Trường Đại Học Công Nghiệp TP.HCM

Khoa Công Nghệ Thông tin

Báo cáo môn học

ĐẢM BẢO CHẤT LƯỢNG VÀ KIỂM THỬ PHẦN MỀM

Tên đề tài:

Tìm hiểu công cụ hỗ trợ kiểm thử.

Nhóm: 8

1. Nguyễn Thị Nga
2. Nguyễn Đa Nghiêm
3. Nguyễn Đăng Nguyên
4. Phan Nguyễn Khôi Nguyên
5. Phạm Thị Thùy Nhi

MỤC LỤC

CHƯƠNG 1. MỞ ĐẦU

Trong phát triển phần mềm hiện đại, việc đảm bảo chất lượng phần mềm là một yếu tố then chốt để tạo ra các sản phẩm ổn định và đáp ứng yêu cầu người dùng. Để đạt được điều này, các công cụ kiểm thử tự động đã trở thành công cụ hỗ trợ không thể thiếu trong việc kiểm tra và xác minh các chức năng của phần mềm.

Bốn công cụ kiểm thử phổ biến được sử dụng rộng rãi trong cộng đồng phát triển phần mềm là **unittest**, **pytest**, **Selenium**, và **Behave**. Mỗi công cụ này có những ưu điểm riêng biệt và hỗ trợ kiểm thử ở các mức độ khác nhau, từ kiểm thử đơn vị đến kiểm thử giao diện người dùng (UI) và kiểm thử hành vi (BDD). Mục tiêu của bài tiểu luận này là phân tích và so sánh bốn công cụ trên, từ đó hiểu rõ hơn về cách chúng hỗ trợ các quy trình kiểm thử phần mềm trong các dự án phát triển phần mềm hiện đại.

CHƯƠNG 2. NỘI DUNG

1. Giới thiệu

1.1. Unittest

Unittest là một framework kiểm thử được tích hợp sẵn trong Python, giúp lập trình viên dễ dàng viết và thực hiện các kiểm thử đơn vị (unit tests). Là một phần của thư viện chuẩn của Python, unittest hỗ trợ viết các bài kiểm thử đơn giản và tích hợp vào quy trình phát triển phần mềm.

***Dùng để làm gì**:*

unittest cung cấp cơ chế tự động hóa kiểm thử các chức năng trong chương trình Python. Công cụ này giúp đảm bảo rằng các thành phần của ứng dụng hoạt động đúng đắn, đặc biệt trong giai đoạn phát triển và bảo trì phần mềm.

***Của công ty nào**:*

Phát triển bởi Python Software Foundation (PSF).

***Download ở đâu**:*

Là một phần của Python, vì vậy không cần tải riêng biệt, chỉ cần cài đặt Python là có thể sử dụng unittest.

***Bản quyền**:*

Python và unittest là mã nguồn mở, theo giấy phép Python Software Foundation License.

1.2. Pytest

Pytest là một framework kiểm thử mã nguồn mở mạnh mẽ và phổ biến trong Python, giúp các lập trình viên viết và thực thi các bài kiểm thử (test) một cách dễ dàng và hiệu quả

Với pytest, người dùng có thể dễ dàng kiểm tra mã nguồn và phát hiện lỗi một cách nhanh chóng. Công cụ này hỗ trợ nhiều loại bài kiểm tra khác nhau, từ kiểm thử đơn vị, kiểm thử tích hợp cho đến kiểm thử hệ thống.

Một trong những điểm mạnh của pytest là cú pháp đơn giản và dễ hiểu, giúp lập trình viên tiết kiệm thời gian trong việc viết và duy trì các bài kiểm tra. Điều này

giúp tăng cường năng suất và giảm bớt gánh nặng cho các nhóm phát triển phần mềm.

Công ty sở hữu

Pytest là một dự án mã nguồn mở và **không thuộc sở hữu của bất kỳ công ty nào**. Ban đầu, nó được phát triển bởi **Holger Krekel** và hiện tại được duy trì bởi cộng đồng lập trình viên trên toàn thế giới.

Bản quyền

Pytest được phát hành dưới Giấy phép MIT (MIT License), một giấy phép mã nguồn mở cho phép người dùng tự do sử dụng, sao chép, sửa đổi, hợp nhất, xuất bản và phân phối phần mềm. Điều này có nghĩa là chúng ta có thể sử dụng pytest trong các dự án cá nhân và thương mại mà không gặp phải

1.3. Selenium

Selenium là một công cụ tự động hóa thử nghiệm phần mềm được sử dụng để kiểm tra ứng dụng web. Nền tảng cho phép người dùng tạo và thực thi các kịch bản thử nghiệm tự động để kiểm tra tính năng của các ứng dụng web trên nhiều trình duyệt hoặc nhiều nền tảng khác nhau.

Selenium cung cấp nhiều ngôn ngữ lập trình giúp cho việc viết và triển khai các kịch bản thử nghiệm trở nên linh hoạt hơn. Bên cạnh đó, Selenium còn hoạt động trên nhiều nền tảng khác nhau, bao gồm:

- Trình duyệt Web: Chrome, Firefox, Internet Explorer, Edge, Safari và Opera.
- Hệ điều hành: Windows, macOS và Linux.
- Ngôn ngữ lập trình: Java, C#, Python, Ruby, JavaScript (sử dụng Node.js), PHP và Perl.

Người/Công ty phát triển

Selenium được phát triển bởi Jason Huggins vào năm 2004

Download ở đâu?

Có thể tải Selenium trên trang chủ của dự án tại https://www.selenium.dev/downloads/.

Bản quyền

Selenium được phát hành theo giấy phép Apache 2.0, cho phép người dùng tự do sử dụng, sao chép, sửa đổi và phân phối mã nguồn

1.4. Behave

Behave là một framework kiểm thử theo mô hình BDD (Behavior Driven Development), giúp các nhóm phát triển phần mềm hợp tác chặt chẽ với các bên không phải lập trình viên để viết kiểm thử trong các dự án phần mềm.

Dùng để làm gì:

Viết kiểm thử theo cách BDD bằng ngôn ngữ tự nhiên, giúp các bên không phải lập trình viên dễ dàng tham gia vào quá trình kiểm thử.

Của công ty nào:

Behave là mã nguồn mở, phát triển bởi cộng đồng.

Download ở đâu:

Tải về từ PyPI.

Bản quyền:

Mã nguồn mở, giấy phép MIT.

2. Chức năng

2.1. Unitest

2.1.1. Chức năng

- Kiểm thử đơn vị các chức năng trong Python.
- Tích hợp với các công cụ CI/CD (Continuous Integration/Continuous Deployment).
- Tự động hóa kiểm thử khi thay đổi mã nguồn.

2.1.2. Ưu điểm

- Tích hợp sẵn trong Python, không cần cài đặt thêm.
- Dễ sử dụng với cú pháp đơn giản.
- Tính tương thích cao, dễ dàng tích hợp vào các hệ thống phát triển phần mềm hiện đại.

2.2. Pytest

2.2.1. Chức năng

- PyTest thích hợp cho việc kiểm thử tự động trong các dự án Python, bao gồm ứng dụng web, dịch vụ web, và các thư viện Python. Mục đích sử dụng PyTest là viết và thực thi các bài kiểm tra tự động để đảm bảo tính đúng đắn, chức năng và hiệu suất của mã nguồn Python. PyTest có thể được sử dụng để kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hệ thống trong quy trình phát triển phần mềm.
- Kiểm thử theo nhóm và tùy chỉnh: pytest cho phép chạy một nhóm các test case hoặc chỉ định các test case cụ thể thông qua các dấu hiệu (markers) hoặc các tùy chọn dòng lệnh. Điều này giúp bạn dễ dàng quản lý các bài kiểm thử phức tạp.
- Hỗ trợ fixture: pytest hỗ trợ "fixtures" giúp bạn chuẩn bị môi trường cho các test case (ví dụ: kết nối cơ sở dữ liệu, thiết lập dữ liệu ban đầu). Fixtures được thiết kế để tái sử dụng, làm cho mã kiểm thử trở nên sạch và rõ ràng hơn.
- Xử lý ngoại lệ và kiểm tra lỗi: pytest có khả năng kiểm tra các ngoại lệ và lỗi, giúp đảm bảo các chức năng hoạt động đúng ngay cả khi xảy ra lỗi.
- Tạo báo cáo kiểm thử chi tiết: pytest hỗ trợ tạo báo cáo chi tiết về các kết quả kiểm thử, giúp bạn dễ dàng xem những gì đã thành công hay thất bại.
- Hỗ trợ cho plugin: pytest có một hệ sinh thái plugin phong phú (như pytest-django cho Django, pytest-flask cho Flask), giúp tích hợp với các framework khác và mở rộng chức năng của pytest theo nhu cầu.
- Kiểm thử song song: Với các plugin như pytest-xdist, bạn có thể chạy các test case song song để tiết kiệm thời gian kiểm thử.

2.2.2. Ưu điểm

- Fixtures: Đây là cơ chế cho phép thiết lập và dọn dẹp trước và sau khi thực thi các bài kiểm thử. Fixtures giúp bạn tạo ra các điều kiện kiểm thử trước cho nhiều bài kiểm tra khác nhau, giảm thiểu mã lặp lại.
- Parametrization: Tính năng này cho phép bạn dễ dàng chạy cùng một bài kiểm thử với nhiều tập dữ liệu khác nhau, giúp tăng cường khả năng kiểm thử mà không cần phải tạo nhiều hàm kiểm thử riêng biệt.

- Tích hợp với các công cụ khác: pytest có thể dễ dàng tích hợp với các công cụ như coverage (đo lường độ bao phủ test), tox (kiểm tra đa phiên bản Python), và nhiều công cụ khác.
- Báo cáo chi tiết và tùy chỉnh: pytest hỗ trợ việc tạo ra báo cáo chi tiết và có thể cấu hình cho các nhu cầu cụ thể của dự án bạn.
- Hỗ trợ kiểm thử tự động và CI/CD: pytest tích hợp tốt với các công cụ CI/CD như Jenkins, GitLab CI, và GitHub Actions, giúp tự động hóa kiểm thử dễ dàng.
- Được cộng đồng rộng rãi sử dụng: pytest có một cộng đồng lớn và tài liệu phong phú, dễ dàng tìm thấy tài liệu và hỗ trợ khi gặp vấn đề.
- Dễ sử dụng và học: pytest có cú pháp đơn giản, dễ hiểu, không cần nhiều thiết lập. Người mới bắt đầu cũng có thể nhanh chóng làm quen và viết test case hiệu quả.
- Cú pháp ngắn gọn, rõ ràng: Không yêu cầu cấu trúc phức tạp như các framework khác, pytest cho phép bạn viết test case ngắn gọn hơn, giúp mã kiểm thử dễ đọc và bảo trì.

2.3. Selenium

2.3.1. Chức năng

Tối ưu cho ứng dụng web

Phiên bản Selenium mới nhất được tối ưu hóa để cung cấp các tính năng và công cụ cần thiết trong việc kiểm thử, tương tác với ứng dụng web. Cụ thể, Selenium đưa ra các phương pháp và hàm để tương tác với các thành phần web như textbox, button, dropdown…

Ngoài ra, Selenium cho phép kiểm tra tính đúng đắn và hiệu suất của các tính năng trên ứng dụng web bằng việc thực thi các kịch bản thử nghiệm tự động. Điều này đảm bảo rằng ứng dụng web có thể hoạt động một cách mượt mà trên nhiều trình duyệt và nền tảng khác nhau.

Hỗ trợ đa nền tảng

Selenium được thiết kế để hỗ trợ nhiều nền tảng khác nhau. Trong đó bao gồm các trình duyệt web phổ biến như Chrome, Firefox, Internet Explorer, Edge, Safari và Opera trên các hệ điều hành Windows, MacOS và Linux. Người dùng có thể kiểm

thử tính đúng đắn và khả năng tương thích của ứng dụng web trên nhiều môi trường khác nhau một cách toàn diện hơn.

Ngôn ngữ lập trình đa dạng

Selenium hỗ trợ nhiều ngôn ngữ lập trình phổ biến như Java, C#, Python, Ruby, JavaScript (sử dụng Node.js), PHP và Perl. Công nghệ cho phép người dùng lựa chọn ngôn ngữ lập trình ưa thích để viết các kịch bản thử nghiệm. Từ đó tạo điều kiện thuận lợi cho việc sử dụng và triển khai Selenium trong môi trường phát triển phần mềm.

Tích hợp với các công cụ kiểm thử

Selenium được tích hợp tốt với các công cụ quản lý kiểm thử như JUnit và TestNG cũng như các công cụ CI/CD là Jenkins. Hoạt động này kiến tạo quy trình kiểm thử tự động liền mạch và hiệu quả. Thông qua cách kết hợp Selenium vào quy trình kiểm thử và triển khai dự án phần mềm.

Selenium Grid

Là một tính năng mạnh mẽ của Selenium cho phép người dùng thực thi song song các kịch bản thử nghiệm trên nhiều trình duyệt và nền tảng khác nhau. Điều này làm tăng hiệu suất và giảm thời gian kiểm thử bằng cách chạy các thử nghiệm đồng thời trên nhiều máy. Selenium Grid có khả năng mở rộng và đamt bảo tính nhất quán trong việc kiểm thử phần mềm.

2.3.2. Ưu điểm

- Dễ sử dụng: Selenium IDE có giao diện dễ sử dụng và không đòi hỏi về kiến thức kỹ thuật sâu. Người mới bắt đầu với việc kiểm thử tự động có thể ứng dụng công nghệ một cách vô cùng tiện lợi.
- Tạo kịch bản nhanh: Selenium IDE cho phép người dùng dễ dàng ghi lại các hành động trên trình duyệt và tạo kịch bản kiểm thử một cách nhanh chóng.

- Hỗ trợ nhiều trình duyệt: Selenium IDE hỗ trợ nhiều trình duyệt phổ biến như Chrome và Firefox.
- Là công cụ miễn phí: Selenium IDE là một công cụ mã nguồn mở, miễn phí cho việc sử dụng.

2.4. Behave

2.3.1. Chức năng

- Diễn giải tính năng: Các tệp .feature mô tả các chức năng của phần mềm từ góc độ người dùng, thường sử dụng cú pháp Gherkin (với các từ khóa như Given, When, Then).
- Chuyển đổi ngôn ngữ tự nhiên thành mã kiểm thử: Behave tự động liên kết các câu lệnh trong kịch bản với các hàm Python được định nghĩa trước. Mỗi câu trong kịch bản sẽ tương ứng với một phương thức hoặc hàm trong mã.
- Tích hợp BDD: Tạo môi trường giao tiếp dễ dàng giữa các bộ phận như phát triển, kiểm thử, và quản lý sản phẩm, bằng cách sử dụng ngôn ngữ tự nhiên để mô tả các trường hợp kiểm thử.
- Chạy kịch bản kiểm thử: Behave có thể tự động chạy tất cả các kịch bản kiểm thử hoặc chỉ chạy một phần tùy chọn. Điều này giúp phát hiện sớm các lỗi và cải tiến tính ổn định của phần mềm.

2.3.2. Ưu điểm

- Ngôn ngữ tự nhiên: Behave sử dụng ngôn ngữ tự nhiên trong các kịch bản kiểm thử, giúp cho cả những người không phải lập trình viên (như khách hàng hoặc người quản lý sản phẩm) có thể đọc và hiểu các kịch bản kiểm thử. Điều này rất hữu ích trong môi trường phát triển Agile, nơi các bên liên quan không phải lúc nào cũng có kiến thức kỹ thuật.
- Cú pháp dễ tiếp cận: Cú pháp Gherkin, được sử dụng trong Behave, cho phép mô tả các yêu cầu và hành vi mong đợi của hệ thống theo các bước dễ hiểu như "Given", "When", "Then". Điều này tạo điều kiện cho việc hợp tác giữa các nhóm kỹ thuật và phi kỹ thuật..

3. Cài đặt

3.1. Unitest

Cài đặt Python (vì `unittest` đã có sẵn trong Python).
Môi trường triển khai: Tất cả các hệ điều hành hỗ trợ Python.

3.2. Pytest

Cách 1: Chạy lệnh này trong terminal hoặc command prompt

Cài đặt pytest là một bước đơn giản và có thể được thực hiện dễ dàng thông qua pip, công cụ quản lý gói của Python :

pip install pytest

Sau khi cài đặt thành công, bạn có thể kiểm tra phiên bản pytest đã được cài đặt bằng cách sử dụng lệnh:

pytest --version

Nếu bạn thấy phiên bản hiện lên, điều đó có nghĩa là pytest đã được cài đặt thành công trong môi trường lập trình của bạn.

Cách 2:

Tải xuống từ trang PyPI của pytest: Trên trang này, bạn có thể tải gói pytest dưới dạng file nén. Sau khi tải xuống, bạn cần giải nén và cài đặt thủ công. Tuy nhiên, cách này ít được dùng vì pip install đơn giản hơn.

3.3. Selenium

Bước 1: Cài đặt ngôn ngữ lập trình: Đảm bảo đã cài đặt môi trường phát triển ngôn ngữ bạn muốn dùng (ví dụ: Python, Java).

Bước 2: Cài đặt thư viện Selenium:

- Với Python, chạy lệnh: pip install selenium
- Với Java, tải thư viện Selenium từ Maven Central hoặc Selenium trang chủ.

Bước 3: Cài đặt trình điều khiển : Tùy thuộc vào trình duyệt, tải driver tương ứng (như ChromeDriver, GeckoDriver cho Firefox).

Môi trường triển khai:

Selenium có thể triển khai trên các hệ điều hành Windows, macOS và Linux. Selenium Grid còn hỗ trợ chạy thử nghiệm phân tán trên nhiều máy tính, cho phép kiểm thử đồng thời.

3.4. Behave

Cài đặt Python: Behave yêu cầu Python 3.x, nên bạn cần đảm bảo máy tính đã cài đặt Python.
Cài đặt Behave: Behave có thể dễ dàng cài đặt thông qua pip
Cấu hình thư mục và tệp: Behave yêu cầu một cấu trúc thư mục nhất định cho các tệp .feature và mã kiểm thử Python (thường đặt trong thư mục features/steps).
Thiết lập các thư viện phụ trợ: Nếu kịch bản kiểm thử có yêu cầu cụ thể (như Selenium cho kiểm thử trình duyệt), bạn sẽ cần cài đặt thêm các thư viện đó bằng pip.

Behave hoạt động tốt trong môi trường phát triển tích hợp như PyCharm, Visual Studio Code, và cũng hỗ trợ tích hợp với các công cụ CI/CD như Jenkins để tự động hóa các kịch bản kiểm thử trong quy trình phát triển phần mềm.

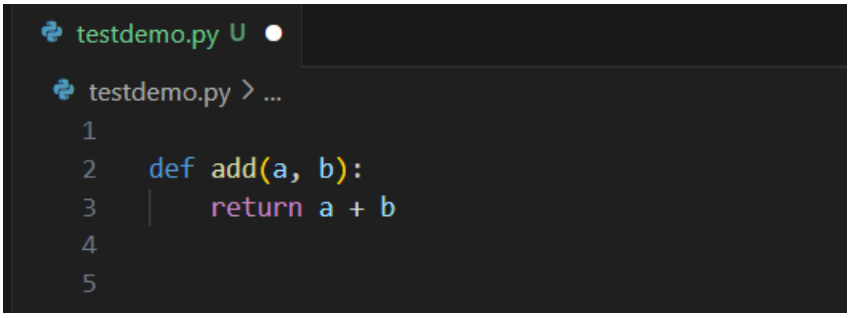
4. Hướng dẫn sử dụng

3.1. UnitTest

```
import unittest
class TestExample(unittest.TestCase):
    def test_addition(self):
            self.assertEqual(1 + 1, 2)
if __name__ == "__main__":
    unittest.main().
```

3.2. Pytest

Ví dụ chúng ta có file testdemo.py chứa hàm add. Và sau đây là cách chúng ta sẽ test hàm add này :

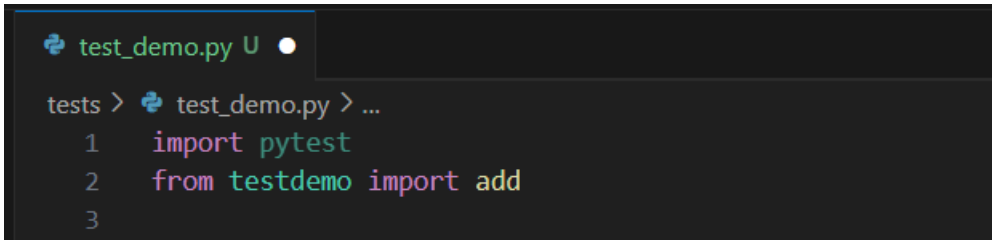


Bước 1 : Tạo file để viết test pytest

Lưu ý : Theo mặc định thì pytest sẽ chỉ nhận dạng được các file có tên bắt đầu với **test_** hoặc kết thúc với **_test** .Ví dụ: test_app.py. Còn đối với các phương thức thì yêu cầu tên của các phương thức bắt đầu bằng "**test**", những phương thức có tên khác sẽ bị bỏ qua.Ví dụ: test_add_task.

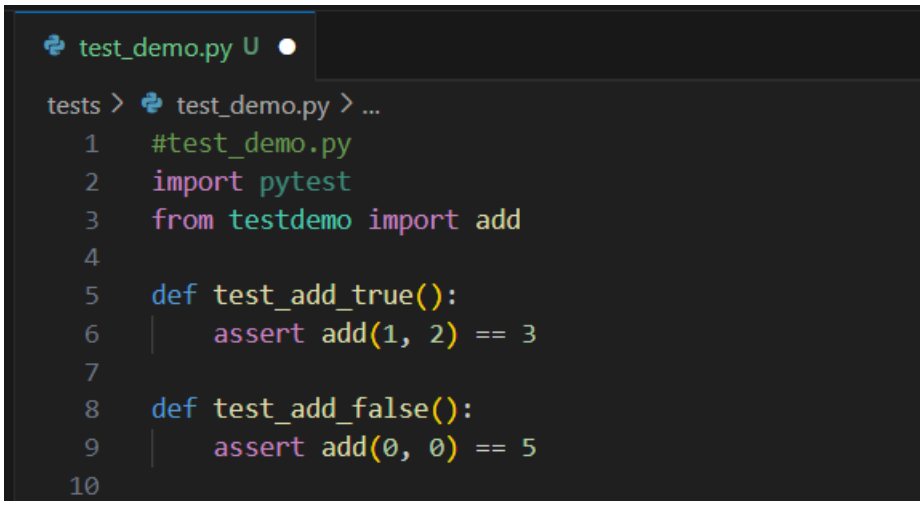
Bước 2 : Import thư viện vào file test

Bạn cần import pytest và import file chứa các phương thức muốn test theo cú pháp **(from tenFile import tenTuDat)**. Như bên dưới :



Ngoài ra bạn có thể import các thư viện khác hỗ trợ cho việc test.

Bước 3 : Viết test



Trong pytest bạn sẽ kiểm tra bằng câu lệnh assert . Có rất nhiều dạng như :

- Assert Cơ Bản : assert 1 + 1 == 2
- Assert Kiểm Tra Trạng Thái : assert x is True
- Assert Kiểm Tra Giá Trị : assert x != "world" (kiểm tra biến x có thể là từ "world")
- Assert Kiểm Tra Danh Sách : assert 2 in my_list (kiểm tra số 2 có trong mảng my_list)
- Assert Kiểm Tra Đối Tượng : assert person1 != person2

Ngoài ra còn có kiểm tra lỗi, kiểu dữ liệu, độ lớn, độ chính xác, độ tương đương ,....

Bước 4 : Chạy test

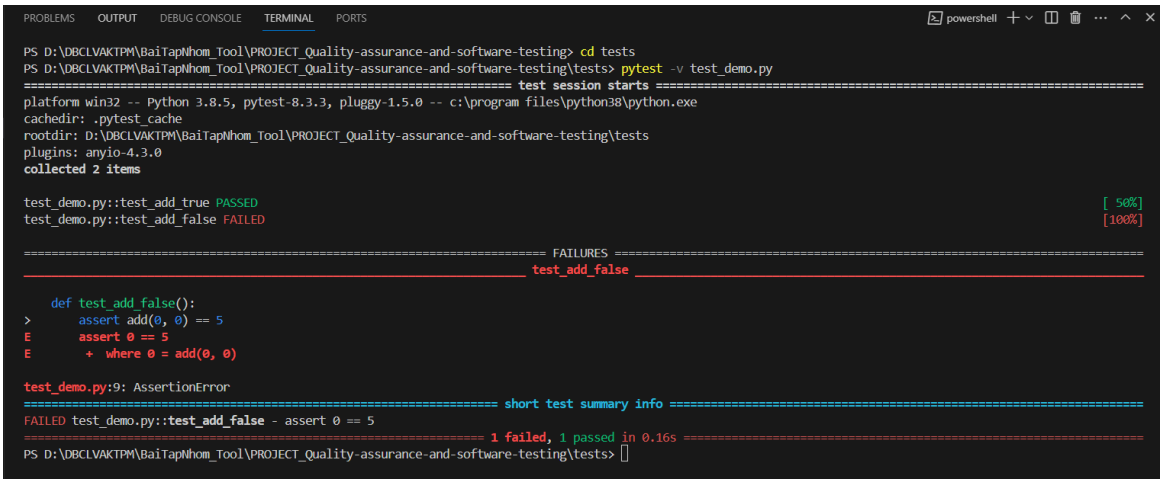
Mở terminal hoặc command prompt và điều hướng đến thư mục chứa các tệp của bạn. Sau đó, chạy lệnh sau để kiểm tra:

pytest -v tenFileTest.py

Ngoài ra nếu bạn muốn chạy hết toàn test các file test. Hãy chạy lệnh sau :

pytest

Bước 5 : Xem kết quả và phân tích



Trong khối đầu tiên là “test session starts” cho biết trạng thái hệ thống gồm: phiên bản Python, phiên bản pytest và các plugin được cài, đường dẫn rootdir hoặc đường dẫn pytest tìm file config và file để kiểm thử và số lượng hàm cần kiểm thử mà pytest tìm ra. Nếu hàm này test thành công sẽ có chữ PASSED còn hàm nào thất bại sẽ có chữ FAILED.

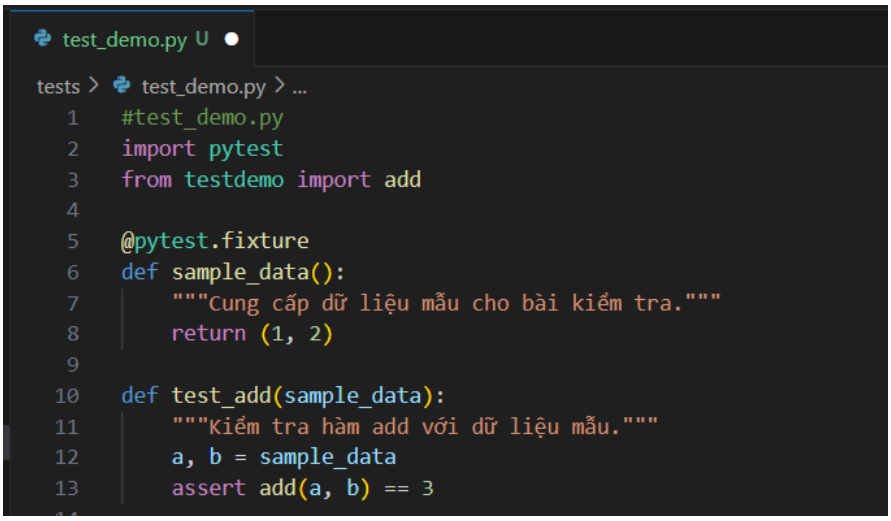
Với mỗi bài kiểm thử bị fail sẽ có thêm thông tin về hàm gây ra lỗi, được hiển thị trong khối FAILURES

Cuối cùng là chi tiết hơn về lỗi trong khối “short test sumary info” , hỗ trợ lập trình viên debug

Hướng dẫn nâng cao :

Trong một số trường hợp chúng ta có thể cần phải cài đặt hoặc cấu hình môi trường trước mới test .Thì pytest có hỗ trợ @pytest.fixture để giúp giải quyết vấn đề này:

Cách sử dụng : @pytest.fixture ở phía trên hàm muốn chỉ định.Như ảnh bên dưới :

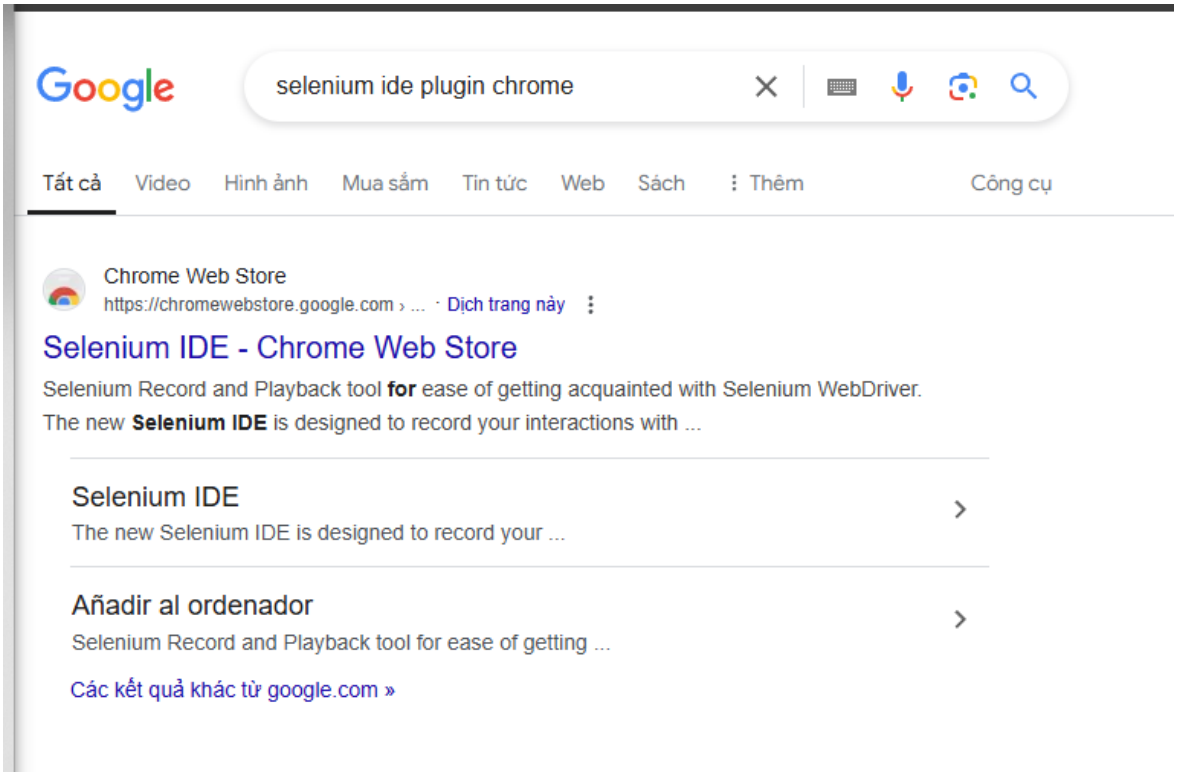


Ngoài ra có thể xác định phạm vi cho fixture để kiểm soát thời điểm nó được tạo ra và hủy bỏ.

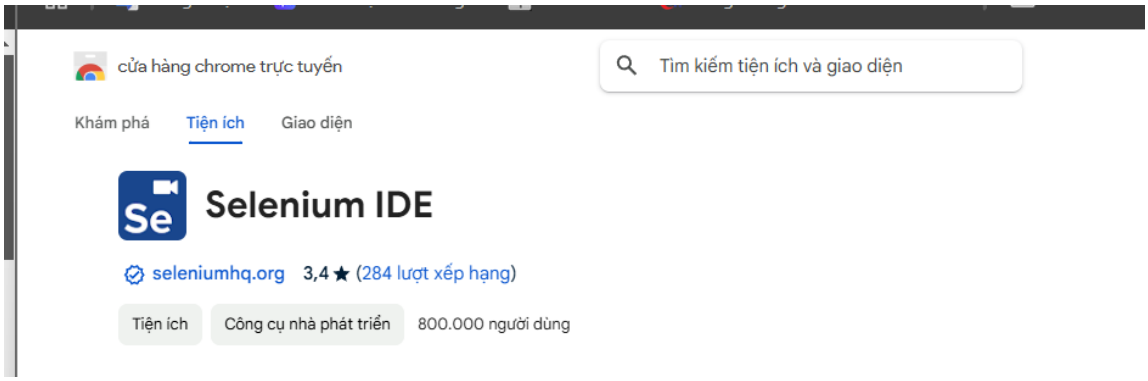
- scope="function" (mặc định): Fixture sẽ được tạo ra cho mỗi hàm kiểm tra.
- scope="class": Fixture sẽ được tạo ra một lần cho mỗi lớp kiểm tra.
- scope="module": Fixture sẽ được tạo ra một lần cho mỗi module (tệp).
- scope="session": Fixture sẽ được tạo ra một lần cho toàn bộ phiên kiểm tra.

3.3. Selenium

Bước 1: Cài đặt plugin Selenium từ Chrome Extention: *gõ từ khóa* ***"selenium ide plugin chrome"***



Bước 2. Add vào chrome



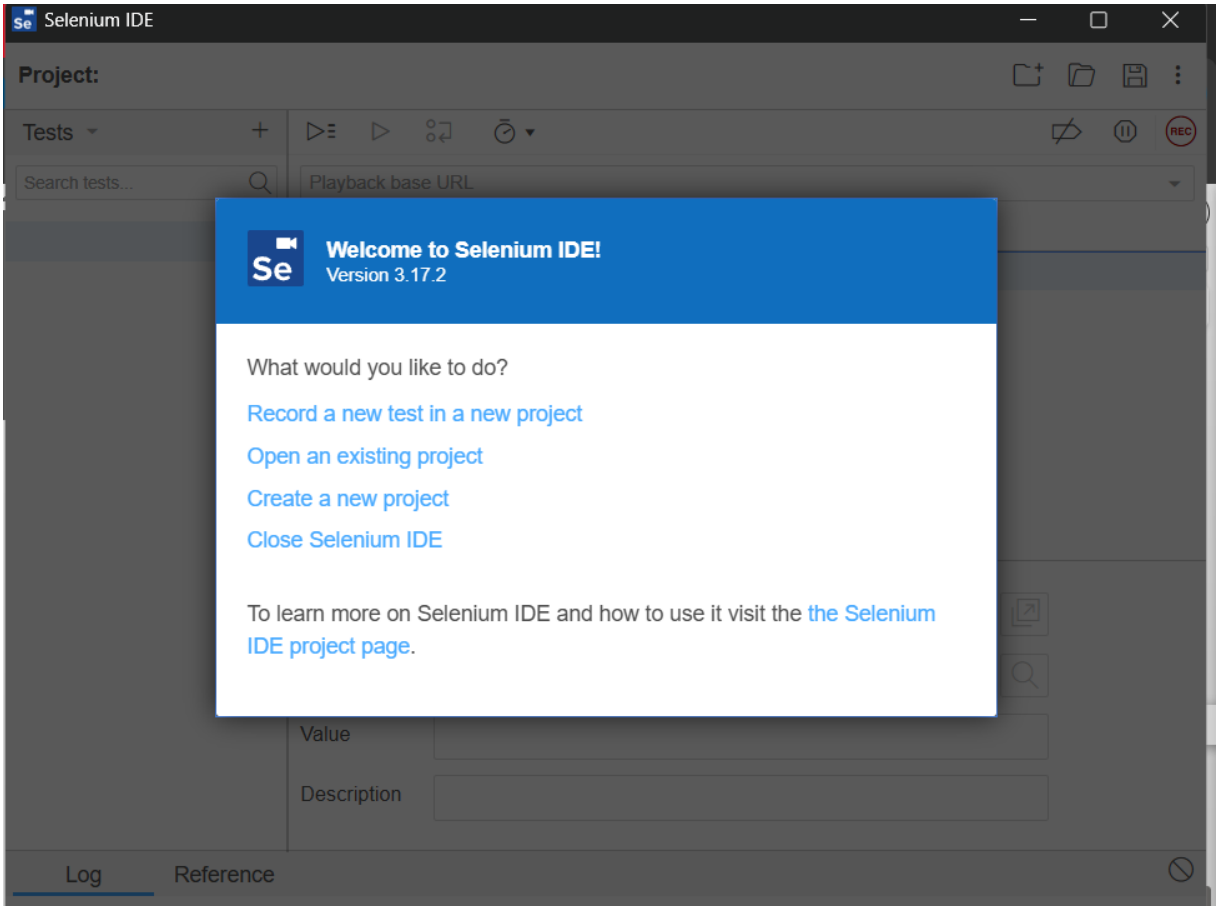
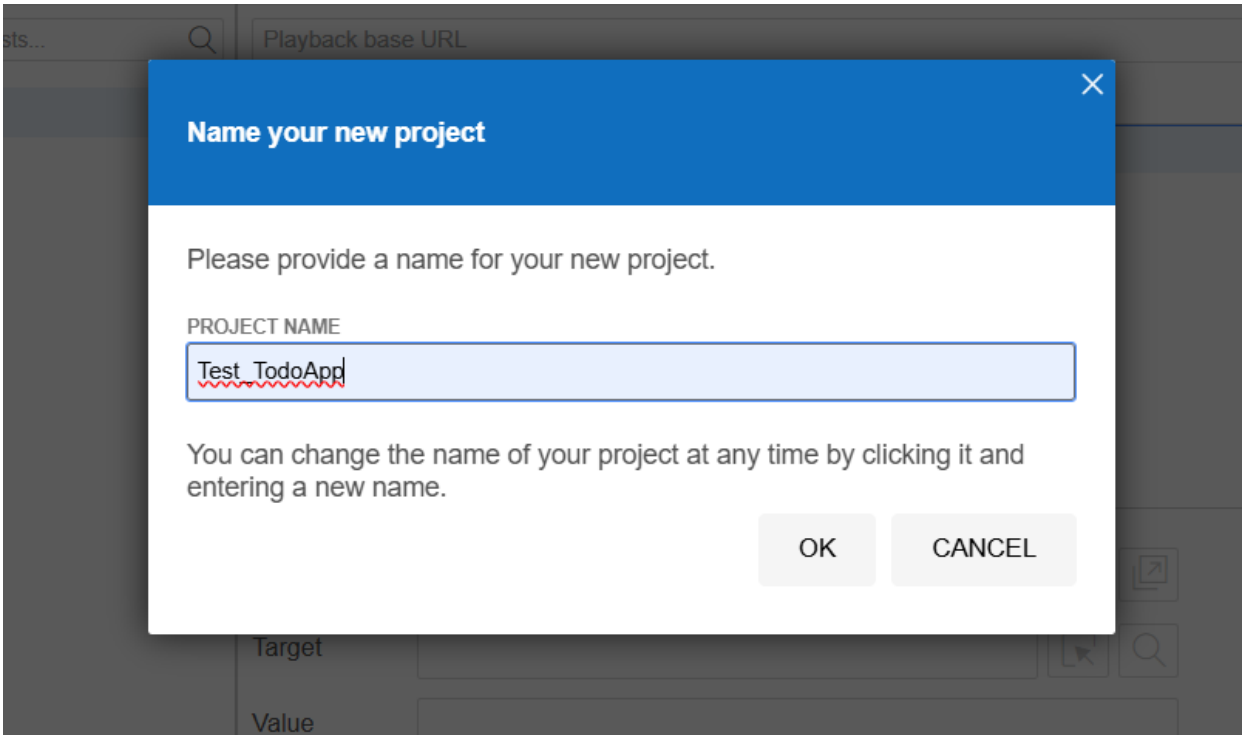
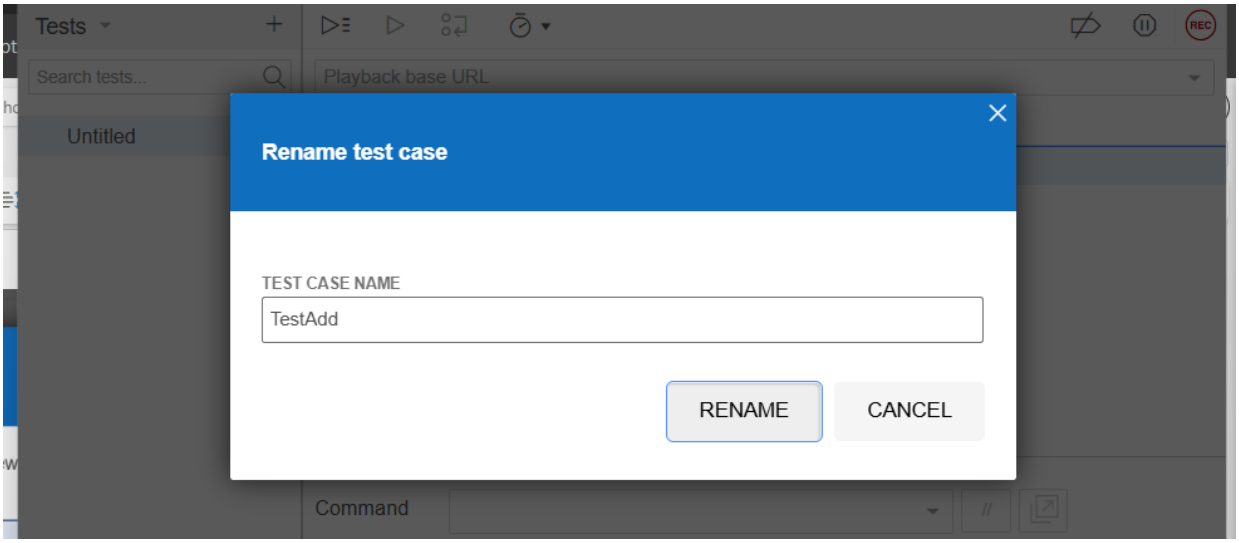


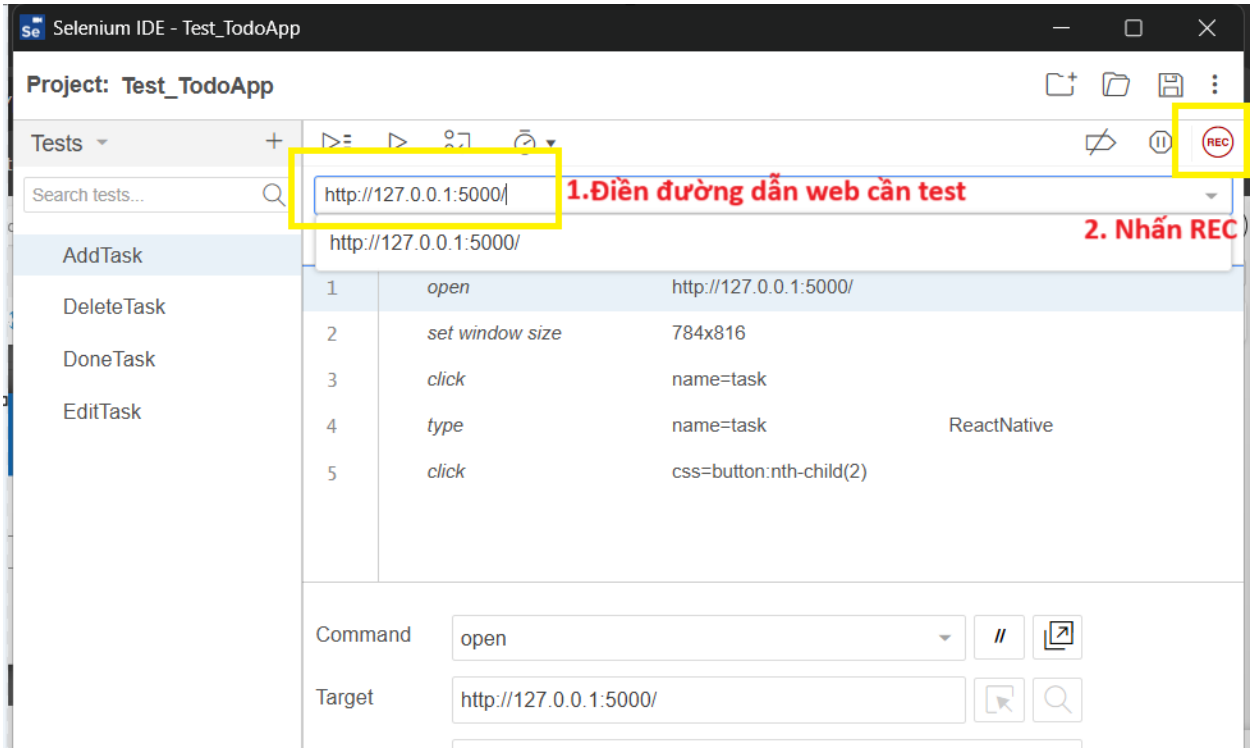
Figure 1: Giao diện selenium

Bước 3. Tạo project, test class

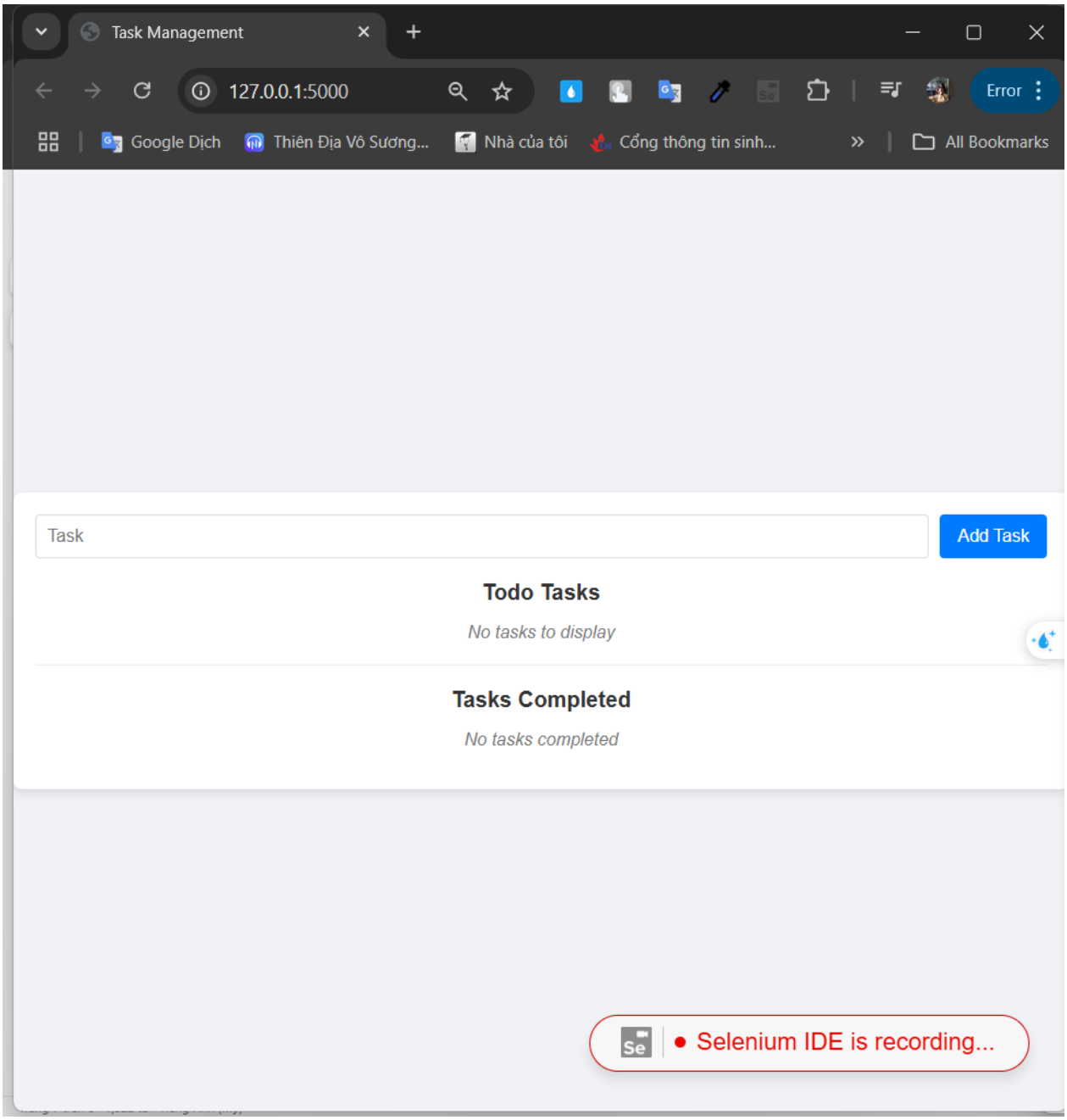
- Chọn "Create a new project" để tạo một project mới
- Có thể mở project có sẵn bằng cách chọn "Open an existing project"


Playback base URL
Name your new project
Please provide a name for your new project.
PROJECT NAME
Test_TodoApp
You can change the name of your project at any time by clicking it and entering a new name.
OK
CANCEL
Target
Value


Tests
Search tests...
Untitled
Playback base URL
Rename test case
TEST CASE NAME
TestAdd
RENAME
CANCEL
Command



Sau khi nhấn REC, giao diện web sẽ được mở và có thể thực hiện test



Thử add một task và tắt record

Ví dụ: Tạo test chức năng addTask với value "DBCLKTPM"

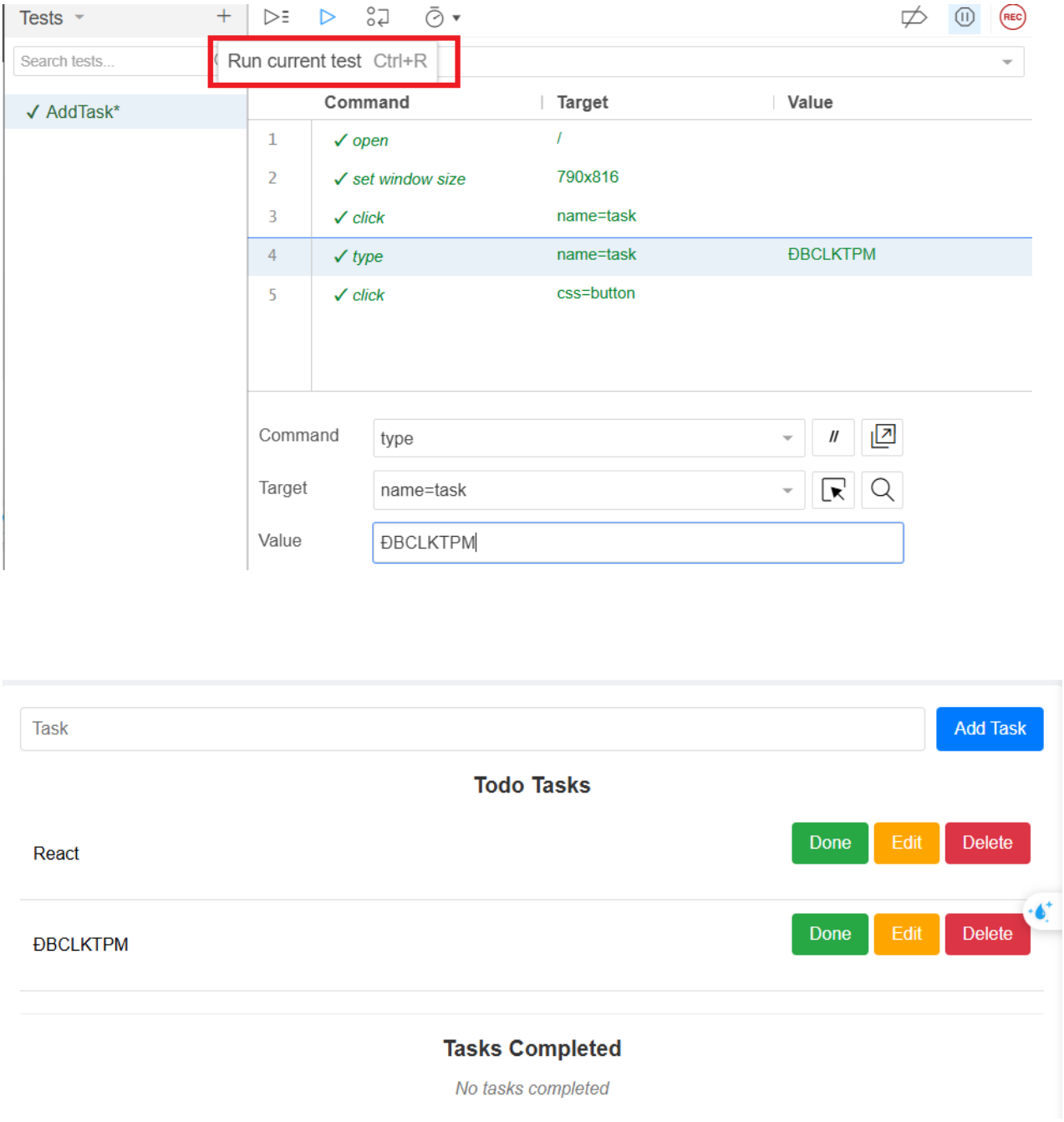
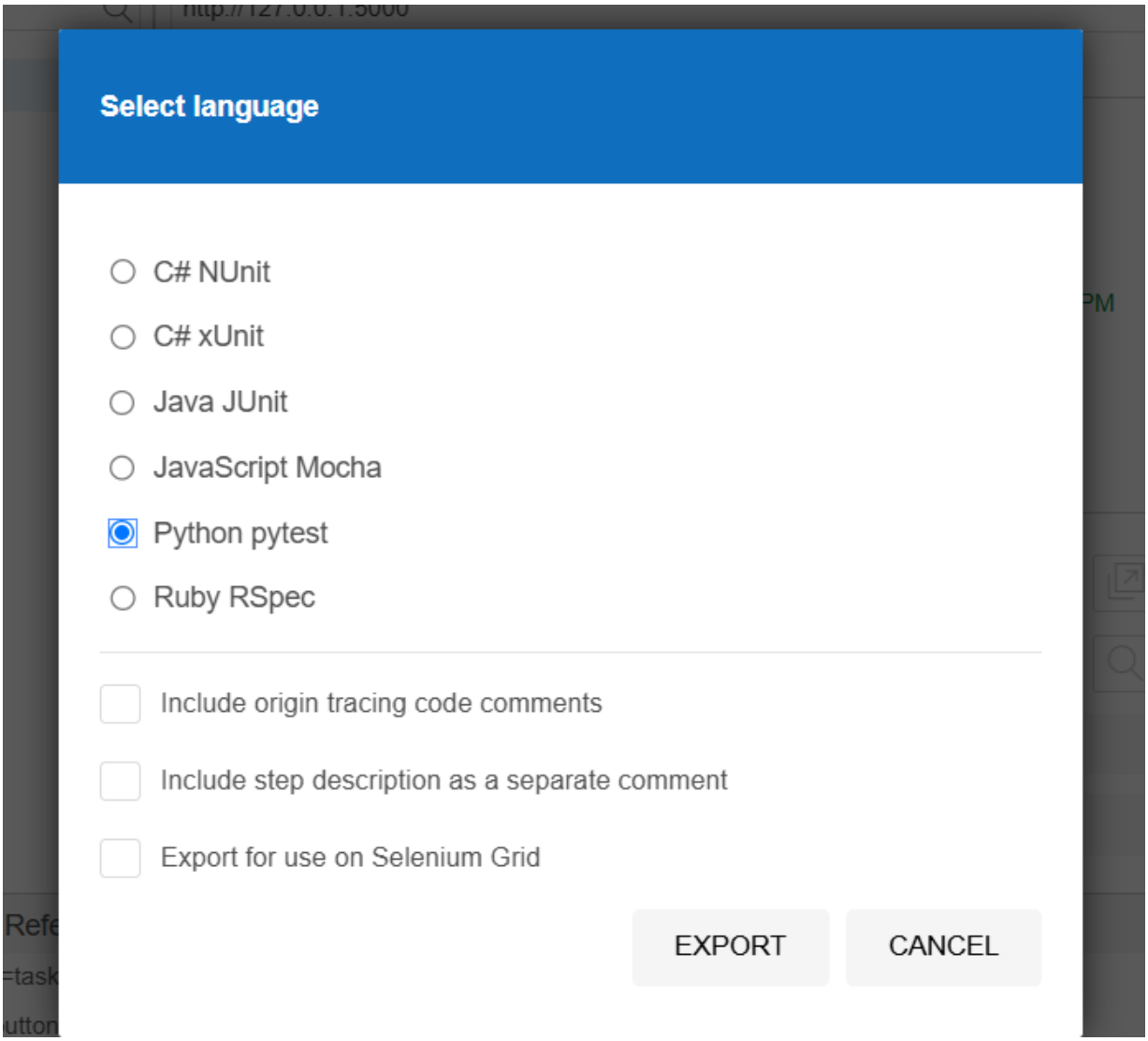


Figure 2 : Kết quả

Có thể export ra file



3.4. Behave

File .feature:

```
Feature: Showing off behave

  Scenario: run a simple test
    Given we have behave installed
    When we implement a test
    Then behave will test it for us!
```

File .py:

```
from behave import *

@given('we have behave installed')
```

```
def step_impl(context):
    pass

@when('we implement a test')
def step_impl(context):
    assert True is not False

@then('behave will test it for us!')
def step_impl(context):
    assert context.failed is False
```

5. Đánh giá công cụ tìm hiểu

3.1. Pytest

Tiêu chí	pytest	unittest	nose2
Mức độ phổ biến	Rất phổ biến, cộng đồng lớn và nhiều plugin	Phổ biến, đi kèm với Python	Ít phổ biến hơn, chủ yếu được sử dụng với các dự án cũ
Khả năng viết test	Hỗ trợ viết test linh hoạt, có thể không cần viết lớp (class)	Cần phải viết test trong lớp TestCase	Giống như unittest, nhưng linh hoạt hơn một chút
Fixture	Hỗ trợ fixture mạnh mẽ, có thể dùng decorator như @pytest.fixture	Có fixture nhưng hạn chế hơn	Có fixture, hỗ trợ tốt nhưng không đa dạng như pytest
Tích hợp plugin	Có rất nhiều plugin và dễ mở rộng	Ít plugin tích hợp hơn	Tích hợp plugin khá tốt, nhưng không phổ biến như pytest
Kiểm thử tham số (Parametrized testing)	Hỗ trợ tốt với decorator @pytest.mark.parametrize	Không có sẵn, cần tạo vòng lặp thủ công	Có hỗ trợ thông qua một số plugin
Báo cáo kết quả	Có báo cáo đơn giản, có thể mở rộng với plugin	Báo cáo kết quả cơ bản	Báo cáo đơn giản, có thể dùng plugin để mở rộng
Cấu trúc thư mục tự động	Tự động phát hiện file và thư mục theo cấu trúc dễ dàng	Cần cấu hình thêm để nhận diện tự động	Tự động phát hiện nhưng không mạnh bằng pytest

Thời gian học	Mất thời gian ban đầu cho tính năng nâng cao	Dễ học, tích hợp sẵn trong Python	Dễ học, phù hợp với người đã quen với unittest
Hỗ trợ kiểm thử nâng cao (e.g., E2E)	Không tối ưu cho E2E	Hạn chế	Hạn chế, chủ yếu cho unit test
Độ ổn định	Rất ổn định, cập nhật thường xuyên	Rất ổn định, có trong Python tiêu chuẩn	Ổn định nhưng ít cập nhật hơn
Cách cài đặt	Cài đặt thông qua pip install pytest	Được cài sẵn với Python	Cài đặt thông qua pip install nose2

3.2. Selenium

Tính năng	Selenium	Watir	SilkTest
Ngôn ngữ lập trình	Hỗ trợ đa ngôn ngữ (Java, Python, C#, Ruby, etc.)	Ruby là chính, nhưng có thể kết hợp với Selenium để hỗ trợ ngôn ngữ khác	Chủ yếu là Silk4J (Java), Silk4NET (C#), Silk4J (Ruby)
Trình duyệt hỗ trợ	Đa trình duyệt (Chrome, Firefox, Safari, Edge)	Đa trình duyệt (thông qua WebDriver)	Đa trình duyệt, hỗ trợ tốt cho cả Internet Explorer
Môi trường triển khai	Hỗ trợ đa nền tảng (Windows, macOS, Linux)	Chủ yếu trên Windows, nhưng có thể mở rộng với Selenium	Chủ yếu trên Windows, tích hợp tốt với các công cụ của Micro Focus
Mức độ sử dụng	Rất phổ biến, cộng đồng lớn, tài liệu phong phú	Được dùng chủ yếu bởi người dùng Ruby	Được sử dụng rộng rãi trong doanh nghiệp lớn
Khả năng mở rộng	Cao, có thể tích hợp với CI/CD và Selenium Grid	Mức độ mở rộng vừa phải, phù hợp với các dự án nhỏ và trung bình	Tích hợp tốt với các công cụ trong hệ sinh thái Micro Focus
Chi phí	Miễn phí (mã nguồn mở)	Miễn phí (mã nguồn mở)	Thương mại, có phí bản quyền và hỗ trợ

CHƯƠNG 3. KẾT LUẬN

Sau khi phân tích các công cụ kiểm thử **unittest**, **pytest**, **Selenium**, và **Behave**, chúng ta có thể nhận thấy mỗi công cụ có những đặc điểm và ưu điểm riêng phù hợp với các nhu cầu kiểm thử khác nhau trong quy trình phát triển phần mềm.

unittest và **pytest** là lựa chọn lý tưởng cho kiểm thử đơn vị và kiểm thử tích hợp, đặc biệt khi phát triển ứng dụng Python. Cả hai công cụ đều có khả năng dễ dàng cài đặt và triển khai, hỗ trợ việc kiểm tra các module và chức năng của phần mềm.

Selenium là công cụ mạnh mẽ cho kiểm thử giao diện người dùng (UI) trên các ứng dụng web, giúp kiểm thử tự động các tác vụ người dùng trên trình duyệt. Tuy nhiên, Selenium yêu cầu cài đặt phức tạp và môi trường triển khai chi tiết hơn so với unittest và pytest.

Behave, với phương pháp BDD, mang đến một cách tiếp cận dễ đọc và hiểu cho việc mô tả các yêu cầu và hành vi của phần mềm, từ đó tăng cường sự hợp tác giữa các đội ngũ kỹ thuật và phi kỹ thuật. Tuy nhiên, Behave cũng có một số thách thức trong việc triển khai và yêu cầu người sử dụng hiểu rõ phương pháp BDD.

Việc lựa chọn công cụ kiểm thử phù hợp phụ thuộc vào yêu cầu cụ thể của từng dự án và môi trường phát triển. Các công cụ này đều có những ưu điểm và nhược điểm riêng, và trong nhiều trường hợp, chúng có thể được kết hợp để tận dụng tối đa khả năng của từng công cụ. Tóm lại, việc lựa chọn công cụ kiểm thử đúng đắn sẽ giúp tăng cường chất lượng phần mềm, giảm thiểu lỗi và cải thiện hiệu quả trong quá trình phát triển phần mềm.