**BÁO CÁO BÀI TẬP LỚN KIỂM THỬ PHẦN MỀM**

**Giáo viên hướng dẫn:** Nguyễn Thị Ngọc Thanh

**Sinh viên thực hiện:** Lâm Huỳnh Chấn Nguyên (36)

**Mã số sinh viên**: 2151050284

**Khoa:** Công nghệ thông tin

**MỤC LỤC**

[NUNIT TEST TÍNH DIỆN TÍCH – CHU VI Hình Chữ Nhật 1](#_Toc166618902)

[Decision Tree: 1](#_Toc166618903)

[**Tạo môi trường** 1](#_Toc166618904)

[- Tạo giao diện tính diện tích hình chữ nhật bằng ngôn ngữ C# 1](#_Toc166618905)

[+ Sử dụng file Windows Forms App(.NET Framework) 1](#_Toc166618906)

[Viết code để xử lí tính toán trong chương trình 2](#_Toc166618907)

[Viết code xử lí nút tính Diện Tích và Chu Vi của chương trình 3](#_Toc166618908)

[Add Class, đặt tên là Rectangle, để hổ trợ tính toán 4](#_Toc166618909)

[Code trong Class Rectangle 5](#_Toc166618910)

[Tạo Project để kiểm thử phần mềm trên 5](#_Toc166618911)

[**Tại project Unit Test, thực hiện Add Reference để tham chiếu đến project cần thực hiện Unit Test** 8](#_Toc166618912)

[Sau khi Add Reference thành công sẽ hiển thị tên Project đã Add 8](#_Toc166618913)

[Phải Using Project trước khi code 8](#_Toc166618914)

[**Mô tả Unit test:** 9](#_Toc166618915)

[**Các Test Case của chức năng tính chu vi:** 9](#_Toc166618916)

[**Test Report của chức năng tính chu vi Hình chữ nhật:** 9](#_Toc166618917)

[**Các Test Case của chức năng tính diện tích:** 9](#_Toc166618918)

[**Test Report của chức năng tính diện tích Hình chữ nhật:** 10](#_Toc166618919)

[Viết Test Case cho 2 chức năng 10](#_Toc166618920)

[**Test bằng cách gán bộ dữ liệu 5,5 cho 2 test case** 10](#_Toc166618921)

[Kết quả test 11](#_Toc166618922)

[**Test bằng File CSV** 11](#_Toc166618923)

[Chức năng 1: Tính Diện Tích 11](#_Toc166618924)

[Chức năng 2: Tính chu vi 13](#_Toc166618925)

[SELENIUM – Kiểm đăng nhập và tạo thùng chứa trên Github 16](#_Toc166618926)

[**Mô tả trang web Github:** GitHub là một nền tảng lưu trữ mã nguồn mở và hợp tác trực tuyến. Trang web này cung cấp một môi trường cho các nhà phát triển phát triển, quản lý, và hợp tác trên các dự án phần mềm. 16](#_Toc166618927)

[Tạo Môi Trường Test 17](#_Toc166618928)

[Tạo 1 Window Forms App (.NET Framework) có tên là TestSelenium 17](#_Toc166618929)

[Thiết kế giao diện Windows Forms 17](#_Toc166618930)

[Kiểm tra phiên bản google để cài Selenium: 17](#_Toc166618931)

[Add Extensions Selenium vào Project 18](#_Toc166618932)

[Add Selenium.WebDriver vào Project 18](#_Toc166618933)

[Add Selenium.WebDriver.ChromeDriver 18](#_Toc166618934)

[DECSION TREE của chức năng Đăng nhập 19](#_Toc166618935)

[DECSION TREE của chức năng Tạo thùng chứa 19](#_Toc166618936)

[**Liệt kê các Test Case của chức năng 1: Đăng nhập trang Github** 19](#_Toc166618937)

[**Test Report của chức năng Đăng nhập** 19](#_Toc166618938)

[**Liệt kê các Test Case của chức năng 2: Tạo thùng chứa** 20](#_Toc166618939)

[**Test Report của chức năng Tạo thùng chứa** 20](#_Toc166618940)

[Chức năng 1: Đăng nhập trang GitHub 20](#_Toc166618941)

[Tìm các Element của từng vị trí để thực hiện Test 20](#_Toc166618942)

[Test Case 1: Đăng nhập nhưng Thiếu Username 21](#_Toc166618943)

[Test Case 2 : Đăng nhập nhưng Thiếu Password 22](#_Toc166618944)

[Test Case 3 : Đăng nhập nhưng Sai Username 23](#_Toc166618945)

[Test Case 4 : Đăng nhập nhưng Sai Password 24](#_Toc166618946)

[Test Case 5 : Đăng nhập hợp lệ 25](#_Toc166618947)

[Chức năng 2: Tạo Repositories 26](#_Toc166618948)

[Ta tiếp tục thực hiện tìm các Element ở các nút Button để tạo thùng chứa 26](#_Toc166618949)

[Test Case 1: Tạo thùng chứa hợp lệ 29](#_Toc166618950)

[Test Case 2: Tạo thùng chứa nhưng không nhập Repository Name 30](#_Toc166618951)

[Test Case 3: Tạo thùng chứa nhưng có tên trùng với thùng chứa đã tạo trước đó 31](#_Toc166618952)

[API TỰ TẠO VÀ TEST TRÊN POSTMAN 32](#_Toc166618953)

[Chuẩn bị môi trường test 32](#_Toc166618954)

[Download ứng dụng Postman 32](#_Toc166618955)

[Tạo một API để test 32](#_Toc166618956)

[Download Node.js 32](#_Toc166618957)

[Vào cmd để cài đặt JSON SERVER 33](#_Toc166618958)

[Test trên API vừa tạo 39](#_Toc166618959)

[Vào Postman. Tạo mới Workspace 39](#_Toc166618960)

[Thiết lập biến địa chỉ. 40](#_Toc166618961)

[Thực hiện viết các test case cho Request GetProducts 41](#_Toc166618962)

[Tiến hành tạo GET để test API. 41](#_Toc166618963)

[Viết một số test case đơn giản: 44](#_Toc166618964)

[Test case 1.1: kiểm tra xem biến địa chỉ có đúng đường link localhost không 44](#_Toc166618965)

[Test case 1.2: kiểm tra Status Code có phải là 200 hay không 45](#_Toc166618966)

[Thực hiện viết các test case cho Request PostProducts 45](#_Toc166618967)

[Tiến hành tạo POST để test API. 45](#_Toc166618968)

[Viết một số test case đơn giản 47](#_Toc166618969)

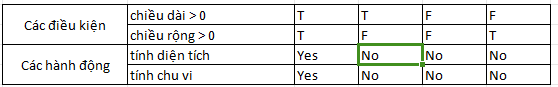
[Test case 2.1: Kiểm tra thời gian phản hồi có nhỏ hơn 8ms không 47](#_Toc166618970)

[Test case 2.2: Kiểm tra Status có phải 201 202 không 47](#_Toc166618971)

# **NUNIT TEST TÍNH DIỆN TÍCH – CHU VI Hình Chữ Nhật**

**NUnit Test Phần mềm tính diện tích Hình Chữ Nhật**

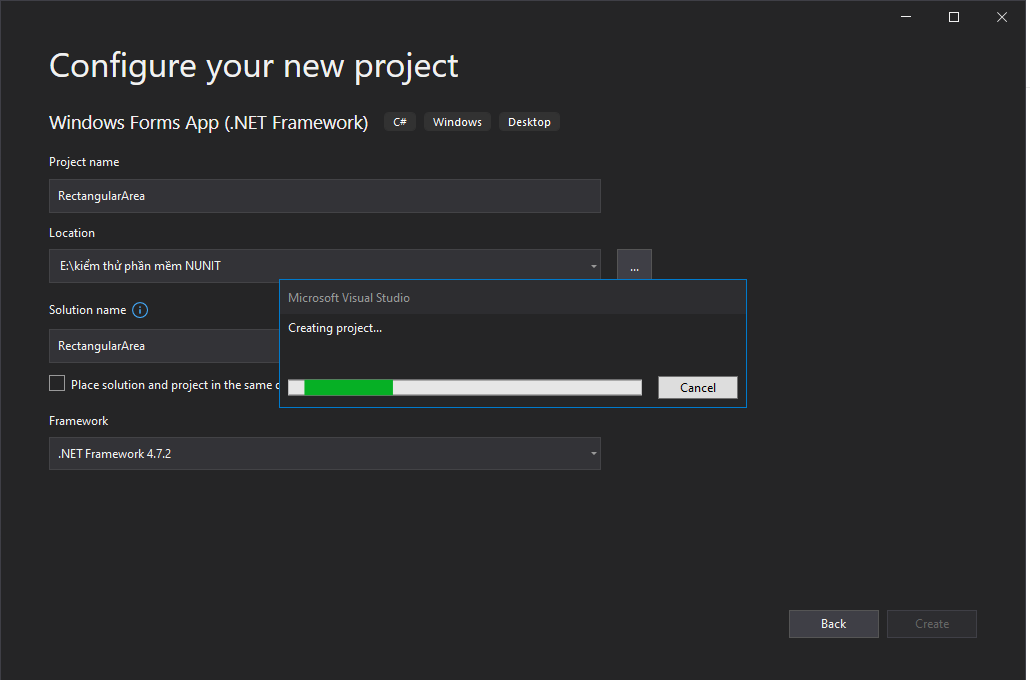
# Decision Tree:



## **Tạo môi trường**

## - Tạo giao diện tính diện tích hình chữ nhật bằng ngôn ngữ C#

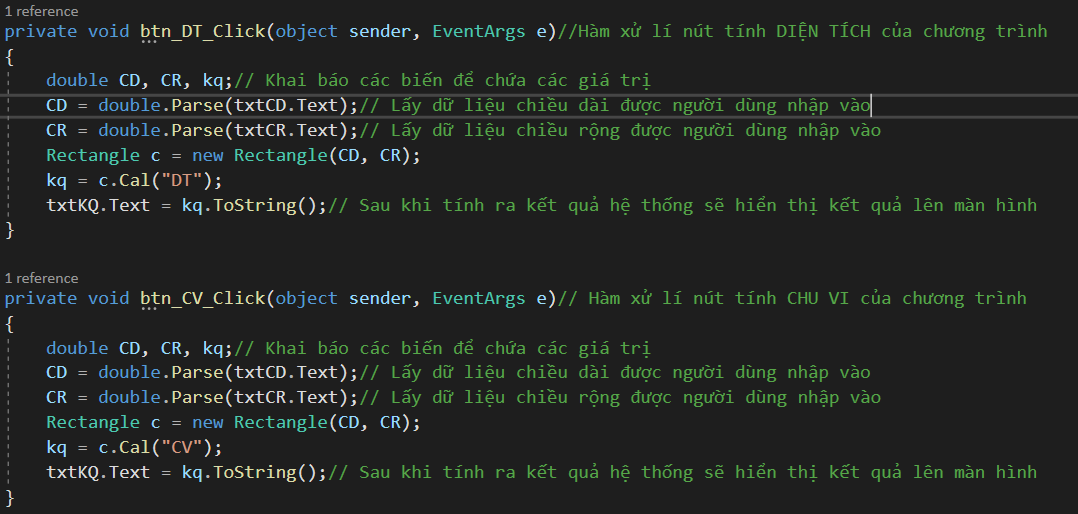
### + Sử dụng file Windows Forms App(.NET Framework)



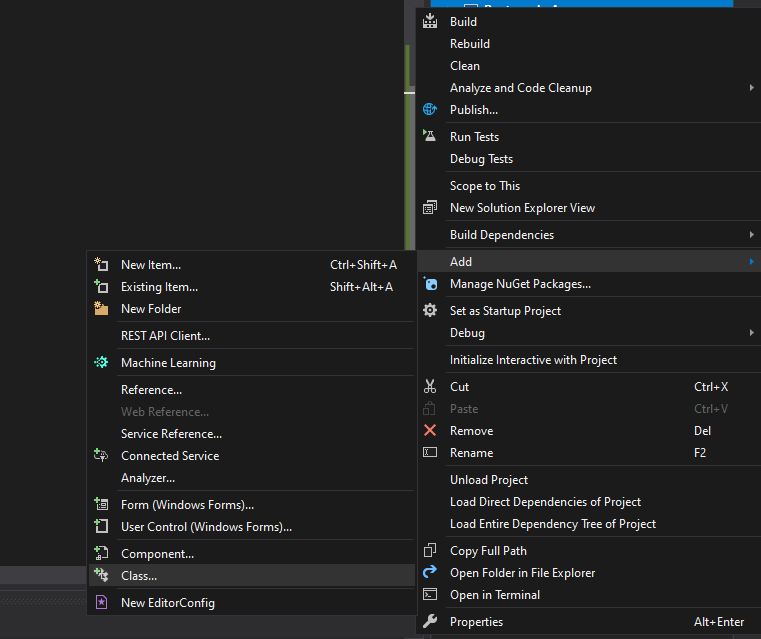


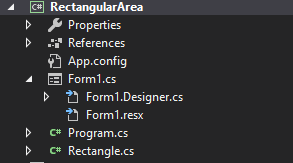
# **Viết code để xử lí tính toán trong chương trình**

## Viết code xử lí nút tính Diện Tích và Chu Vi của chương trình

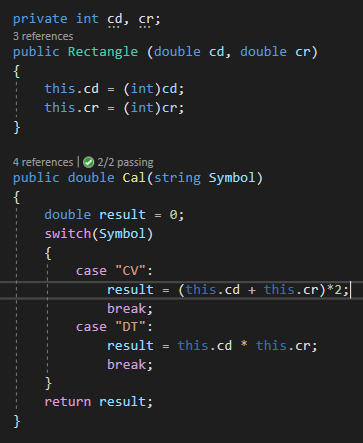


# **Add Class, đặt tên là Rectangle, để hổ trợ tính toán**



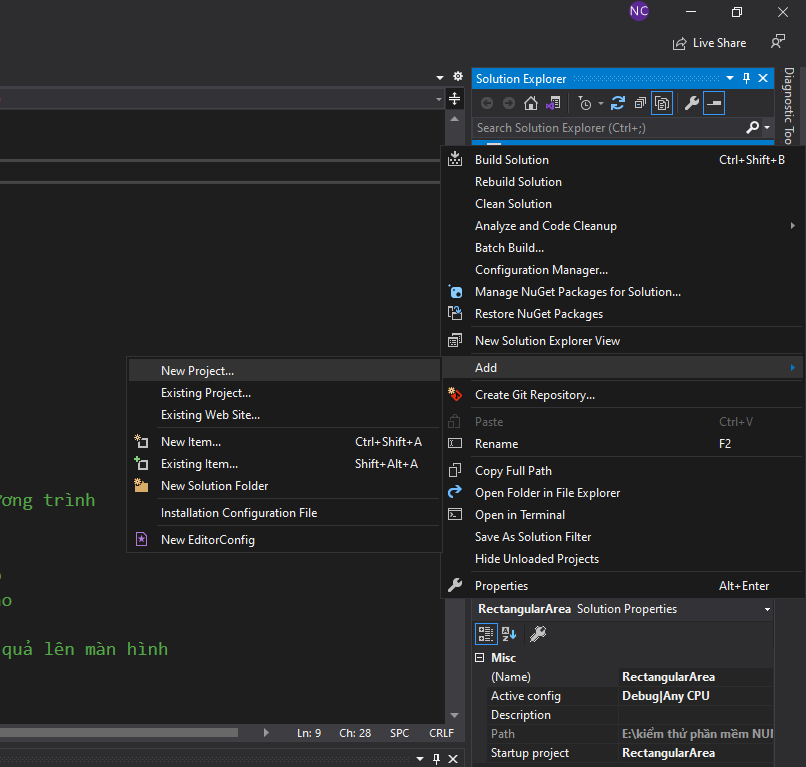


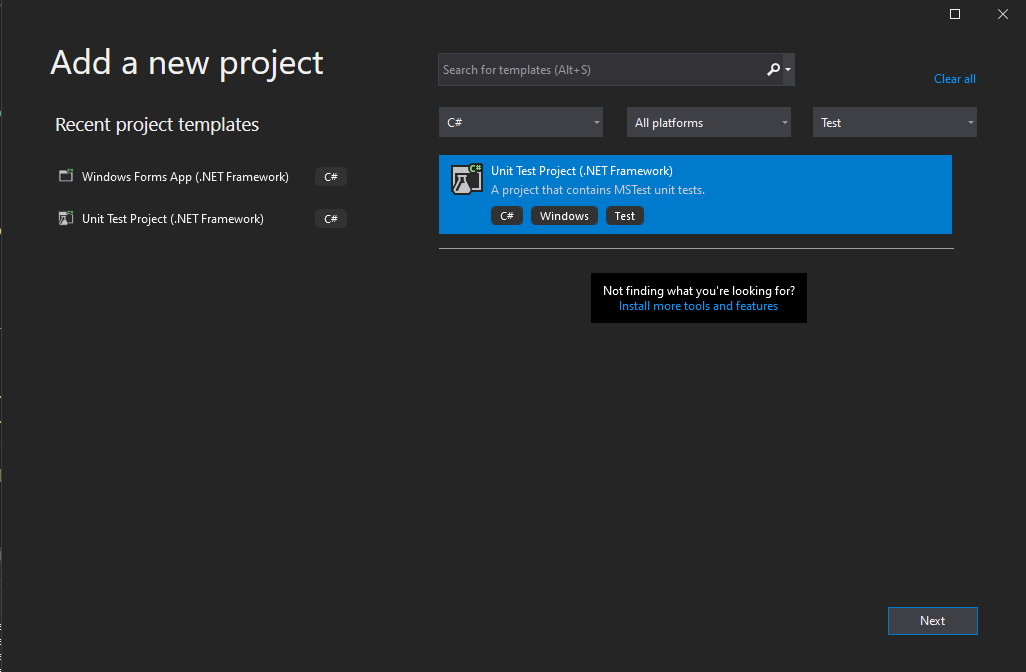
## Code trong Class Rectangle



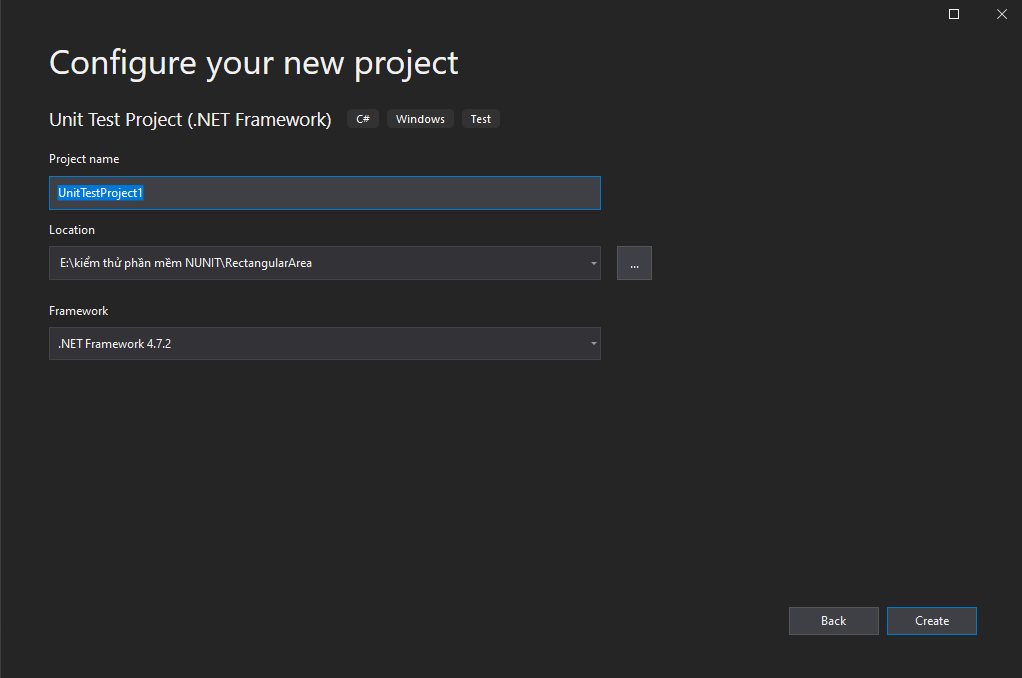
# **Tạo Project để kiểm thử phần mềm trên**

Click chuột phải Solution > Add > New Projects... Sau đó chọn loại project là “Unit Test Project” và đặt tên là RectangularAreaTester

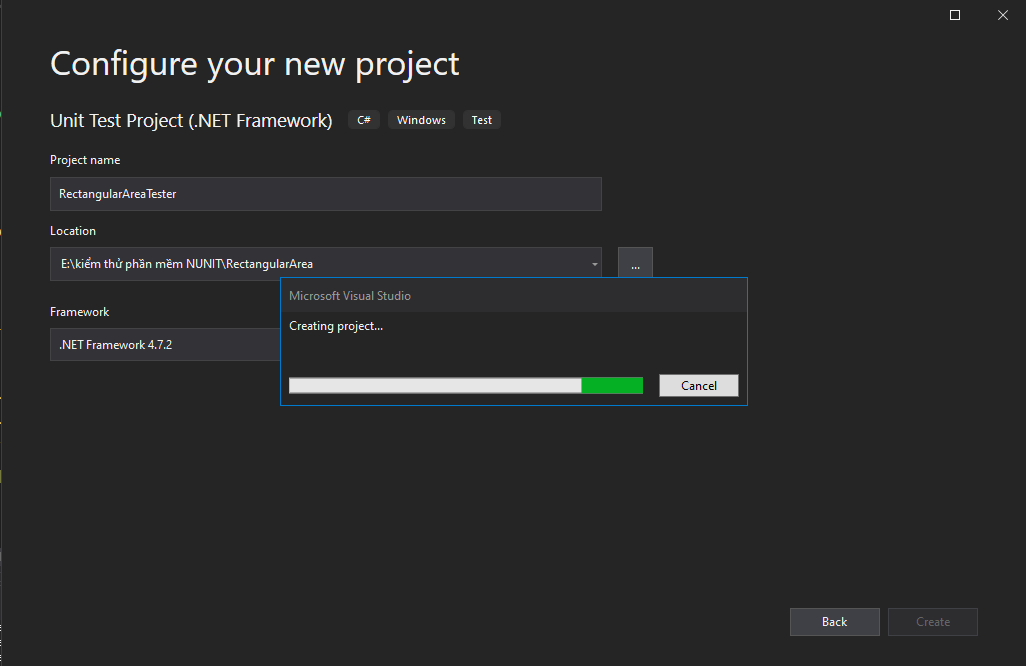




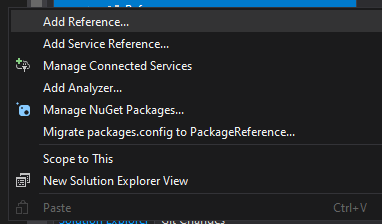
Sau đó nhấn Next

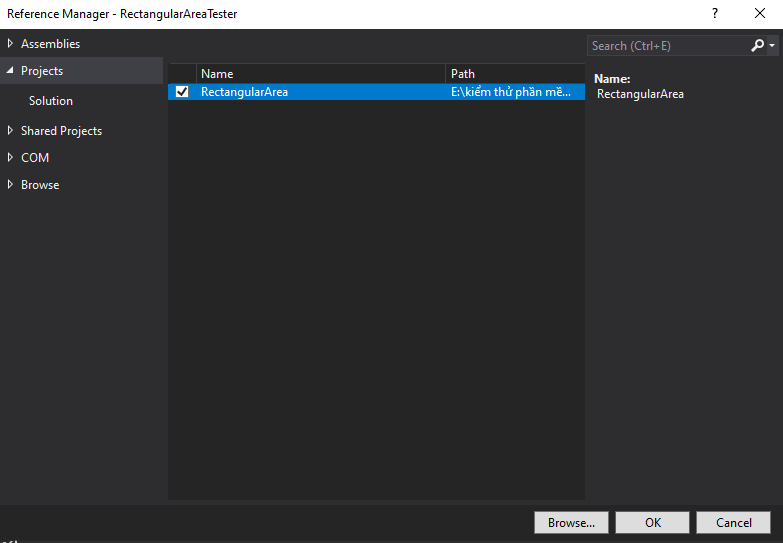


Đặt tên Project theo yêu cầu và sau đó nhất Create

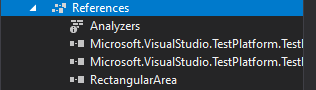


## **Tại project Unit Test, thực hiện Add Reference để tham chiếu đến project cần thực hiện Unit Test**

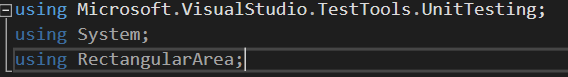




### Sau khi Add Reference thành công sẽ hiển thị tên Project đã Add



### Phải Using Project trước khi code



## **Mô tả Unit test:**

Unit test này dùng để tính chu vi và diện tích hình chữ nhật. Người dùng nhập chiều dài, chiều rộng hình chữ nhật và các thuật toán sẽ tính toán để cho ra chu vi và diện tích hình chữ nhật.

Công thức của:

Chu vi Hình chữ nhật: (Chiều dài + Chiều rộng)x2

Diện tích Hình chữ nhật: Chiều dài x Chiều rộng

Dựa vào mô tả trên mình sẽ viết test case cho 2 chứ năng Tính chu vi và Tính diện tích hình chữ nhật.

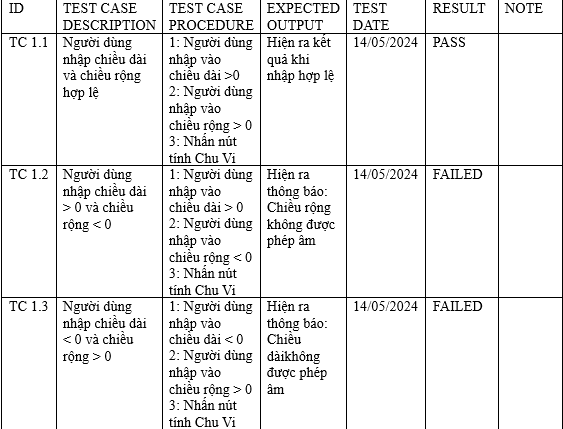
## **Các Test Case của chức năng tính chu vi:**

+Test Case 1.1: Test\_TinhChuVi\_HCN\_CDvaCRlonhon0();

+Test Case 1.2: Test\_TinhChuVi\_HCN\_CDlonhon0\_CRnhohon0();

+Test Case 1.3: Test\_TinhChuVi\_HCN\_CDnhohon0\_CRlonhon0();

## **Test Report của chức năng tính chu vi Hình chữ nhật:**



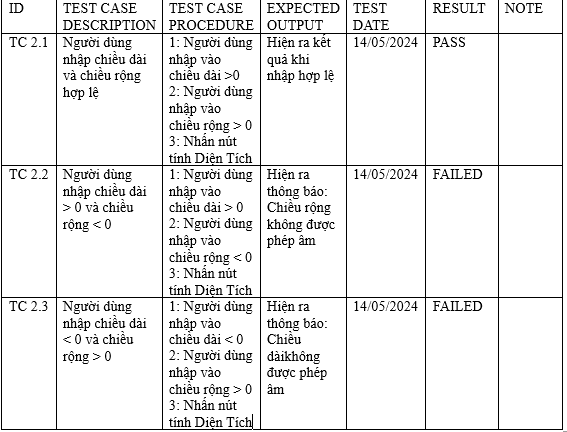
## **Các Test Case của chức năng tính diện tích:**

+Test Case 2.1: Test\_TinhDienTich\_HCN\_CDvaCRlonhon0();

+Test Case 2.2: Test\_TinhDienTich\_HCN\_CDnhohon0\_CRlonhon0();

+Test Case 2.3 Test\_TinhDienTich\_HCN\_CDlonhon0\_CRnhohon0();

## **Test Report của chức năng tính diện tích Hình chữ nhật:**



## Viết Test Case cho 2 chức năng

### **Test bằng cách gán bộ dữ liệu 5,5 cho 2 test case**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using RectangularArea;

namespace RectangularAreaTester

{

[TestClass]

public class UnitTest1

{

private Rectangle r; //36\_Nguyen\_UnitTest

[TestInitialize]

public void SetUp() // khai báo sẵn các biến để dùng test

{

r = new Rectangle(5, 5);

}

[TestMethod] // Test case tính diện tích hình chữ nhật

public void Test\_TinhDienTich\_HCN()

{

double expected, actual;

expected = 25;

actual = r.Cal("DT");

Assert.AreEqual(expected, actual);

}

[TestMethod] // Test case tính chu vi hình chữ nhật

public void Test\_TinhChuVi\_HCN()

{

double expected, actual;

expected = 20;

actual = r.Cal("CV");

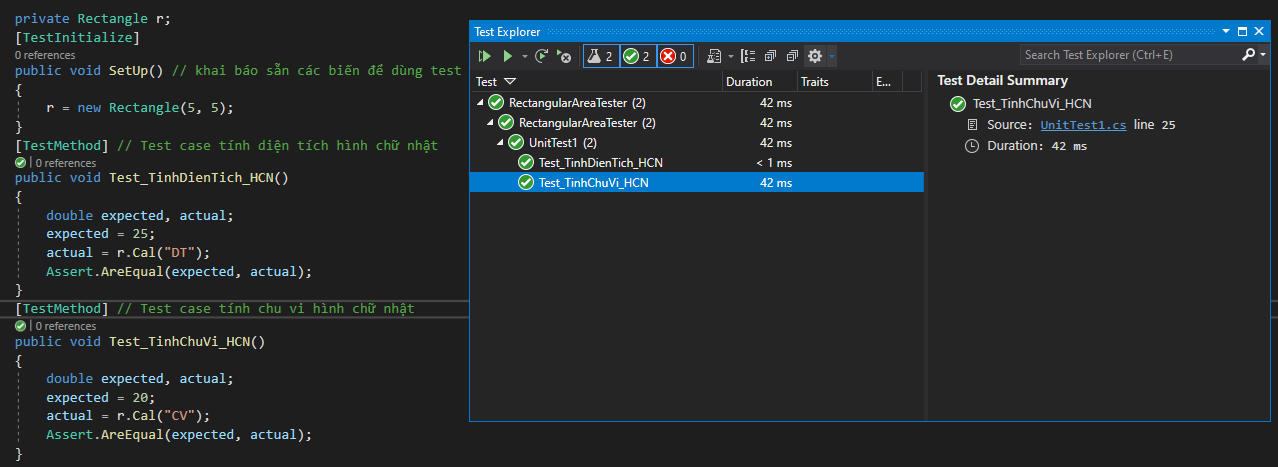
Assert.AreEqual(expected, actual);

}

}

}

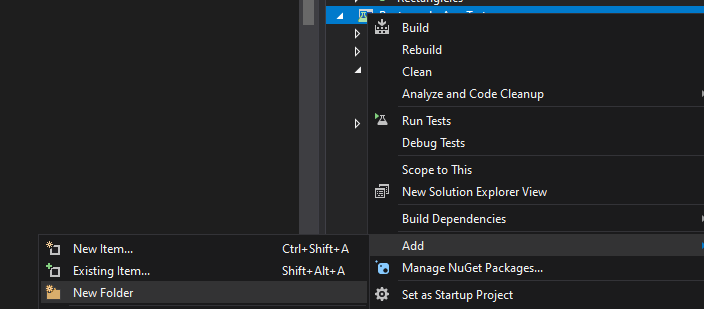
### Kết quả test



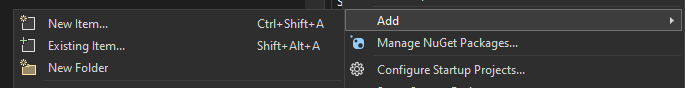
### **Test bằng File CSV**

### Chức năng 1: Tính Diện Tích

Add 1 new folder vào Rectangular Test



Tạo 1 Folder có tên là DataDienTich

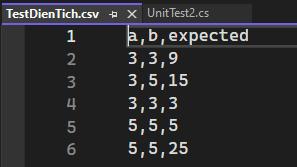




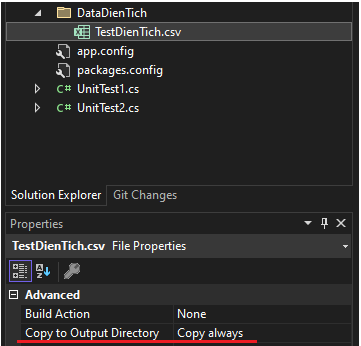
Sau khi tạo 1 Folder thì add File TestDienTich.csv vào trong Folder



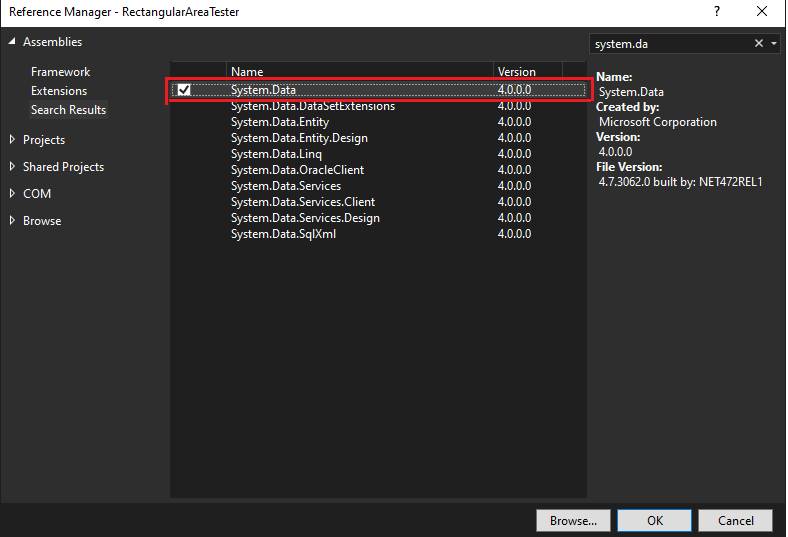
Chuẩn bị dữ liệu trong File TestData.csv để bắt đầu chạy Test



Cho phép File TestDienTich.csv được copy vào Output Directory để chạy Test



Add Thư viện System.Data để đọc được File CSV



Viết code để liên kết file TestDienTich.csv

public TestContext TestContext { get; set; }

[TestMethod]

[DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",

@".\DataDienTich\TestDienTich.csv",

"TestDienTich#csv",

DataAccessMethod.Sequential)]

Viết code để lấy dữ liệu trong file ra để thực hiện Test ( Chức năng: Tính diện tích)

public void TinhDienTichData()

{

double a, b, expected, actual;

a = double.Parse(TestContext.DataRow[0].ToString());

b = double.Parse(TestContext.DataRow[1].ToString());

expected = double.Parse(TestContext.DataRow[2].ToString());

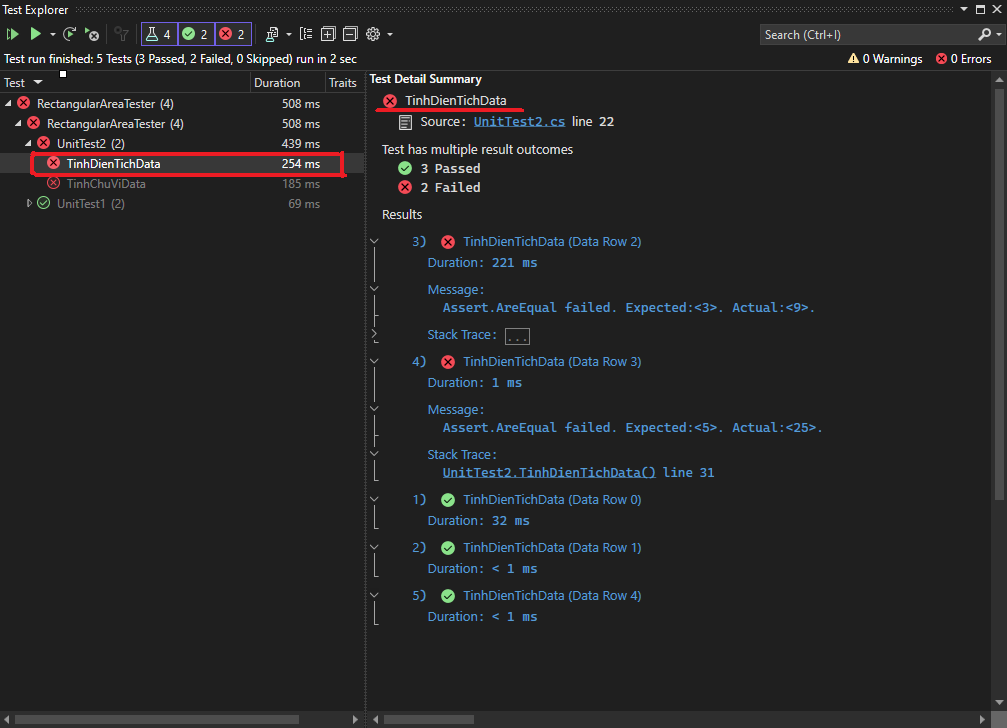
Rectangle r = new Rectangle(a, b);

actual = r.Cal("DT");

Assert.AreEqual(expected, actual);

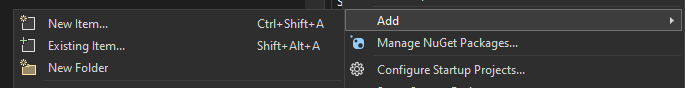
}

Kết quả khi chạy test



### Chức năng 2: Tính chu vi

Tạo 1 Folder có tên là DataChuVi

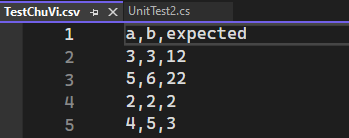




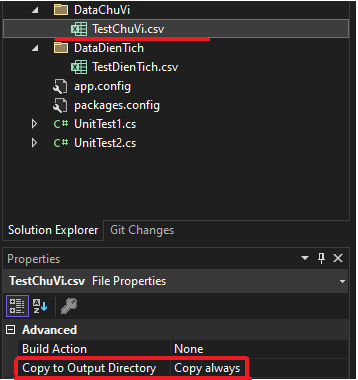
Sau khi tạo 1 Folder thì add File TestChuVi.csv vào trong Folder



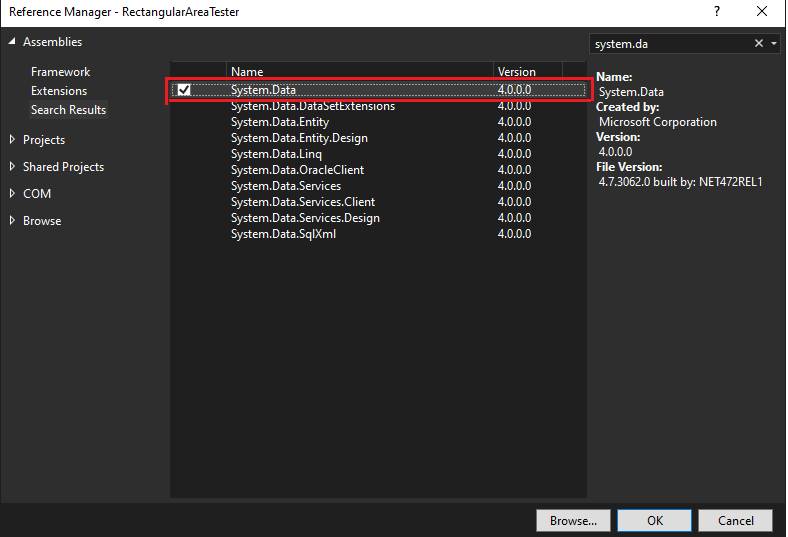
Chuẩn bị dữ liệu trong File TestData.csv để bắt đầu chạy Test



Cho phép File TestChuVi.csv được copy vào Output Directory để chạy Test



Add Thư viện System.Data để đọc được File CSV



Viết code để liên kết file TestChuVi.csv

[TestMethod]

[DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",

@".\DataChuVi\TestChuVi.csv",

"TestChuVi#csv",

DataAccessMethod.Sequential)]

Viết code để lấy dữ liệu trong file ra để thực hiện Test ( Chức năng: Tính Chu Vi)

public void TinhChuViData()

{

double a, b, expected, actual;

a = double.Parse(TestContext.DataRow[0].ToString());

b = double.Parse(TestContext.DataRow[1].ToString());

expected = double.Parse(TestContext.DataRow[2].ToString());

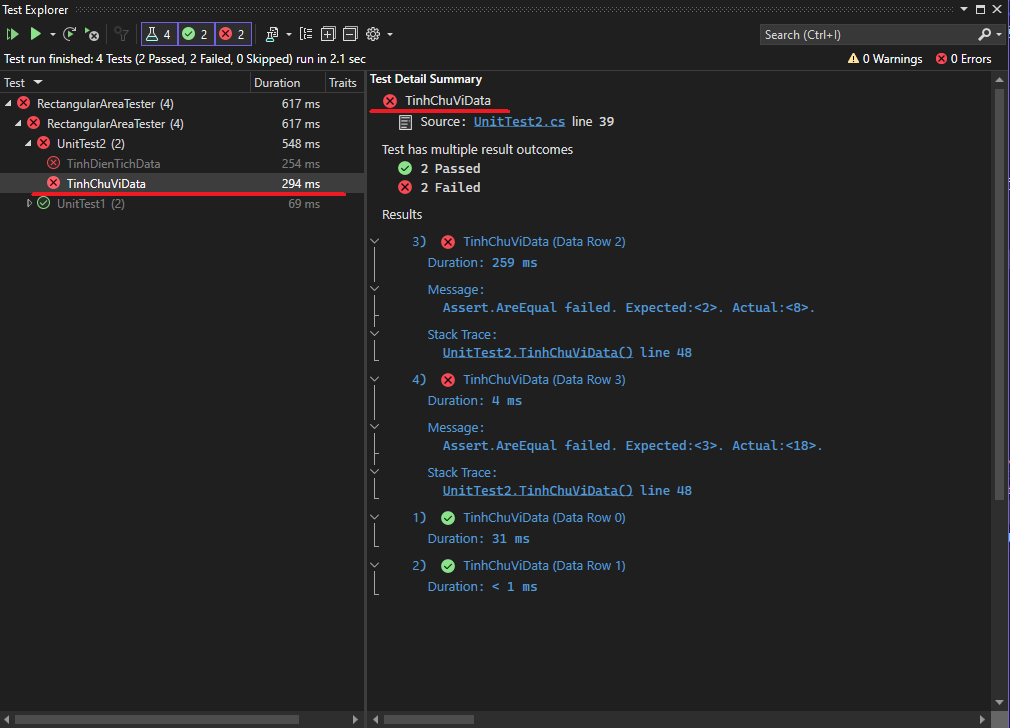
Rectangle r = new Rectangle(a, b);

actual = r.Cal("CV");

Assert.AreEqual(expected, actual);

}

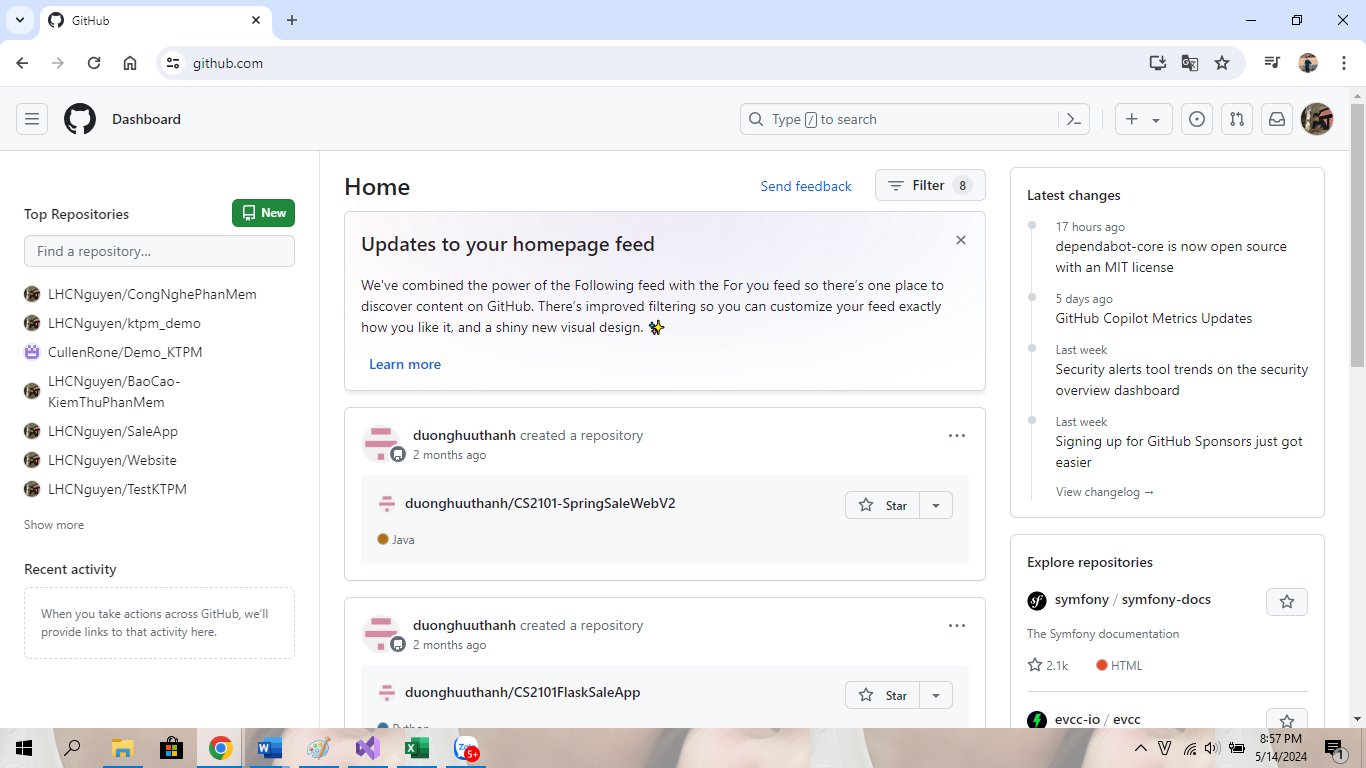
Kết quả khi chạy test



# **SELENIUM – Kiểm đăng nhập và tạo thùng chứa trên Github**

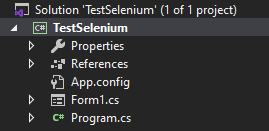
## **Mô tả trang web Github:** GitHub là một nền tảng lưu trữ mã nguồn mở và hợp tác trực tuyến. Trang web này cung cấp một môi trường cho các nhà phát triển phát triển, quản lý, và hợp tác trên các dự án phần mềm.

Giao diện của trang web:

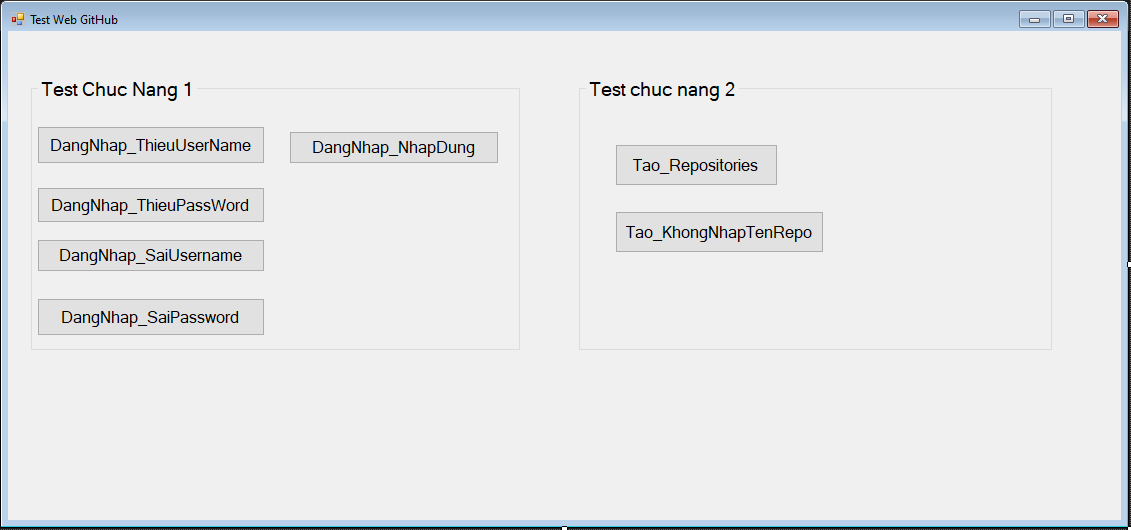


## Tạo Môi Trường Test

## Tạo 1 Window Forms App (.NET Framework) có tên là TestSelenium

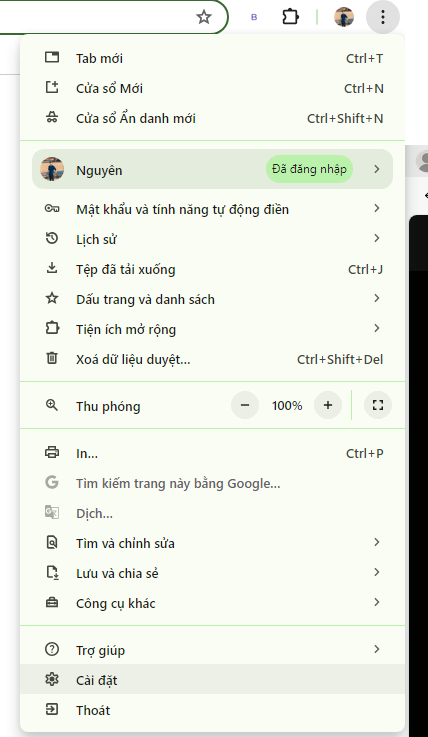
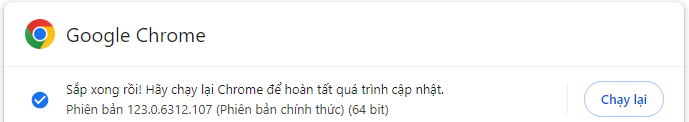


## Thiết kế giao diện Windows Forms



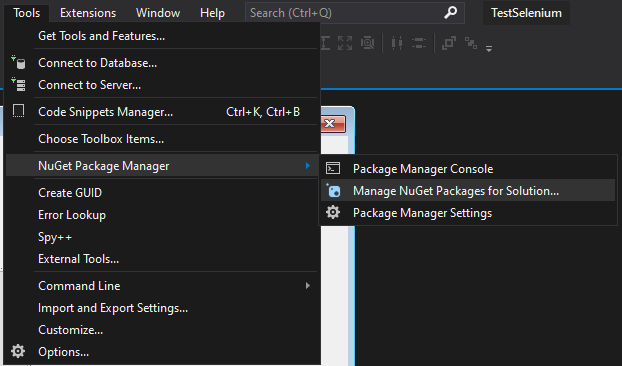
## Kiểm tra phiên bản google để cài Selenium:

Vào cài đặt -> Giới thiệu về Chrome

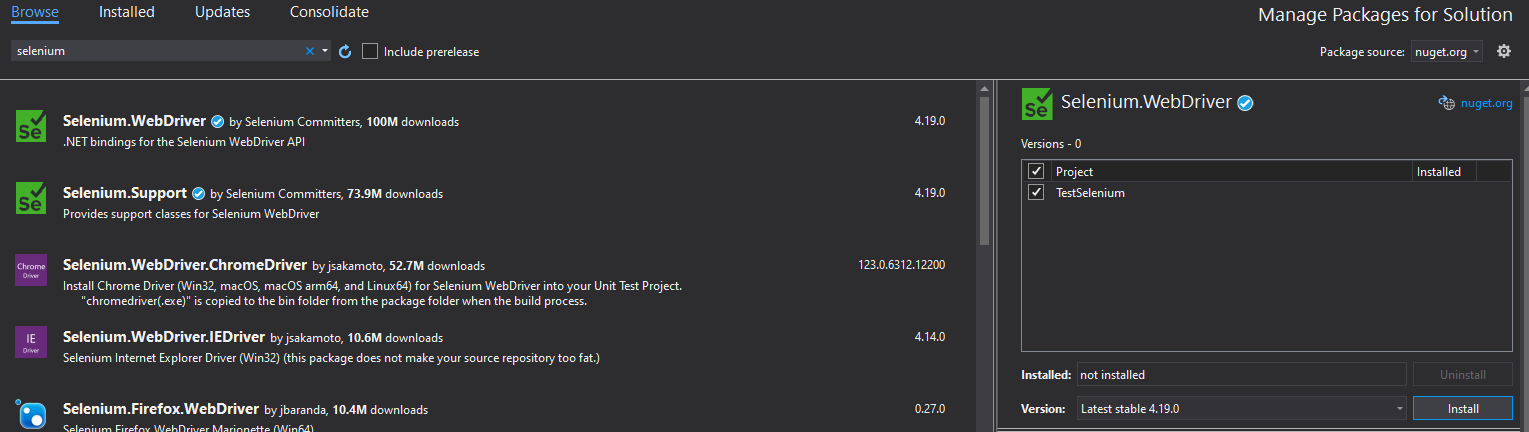




## Add Extensions Selenium vào Project

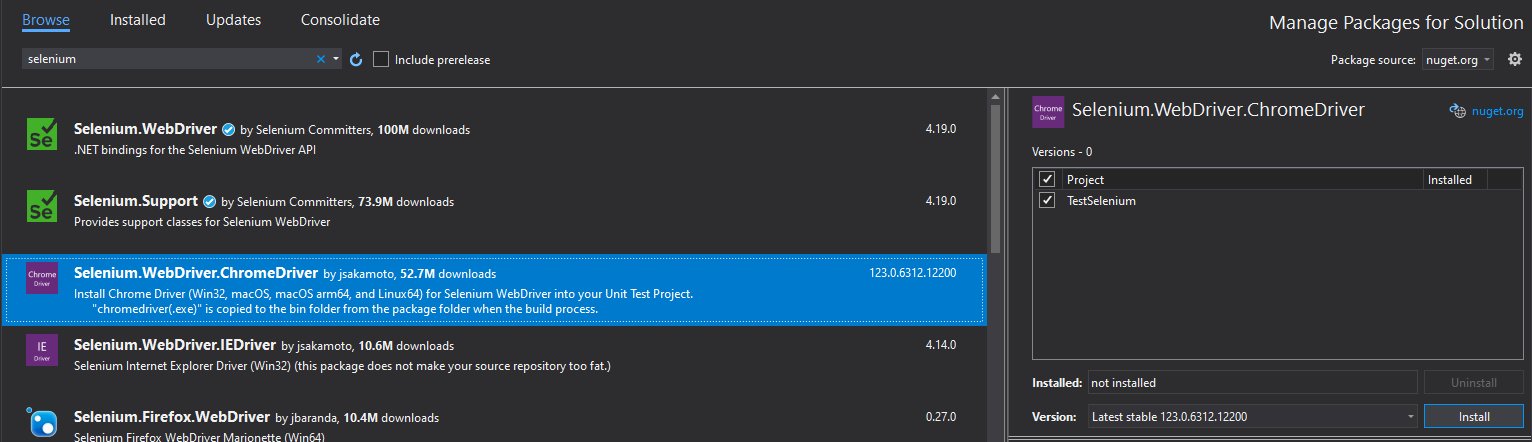


## Add Selenium.WebDriver vào Project



## Add Selenium.WebDriver.ChromeDriver

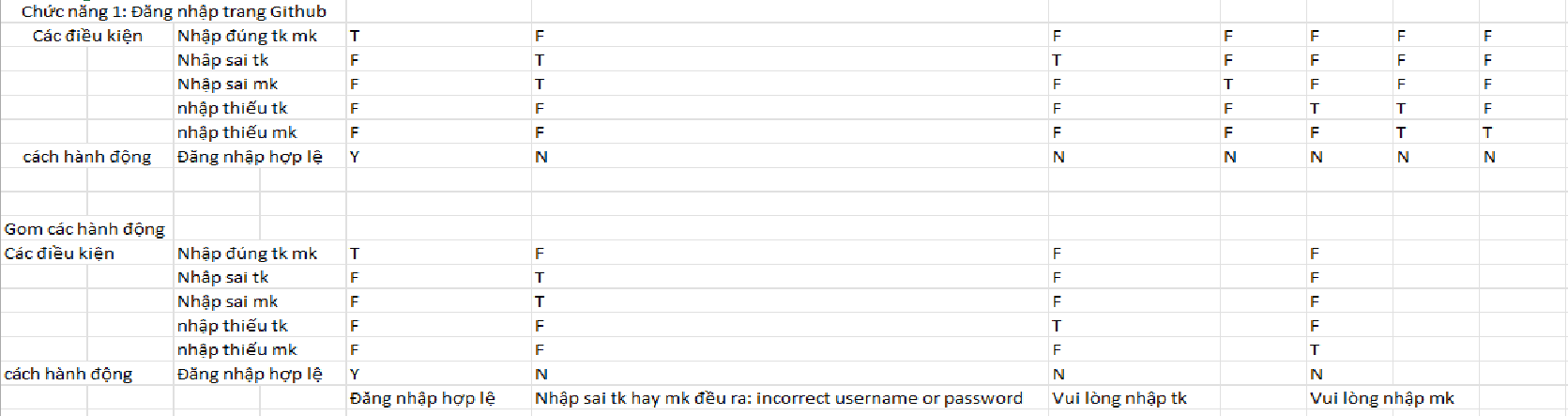
Phải so sánh với phiên bản google để cài Selenium.WebDriver.ChromeDriver cho phù hợp



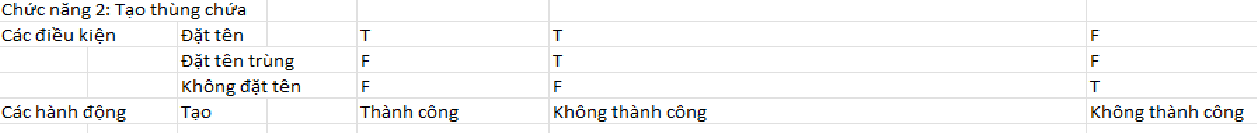
Phải Using vào Project



## DECSION TREE của chức năng Đăng nhập



## DECSION TREE của chức năng Tạo thùng chứa



## **Liệt kê các Test Case của chức năng 1: Đăng nhập trang Github**

Testcase 1.1: DangNhap\_Thieu\_UserName\_Click();

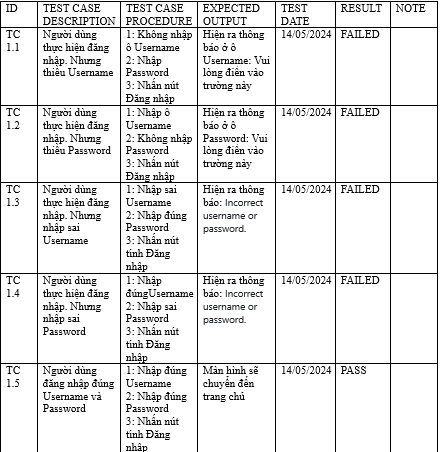
Testcase 1.2: DangNhap\_ThieuPassWord\_Click ();

Testcase 1.3: DangNhap\_SaiUsername\_Click ();

Testcase 1.4: DangNhap\_SaiPassword\_Click ();

Testcase 1.5: DangNhap\_NhapDung\_Click ();

## **Test Report của chức năng Đăng nhập**



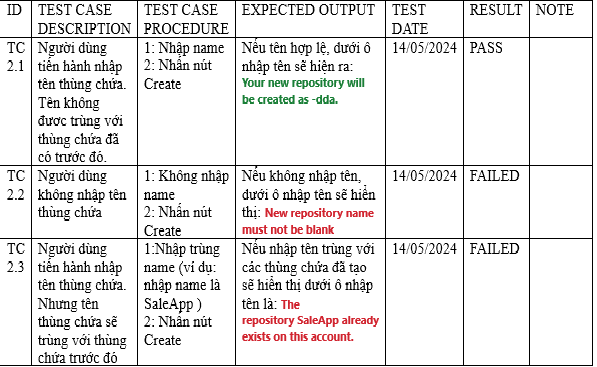
## **Liệt kê các Test Case của chức năng 2: Tạo thùng chứa**

Testcase 2.1: Tao\_Repositories\_Click();

Testcase 2.2: Tao\_KhongNhapTenRepo\_Click();

Testcase 2.3: Tao\_RepoCoTenTrung\_Click();

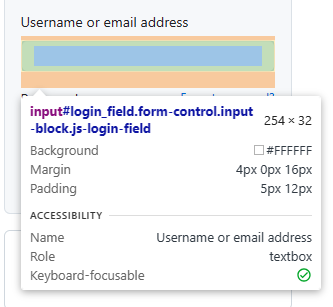
## **Test Report của chức năng Tạo thùng chứa**



## Chức năng 1: Đăng nhập trang GitHub

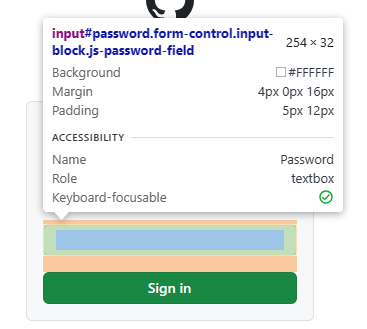
## Tìm các Element của từng vị trí để thực hiện Test

Tìm Name của chổ đăng nhập Username



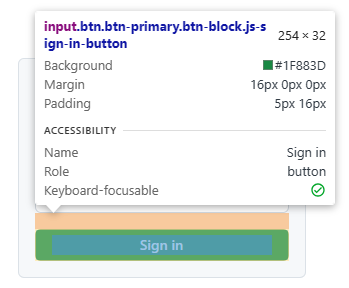


Tìm Name của chổ đăng nhập Password





Tìm Name ở nút Sign in





### Test Case 1: Đăng nhập nhưng Thiếu Username

Thiết kế giao diện: nút Button và đặt tên như hình.





Viết code xử lí:

private void DangNhap\_Thieu\_UserName\_Click(object sender, EventArgs e)

{

ChromeDriverService Chrome\_Nguyen\_36 = ChromeDriverService.CreateDefaultService();

Chrome\_Nguyen\_36.HideCommandPromptWindow = true;

IWebDriver Nguyen\_36 = new ChromeDriver(Chrome\_Nguyen\_36);

Nguyen\_36.Navigate().GoToUrl("https://github.com/login");

IWebElement taikhoan = Nguyen\_36.FindElement(By.Name("login"));

taikhoan.SendKeys("");

IWebElement matkhau = Nguyen\_36.FindElement(By.Name("password"));

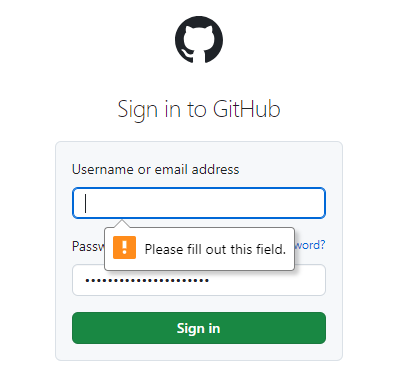
matkhau.SendKeys("lamhuynhchannguyen2003");

IWebElement hoanthanh = Nguyen\_36.FindElement(By.Name("commit"));

hoanthanh.Click();

}

Kết quả khi chạy test case 1:



### Test Case 2 : Đăng nhập nhưng Thiếu Password

Thiết kế giao diện: nút Button và đặt tên như hình.



Viết code xử lý:

private void DangNhap\_ThieuPassWord\_Click(object sender, EventArgs e)

{

ChromeDriverService Chrome\_Nguyen\_36 = ChromeDriverService.CreateDefaultService();

Chrome\_Nguyen\_36.HideCommandPromptWindow = true;

IWebDriver Nguyen\_36 = new ChromeDriver(Chrome\_Nguyen\_36);

Nguyen\_36.Navigate().GoToUrl("https://github.com/login");

IWebElement taikhoan = Nguyen\_36.FindElement(By.Name("login"));

taikhoan.SendKeys("LHCNguyen");

IWebElement matkhau = Nguyen\_36.FindElement(By.Name("password"));

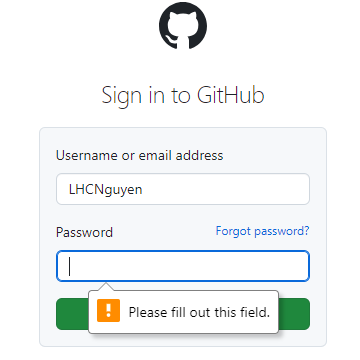
matkhau.SendKeys("");

IWebElement hoanthanh = Nguyen\_36.FindElement(By.Name("commit"));

hoanthanh.Click();

}

Kết quả khi chạy Test Case 2:



### Test Case 3 : Đăng nhập nhưng Sai Username

Thiết kế giao diện: nút Button và đặt tên như hình.



Viết code xử lý:

private void DangNhap\_SaiUsername\_Click(object sender, EventArgs e)

{

ChromeDriverService Chrome\_Nguyen\_36 = ChromeDriverService.CreateDefaultService();

Chrome\_Nguyen\_36.HideCommandPromptWindow = true;

IWebDriver Nguyen\_36 = new ChromeDriver(Chrome\_Nguyen\_36);

Nguyen\_36.Navigate().GoToUrl("https://github.com/login");

IWebElement taikhoan = Nguyen\_36.FindElement(By.Name("login"));

taikhoan.SendKeys("LHCNguyen.");

IWebElement matkhau = Nguyen\_36.FindElement(By.Name("password"));

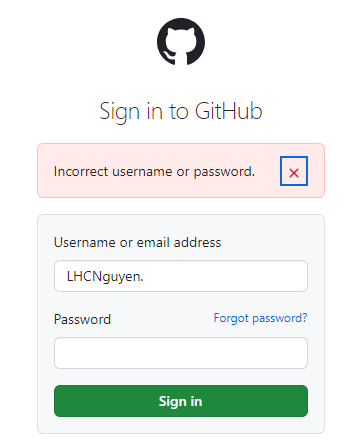
matkhau.SendKeys("lamhuynhchannguyen2003");

IWebElement hoanthanh = Nguyen\_36.FindElement(By.Name("commit"));

hoanthanh.Click();

}

Kết quả khi chạy Test Case 3:



### Test Case 4 : Đăng nhập nhưng Sai Password

Thiết kế giao diện: nút Button và đặt tên như hình.



Viết code xử lý:

private void DangNhap\_SaiPassword\_Click(object sender, EventArgs e)

{

ChromeDriverService Chrome\_Nguyen\_36 = ChromeDriverService.CreateDefaultService();

Chrome\_Nguyen\_36.HideCommandPromptWindow = true;

IWebDriver Nguyen\_36 = new ChromeDriver(Chrome\_Nguyen\_36);

Nguyen\_36.Navigate().GoToUrl("https://github.com/login");

IWebElement taikhoan = Nguyen\_36.FindElement(By.Name("login"));

taikhoan.SendKeys("LHCNguyen");

IWebElement matkhau = Nguyen\_36.FindElement(By.Name("password"));

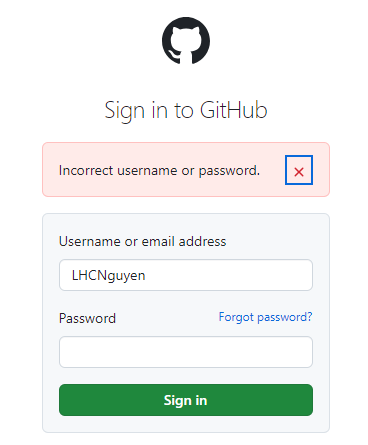
matkhau.SendKeys("lamhuynhchannguyen2003.");

IWebElement hoanthanh = Nguyen\_36.FindElement(By.Name("commit"));

hoanthanh.Click();

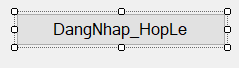
}

Kết quả khi chạy Test Case 4:



### Test Case 5 : Đăng nhập hợp lệ

Thiết kế giao diện: nút Button và đặt tên như hình.



Viết code xử lý:

private void DangNhap\_NhapDung\_Click(object sender, EventArgs e)

{

ChromeDriverService Chrome\_Nguyen\_36 = ChromeDriverService.CreateDefaultService();

Chrome\_Nguyen\_36.HideCommandPromptWindow = true;

IWebDriver Nguyen\_36 = new ChromeDriver(Chrome\_Nguyen\_36);

Nguyen\_36.Navigate().GoToUrl("https://github.com/login");

IWebElement taikhoan = Nguyen\_36.FindElement(By.Name("login"));

taikhoan.SendKeys("LHCNguyen");

IWebElement matkhau = Nguyen\_36.FindElement(By.Name("password"));

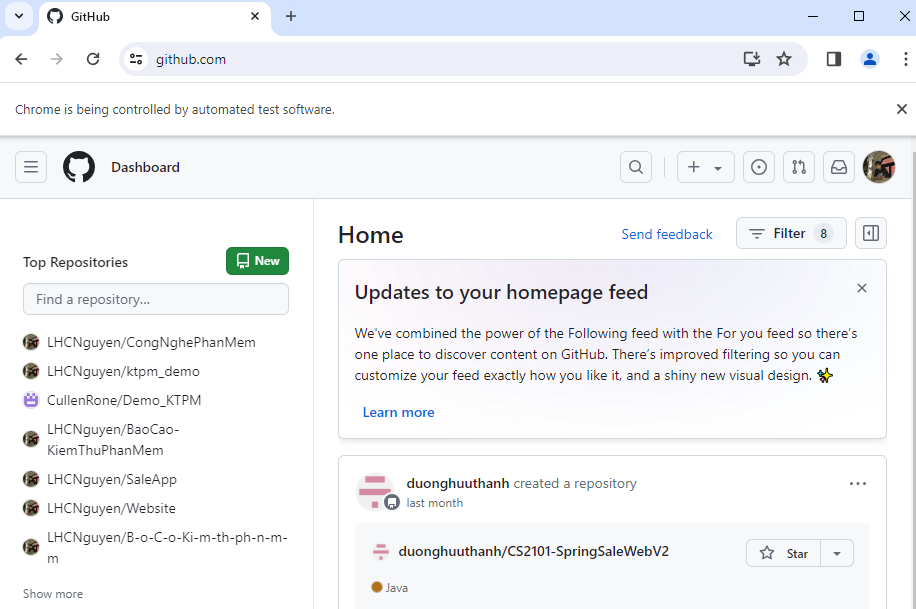
matkhau.SendKeys("lamhuynhchannguyen2003");

IWebElement hoanthanh = Nguyen\_36.FindElement(By.Name("commit"));

hoanthanh.Click();

}

Kết quả khi chạy Test Case 5:

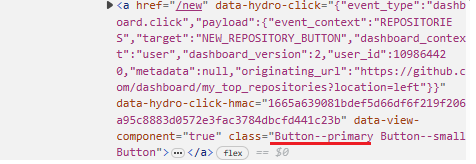
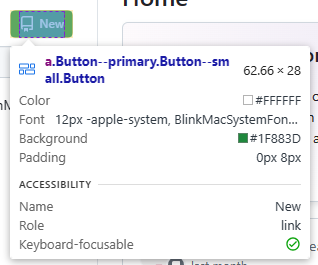


## Chức năng 2: Tạo Repositories

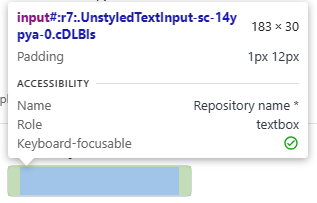
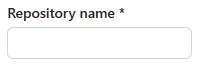
Sau khi thực hiện đăng nhập thành công ở Chức Năng 1

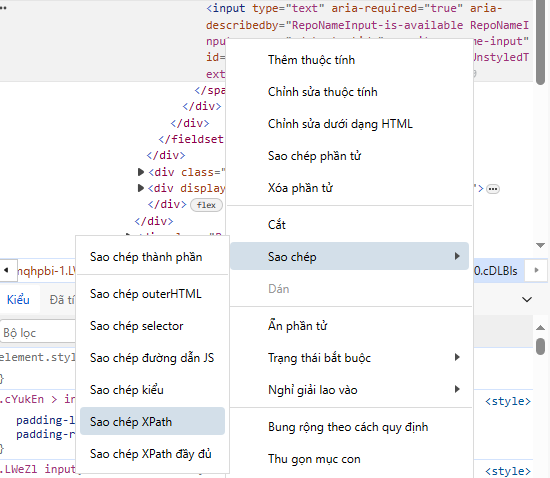
## Ta tiếp tục thực hiện tìm các Element ở các nút Button để tạo thùng chứa

Tìm Element ở nút New

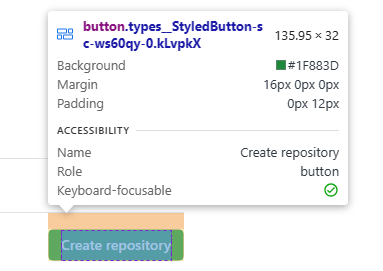


Tìm Element ở Repositories Name: copy XPath



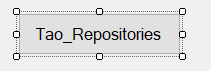


Tìm Element ở nút Create repository



### Test Case 1: Tạo thùng chứa hợp lệ

Thiết kế giao diện: nút Button và đặt tên như hình.



Viết code xử lý:

private void Tao\_Repositories\_Click(object sender, EventArgs e)

{

ChromeDriverService Chrome\_Nguyen\_36 = ChromeDriverService.CreateDefaultService();

Chrome\_Nguyen\_36.HideCommandPromptWindow = true;

IWebDriver Nguyen\_36 = new ChromeDriver(Chrome\_Nguyen\_36);

Nguyen\_36.Navigate().GoToUrl("https://github.com/login");

IWebElement taikhoan = Nguyen\_36.FindElement(By.Name("login"));

taikhoan.SendKeys("LHCNguyen");

IWebElement matkhau = Nguyen\_36.FindElement(By.Name("password"));

matkhau.SendKeys("lamhuynhchannguyen2003");

IWebElement hoanthanh = Nguyen\_36.FindElement(By.Name("commit"));

hoanthanh.Click();

IWebElement newRepositories = Nguyen\_36.FindElement(By.ClassName("Button--primary"));

newRepositories.Click();

IWebElement nameRepo = Nguyen\_36.FindElement(By.XPath("//\*[@id=\":r7:\"]"));

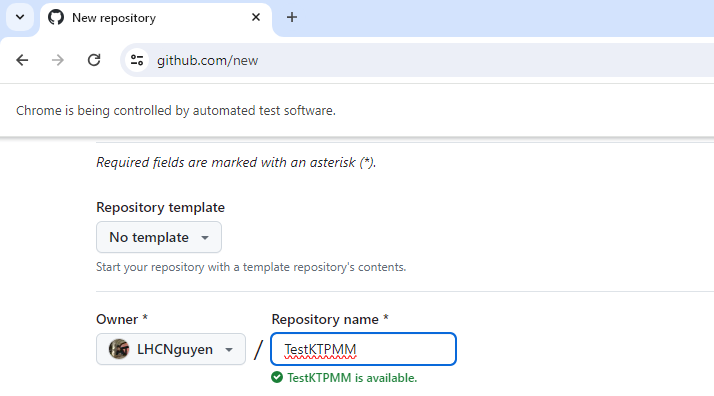
nameRepo.SendKeys("TestKTPM");

IWebElement createRepo = Nguyen\_36.FindElement(By.ClassName("kLvpkX"));

createRepo.Click();

}

Kết quả khi chạy Test Case 1:



### Test Case 2: Tạo thùng chứa nhưng không nhập Repository Name

Thiết kế giao diện: nút Button và đặt tên như hình



Viết code xử lý:

private void Tao\_KhongNhapTenRepo\_Click(object sender, EventArgs e)

{

ChromeDriverService Chrome\_Nguyen\_36 = ChromeDriverService.CreateDefaultService();

Chrome\_Nguyen\_36.HideCommandPromptWindow = true;

IWebDriver Nguyen\_36 = new ChromeDriver(Chrome\_Nguyen\_36);

Nguyen\_36.Navigate().GoToUrl("https://github.com/login");

IWebElement taikhoan = Nguyen\_36.FindElement(By.Name("login"));

taikhoan.SendKeys("LHCNguyen");

IWebElement matkhau = Nguyen\_36.FindElement(By.Name("password"));

matkhau.SendKeys("lamhuynhchannguyen2003");

IWebElement hoanthanh = Nguyen\_36.FindElement(By.Name("commit"));

hoanthanh.Click();

IWebElement newRepositories = Nguyen\_36.FindElement(By.ClassName("Button--primary"));

newRepositories.Click();

IWebElement nameRepo = Nguyen\_36.FindElement(By.XPath("//\*[@id=\":r7:\"]"));

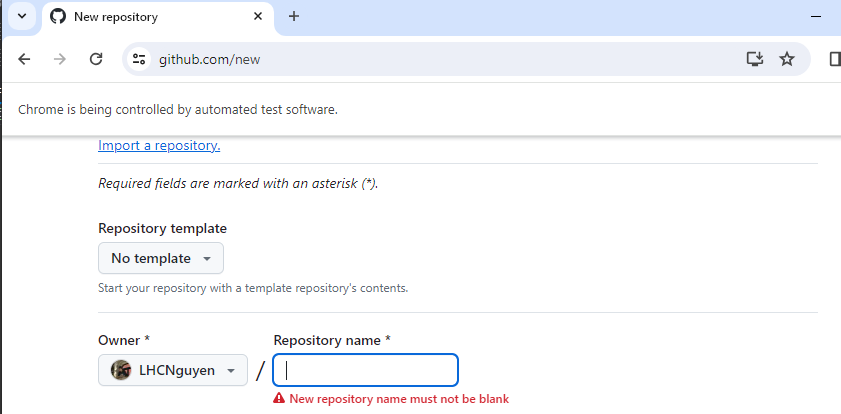
nameRepo.SendKeys("");

IWebElement createRepo = Nguyen\_36.FindElement(By.ClassName("kLvpkX"));

createRepo.Click();

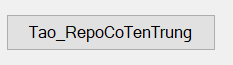
}

Kết quả khi chạy Test Case 2:



### Test Case 3: Tạo thùng chứa nhưng có tên trùng với thùng chứa đã tạo trước đó

Thiết kế giao diện: nút Button và đặt tên như hình



Viết code xử lý:

private void Tao\_RepoCoTenTrung\_Click(object sender, EventArgs e)

{

ChromeDriverService Chrome\_Nguyen\_36 = ChromeDriverService.CreateDefaultService();

Chrome\_Nguyen\_36.HideCommandPromptWindow = true;

IWebDriver Nguyen\_36 = new ChromeDriver(Chrome\_Nguyen\_36);

Nguyen\_36.Navigate().GoToUrl("https://github.com/login");

IWebElement taikhoan = Nguyen\_36.FindElement(By.Name("login"));

taikhoan.SendKeys("LHCNguyen");

IWebElement matkhau = Nguyen\_36.FindElement(By.Name("password"));

matkhau.SendKeys("lamhuynhchannguyen2003");

IWebElement hoanthanh = Nguyen\_36.FindElement(By.Name("commit"));

hoanthanh.Click();

IWebElement newRepositories = Nguyen\_36.FindElement(By.ClassName("Button--primary"));

newRepositories.Click();

IWebElement nameRepo = Nguyen\_36.FindElement(By.XPath("//\*[@id=\":r7:\"]"));

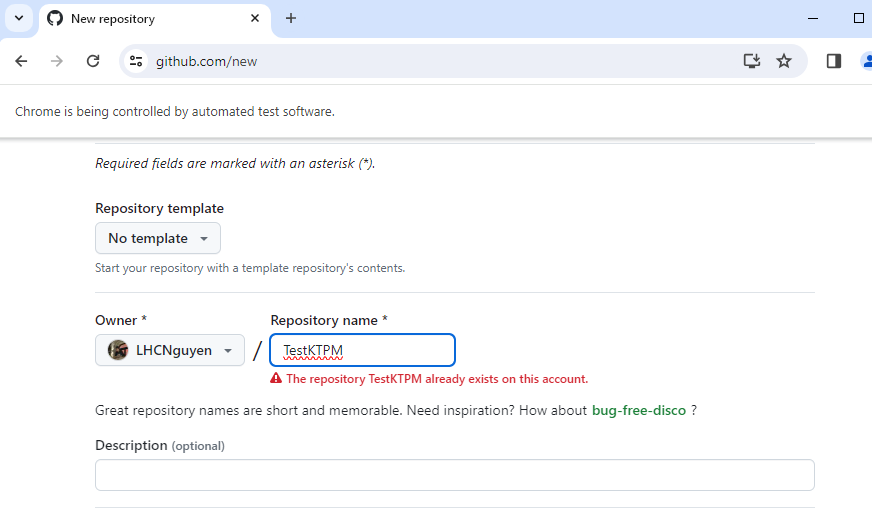
nameRepo.SendKeys("TestKTPM");

IWebElement createRepo = Nguyen\_36.FindElement(By.ClassName("kLvpkX"));

createRepo.Click();

}

Kết quả khi chạy Test Case 3:



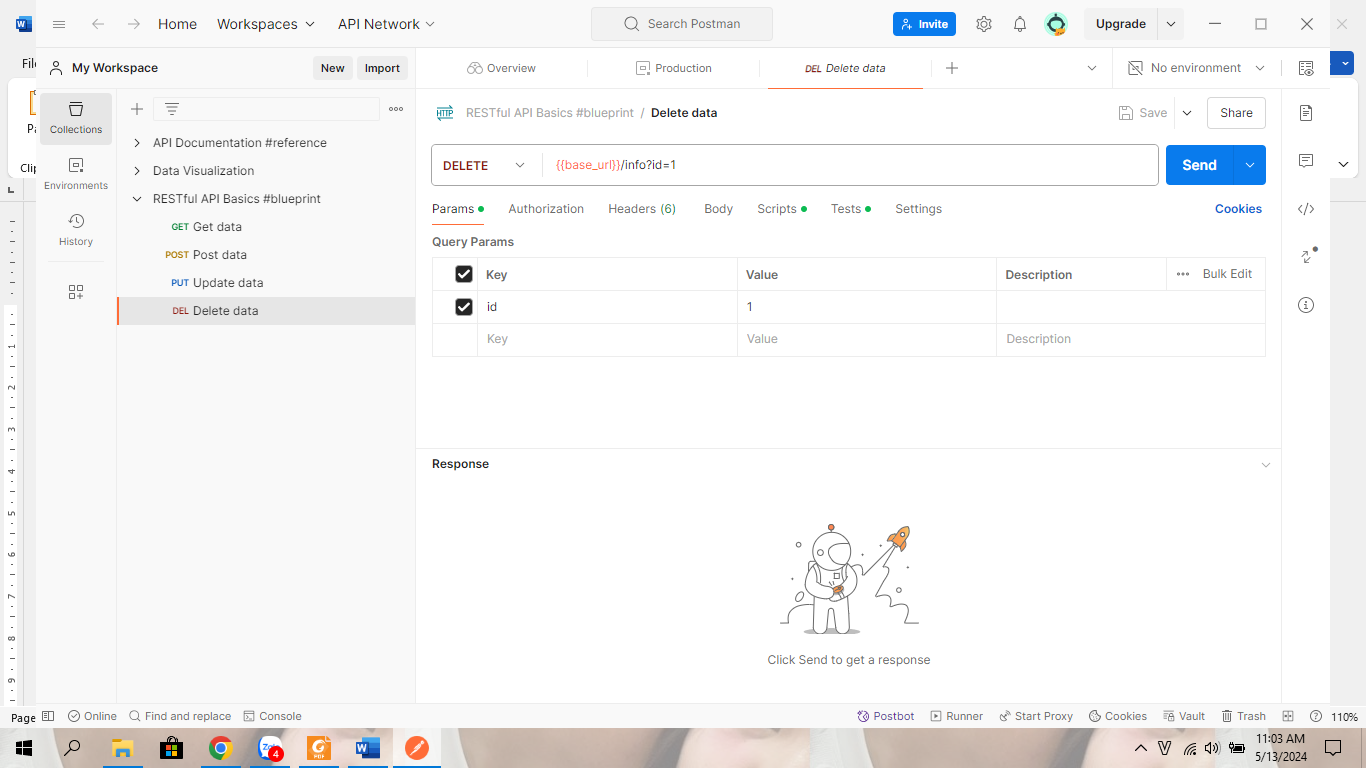
# **API TỰ TẠO VÀ TEST TRÊN POSTMAN**

## Chuẩn bị môi trường test

## Download ứng dụng Postman

tại đây: <https://www.postman.com/downloads/>

Sau khi tải về và tiến hành đăng nhập



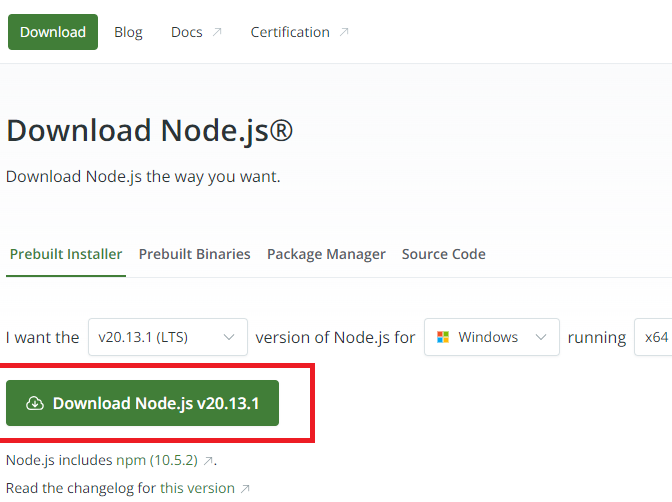
Sau khi cài đặt App Postman thành công

## Tạo một API để test

Đầu tiên cài JSON SERVER

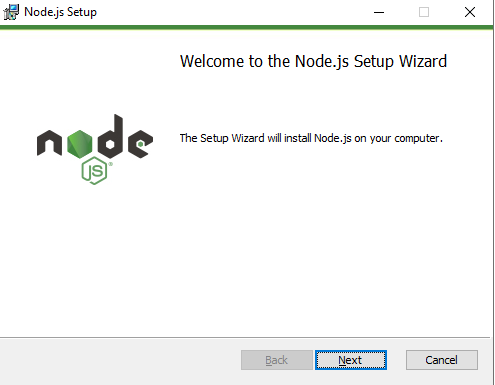
## Download Node.js

qua link sau: <https://nodejs.org/en/download>



Nhấn Download để tải về máy và cài đặt

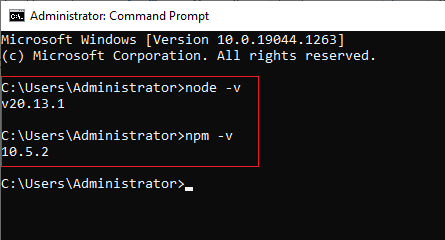
Sau khi tải về, cứ nhấn next cho đến khi finish



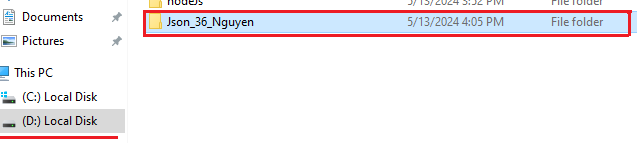
Sau khi cài đặt hoàn tất

## Vào cmd để cài đặt JSON SERVER

kiểm tra thử phiên bản của JSON SERVER: node -v và npm -v

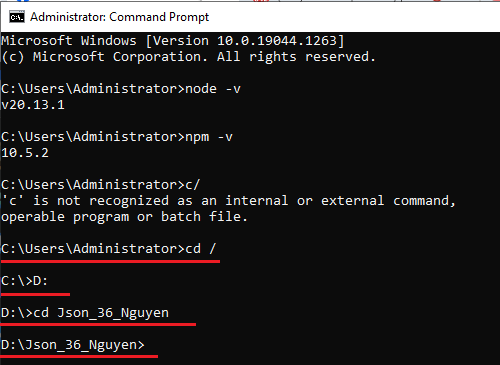


Sau đó tạo 1 folder để cài gói NODE package manager (NPM): D:\Json\_36\_Nguyen

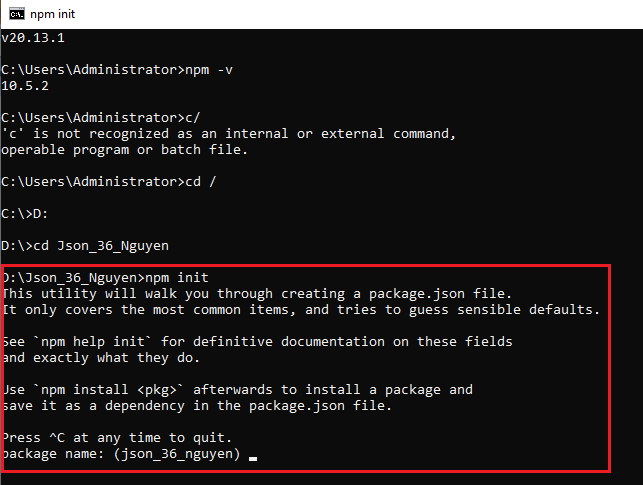


Sau khi tạo xong 1 folder để chứa gói NODE, vào cmd để thực hiện cài đặt: npm init

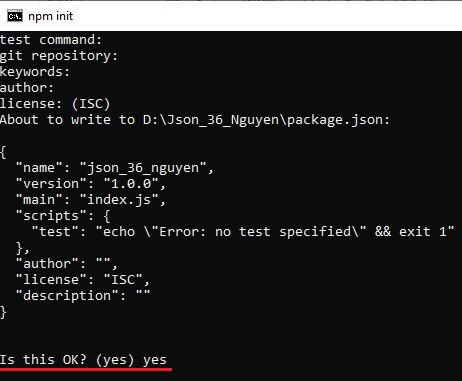
Vì chỉ cài gói NODE trong 1 folder nên phải di chuyển đến đúng folder muốn cài



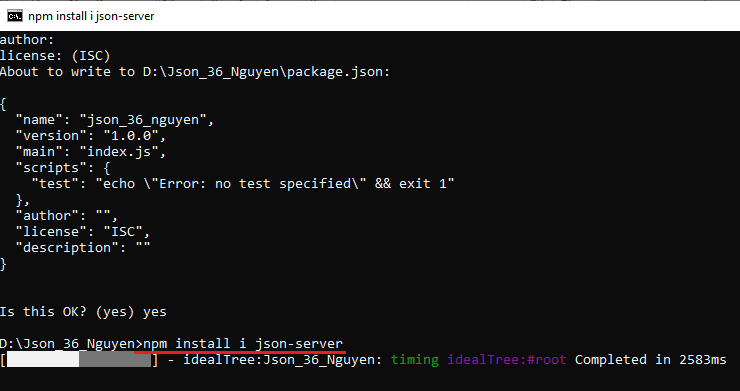
Sau khi di chuyển đến đúng folder muốn cài thì sử dụng lệnh: npm init



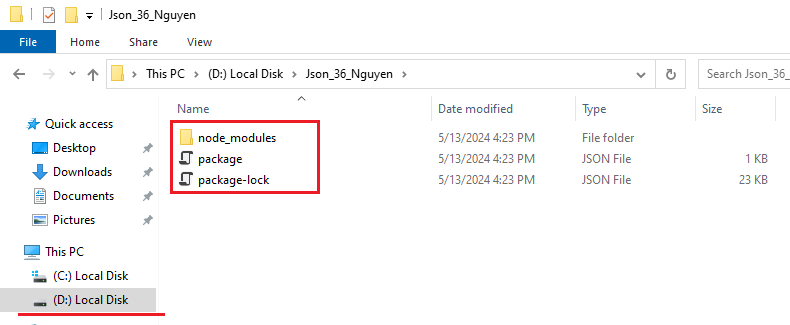
Đến đây cứ nhấn enter cho đến khi hiện chữ Yes: phải nhập chữ Yes vào. Nếu không nhập sẽ bị báo lỗi



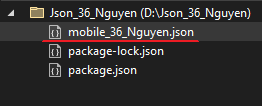
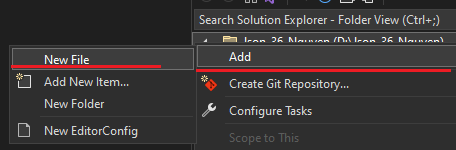
Sau đó gõ: npm install I json-server



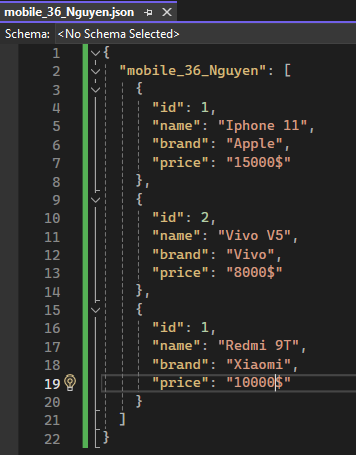
Sau khi hoản tất, vào folder D:\Json\_36\_Nguyen để kiểm tra Package đã cài



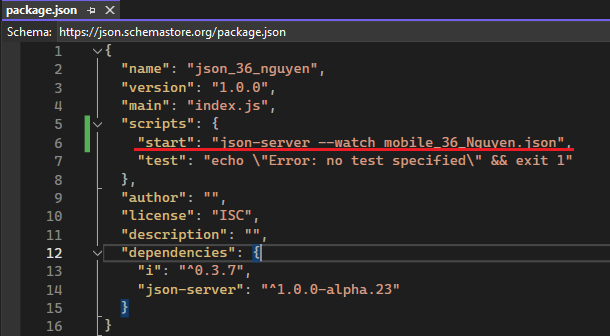
Tạo file mobile\_36\_Nguyen.json thành API trên server ảo của mình



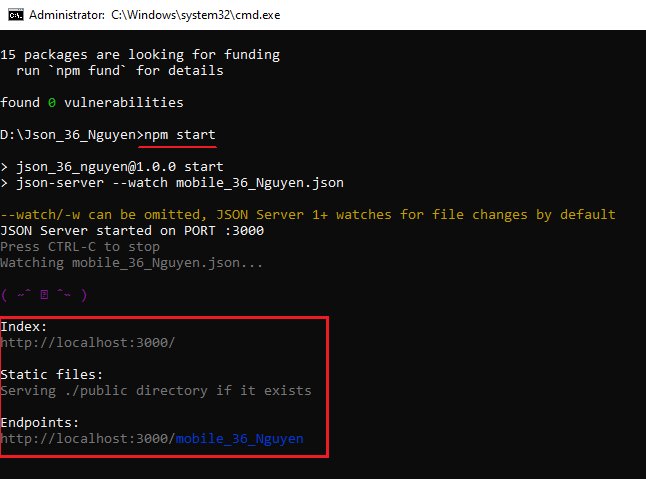
Sau khi thêm file mobile\_36\_Nguyen,json ta tiến hành mở file và tạo cái trường trong file gồm: id, name, brand, price



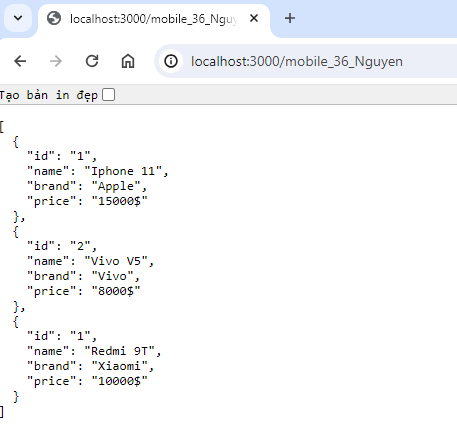
Sau đó mở file package.json, để thêm dòng: "start": "json-server --watch mobile\_36\_Nguyen.json" ở trong scripts.



Quay lại cmd gõ: npm start. Sau đó copy đường link dán vào web rồi chạy.



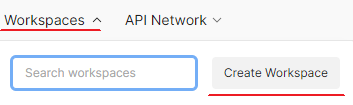
Sau khi open sẽ thấy API của mình mới tạo ra



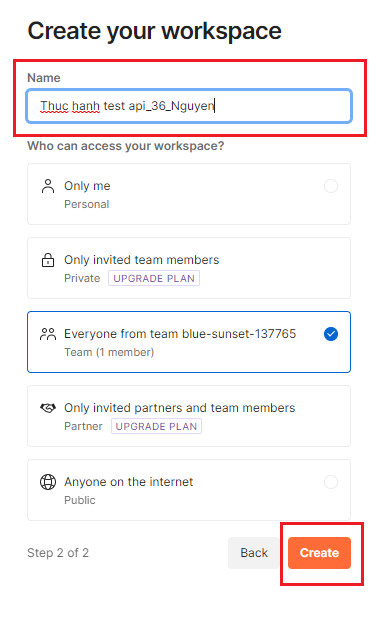
Sau khi tạo API hoàn tất

## Test trên API vừa tạo

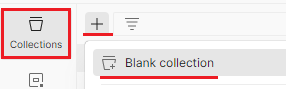
## Vào Postman. Tạo mới Workspace



Đặt tên workspace là: thuc hanh test api\_36\_Nguyen. Nhấn Create để tạo



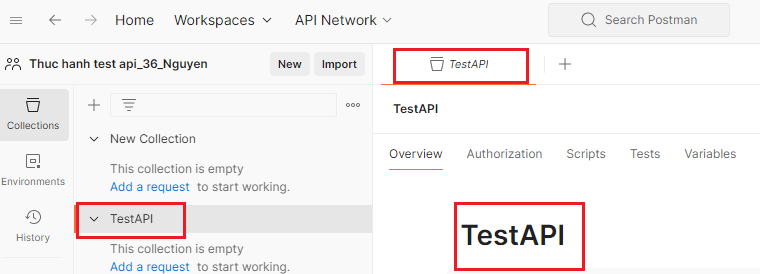
Sau khi tạo thành công Workspace. Chúng ta sẽ tạo 1 Collections: Nhấn vào Collections => Nhấn dấu cộng => Chọn Blank Collection



Đặt tên Collection là: TestAPI



Tạo thành công

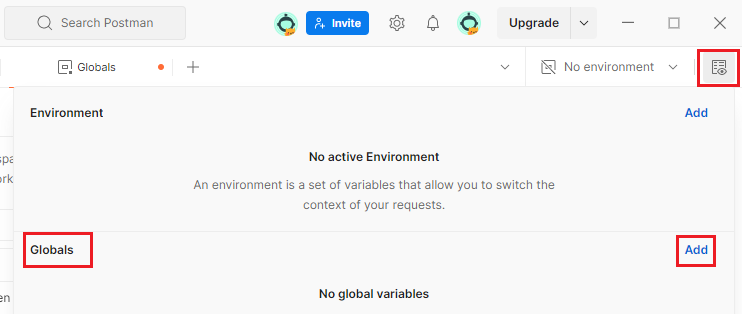


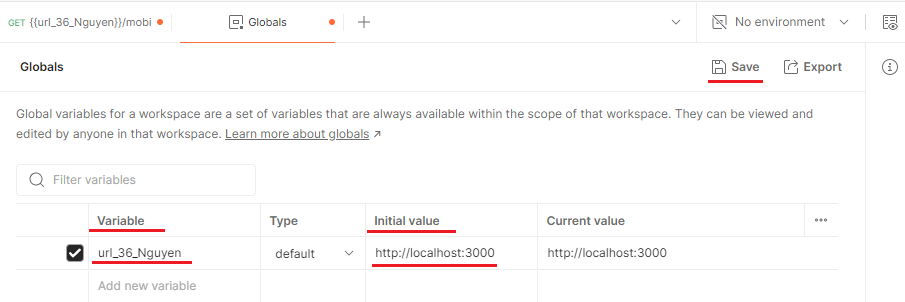
### Thiết lập biến địa chỉ.

Mục đích thiết lập biến địa chỉ là để dễ dàng sửa đổi url khi bị thay đổi. Thay vì phải truy cập từng cái phương thức để sửa lại url thì chỉ cần vào biến địa chỉ để sửa url. Biến địa chỉ sẽ tự động cập nhật url trong các phương thức: get, post……

Cách cài đặt:

Nhấn vào nút Enviroment quick look=> Ở ô Globals nhấn Add=> Ở cộtVariables nhập tên biến là: url\_36\_Nguyen => Dán đường link API: (http://localhost:3000/) đã tạo ở trên vào ô Initial Value => Sau đó nhấn save

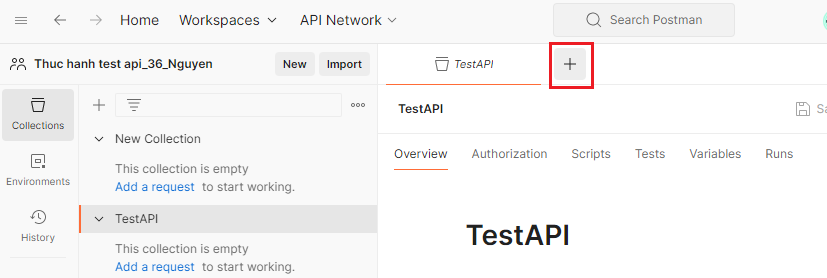




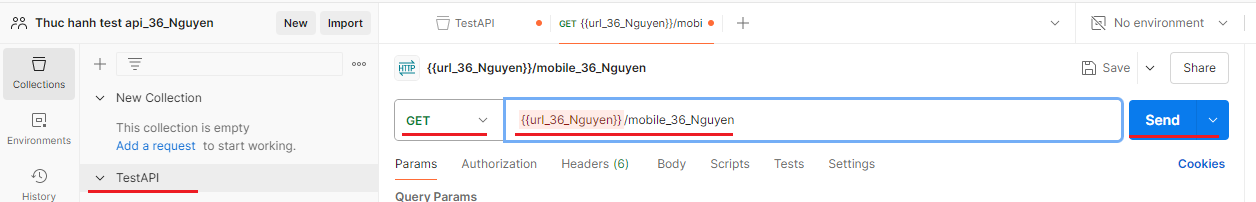
## Thực hiện viết các test case cho Request GetProducts

### Tiến hành tạo GET để test API.

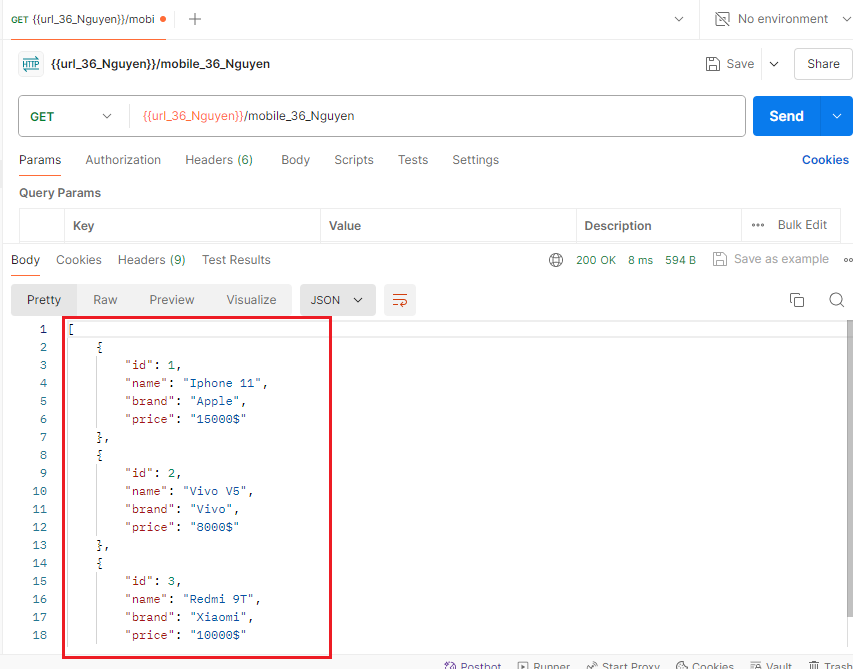
Nhấn vào dấu cộng để tạo GET



Sau đó chọn GET, sử dụng biến địa chỉ mới tạo ở trên, Nhấn Send để lấy dữ liệu API về



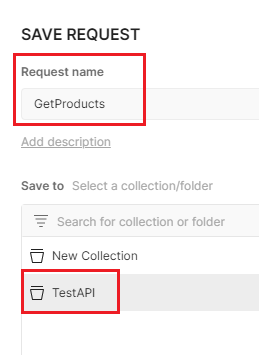
Dữ liệu được lấy về



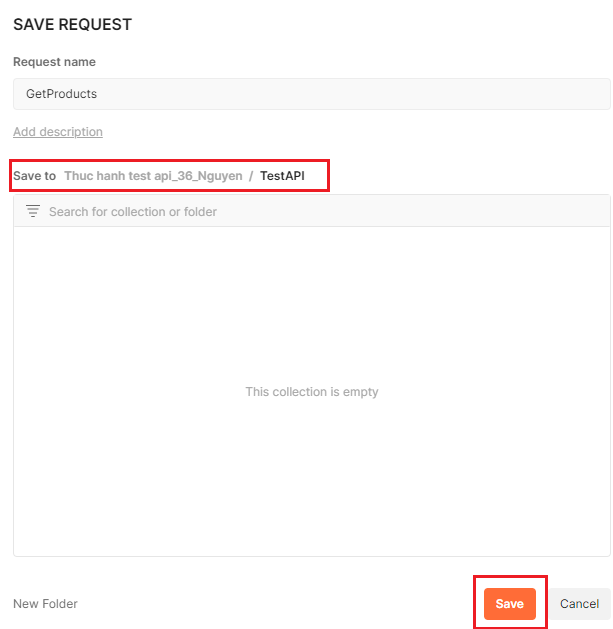
Sau đó nhấn save để lưu.



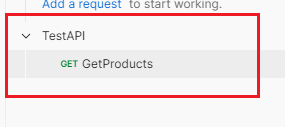
Đặt tên là GetProducts và chọn lưu ở TestAPI đã tạo ở trên.



Sau đó nhấn save



Save thành công

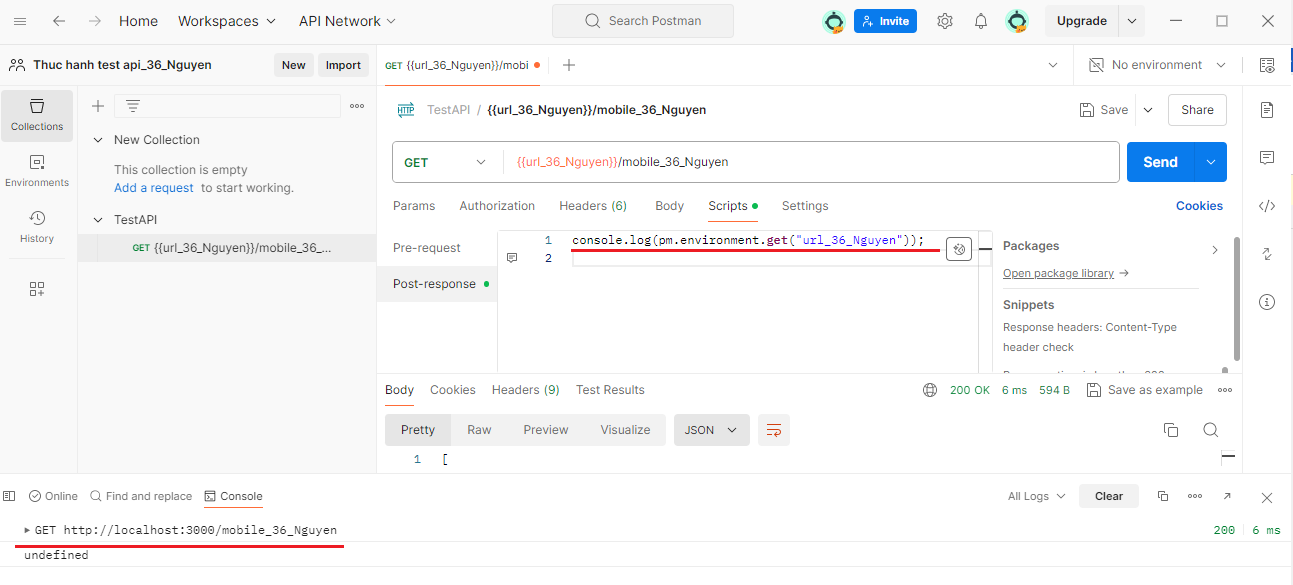


## Viết một số test case đơn giản:

### Test case 1.1: kiểm tra xem biến địa chỉ có đúng đường link localhost không

console.log(pm.environment.**get**("url\_36\_Nguyen"));

Hình ảnh và kết quả test case 1.1

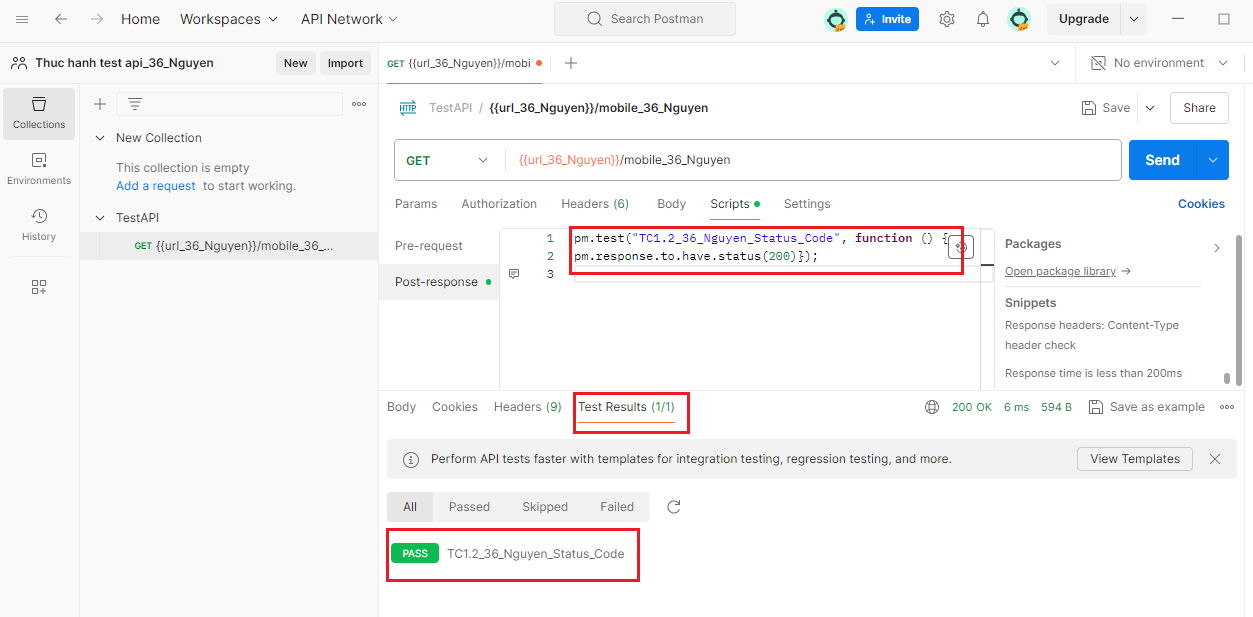


### Test case 1.2: kiểm tra Status Code có phải là 200 hay không

pm.test("TC1.2\_36\_Nguyen\_Status\_Code", **function** () {

pm.response.to.have.status(200)});

Hình ảnh và kết quả test case 1.2

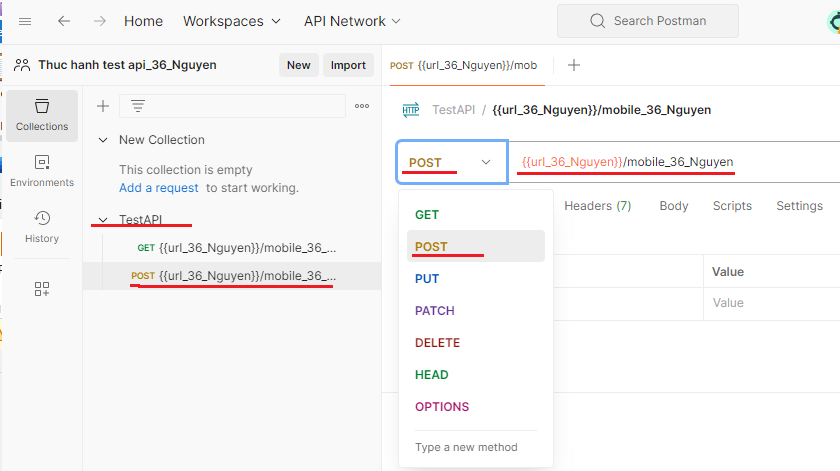


## Thực hiện viết các test case cho Request PostProducts

### Tiến hành tạo POST để test API.

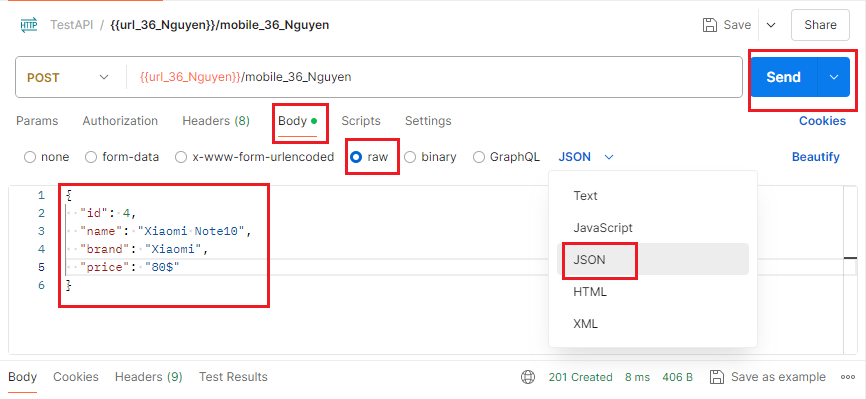
Nhấn vào dấu cộng để tạo POST

Các bước tạo: nhấn vào dấu cộng => Sau đó chọn POST => sử dụng biến địa chỉ đã tạo: ({{url\_36\_Nguyen}}/mobile\_36\_Nguyen) Sau đó nhấn Ctrl + S để lưu Request.



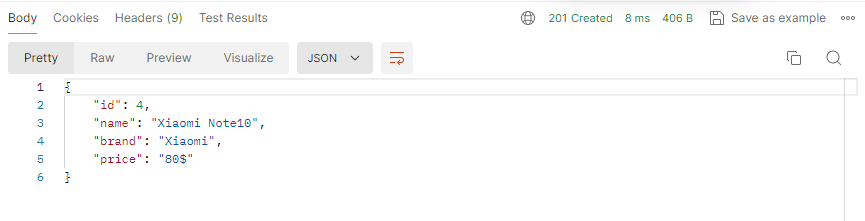
Sau khi tạo xong ta nhấn vào: Body => Chọn raw => chọn JSON nhập các values tương ứng đã tạo API để thêm vào.

API đã tạo gồm 4 values: id ,name, brand, price. Sau khi thêm tất cả các values nhấn Send để gửi.



Kết quả sau khi nhấn Send

Dữ liệu đã được thêm vào



## Viết một số test case đơn giản

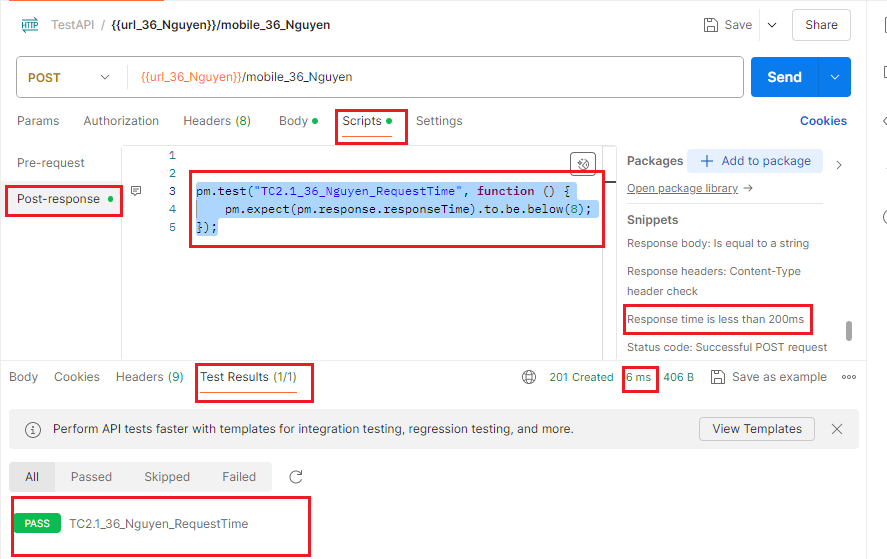
### Test case 2.1: Kiểm tra thời gian phản hồi có nhỏ hơn 8ms không

pm.test("TC2.1\_36\_Nguyen\_RequestTime", **function** () {

    pm.expect(pm.response.responseTime).to.be.below(8);

});

Hình ảnh và kết quả test case 2.1:



### Test case 2.2: Kiểm tra Status có phải 201 202 không

pm.test("TC\_2.2\_36\_Nguyen\_Status\_201\_202", **function** () {

    pm.expect(pm.response.code).to.be.oneOf([201, 202]);

});

Hình ảnh và kết quả test case 2.2:

