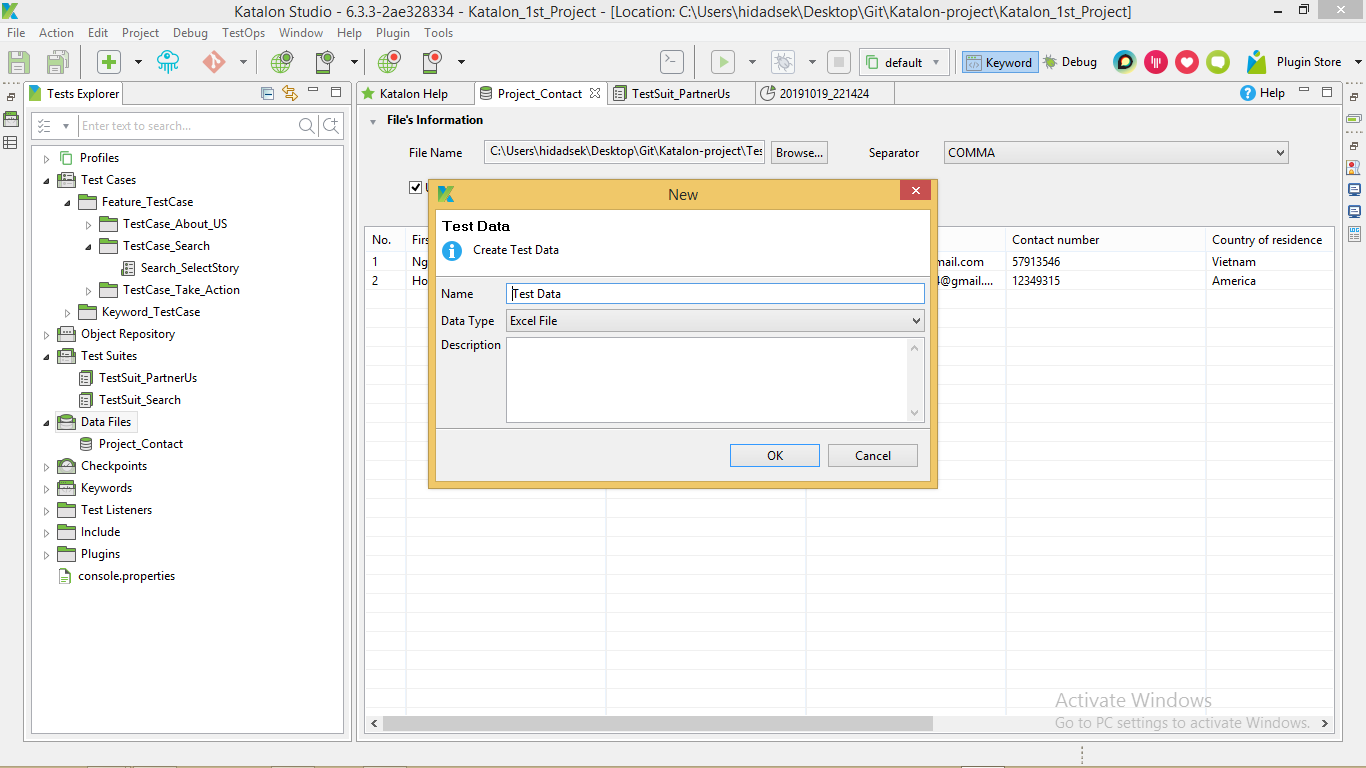


#### Tạo data bằng file excel

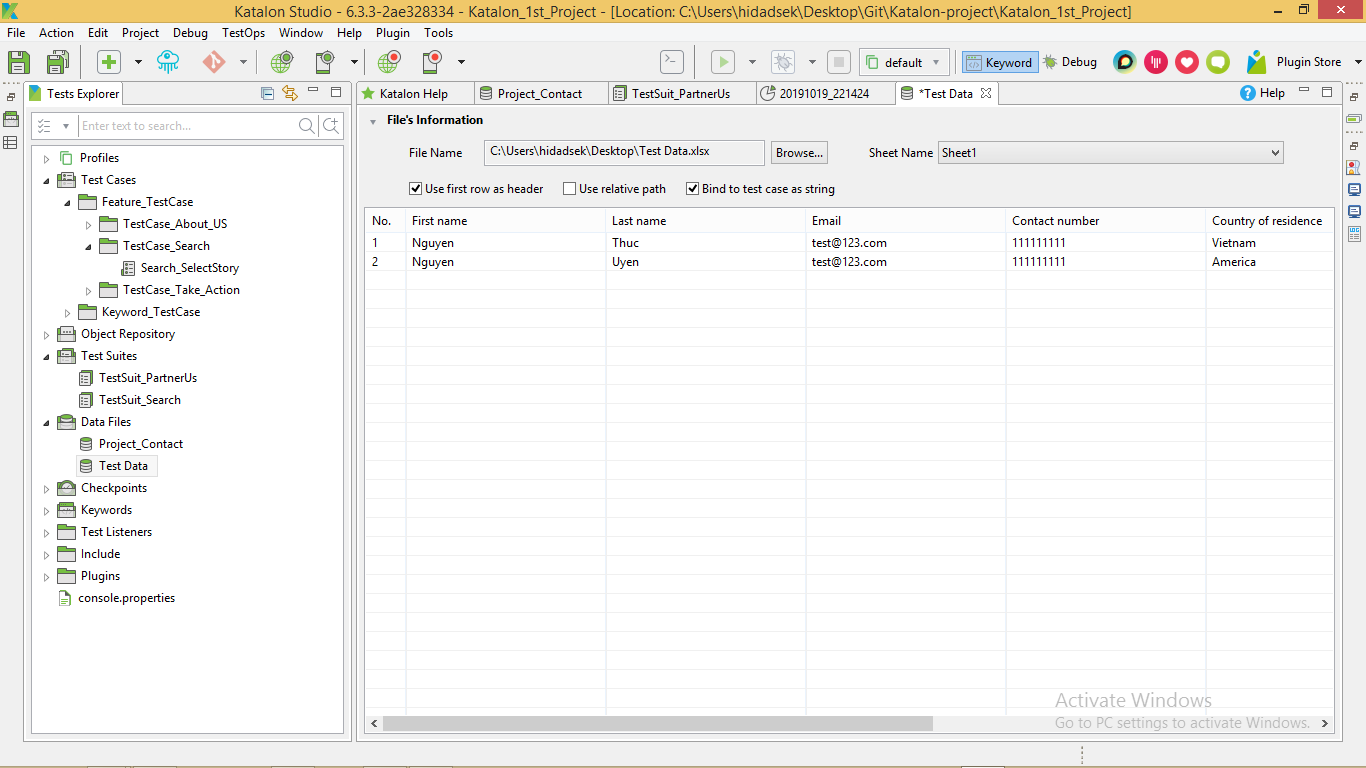
* Step 1: Chuẩn bị file Excel.



* Step 2: Chọn File > New > Test Data. Chọn Data Type là Excel File. Nhấn OK.

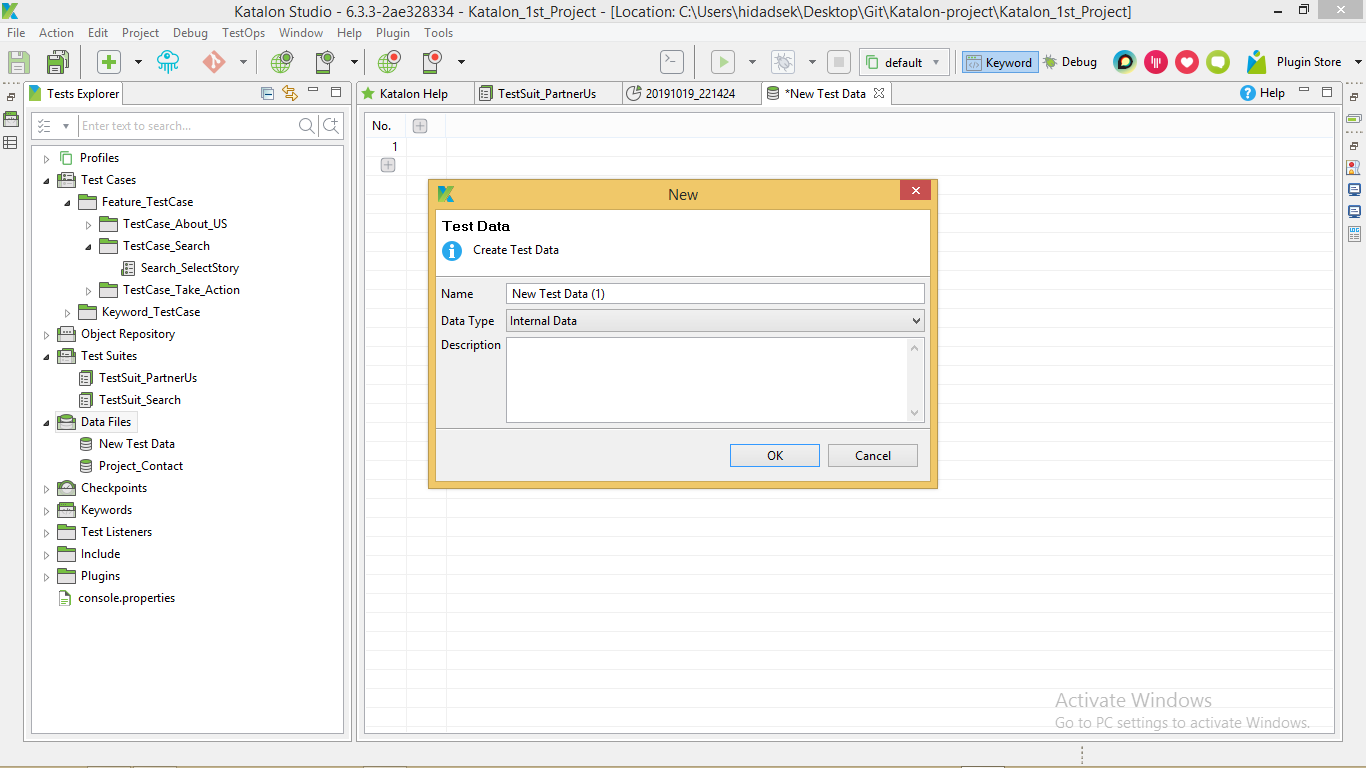


* Step 3: Nhấn Browse và chọn file

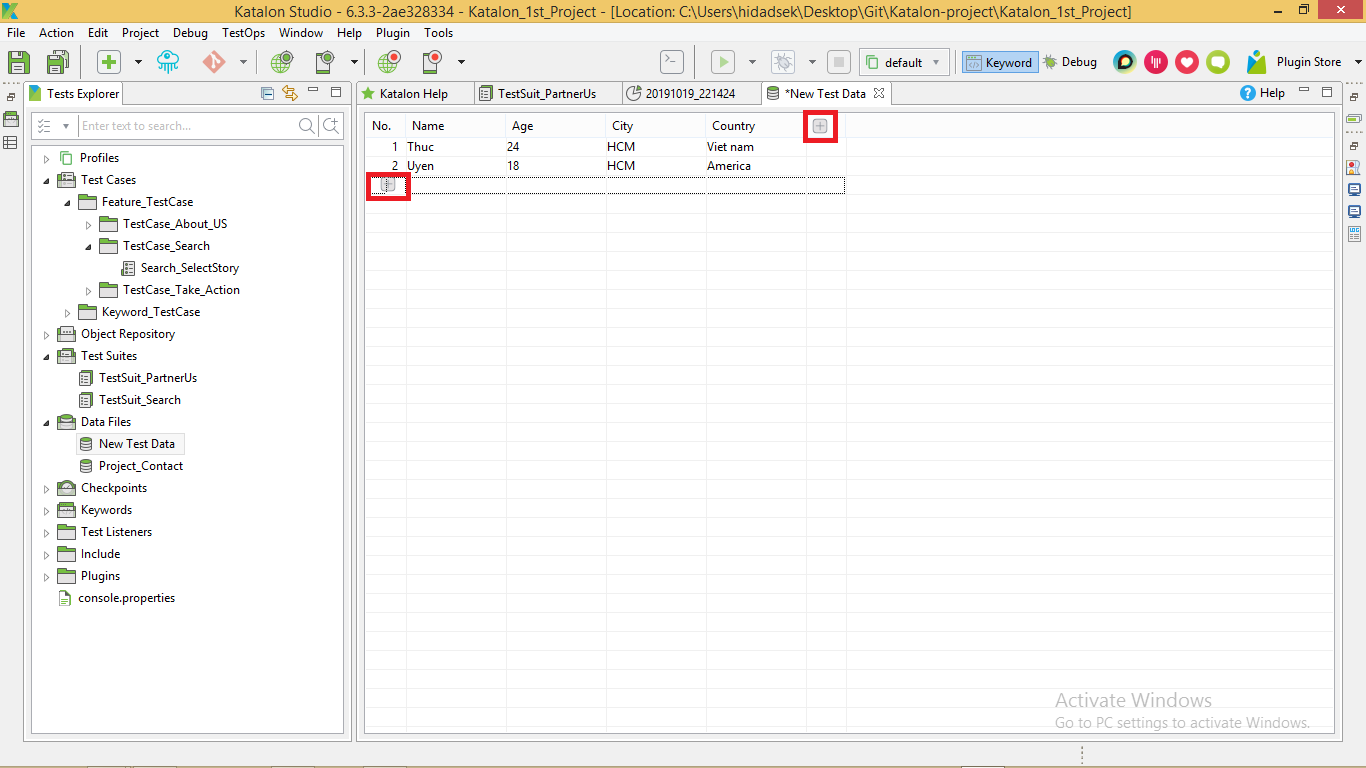


#### Tạo data bằng internal table

* Step 1: Chọn File > New > Test Data. Chọn Data Type là Internal Data. Nhấn OK.

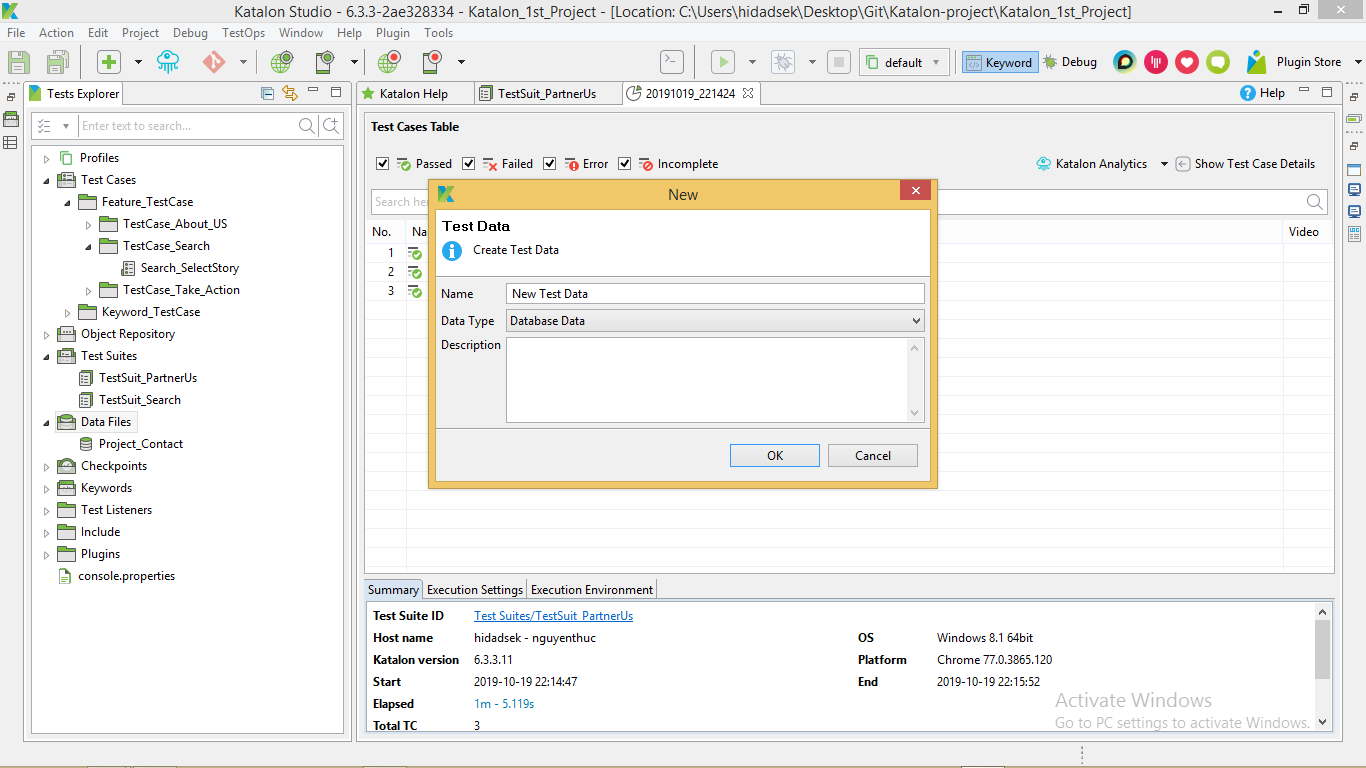


* Step 2: Thêm cột và dòng bằng chách nhấn vào dấu +.

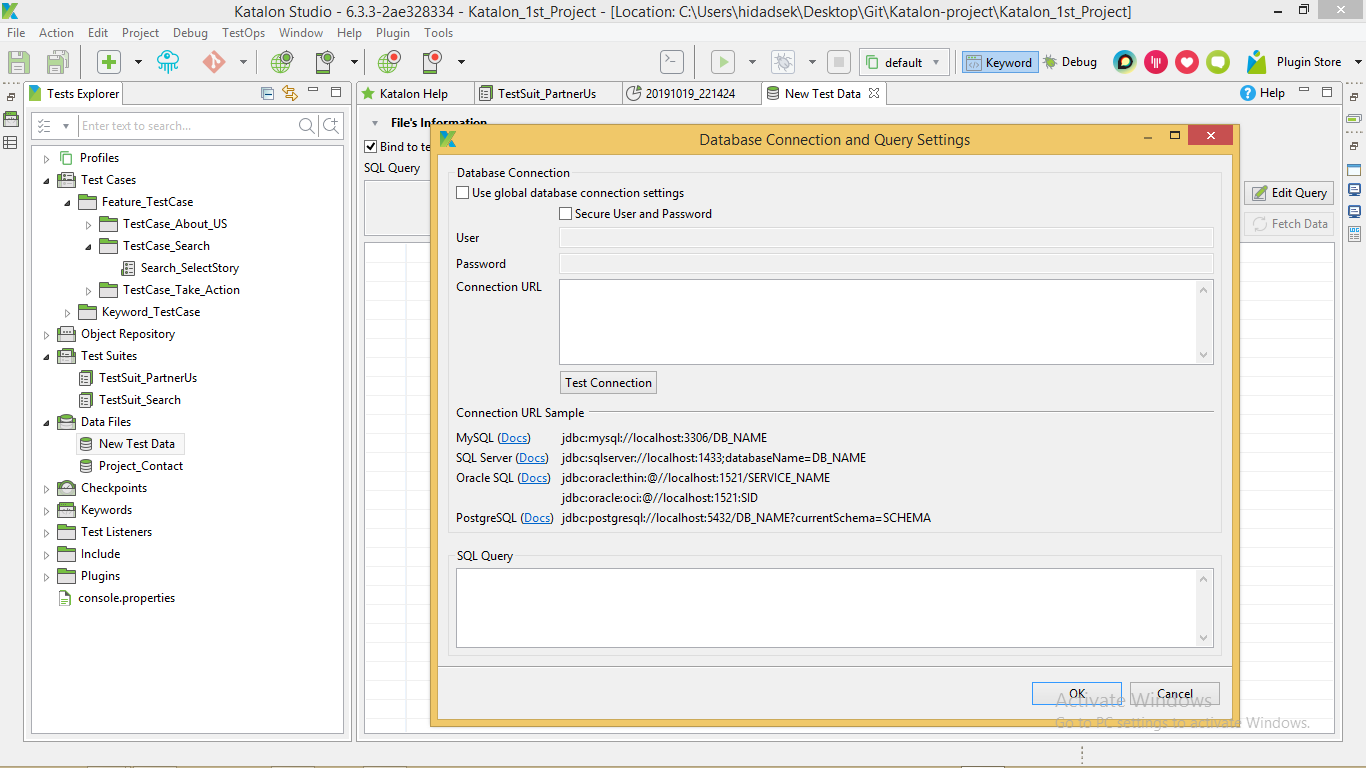


#### Tạo data bằng cách database

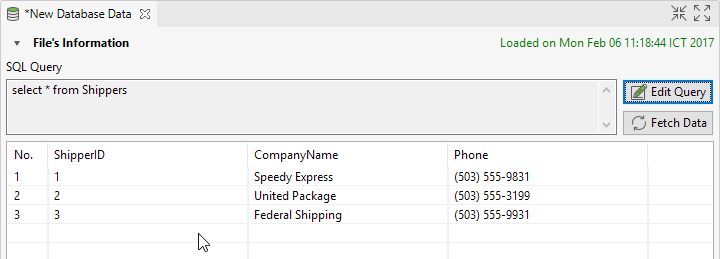
* Step 1: Chọn File > New > Test Data. Chọn Data Type là Database Data. Nhấn OK.



* Step 2: Nhấn Edit Query để mở dialog và nhập Connection URL



* Step 3: Sau khi nhập URL, nhập query và nhấn OK. Database sẽ được hiện ở phần preview



# Áp dụng Katalo kiểm thử ứng dụng web Our Better World

## Test Approach

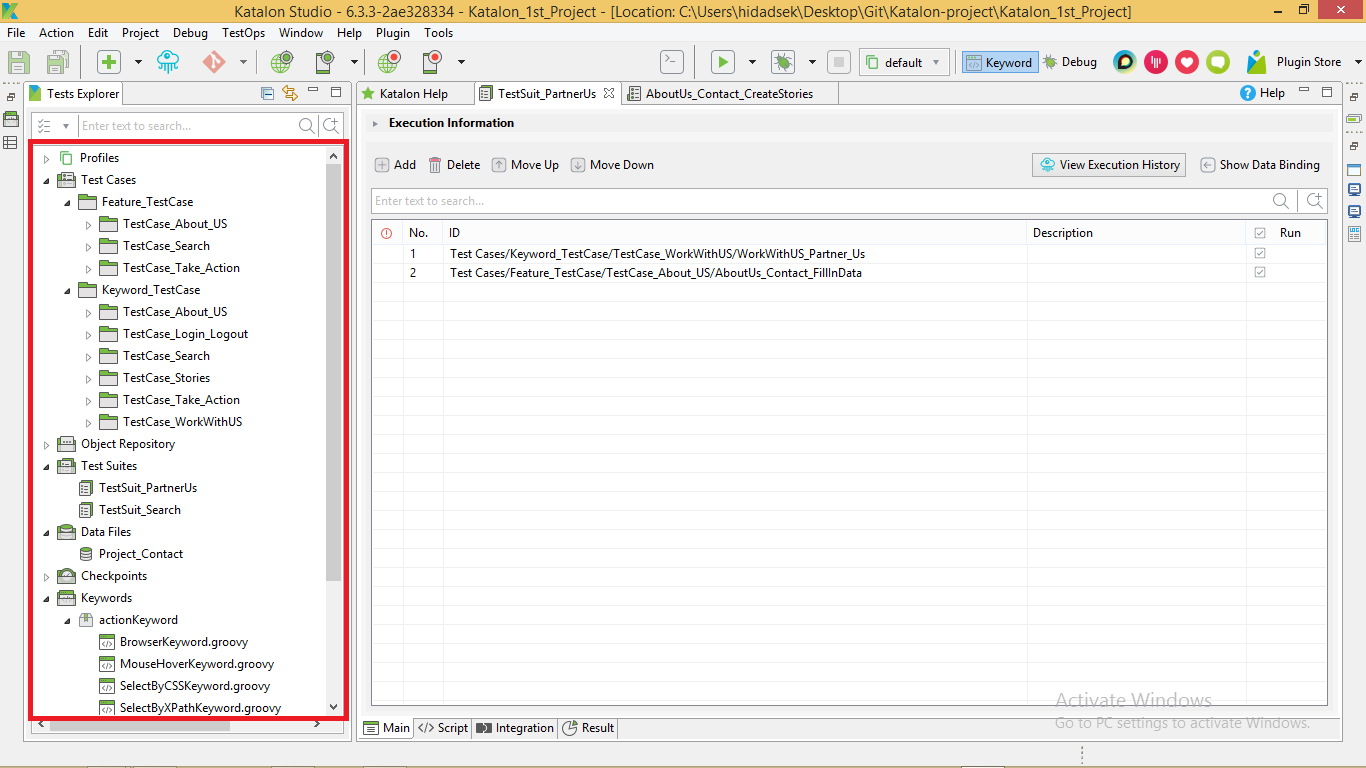
Hướng tiếp cận viết test case sẽ tập trung theo hướng Graphical user interface testing hay còn biết đến là test về UI. Các test case sẽ được viết theo hướng custom keyword để dễ tái sử dụng cũng như customize. Để dễ quản lý test case, em sẽ chia thành 3 tầng.

* Tầng 1: Custom-keyword Function. Đây sẽ là tầng chứa các code hàm custom để dùng cho các test case. Các hàm trong đây thường sẽ không hard data, chủ yếu là hàm xử lý và các Xpath của UI.
* Tầng 2: Feature Test Case. Tầng này sẽ bao gồm các test case có các step sử dụng nhiều lần. Các test case trong tầng này sẽ chỉ bao gồm các step để đúng chức năng của Feature, không bao gồm việc đóng mở browser…
* Tầng 3: Keyword Test Case. Tầng này sẽ bao gồm các end-to-end test case. Mỗi test case là một luồng test độc lập với 1 bộ data riêng không bị impact bởi test case khác.

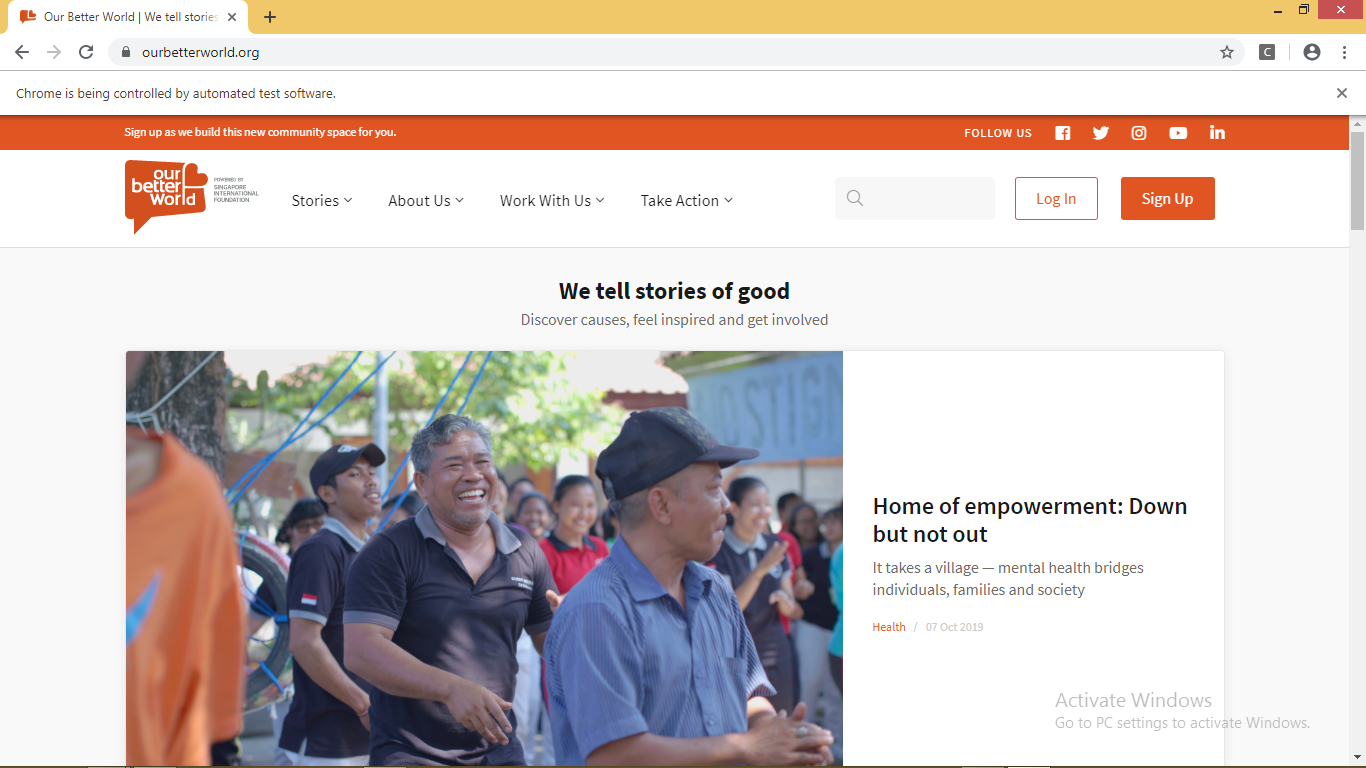
Riêng các test case cần chạy nhiều lần với nhiều bộ data khác nhau theo hướng data-driven, Katalon có cung cấp phần bind data trong Test Suites nên em sẽ dùng Test Suites để chạy.

## Hình ảnh demo

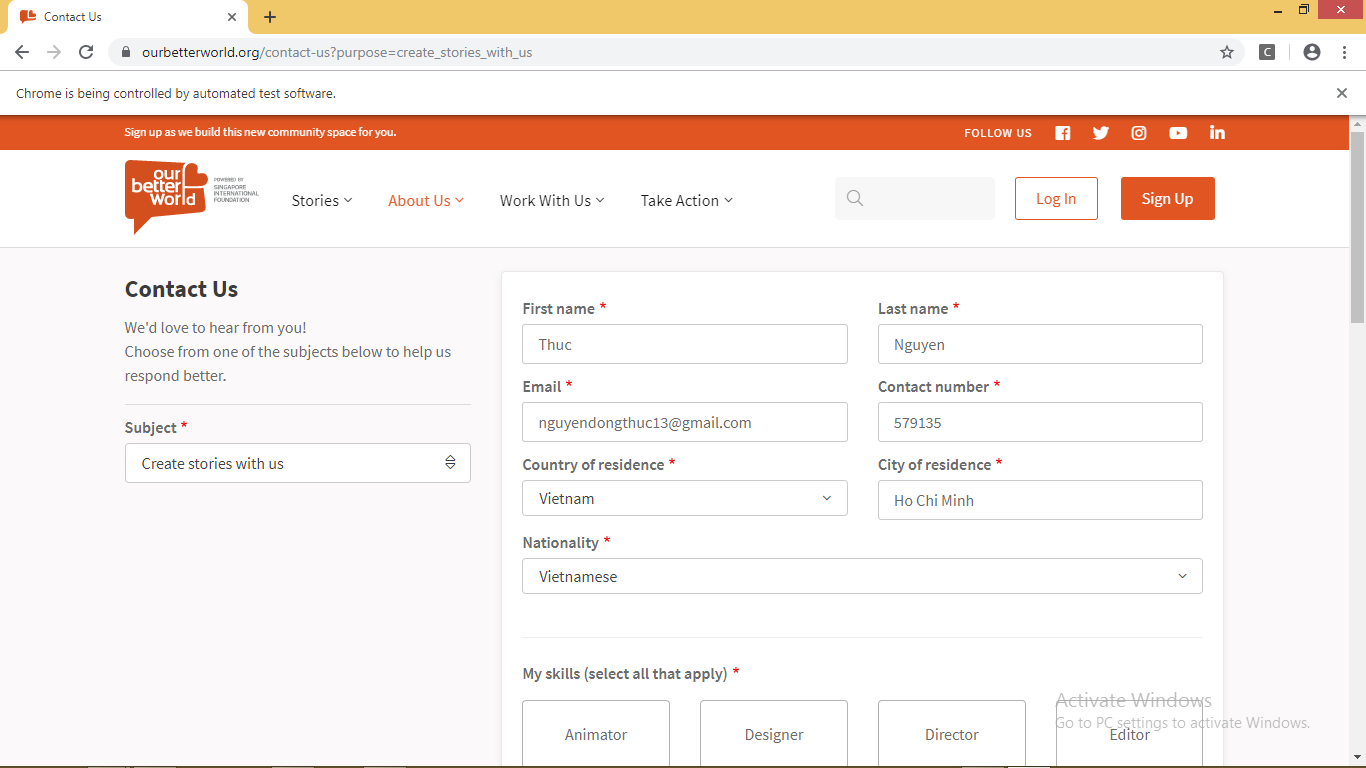
* Cấu trúc test case



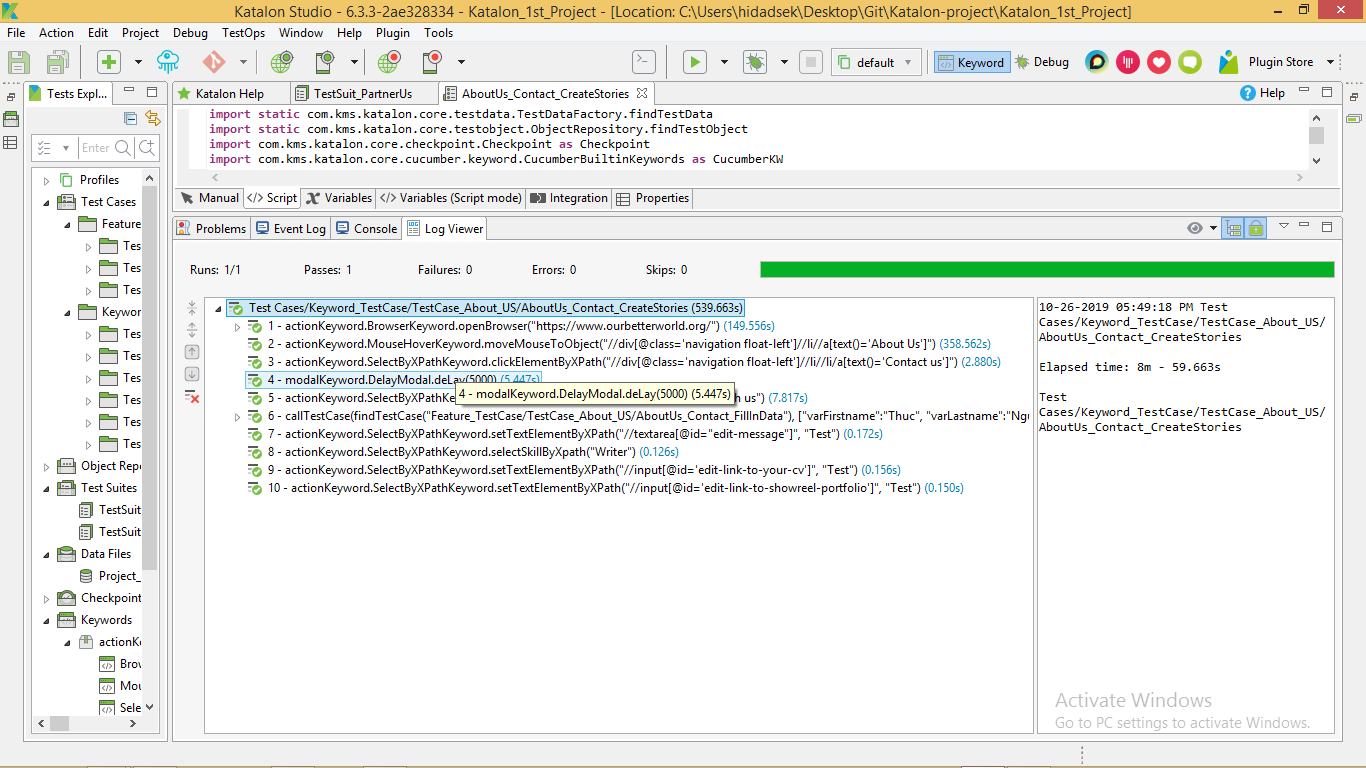
* Trang web our better world



* Automate testing



* Kết quả trả về



# Kinh nghiệm và bài học

Trong quá trình làm đồ án, nhờ sự giúp đỡ của thầy cô, em đã rút được kinh nghiệm:

* Em đã biết cách thiết kế một frame work đơn giản để chạy automation test.
* Em đã nghiên cứu và hoàn thiện các kỹ năng chạy test như bắt xpath, code… cũng như xử lý các tình huống khó khăn trong testing.

Đồng thời em cũng đã thấy được các mặt hạn chế của bản thân cũng như hạn chế Katalon:

* Các test case em viết còn sơ sài, có nhiều trường hợp em không thể nghĩ ra được cách giải quyết như việc thoát qua được capcha…
* Katalon cũng còn rất nhiều thiếu sót. Các biến mà Katalon hỗ trợ vẫn còn ít chưa hỗ trợ được các tình năng như code expression. UI của Katalon cũng còn khá đơn giản không phù hợp với những người không quen việc coding.

Kết luận

Kết luận lại, em nghĩ kiểm thử tự động (automation testing) là một bước đột phá lớn trong ngành kiểm thử. Như trong phần demo, nếu phải kiểm thử thủ công (manual testing) thì em sẽ mất rất nhiều thời gian và công sức để chạy mỗi ngày, trong khi đó với kiểm thử tự thì em chỉ cần để máy chạy và có thể làm các công việc khác, tiết kiểm được công sức, chi phí và thời gian.Vớinhưng ưu điểm trên, em tin là kiểm thử tự đông sẽ ngày càng phát triển và sẽ là một kỹ năng cần có của mỗi kiểm thử viên trong tương lai.