

BÁO CÁO ĐỀ TÀI

KIỂM THỬ WEBSITE THEBLUETSHIRT

Mục lục

Chương 1: NUNIT	1
1. Mô tả bài toán giải phương trình bậc 2	1
2. Xây dựng test case	1
3. Tạo Windows Form giải phương trình bậc 2	2
Bước 1: Mở Visual studio 2019, chọn Create a new project	2
Bước 2: Tạo 1 project C# là Windows Form Application (.NET Framework) -> next	2
Bước 3: Đặt tên project, chọn đường dẫn lưu trữ, đặt tên solution và chú ý Framework -> Create	3
Bước 4: Thiết kế giao diện giải phương trình bậc 2	4
Bước 5: Tạo 1 class: Click chuột phải vào project -> Add -> Class -> Đặt tên cho class -> Add	5
Bước 6: Tại class GiaiPTB2_45_Nhan, viết code giải phương trình bậc 2	6
Bước 7: Vào Form1_45_Nhan viết code sự kiện click chuột vào nút Giải_45_Nhan , để giải phương trình bậc 2	7
4. Tạo project kiểm thử để kiểm thử các test case của phương trình bậc 2: .	9
Bước 1: Click chuột phải vào Solution ‘KiemThuBTL_45_Nhan’ -> Add -> New Project	9

Bước 2: Tìm “unit” và chọn Unit Test Project (.NET Framework)	10
Bước 3: Đặt tên project, kiểm tra đường dẫn, chú ý Framework -> Create	11
Bước 4: Thực hiện Add Reference để tham chiếu đến project cần thực hiện Unit Test:	11
Bước 5: Viết code các test case kiểm thử	12
Bước 6: Chạy các test case đã viết: Nhấn Test -> Run All Test, xem kết quả chạy các test case	14
5. Thực thi test với dữ liệu test có sẵn	15
Bước 1: Tại project PTB2Tester_45_Nhan, tạo 1 thư mục Data_45_Nhan	15
Bước 2: Click chuột phải vào thư mục Data_45_Nhan -> Add -> New Item -> Đặt tên tập tin có đuôi là .csv -> Add	15
Bước 3: Click chuột phải vào tập tin DataPTB2_45_Nhan.csv -> Properties thiết lập thuộc tính “Copy to Output Directory” thành “Copy always” để tập tin này được sao chép vào thư mục bin khi thực hiện build project	16
Bước 4: Tìm và thực hiện Add Reference “System.Data” vào project unit test	16
Bước 5: Nhập dữ liệu vào tập tin DataPTB2_45_Nhan.csv	17
Bước 6: Viết test case sử dụng dữ liệu này để kiểm thử	17
Bước 7: Nhấn Test -> Run All Test, xem kết quả chạy các test case ..	18
Chương 2: SELENIUM WEBDRIVER	19
1. Mô tả chức năng đăng nhập	19
2. Xây dựng test case	20

3. Tạo project unit test Thebluetshirt_45_Nhan để kiểm thử các test case của chức năng đăng nhập	21
Bước 1: Tương tự như vậy, click chuột phải vào Solution ‘KiemThuBTL_45_Nhan’ -> Add -> New Project -> Tìm và chọn Unit Test Project (.NET Framework) -> Next -> Đặt tên project, kiểm tra đường dẫn, chú ý Framework -> Create	21
Bước 2: Kiểm tra Browser: Chrome trên máy tính	21
Bước 3: Cài đặt thư viện Selenium	22
Bước 4: Tại vùng sử dụng thư viện, thêm các thư viện như sau:	26
4. Lấy các elements	27
Bước 1: Click phải chuột vào dấu X trên thẻ “div” -> nhấn Inspect..	27
Bước 2: Lấy element (LinkText) của nút “TÀI KHOẢN”	28
Bước 3: Lấy element Id của input “Email”	28
Bước 4: Lấy element XPath của input “Mật khẩu”	29
Bước 5: Lấy element ClassName của nút “ĐĂNG NHẬP”	29
5. Chuẩn bị trước khi test các test case	30
5.1. Thiết lập các biến và phương thức cho class kiểm thử chức năng đăng nhập (UnitTestDangNhap_45_Nhan)	30
5.2. Test với dữ liệu test có sẵn:	31
6. Viết các test case cho chức năng đăng nhập	32
6.1. Test case 1: TestDNThanhCong_45_Nhan()	32
6.2. Các bước thực thi test case với dữ liệu có sẵn	33
Bước 1: Tạo 1 tập tin DataDNThanhCong_45_Nhan.csv, nhập dữ liệu đầu vào	33

Bước 2: Chính thuộc tính “Copy to Output Driectory” thành “Copy always”	34
Bước 3: Viết code cho test case	34
Bước 4: Kết quả chạy test case	35
6.3. Test case 2: TestDNSaiEmail_45_Nhan()	35
6.4. Các bước thực thi	36
Bước 1: Tạo 1 tập tin DataDNSaiEmail_45_Nhan.csv, nhập dữ liệu đầu vào	36
Bước 2: Chính thuộc tính “Copy to Output Driectory” thành “Copy always”	36
Bước 3: Bắt element ClassName của cảnh báo	37
Bước 4: Viết code cho test case	37
Bước 5: Kết quả chạy test case:	38
6.5. Test case 3: TestDNSaiMatKhau_45_Nhan()	38
6.6. Các bước thực thi	39
Bước 1: Tạo 1 tập tin DataDNSaiMatKhau_45_Nhan.csv, nhập dữ liệu đầu vào	39
Bước 2: Chính thuộc tính “Copy to Output Driectory” thành “Copy always”	40
Bước 3: Bắt element Xpath của cảnh báo	40
Bước 4: Viết code cho test case	41
Bước 5: Kết quả chạy test case:	41
6.7. Test case 4: TestDNEmailRong_45_Nhan()	42
6.8. Các bước thực thi	43

Bước 1: Tạo 1 tập tin DataDNEmailRong_45_Nhan.csv, nhập dữ liệu đầu vào	43
Bước 2: Chính thuộc tính “Copy to Output Driectory” thành “Copy always”	43
Bước 3: Bắt element Id của cảnh báo thanh input Email	43
Bước 4: Viết code cho test case	44
Bước 5: Kết quả chạy test case	45
6.9. Test case 5: TestDNMatKhauRong_45_Nhan()	45
6.10. Các bước thực thi	46
Bước 1: Tạo 1 tập tin DataDNMatKhauRong_45_Nhan.csv, nhập dữ liệu đầu vào	46
Bước 2: Chính thuộc tính “Copy to Output Driectory” thành “Copy always”	46
Bước 3: Bắt element CssSelector của cảnh báo thanh input Mật khẩu	47
Bước 4: Viết code cho test case	47
Bước 5: Kết quả chạy test case	48
7. Mô tả chức năng tìm kiếm và thêm vào giỏ hàng	49
8. Xây dựng test case	50
9. Tạo 1 tập tin UnitTestTKVaThemGH_45_Nhan.cs để kiểm thử các test case của chức năng tìm kiếm và thêm vào giỏ hàng	51
Bước 1: Click chuột phải project Thebluetshirt_45_Nhan -> Add -> Unit Test	51

Bước 2: Click chuột phải vào unit test vừa tạo -> Rename thành UnitTestTKVaThemGH_45_Nhan.cs	51
Bước 3: Tại vùng sử dụng thư viện, thêm các thư viện như sau:	52
10. Lấy các elements	52
Bước 1: Lấy element ClassName của nút “TÌM KIẾM”	52
Bước 2: Lấy element CssSelector của thanh input Search	53
Bước 3: Lấy element Name của input Search.....	53
11. Chuẩn bị trước khi test các test case	54
12. Viết các test case cho chức năng tìm kiếm và thêm vào giỏ hàng.....	56
12.1. Test case 1: TestTKCoSPThemGH_45_Nhan()	56
12.2. Các bước thực thi test case	57
Bước 1: Bắt element Xpath của kết quả tìm kiếm	57
Bước 2: Bắt element Xpath của sản phẩm tìm thấy đầu tiên	58
Bước 3: Bắt element Id của button thêm vào giỏ	58
Bước 4: Viết code cho test case	59
Bước 5: Kết quả chạy test case	60
12.3. Test case 2: TestTKKhongSP_45_Nhan()	60
12.4. Các bước thực thi test case	61
Bước 1: Bắt element Xpath của kết quả tìm kiếm	61
Bước 2: Viết code cho test case	61
Bước 3: Kết quả chạy test case	62
12.5. Test case 3: TestTKRong_45_Nhan()	62
12.6. Các bước thực thi test case	63

Bước 1: Bắt element ClassName của cảnh báo	63
Bước 2: Viết code cho test case	64
Bước 3: Kết quả chạy test case	64
1. Test API bằng Postman	65
1.1. Chuẩn bị	65
Bước 1: Cài đặt postman, mở app Postman, tạo 1 Workspace mới	65
Bước 2: Đặt tên Workspace, có thể viết description, chọn type, nhấn create workspaces	65
Bước 3: Tạo 1 collections mới	67
Bước 4: Đặt tên, có thể viết description -> Create	67
1.2. Phương thức GET	68
Bước 1: Click phải vào collection vừa tạo -> Add Request.....	68
Bước 2: Đặt tên request -> Save vào collection	69
Bước 3: Copy đường dẫn vào thanh URL để lấy list posts -> Save -> Send, xem kết quả ở phần “Body”	70
Bước 4: Chọn tab Tests -> viết testscript cho phương thức GET	71
Bước 5: Nhấn Save -> Send -> Xem ở tab Test Results	72
1.3. Phương thức POST	73
Bước 1: Tạo 1 request dạng POST	73
Bước 2: Copy đường dẫn vào URL, mở tab Body -> raw-> dán code từ trang web vào -> Save -> Send	73
Bước 3: Chọn tab Tests -> viết testscript cho phương thức POST	73
Bước 4: Nhấn Save -> Send -> Xem ở tab Test Results	74

1.4. Phương thức PUT	74
Bước 1: Tạo 1 resquest dạng PUT	74
Bước 2: Copy đường dẫn vào URL, mở tab Body -> raw-> dán code từ trang web vào -> Save -> Send	74
Bước 3: Chọn tab Tests -> viết testscript cho phương thức PUT	75
Bước 4: Nhấn Save -> Send -> Xem ở tab Test Results	76
1.5. Phương thức DELETE	76
Bước 1: Tạo 1 resquest dạng DELETE	76
Bước 2: Copy đường dẫn vào URL -> Save -> Send	76
Bước 3: Chọn tab Tests -> viết testscript cho phương thức DELETE	77
Bước 4: Nhấn Save -> Send -> Xem ở tab Test Results	77
2. Tạo API với Nodejs	78
2.1. Lấy link api của riêng mình bằng nodejs	78
Bước 1: Truy cập vào link https://nodejs.org/en/download, tải file cài đặt, nhấn next -> đến cuối -> finish	78
Bước 2: Mở cmd, vào thư mục đã cài đặt để kiểm tra version ...	78
Bước 3: Cài đặt gói (npm init) NODE package manager(NPM): Tạo folder D:\installnodejs_45_Nhan\jsonserver_45_Nhan	79
Bước 4: Enter đến cuối nhập “yes” -> enter để tạo project	79
Bước 5: Sau đó gõ npm install i json-server	80
Bước 6: Tạo file ualbums_45_Nhan.json thành API trên server ảo của mình	81
Bước 7: Mở file package.json thêm dòng start vào	83

Bước 8: Mở cmd, chạy lệnh	84
Bước 9: Thu được API, chạy web kiểm tra	84
2.2. Test trên api vừa tạo	85
● GET	86
Bước 1: Add Request GET, copy đường dẫn vào thanh URL -> Save -> Send, xem kết quả ở phần “Body”	86
Bước 2: Chọn tab Tests -> viết testscript cho phương thức GET	86
Bước 3: Nhấn Save -> Send -> Xem ở tab Test Results	87
● POST	88
Bước 1: Add Request POST, copy đường dẫn vào URL, mở tab Body -> raw-> dán code từ trang web vào -> Save -> Send	88
Bước 2: Chọn tab Tests -> viết testscript cho phương thức POST	88
Bước 3: Nhấn Save -> Send -> Xem ở tab Test Results	89
● PUT	90
Bước 1: Add Request PUT, copy đường dẫn vào URL, mở tab Body -> raw-> dán code từ trang web vào -> Save -> Send	90
Bước 2: Chọn tab Tests -> viết testscript cho phương thức PUT	91
Bước 3: Nhấn Save -> Send -> Xem ở tab Test Results	91
● DELETE	92
Bước 1: Add Request POST, copy đường dẫn vào URL -> Save -> Send	92
Bước 2: Chọn tab Tests -> viết testscript cho phương thức DELETE	93
Bước 3: Nhấn Save -> Send -> Xem ở tab Test Results	93

Chương 1: NUNIT

1. Mô tả bài toán giải phương trình bậc 2

- Phương trình bậc 2 là phương trình có dạng tổng quát $ax^2 + bx + c = 0$ (*) với a, b, c là các hằng số và $a \neq 0$. x là giá trị cần tìm sao cho khi thay vào phương trình (*) được kết quả bằng 0.
- Để giải phương trình bậc 2, cần tính $\Delta = b^2 - 4*a*c$, để xác định tính chất của nghiệm:

- Nếu $\Delta > 0$: phương trình có 2 nghiệm phân biệt, $x_1 = \frac{-b + \sqrt{\Delta}}{2a}$ và

$$x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

- Nếu $\Delta < 0$: phương trình vô nghiệm
- Nếu $\Delta = 0$: phương trình có nghiệm kép, $x_1 = x_2 = \frac{-b}{2a}$

2. Xây dựng test case

- Với mô tả của bài toán trên, em xây dựng 4 test case và ghi chú như sau:

Test case 1: Test2Nghiem_45_Nhan()

Test case 2: TestVoNghiem_45_Nhan()

Test case 3: TestNghiemKep_45_Nhan()

Test case 4: TestAKhac0_45_Nhan()

		Test case 1	Test case 2	Test case 3	Test case 4
Các điều kiện	$a \neq 0?$	T	T	T	F
	$\Delta > 0?$	T	F	F	-
	$\Delta < 0?$	F	T	F	-
	$\Delta = 0?$	F	F	T	-
Các hành động	Kết quả	KQ1	KQ2	KQ3	KQ4

Hình 1.2.1: Các test case cho bài toán giải phương trình bậc 2

Xác định các điều kiện: $a \neq 0$, $\Delta > 0$, $\Delta < 0$ và $\Delta = 0$

Mỗi đầu vào trên nhận 1 trong 2 giá trị: đúng (T), sai (F)

KQ1: phương trình có 2 nghiệm phân biệt, $x_1 = \frac{-b + \sqrt{\Delta}}{2a}$ và $x_2 = \frac{-b - \sqrt{\Delta}}{2a}$

KQ2: phương trình vô nghiệm

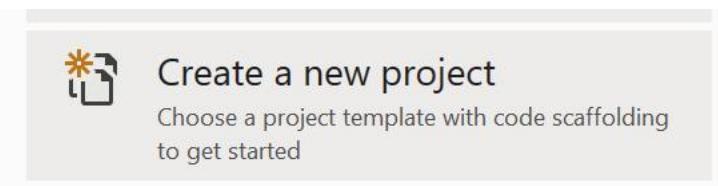
KQ3: phương trình có nghiệm kép, $x_1 = x_2 = \frac{-b}{2a}$

KQ4: in ra câu "Phuong trinh bac 2, a khac 0"

Hình 1.2.2: Ghi chú test case cho bài toán giải phương trình bậc 2

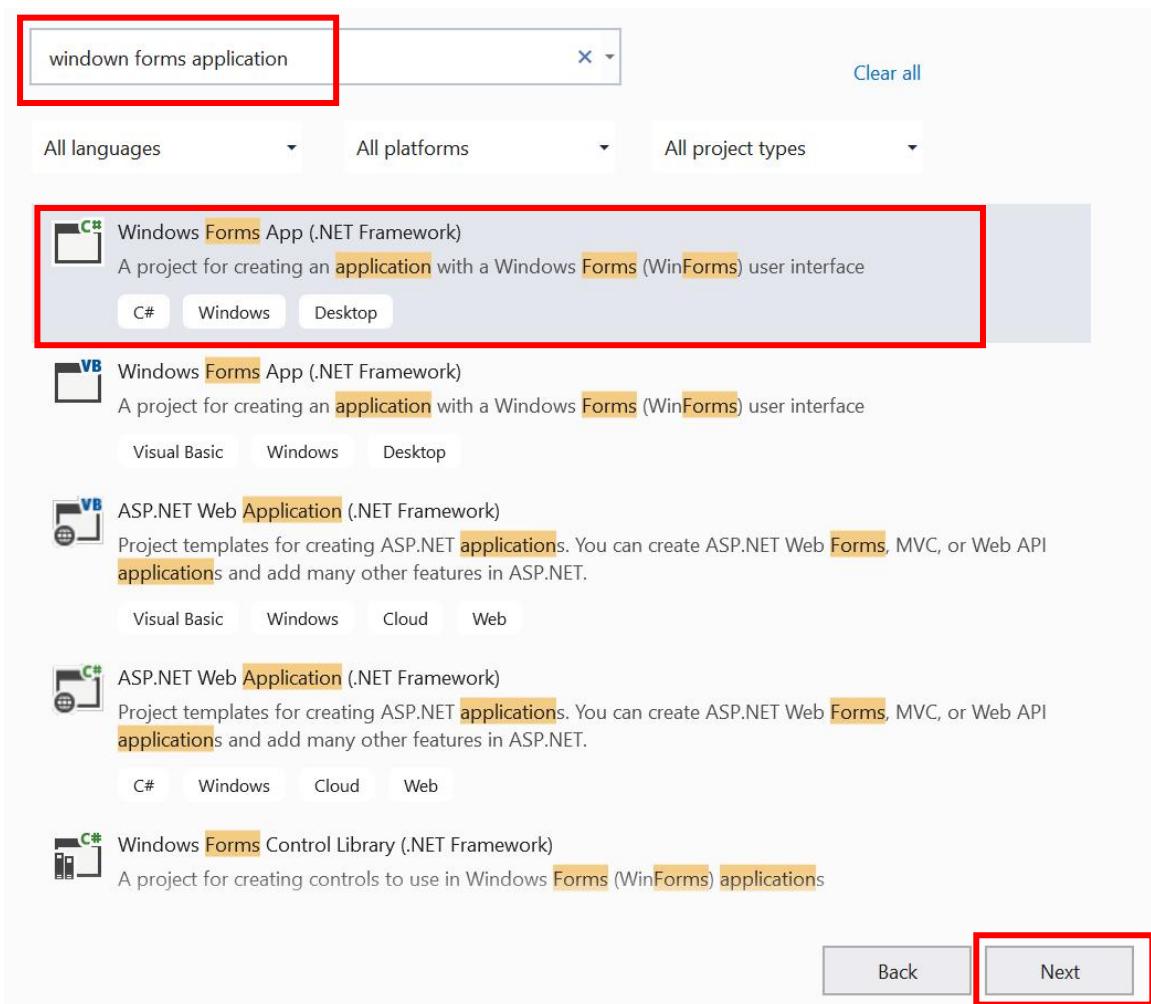
3. Tạo Windows Form giải phương trình bậc 2

Bước 1: Mở Visual studio 2019, chọn **Create a new project**



Hình 1.3.1: Chọn vào "Create a new project"

Bước 2: Tạo 1 project C# là **Windows Form Application (.NET Framework)** -> next

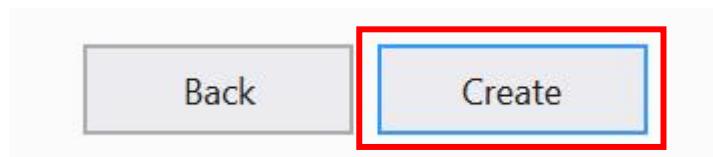


Hình 1.3.2: Chọn dạng project C# là “Windows Form Application”

Bước 3: Đặt tên project, chọn đường dẫn lưu trữ, đặt tên solution và chú ý Framework -> Create

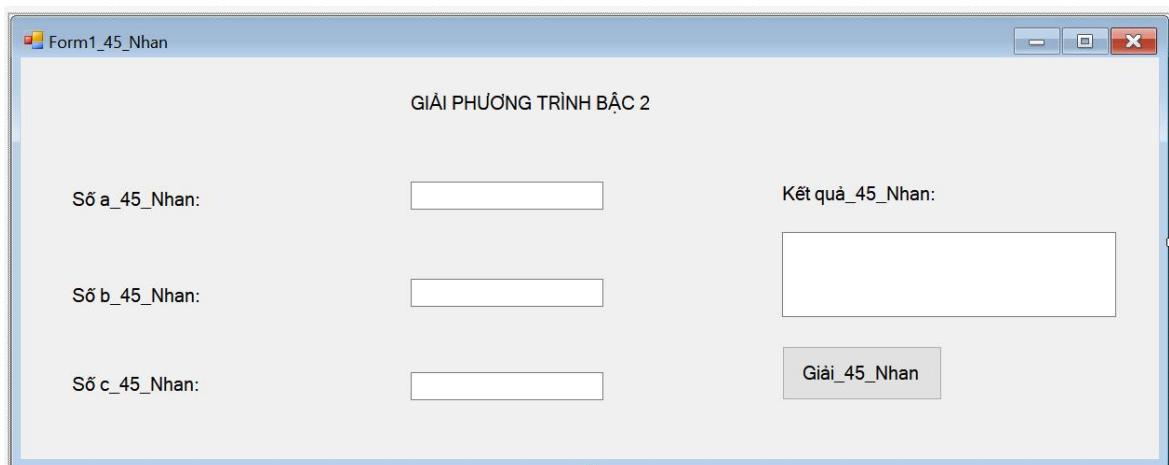


Hình 1.3.3: Tạo thông tin project



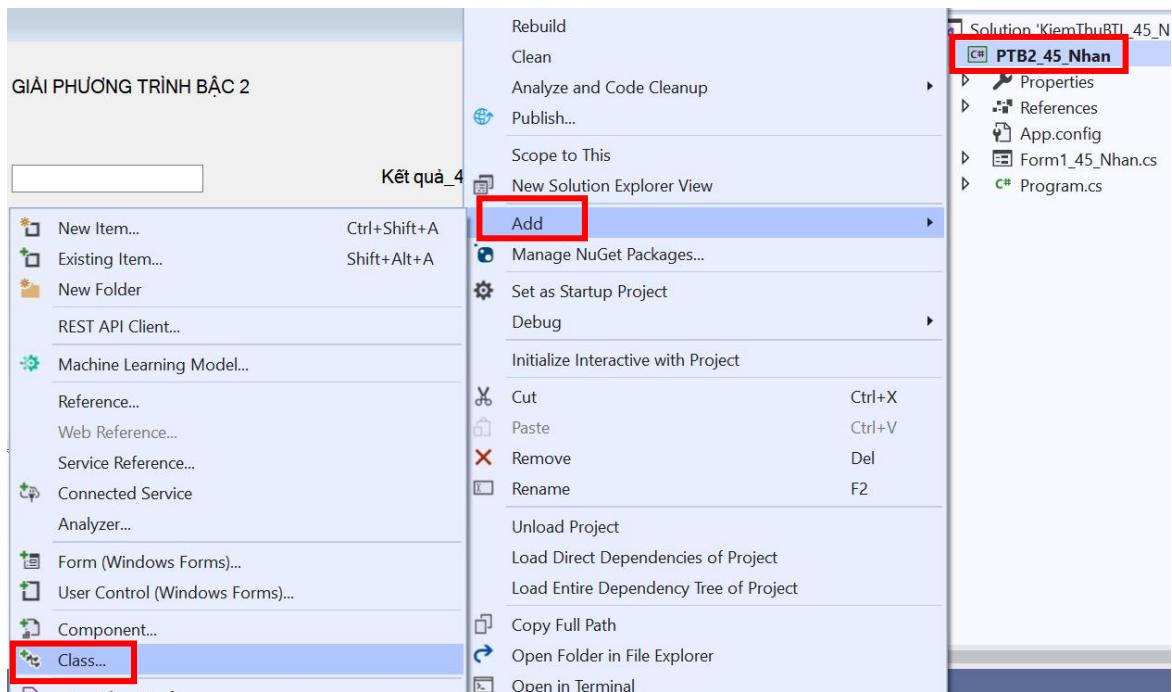
Hình 1.3.4: Chọn vào “Create”

Bước 4: Thiết kế giao diện giải phương trình bậc 2

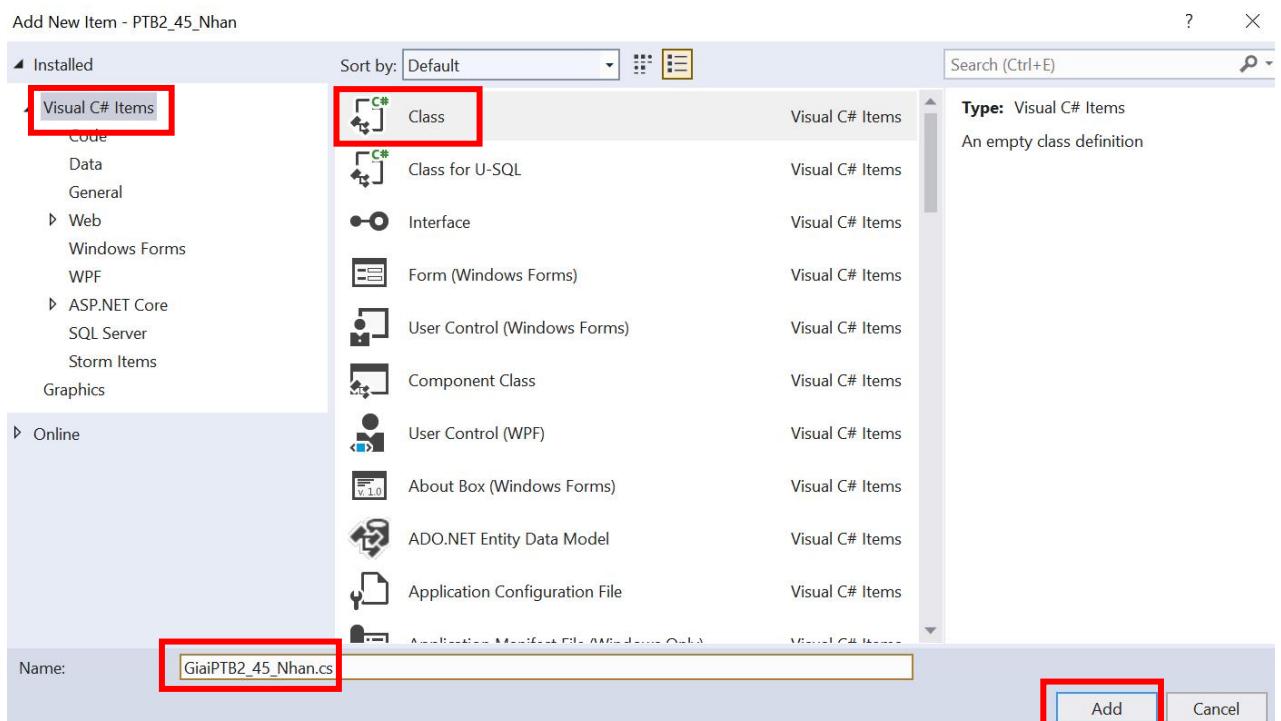


Hình 1.3.5: Giao diện cho bài toán giải phương trình bậc 2

Bước 5: Tạo 1 class: Click chuột phải vào project -> Add -> Class -> Đặt tên cho class -> Add



Hình 1.3.6: Tạo 1 file class



Hình 1.3.7: Đặt tên file class

Bước 6: Tại class GiaiPTB2_45_Nhan, viết code giải phương trình bậc 2

- Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PTB2_45_Nhan
{
    public class GiaiPTB2_45_Nhan
    {
        // Khai báo thuộc tính của lớp gồm 3 biến STT_Ten: 45_Nhan
        private double a_45_Nhan, b_45_Nhan, c_45_Nhan;

        // Tạo phương thức khởi tạo có 3 tham số STT_Ten: 45_Nhan
        public GiaiPTB2_45_Nhan(double a_45_Nhan, double b_45_Nhan, double c_45_Nhan)
        {
            this.a_45_Nhan = a_45_Nhan;
            this.b_45_Nhan = b_45_Nhan;
            this.c_45_Nhan = c_45_Nhan;
        }

        // Tạo phương thức giải bài toán phương trình bậc 2 STT_Ten: 45_Nhan
        public string Giai_45_Nhan()
        {
            string kq_45_Nhan = "";
            if (a_45_Nhan != 0) // Phương trình bậc 2 số a khác 0 STT_Ten: 45_Nhan
            {
                // Math.Pow để tính lũy thừa STT_Ten: 45_Nhan
                double delta_45_Nhan = Math.Pow(b_45_Nhan, 2) - (4 * a_45_Nhan * c_45_Nhan);
            }
        }
}
```

```

        if (delta_45_Nhan > 0) // Phương trình có 2 nghiệm phân biệt STT_Ten: 45_Nhan
        {
            //Math.Round để làm tròn số thực đến số thập phân thứ 2 STT_Ten: 45_Nhan
            double x1_45_Nhan = Math.Round((-b_45_Nhan + Math.Sqrt(delta_45_Nhan)) / (2 * a_45_Nhan), 2);

            double x2_45_Nhan = Math.Round((-b_45_Nhan - Math.Sqrt(delta_45_Nhan)) / (2 * a_45_Nhan), 2);

            kq_45_Nhan = x1_45_Nhan.ToString() + " ; " + x2_45_Nhan.ToString();
        }

        else if (delta_45_Nhan < 0) //Phương trình vô nghiệm STT_Ten: 45_Nhan
        {
            kq_45_Nhan = "Phuong trinh vo nghiem";
        }

        else // delta_45_Nhan = 0 -> phương trình có 1 nghiệm kép STT_Ten: 45_Nhan
        {
            double x_45_Nhan = Math.Round((-b_45_Nhan) / (2 * a_45_Nhan), 2);

            kq_45_Nhan = x_45_Nhan.ToString();
        }

    }

    else
    {
        kq_45_Nhan = "Phuong trinh bac 2, a khac 0";
    }

    return kq_45_Nhan;
}

}

```

Bước 7: Vào Form1_45_Nhan viết code sự kiện click chuột vào nút
Giải_45_Nhan , để giải phương trình bậc 2

- Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PTB2_45_Nhan
{
    public partial class Form1_45_Nhan : Form
    {
        public Form1_45_Nhan()
        {
            InitializeComponent();
        }

        // Khai báo biến STT_Ten: 45_Nhan
        double a_45_Nhan, b_45_Nhan, c_45_Nhan;
        string kq_45_Nhan;

        private void btnGiai_45_Nhan_Click(object sender, EventArgs e)
        {
            try // Bảo đảm người dùng nhập đúng định dạng (số) mới tính toán STT_Ten: 45_Nhan
            {
                a_45_Nhan = double.Parse(txtA_45_Nhan.Text);
                b_45_Nhan = double.Parse(txtB_45_Nhan.Text);
                c_45_Nhan = double.Parse(txtC_45_Nhan.Text);

                GiaiPTB2_45_Nhan pt_45_Nhan = new GiaiPTB2_45_Nhan(a_45_Nhan, b_45_Nhan,
c_45_Nhan);

                // Gọi phương thức Giai_45_Nhan() của class PTB2_45_Nhan STT_Ten: 45_Nhan
            }
        }
    }
}

```

```

        kq_45_Nhan = pt_45_Nhan.Giai_45_Nhan();

        // Ghi kết quả giải được vào txt_45_Nhan STT_Ten: 45_Nhan

        txtKq_45_Nhan.Text = kq_45_Nhan;

    }

    catch (Exception ex)

    {

        // Trả về câu thông báo lỗi nhập không đúng định dạng STT_Ten: 45_Nhan

        txtKq_45_Nhan.Text = ex.Message;

    }

}

}

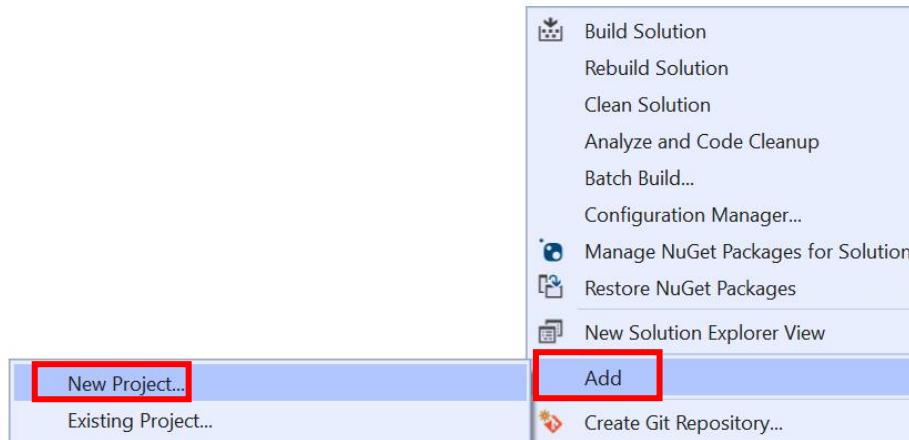
```

4. Tạo project kiểm thử để kiểm thử các test case của phương trình bậc 2:

Bước 1: Click chuột phải vào Solution ‘KiemThuBTL_45_Nhan’ -> Add -> New Project



Hình 1.4.1: Click phải vào Solution



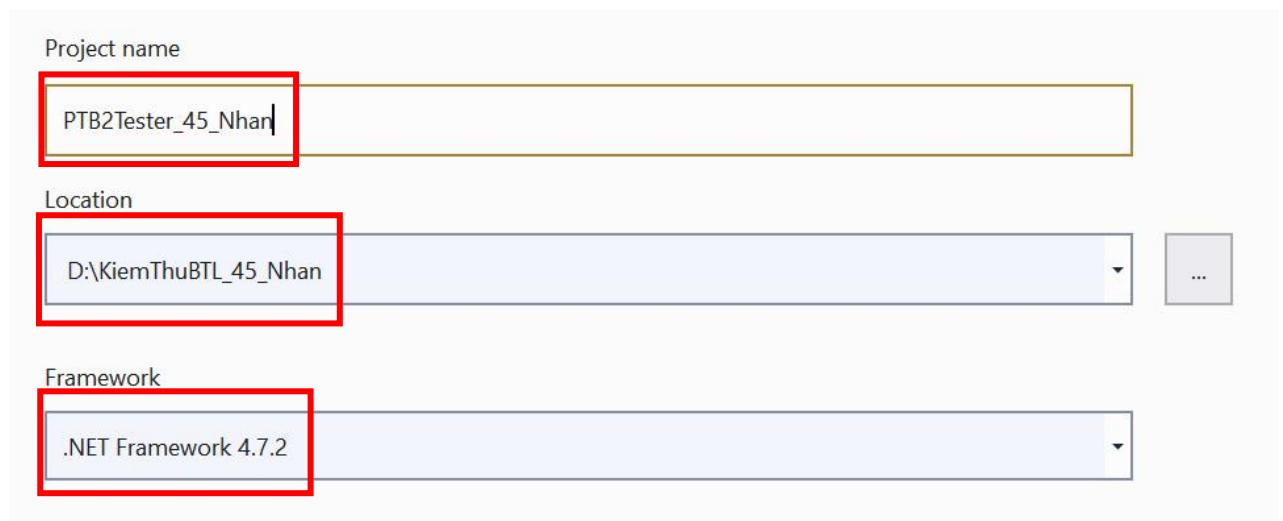
Hình 1.4.2: Chọn “New Project”

Bước 2: Tìm “unit” và chọn Unit Test Project (.NET Framework)

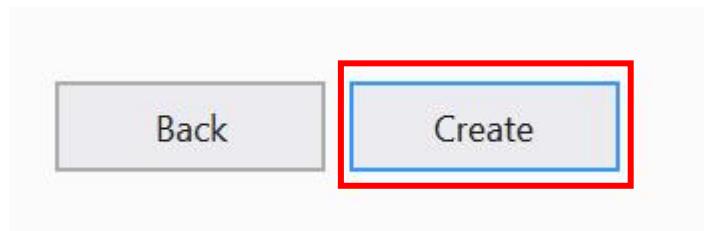
A screenshot of the 'Create New Project' dialog in Visual Studio. The search bar at the top contains the text 'unit' and is highlighted with a red box. Below the search bar are filters for 'All languages', 'All platforms', and 'All project types'. The results section shows several project templates. The first result, 'Unit Test Project (.NET Framework)', is highlighted with a red box. It has a C# icon, a brief description 'A project that contains MSTest unit tests.', and buttons for 'C#' (selected), 'Windows', and 'Test'. Below this are other project types: 'Unit Test Project (.NET Framework)' in VB, 'Native Unit Test Project' in C++, 'U-SQL C# UDO Unit Test Project', and 'U-SQL C# UDO Unit Test Sample Project'. Each has its own icon, description, and language/test platform buttons. In the bottom right corner of the dialog, there is a large blue 'Next' button with a red border.

Hình 1.4.3: Chọn dạng project là Unit Test Project

Bước 3: Đặt tên project, kiểm tra đường dẫn, chú ý Framework -> Create



Hình 1.4.4: Tạo thông tin project



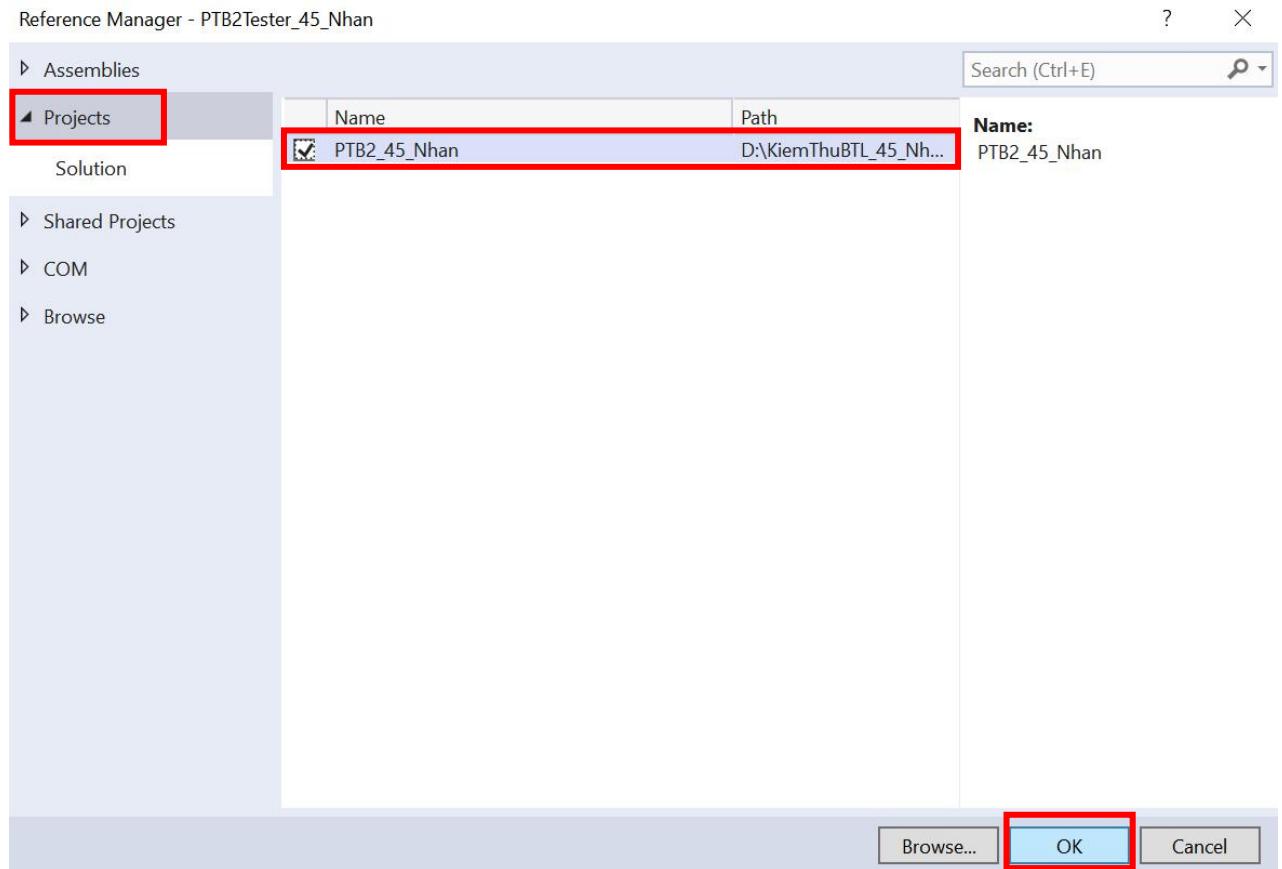
Hình 1.4.5: Nhấn nút “Create”

Bước 4: Thực hiện Add Reference để tham chiếu đến project cần thực hiện Unit Test:

- Click chuột phải Reference -> Add Reference -> chọn project
PTB2_45_Nhan -> Ok



Hình 1.4.6: Add Reference cho project unit test



Hình 1.4.7: Chọn vào project muốn kiểm thử

Bước 5: Viết code các test case kiểm thử

- Code:

```

using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using PTB2_45_Nhan;

namespace PTB2Tester_45_Nhan
{
    [TestClass]
    public class UnitTestPTB2_45_Nhan
    {
        private GiaiPTB2_45_Nhan g_45_Nhan;

        // Phương trình ax^2 + bx + c = 0, nếu a khác 0 STT_Ten: 45_Nhan
    }
}

```

```

//Test Case 1 phương trình có 2 nghiệm STT_Ten: 45_Nhan

[TestMethod]

public void Test2Nghiem_45_Nhan()

{
    string ex_45_Nhan, ac_45_Nhan;

    g_45_Nhan = new GiaiPTB2_45_Nhan(1, 5, 4);

    // Kết quả mong đợi STT_Ten: 45_Nhan

    ex_45_Nhan = "-1; -4";

    // Kết quả thực tế STT_Ten: 45_Nhan

    ac_45_Nhan = g_45_Nhan.Giai_45_Nhan();

    // So sánh 2 kết quả có bằng nhau hay không STT_Ten: 45_Nhan

    Assert.AreEqual(ex_45_Nhan, ac_45_Nhan);

}

//Test Case 2 phương trình vô nghiệm STT_Ten: 45_Nhan

[TestMethod]

public void TestVoNghiem_45_Nhan()

{
    string ex_45_Nhan, ac_45_Nhan;

    g_45_Nhan = new GiaiPTB2_45_Nhan(1, 2, 2);

    ex_45_Nhan = "Phuong trinh vo nghiem";

    ac_45_Nhan = g_45_Nhan.Giai_45_Nhan();

    Assert.AreEqual(ex_45_Nhan, ac_45_Nhan);

}

//Test Case 3 phương trình có nghiệm kép STT_Ten: 45_Nhan

[TestMethod]

public void TestNghiemKep_45_Nhan()

{

```

```

        string ex_45_Nhan, ac_45_Nhan;

        g_45_Nhan = new GiaiPTB2_45_Nhan(1, 0, 0);

        ex_45_Nhan = "0";

        ac_45_Nhan = g_45_Nhan.Giai_45_Nhan();

        Assert.AreEqual(ex_45_Nhan, ac_45_Nhan);

    }

//Test Case 4 kiểm thử trường hợp a = 0 STT_Ten: 45_Nhan

[TestMethod]

public void TestAKhac0_45_Nhan()

{
    string ex_45_Nhan, ac_45_Nhan;

    g_45_Nhan = new GiaiPTB2_45_Nhan(0, 2, 3);

    ex_45_Nhan = "Phuong trinh bac 2, a khac 0";

    ac_45_Nhan = g_45_Nhan.Giai_45_Nhan();

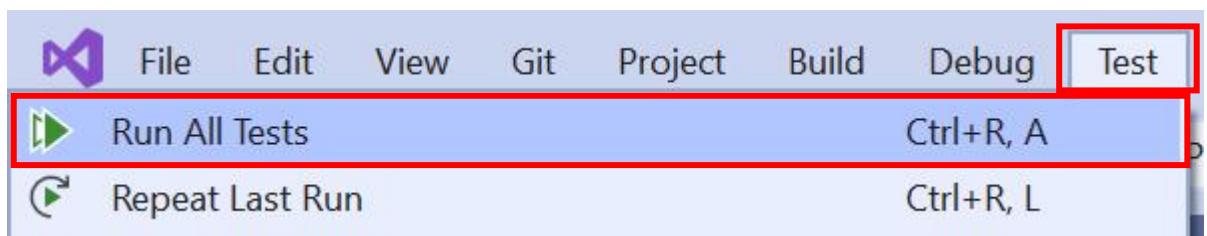
    Assert.AreEqual(ex_45_Nhan, ac_45_Nhan);

}
}

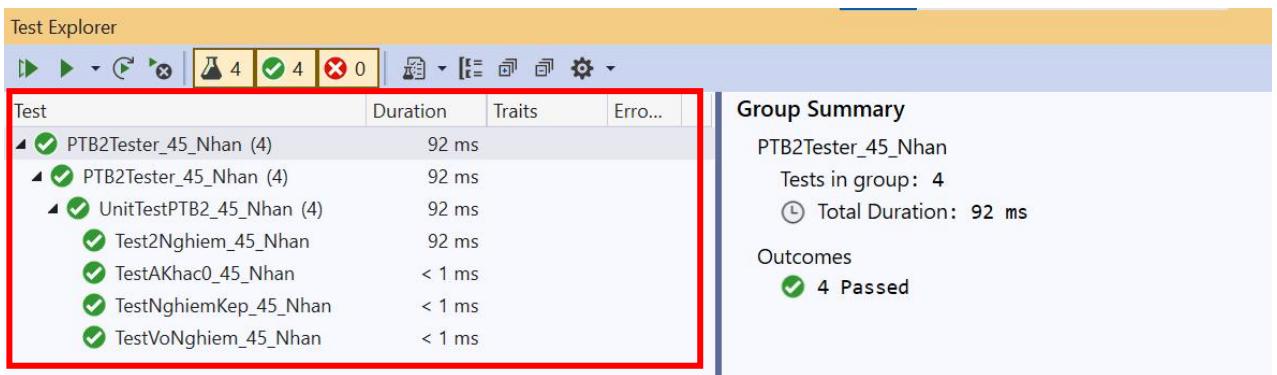
}

```

Bước 6: Chạy các test case đã viết: Nhấn Test -> Run All Test, xem kết quả chạy các test case



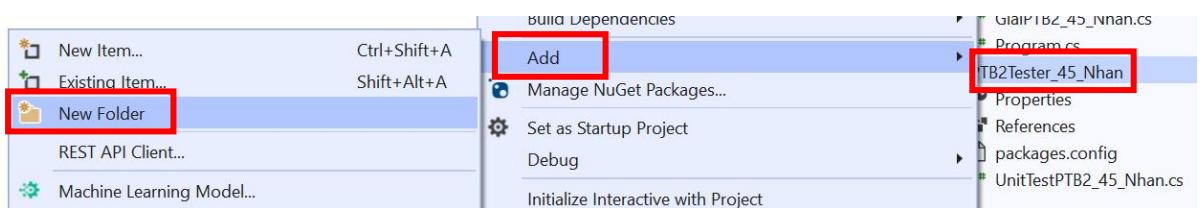
Hình 1.4.8: Chạy các test case



Hình 1.4.9: Các test case đều pass

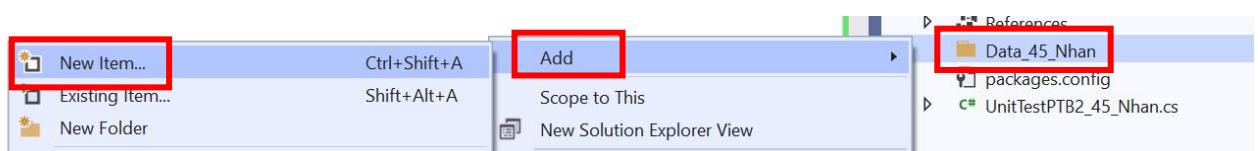
5. Thực thi test với dữ liệu test có sẵn

Bước 1: Tại project PTB2Tester_45_Nhan, tạo 1 thư mục Data_45_Nhan

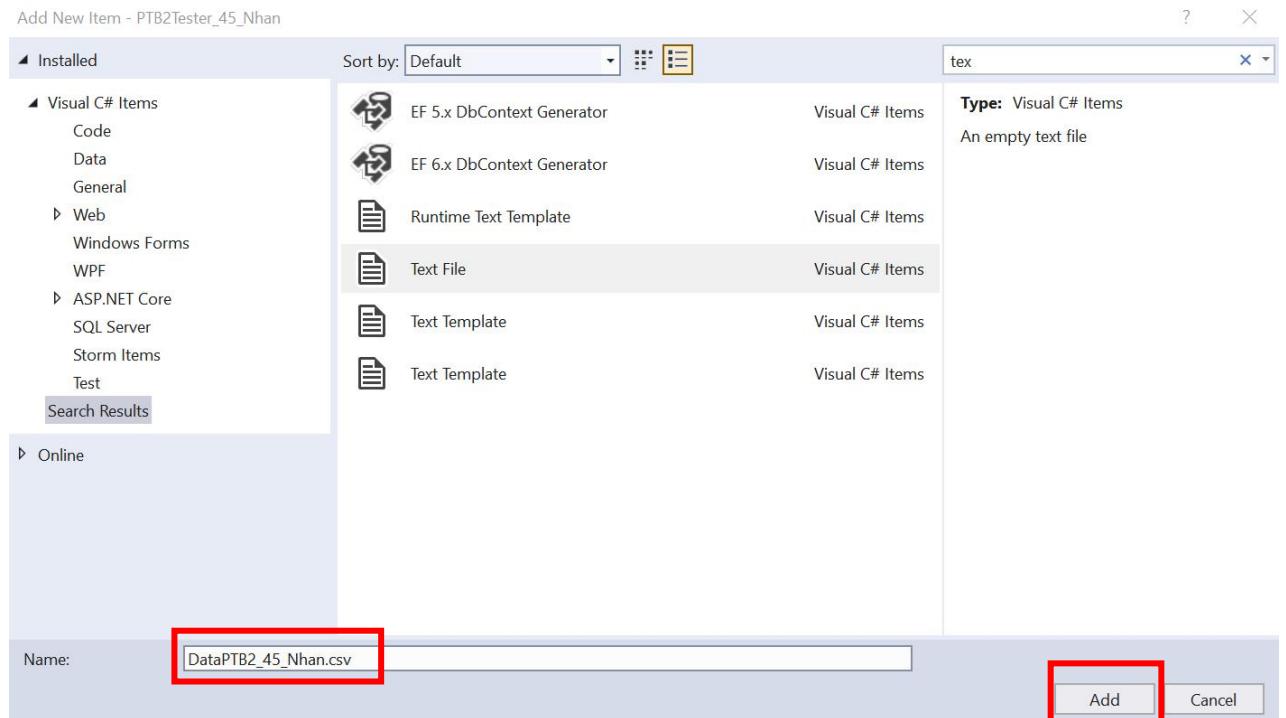


Hình 1.5.1: Tạo thư mục trong project unit test

Bước 2: Click chuột phải vào thư mục Data_45_Nhan -> Add -> New Item ->
Đặt tên tập tin có đuôi là .csv -> Add

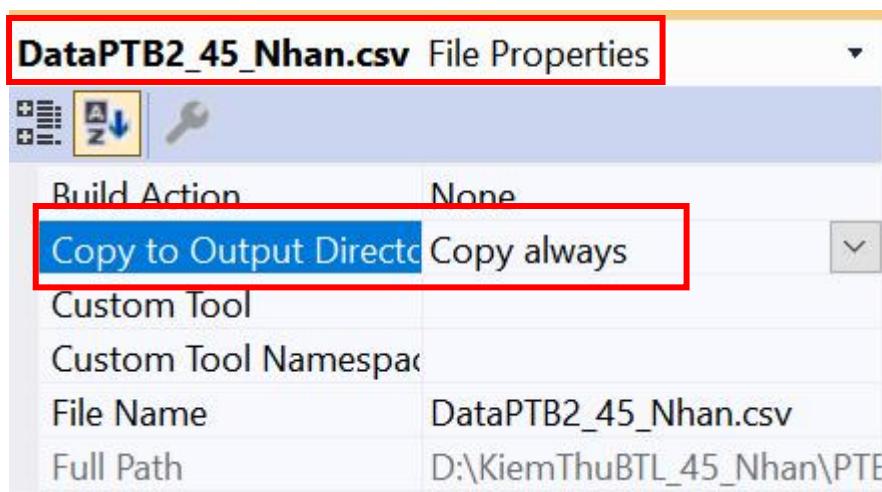


Hình 1.5.2: Tạo tập tin có đuôi là .csv



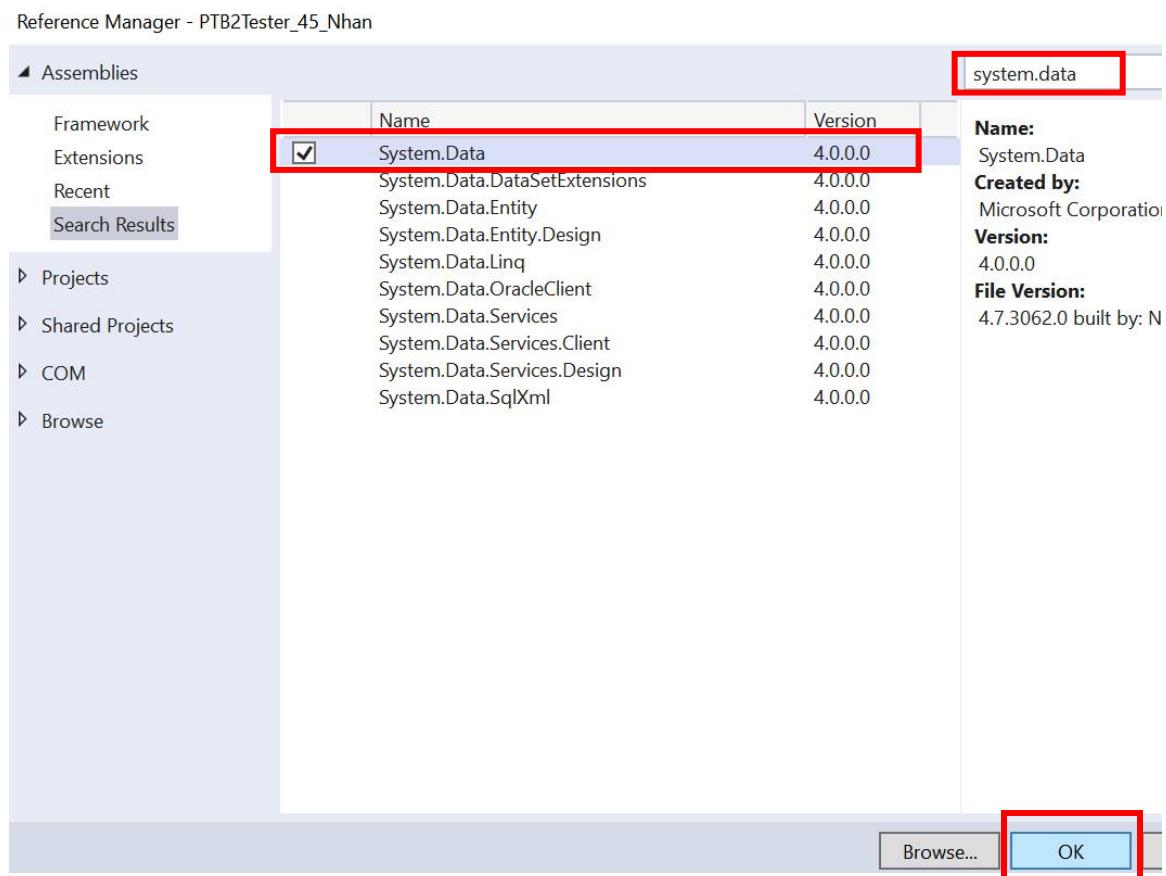
Hình 1.5.3: Đặt tên tập tin csv

Bước 3: Click chuột phải vào tập tin DataPTB2_45_Nhan.csv -> Properties
 thiết lập thuộc tính “Copy to Output Directory” thành “Copy always” để tập
 tin này được sao chép vào thư mục bin khi thực hiện build project



Hình 1.5.4: Thiết lập thuộc tính “Copy to Output Directory” thành “Copy always”

Bước 4: Tìm và thực hiện Add Reference “System.Data” vào project unit test



Hình 1.5.5: Add Reference “System.Data”

Bước 5: Nhập dữ liệu vào tập tin DataPTB2_45_Nhan.csv

- Dòng đầu tiên chứa tên các cột, các dòng còn lại là dữ liệu để kiểm thử

DataPTB2_45_Nhan.csv	
1	a_45_Nhan,b_45_Nhan,c_45_Nhan,kq_45_Nhan
2	1.0,0,0,"0"

Hình 1.5.6: Dữ liệu file csv

Bước 6: Viết test case sử dụng dữ liệu này để kiểm thử

- Code:

```
//Test với file DataPTB2_45_Nhan.csv STT_Ten: 45_Nhan

// Tạo đối tượng TestContext STT_Ten: 45_Nhan

public TestContext TestContext { get; set; }

// Khai báo DataSource STT_Ten: 45_Nhan
```

```

[DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",
@".\Data_45_Nhan\DataPTB2_45_Nhan.csv",
"DataPTB2_45_Nhan#csv", DataAccessMethod.Sequential)]

[TestMethod]

public void TestPTB2_45_Nhan()
{
    // Đọc dữ liệu cột 1 STT_Ten: 45_Nhan
    double a_45_Nhan = double.Parse(TestContext.DataRow[0].ToString());

    // Đọc dữ liệu cột 2 STT_Ten: 45_Nhan
    double b_45_Nhan = double.Parse(TestContext.DataRow[1].ToString());

    // Đọc dữ liệu cột 3 STT_Ten: 45_Nhan
    double c_45_Nhan = double.Parse(TestContext.DataRow[2].ToString());

    // Đọc dữ liệu cột 4, là kết quả kì vọng STT_Ten: 45_Nhan
    string expected_45_Nhan = TestContext.DataRow[3].ToString();

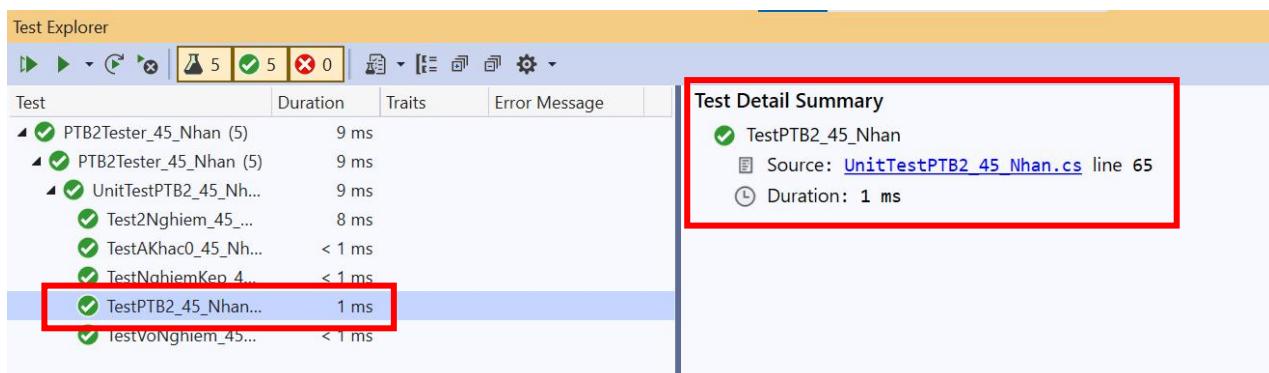
    GiaiPTB2_45_Nhan gai_45_Nhan = new GiaiPTB2_45_Nhan(a_45_Nhan, b_45_Nhan,
c_45_Nhan);

    // Kết quả thực tế STT_Ten: 45_Nhan
    string actual_45_Nhan = gai_45_Nhan.Giai_45_Nhan();

    // So sánh kết quả kì vọng và thực tế
    Assert.AreEqual(expected_45_Nhan, actual_45_Nhan);
}

```

Bước 7: Nhấn Test -> Run All Test, xem kết quả chạy các test case

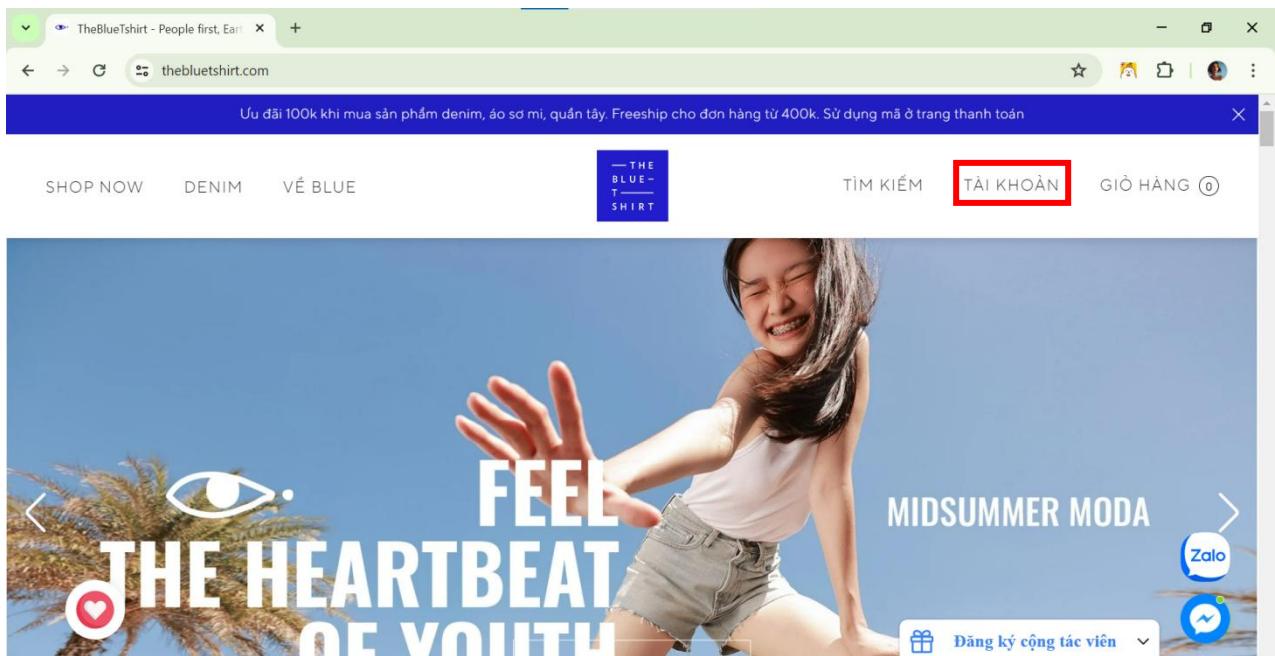


Hình 1.5.7: Test case pass

Chương 2: SELENIUM WEBDRIVER

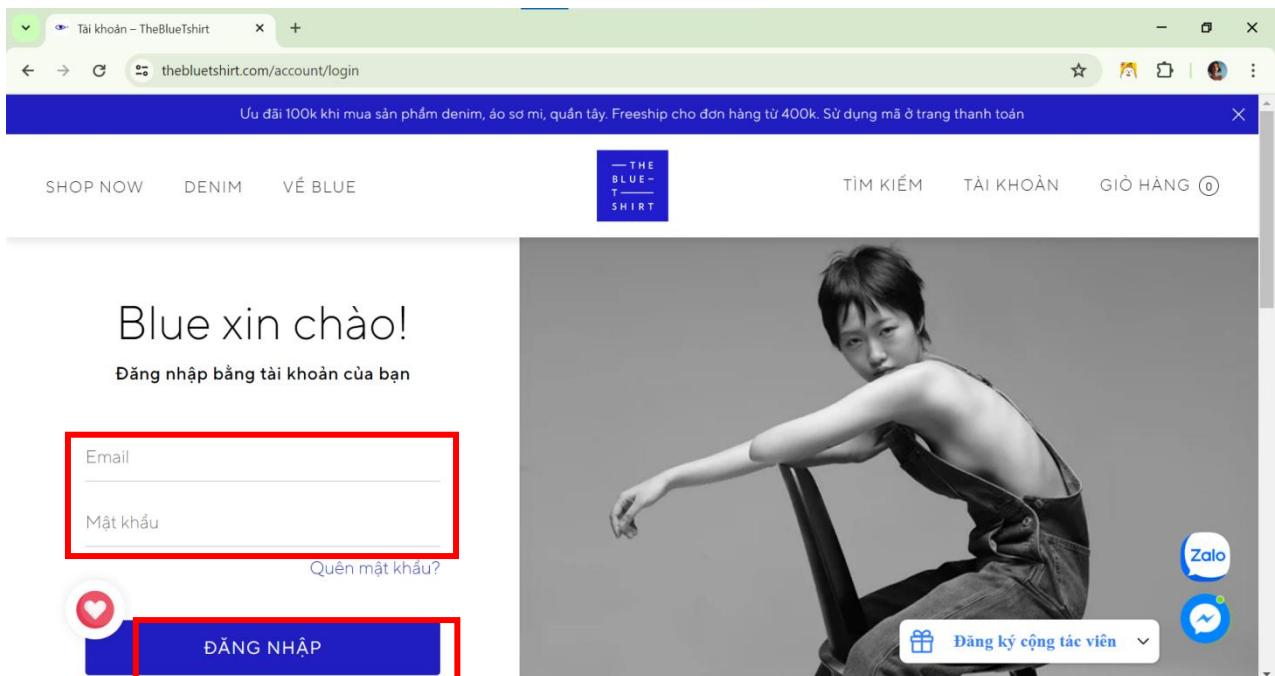
1. Mô tả chức năng đăng nhập

- Chức năng đăng nhập là quy trình cho phép người dùng truy cập vào hệ thống 1 trang web bằng cách xác nhận danh tính của họ thông qua thông tin đăng nhập cá nhân, cụ thể trang web <https://thebluetshirt.com/> là Email và Mật khẩu. Ngoài ra, chức năng đăng nhập còn dùng để bảo vệ dữ liệu cá nhân, bảo đảm tính bảo mật của hệ thống.
- Muốn đăng nhập thành công trước tiên người dùng phải đăng ký thành công tài khoản.
- Nhấn vào “TÀI KHOẢN” để đăng nhập



Hình 2.1.1: Giao diện đầu tiên trang web <https://thebluetshirt.com/>

- Nhập “Email” và “Mật khẩu” sau đó nhấn vào “ĐĂNG NHẬP”



Hình 2.1.2: Giao diện đăng nhập của trang web

2. Xây dựng test case

- Thông qua mô tả và giao diện của trang web đăng nhập, em xây dựng 5 test case và ghi chú như sau:

Test case 1: TestDNThanhCong_45_Nhan()

Test case 2: TestDNSaiEmail_45_Nhan()

Test case 3: TestDNSaiMatKhau_45_Nhan()

Test case 4: TestDNEmailRong_45_Nhan()

Test case 5: TestDNMatKhauRong_45_Nhan()

		Test case 1	Test case 2	Test case 3	Test case 4	Test case 5
Các điều kiện	Email	T	F	T	B	-
	Password	T	-	F	-	B
Các hành động	Đăng nhập thành công	Yes	No	No	No	No

Hình 2.2.1: Các test case cho chức năng đăng nhập

Xác định các điều kiện: Email và Password

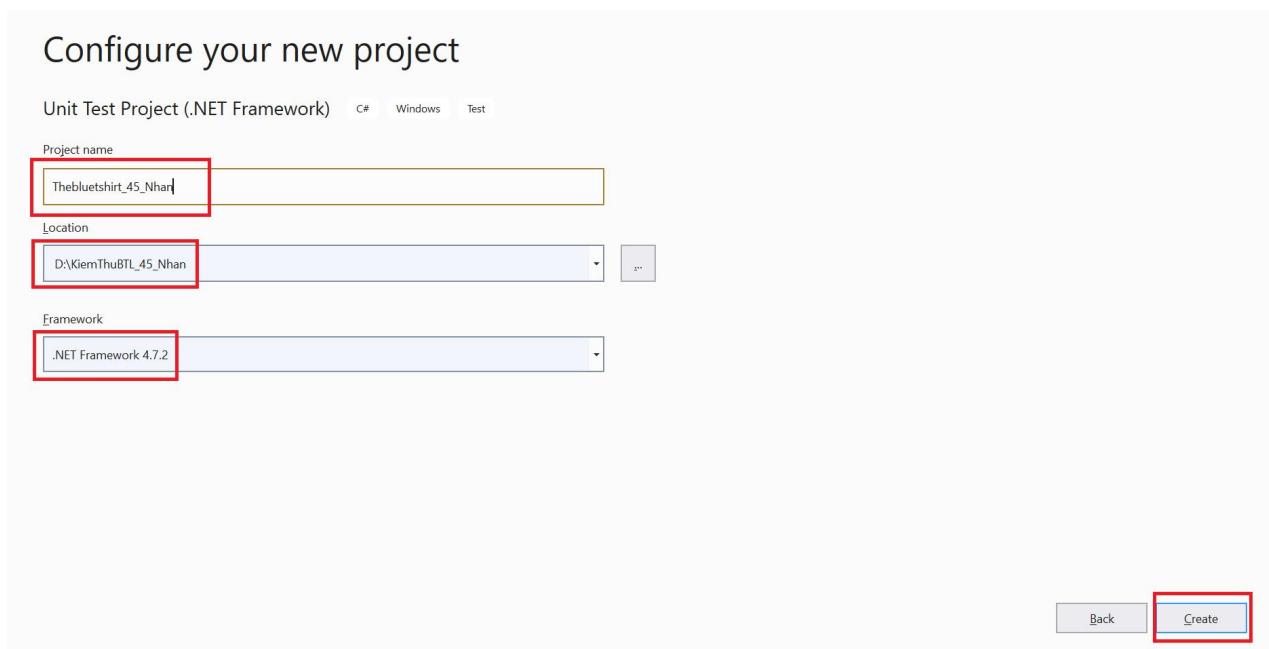
Mỗi đầu vào trên nhận 1 trong 3 giá trị: rỗng (B), hợp lệ (T), không hợp lệ (F)

Hình 2.2.2: Ghi chú test case cho chức năng đăng nhập

3. Tạo project unit test Thebluetshirt_45_Nhan để kiểm thử các test case của chức năng đăng nhập

Bước 1: Tương tự như vậy, click chuột phải vào Solution

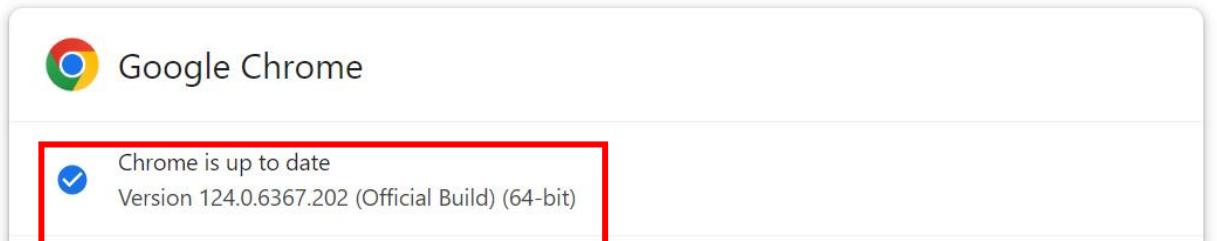
'KiemThuBTL_45_Nhan' -> Add -> New Project -> Tìm và chọn Unit Test Project (.NET Framework) -> Next -> Đặt tên project, kiểm tra đường dẫn, chú ý Framework -> Create



Hình 2.3.1: Tạo project unit test

Bước 2: Kiểm tra Browser: Chrome trên máy tính

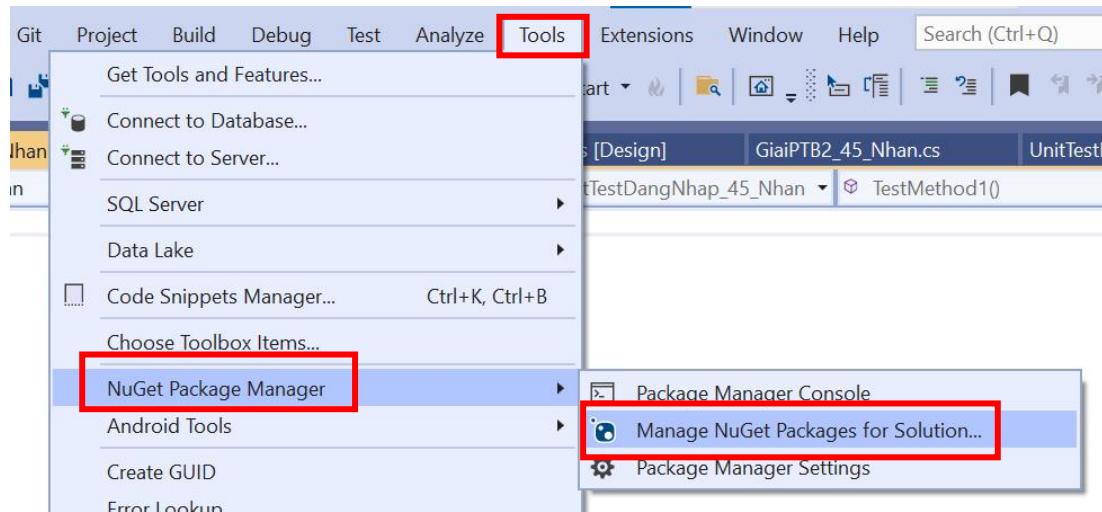
About Chrome



Hình 2.3.2: Phiên bản chrome trên máy tính

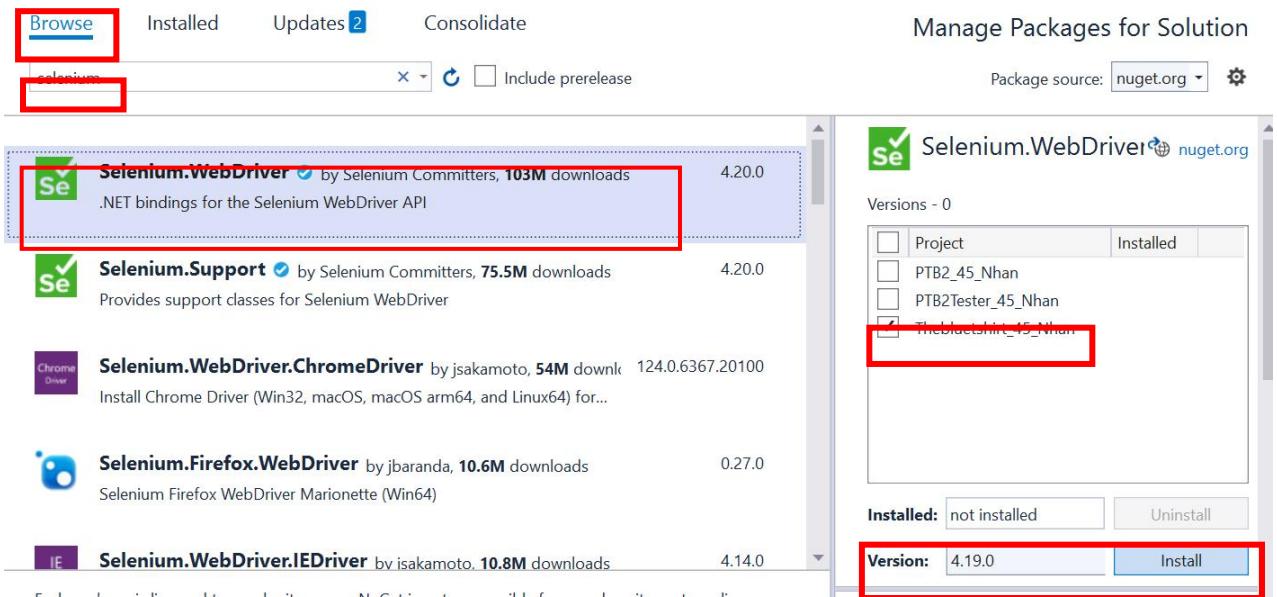
Bước 3: Cài đặt thư viện Selenium

- Chọn Tools -> NuGet Package Manager -> Manage NuGet Packages for Solution...

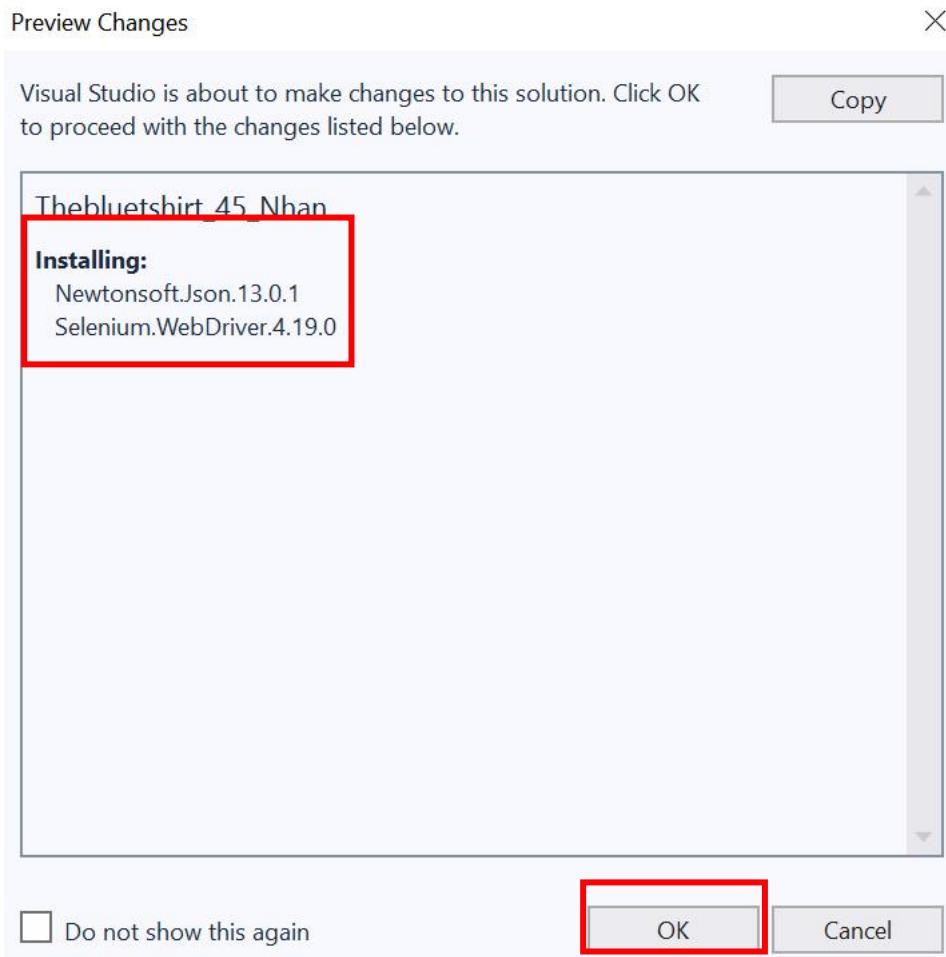


Hình 2.3.3: Mở NuGet để cài đặt selenium

- Tại tab Browse tìm “selenium”, thực hiện cài đặt **Selenium.WebDriver** (Version: **4.19.0**) cho project unit test “Thebluetshirt_45_Nhan -> Install -> OK

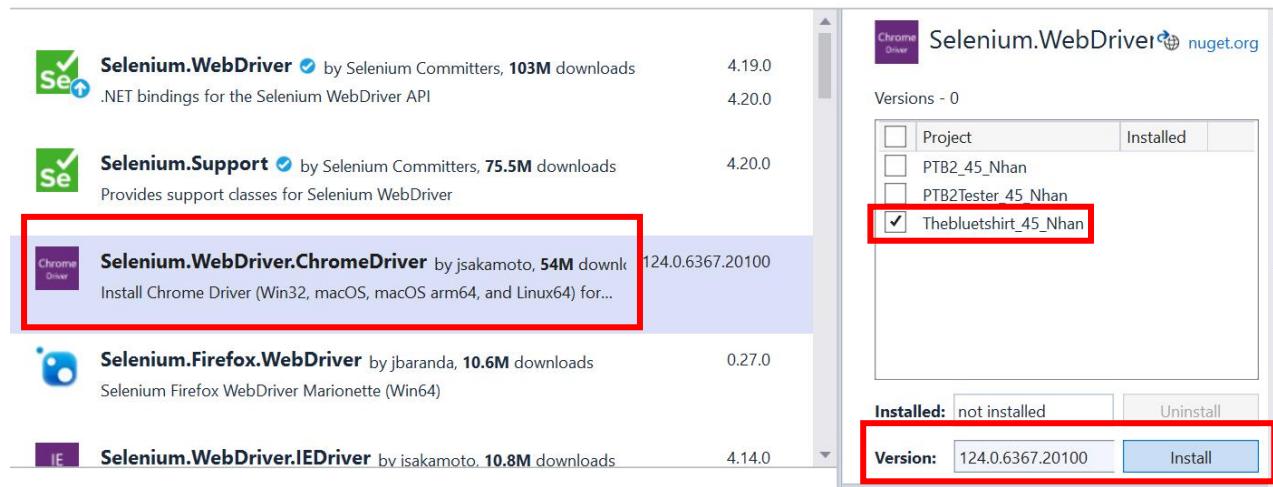


Hình 2.3.4: Cài đặt Selenium.WebDriver



Hình 2.3.5: Xác nhận thông tin cài đặt

- Thực hiện cài đặt **Selenium.WebDriver.ChromeDriver** (Version: **124.6367.20100**) phù hợp với phiên bản Chrome trên máy tính, cho project unit test “Thebluetshirt_45_Nhan -> Install -> OK



Hình 2.3.6: Cài đặt Selenium.WebDriver.ChromeDriver

Preview Changes

X

Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.

Copy

Thebluetshirt_45_Nhan

Installing:

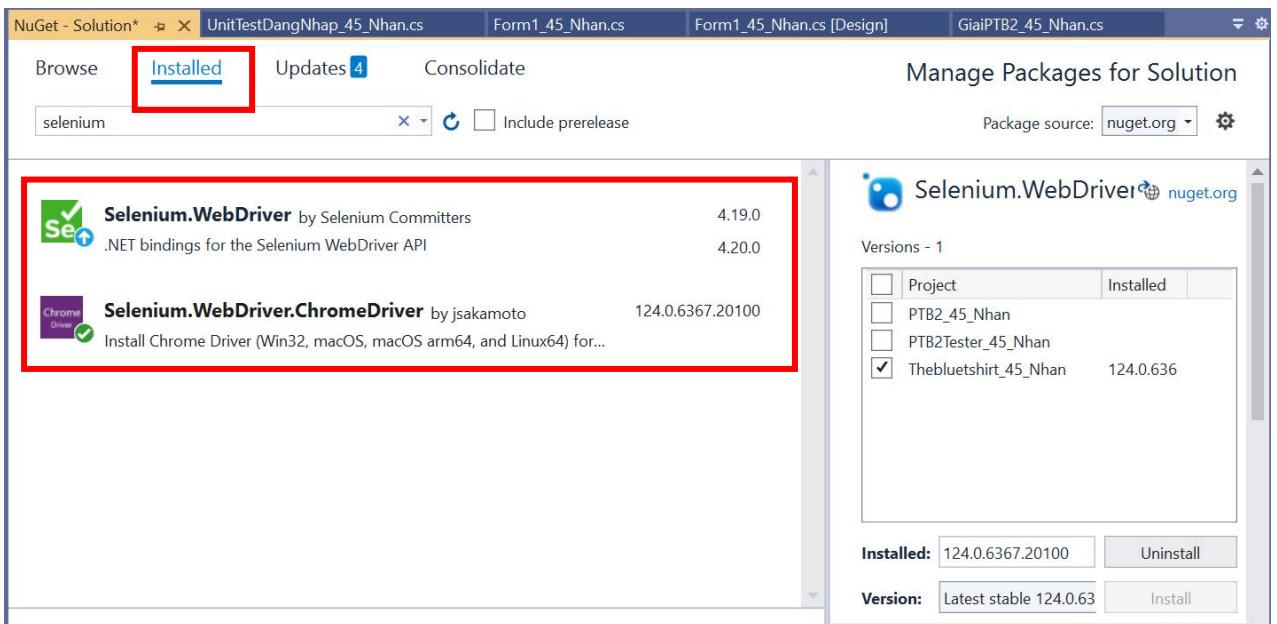
Selenium.WebDriver.ChromeDriver.124.0.6367.20100

Do not show this again

OK

Cancel

Hình 2.3.7: Xác nhận thông tin cài đặt



Hình 2.3.8: Cài đặt thành công

Bước 4: Tại vùng sử dụng thư viện, thêm các thư viện như sau:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
```

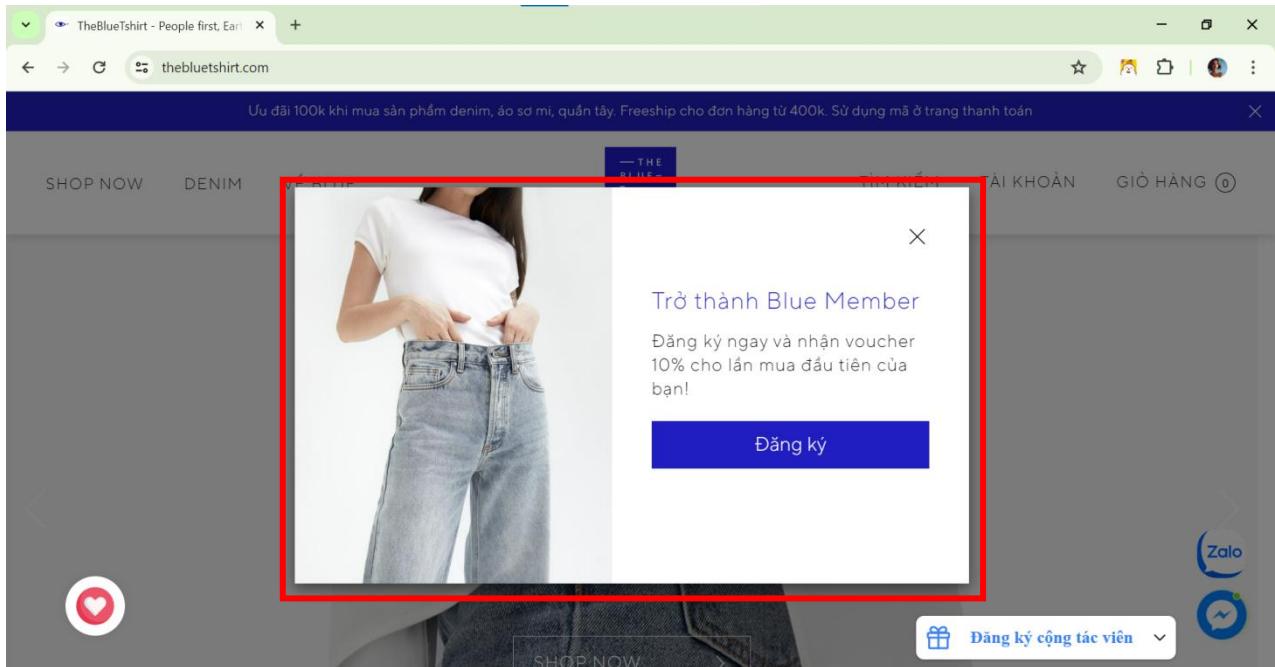
The screenshot shows the 'UnitTestDangNhap_45_Nhan.cs' file in the Visual Studio code editor. The code includes several 'using' statements. A red box highlights the 'using OpenQA.Selenium;' and 'using OpenQA.Selenium.Chrome;' statements, indicating they are the new additions mentioned in the previous step.

```
1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using System;
3  using OpenQA.Selenium;
4  using OpenQA.Selenium.Chrome;
5  using System.Threading;
```

Hình 2.3.9: Thêm thư viện vào code

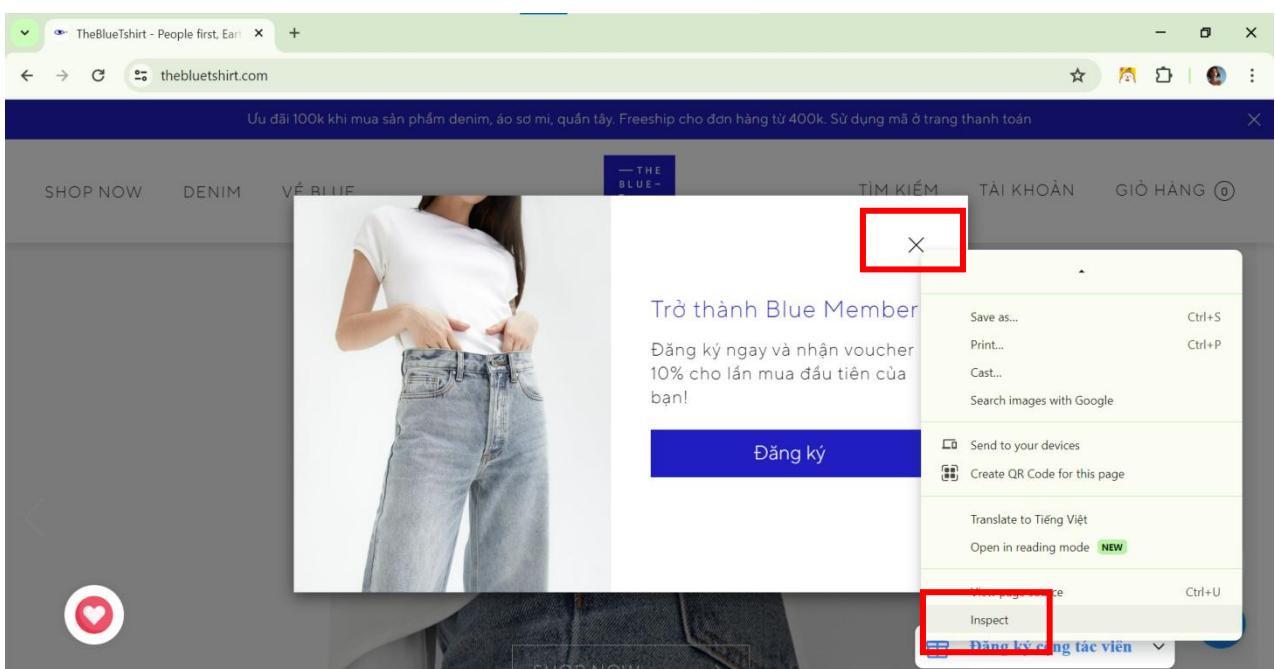
4. Lấy các elements

- Vì vừa vào trang web “<https://thebluetshirt.com/>” trang sẽ hiển thị lên thẻ “div” che phủ các nút, nên phải tắt thẻ div đó đi bằng cách bắt element (**CssSelector**) của dấu X

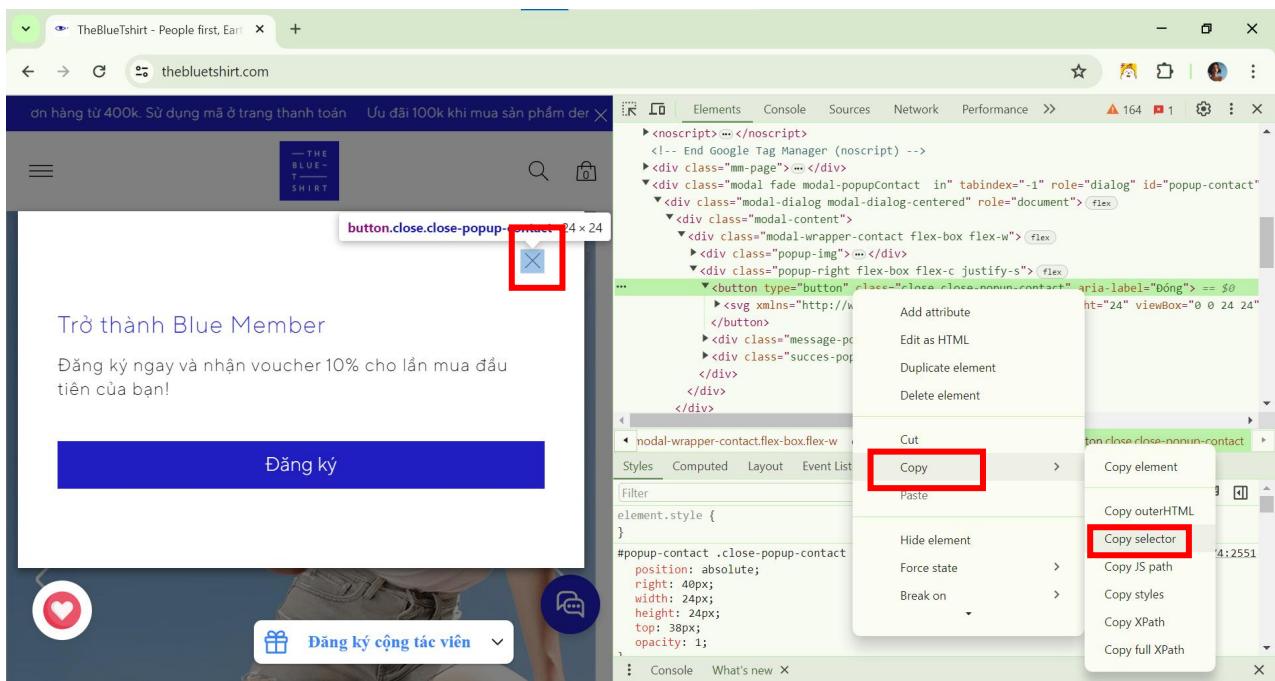


Hình 2.4.1: Thẻ “div” che phủ các nút

Bước 1: Click phải chuột vào dấu X trên thẻ “div” -> nhấn Inspect

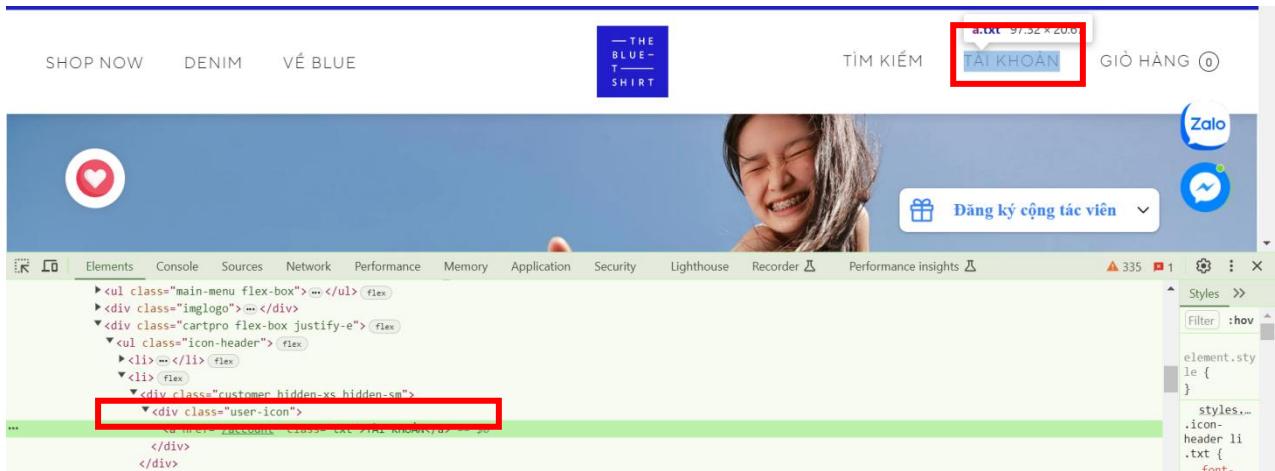


Hình 2.4.2: Chọn Inspect để lấy element



Hình 2.4.3: Copy CssSelector của nút dấu X

Bước 2: Lấy element (LinkText) của nút “TÀI KHOẢN”



Hình 2.4.4: Copy LinkText của nút “TÀI KHOẢN”

Bước 3: Lấy element Id của input “Email”



Hình 2.4.5: Copy Id của input “Email”

Bước 4: Lấy element XPath của input “Mật khẩu”



Hình 2.4.6: Copy XPath của input “Mật khẩu”

Bước 5: Lấy element ClassName của nút “ĐĂNG NHẬP”



Hình 2.4.7: Copy ClassName của nút “ĐĂNG NHẬP”

5. Chuẩn bị trước khi test các test case

5.1. Thiết lập các biến và phương thức cho class kiểm thử chức năng đăng nhập (UnitTestDangNhap_45_Nhan)

- Code:

```
// Tạo biến cục bộ STT_Ten: 45_Nhan

IWebDriver driver_45_Nhan = new ChromeDriver();

// Phương thức này dùng để khởi chạy trang chủ và xóa đi thanh div che phủ các nút
// và nhấp vào TÀI KHOẢN STT_Ten: 45_Nhan

public void SetUp_45_Nhan()

{

    // Chạy trang chủ STT_Ten: 45_Nhan

    driver_45_Nhan.Navigate().GoToUrl("https://thebluetshirt.com/");

    // Web nghỉ 3s STT_Ten: 45_Nhan

    Thread.Sleep(3000);

    // Xóa đi thanh div (nhấp vào dấu X) che phủ các nút STT_Ten: 45_Nhan

    driver_45_Nhan.FindElement(By.CssSelector("#popup-contact > div > div >
" +
    "div.popup-right.flex-box.flex-c.justify-s > button")).Click();

    Thread.Sleep(3000);

    // Click vào nút "TÀI KHOẢN" để đăng nhập STT_Ten: 45_Nhan

    driver_45_Nhan.FindElement(By.LinkText("TÀI KHOẢN")).Click();

    Thread.Sleep(3000);

}

// Phương thức này dùng để đăng nhập trang web với biến STT_Ten: 45_Nhan

public void DangNhap_45_Nhan(string email_45_Nhan, string pass_45_Nhan)

{

    SetUp_45_Nhan();

    // Tự động nhập thông tin vào input Email STT_Ten: 45_Nhan
```

```

driver_45_Nhan.FindElement(By.Id("customer_email")).SendKeys(email_45_Nhan);

// Tự động nhập thông tin vào input Mật khẩu STT_Ten: 45_Nhan

driver_45_Nhan.FindElement(By.XPath("//*[@id='customer_password']")).SendKeys(pass_4
5_Nhan);

// Nhấn nút "ĐĂNG NHẬP" STT_Ten: 45_Nhan

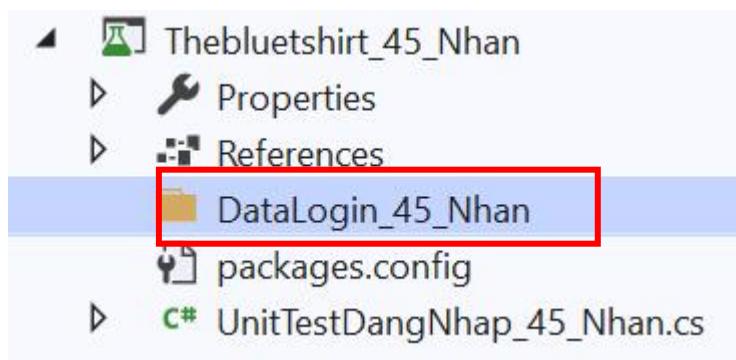
driver_45_Nhan.FindElement(By.ClassName("btn")).Click();

}

```

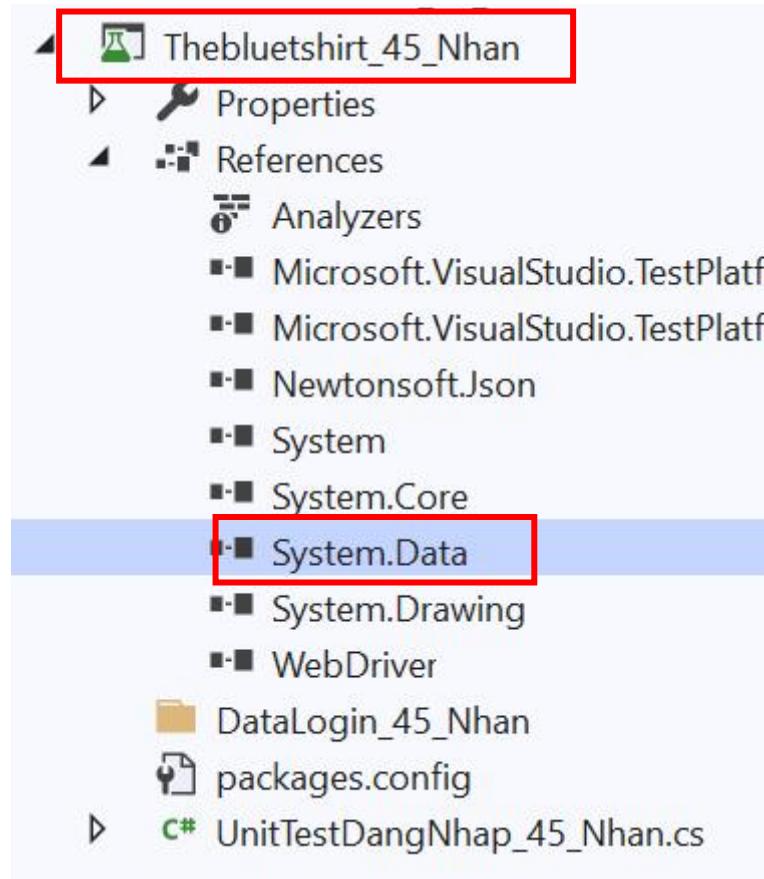
5.2. Test với dữ liệu test có sẵn:

- Tạo thư mục DataLogin_45_Nhan trong project Thebluetshirt_45_Nhan để chứa các file csv cho từng test case



Hình 2.5.1: Tạo thư mục thành công

- Tương tự như phần 5 ở phần I thực hiện Add Reference “System.Data” vào project unit test Thebluetshirt_45_Nhan



Hình 2.5.2: Add Reference thành công

- Tạo đối tượng để đọc file dữ liệu

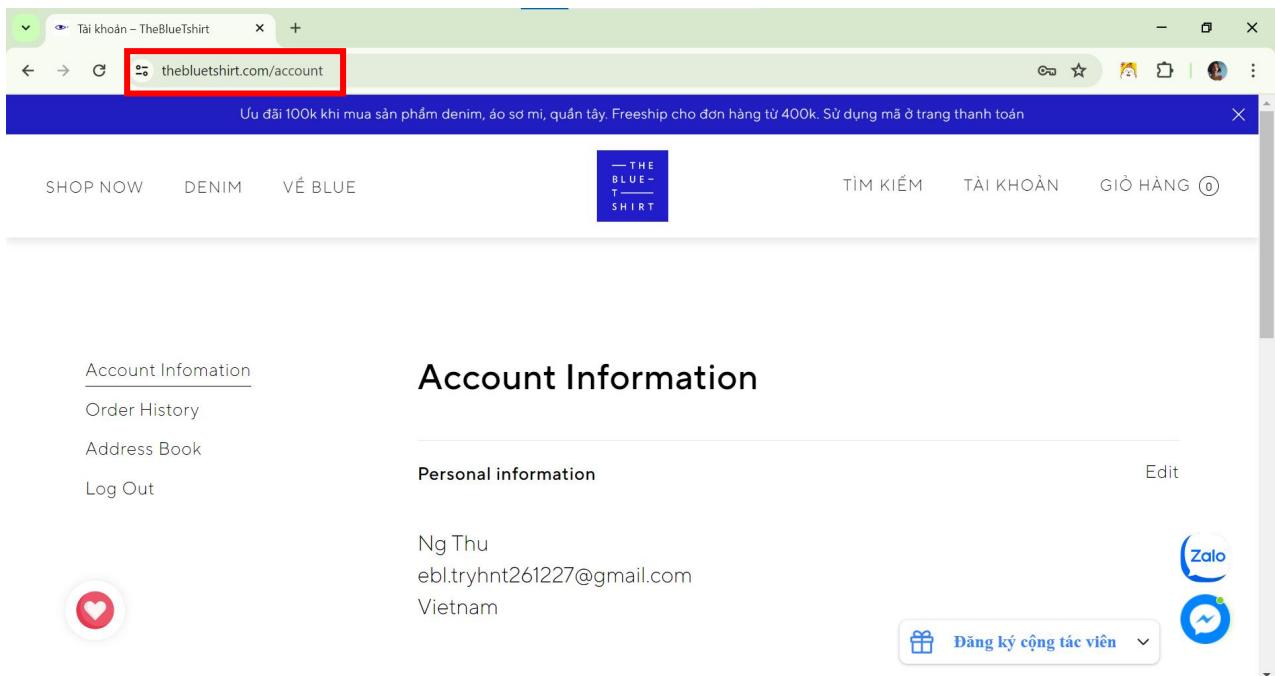
```
// Tạo đối tượng TestContext STT_Ten: 45_Nhan
```

```
public TestContext TestContext { get; set; }
```

6. Viết các test case cho chức năng đăng nhập

6.1. Test case 1: TestDNThanhCong_45_Nhan()

- Test case này kiểm tra xem hành động đăng nhập có thành công hay không, với trường hợp Email và Mật khẩu hợp lệ, tức là người dùng đã đăng ký thành công tài khoản với Email và Mật khẩu.
- Giao diện trang web sau khi đăng nhập thành công



Hình 2.6.1: Giao diện đăng nhập thành công

- Url kì vọng: <https://thebluetshirt.com/account> chuyển đến trang account khi đăng nhập thành công
- Url thực tế: url của trang web sau khi nhấn nút “ĐĂNG NHẬP”
- Nếu 2 Url giống nhau thì test case pass, ngược lại là fail

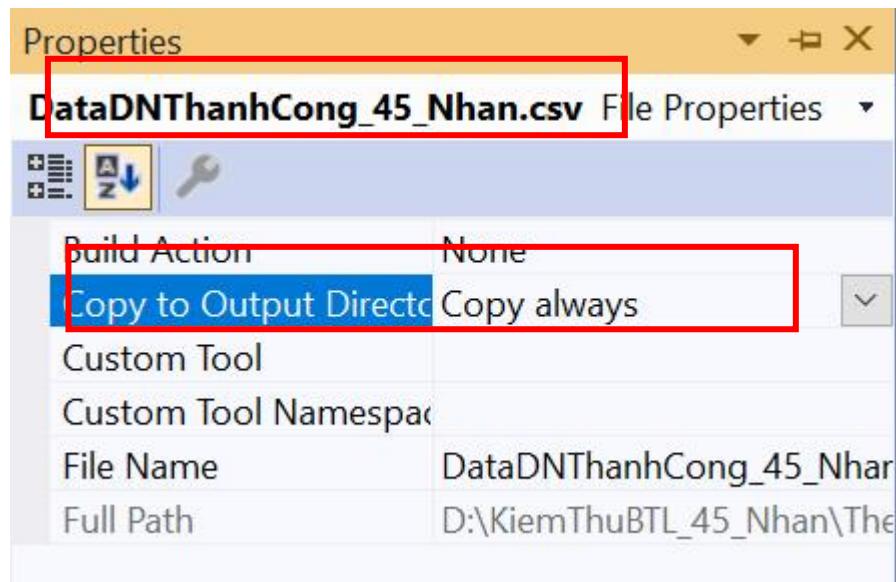
6.2. Các bước thực thi test case với dữ liệu có sẵn

Bước 1: Tạo 1 tập tin DataDNThanhCong_45_Nhan.csv, nhập dữ liệu đầu vào

```
DataDNThanhCong_45_Nhan.csv ✘ X UnitTestDangNhap_45_N
1 Email,Pass
2 "ebl.tryhnt261227@gmail.com","12345"
```

Hình 2.6.2: Thêm dữ liệu đầu vào

Bước 2: Chỉnh thuộc tính “Copy to Output Driectory” thành “Copy always”



Hình 2.6.3: Thay đổi thuộc tính “Copy to Output Driectory” thành “Copy always”

Bước 3: Viết code cho test case

- Code:

```
// Test case 1 email đúng pass đúng STT_Ten: 45_Nhan

// Khai báo DataSource STT_Ten: 45_Nhan

[DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",
    @".\DataLogin_45_Nhan\DataDNThanhCong_45_Nhan.csv",
    "DataDNThanhCong_45_Nhan#csv", DataAccessMethod.Sequential)]

[TestMethod]

public void TestDNThanhCong_45_Nhan()

{
    string e_45_Nhan = TestContext.DataRow[0].ToString();

    string p_45_Nhan = TestContext.DataRow[1].ToString();

    DangNhap_45_Nhan(e_45_Nhan, p_45_Nhan);
}
```

```

    // Nếu đăng nhập thành công đường dẫn url kì vọng sẽ là
    https://thebluetshirt.com/account STT_Ten: 45_Nhan

    string ex_45_Nhan = "https://thebluetshirt.com/account";

    // Lấy đường dẫn trang web sau khi nhấn "ĐĂNG NHẬP" STT_Ten: 45_Nhan
    string ac_45_Nhan = driver_45_Nhan.Url;

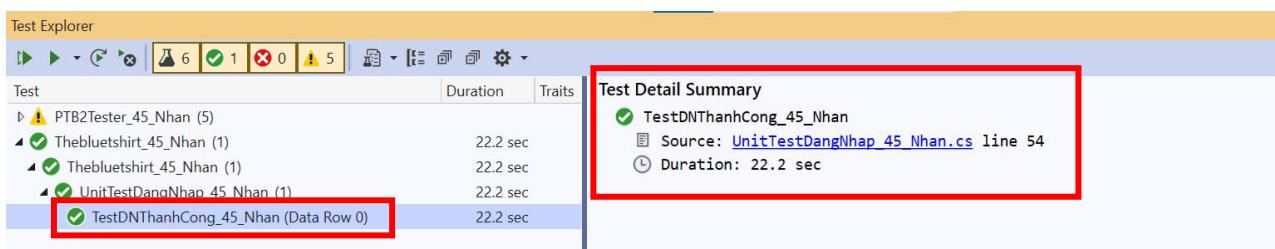
    // So sánh giữa 2 đường dẫn kiểu string STT_Ten: 45_Nhan
    Assert.AreEqual(ex_45_Nhan, ac_45_Nhan);

    // Đóng các cửa sổ liên quan
    driver_45_Nhan.Quit();

}

```

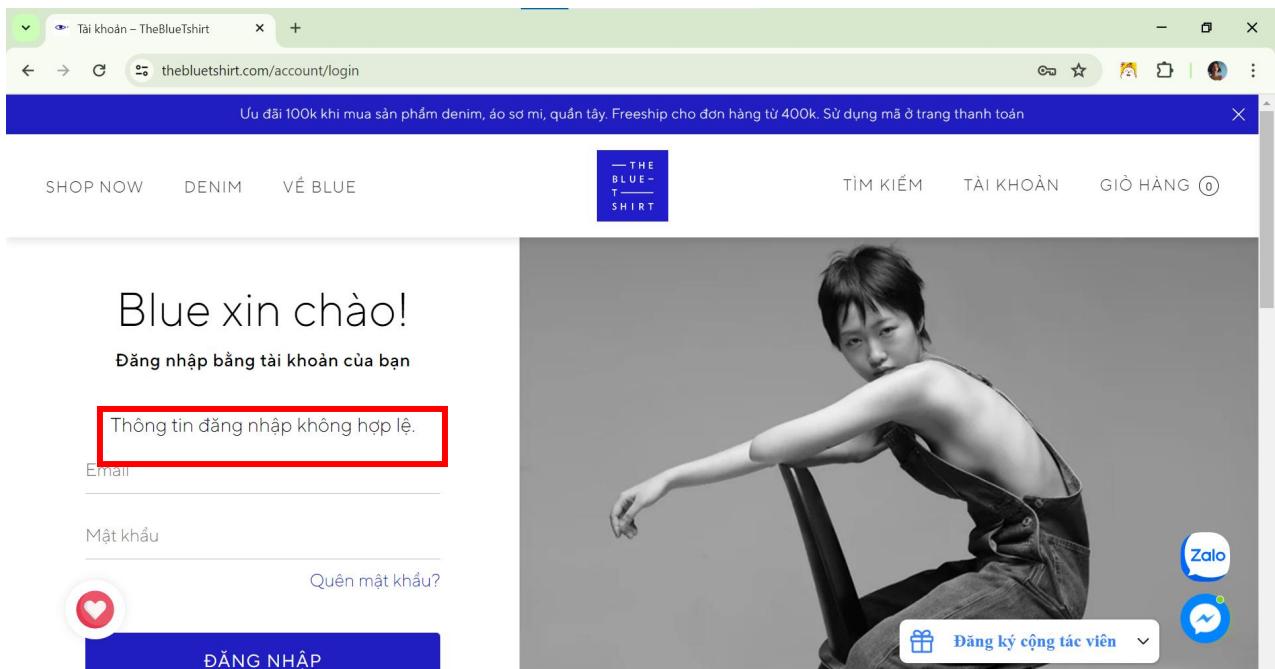
Bước 4: Kết quả chạy test case



Hình 2.6.4: Test case pass

6.3. Test case 2: TestDNSaiEmail_45_Nhan()

- Test case này kiểm tra xem hành động đăng nhập có thất bại hay không, với trường hợp Email không hợp lệ, tức là email chưa đăng ký tài khoản
- Giao diện trang web khi Email không hợp lệ



Hình 2.6.5: Hiển thị cảnh báo

- Kết quả kì vọng: cảnh báo “Thông tin đăng nhập không hợp lệ.”
- Kết quả thực tế: cảnh báo xuất hiện sau khi bấm nút “ĐĂNG NHẬP”
- Nếu 2 kết quả giống nhau thì test case pass, ngược lại là fail

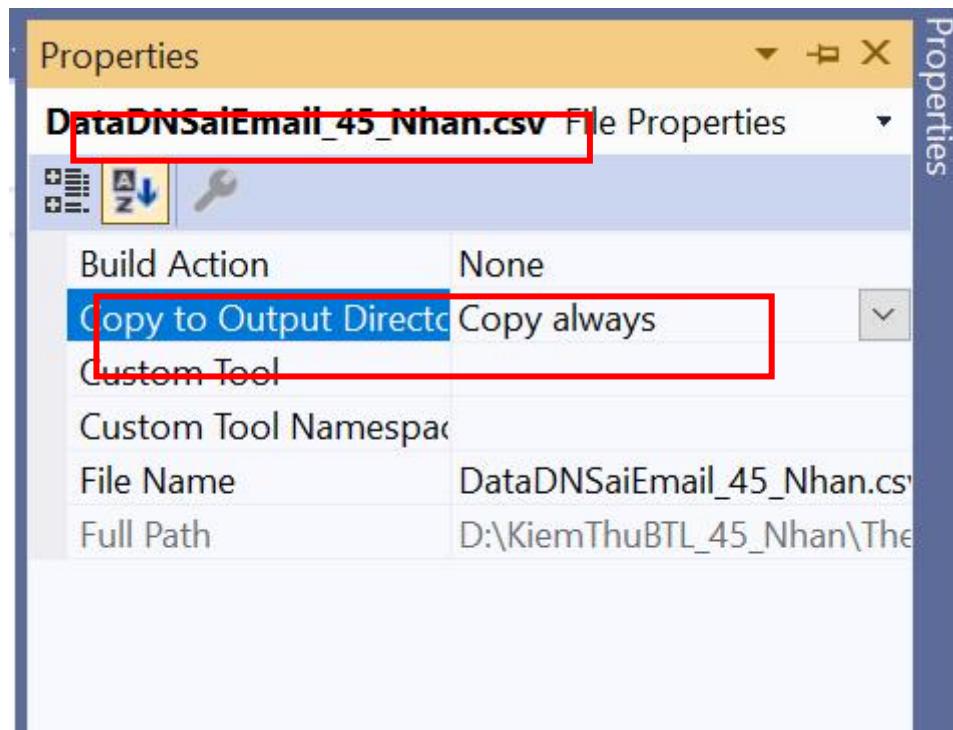
6.4. Các bước thực thi

Bước 1: Tạo 1 tập tin DataDNSaiEmail_45_Nhan.csv, nhập dữ liệu đầu vào

DataDNSaiEmail_45_Nhan.csv	
1	EmailSai,Pass
2	'abc@gmail.com', "23424"
3	'abc@gmail.com', "12345"

Hình 2.6.6: Dữ liệu tập tin csv

Bước 2: Chính thuộc tính “Copy to Output Directory” thành “Copy always”



Hình 2.6.7: Thay đổi thuộc tính “Copy to Output Directory”

Bước 3: Bắt element ClassName của cảnh báo



Hình 2.6.8: Element của cảnh báo

Bước 4: Viết code cho test case

- Code:

```
// Test case 2 email sai pass đúng/sai STT_Ten: 45_Nhan

// Khai báo DataSource STT_Ten: 45_Nhan
```

```

[DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",
    @".\DataLogin_45_Nhan\DataDNSaiEmail_45_Nhan.csv",
    "DataDNSaiEmail_45_Nhan#csv", DataAccessMethod.Sequential)]

[TestMethod]

public void TestDNSaiEmail_45_Nhan()
{
    string e_45_Nhan = TestContext.DataRow[0].ToString();

    string p_45_Nhan = TestContext.DataRow[1].ToString();

    DangNhap_45_Nhan(e_45_Nhan, p_45_Nhan);

    // Cảnh báo kì vọng STT_Ten: 45_Nhan

    string ex_45_Nhan = "Thông tin đăng nhập không hợp lệ.";

    // Lấy cảnh báo sau khi nhấn "ĐĂNG NHẬP" STT_Ten: 45_Nhan

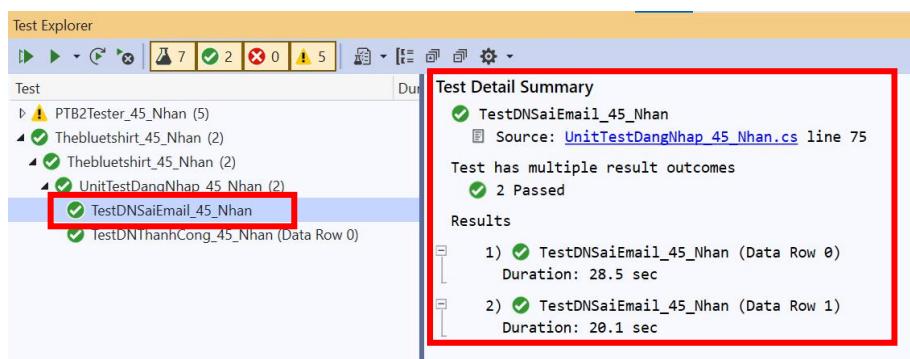
    string ac_45_Nhan = driver_45_Nhan.FindElement(By.ClassName("errors")).Text;

    Assert.AreEqual(ex_45_Nhan, ac_45_Nhan);

    driver_45_Nhan.Quit();
}

```

Bước 5: Kết quả chạy test case:

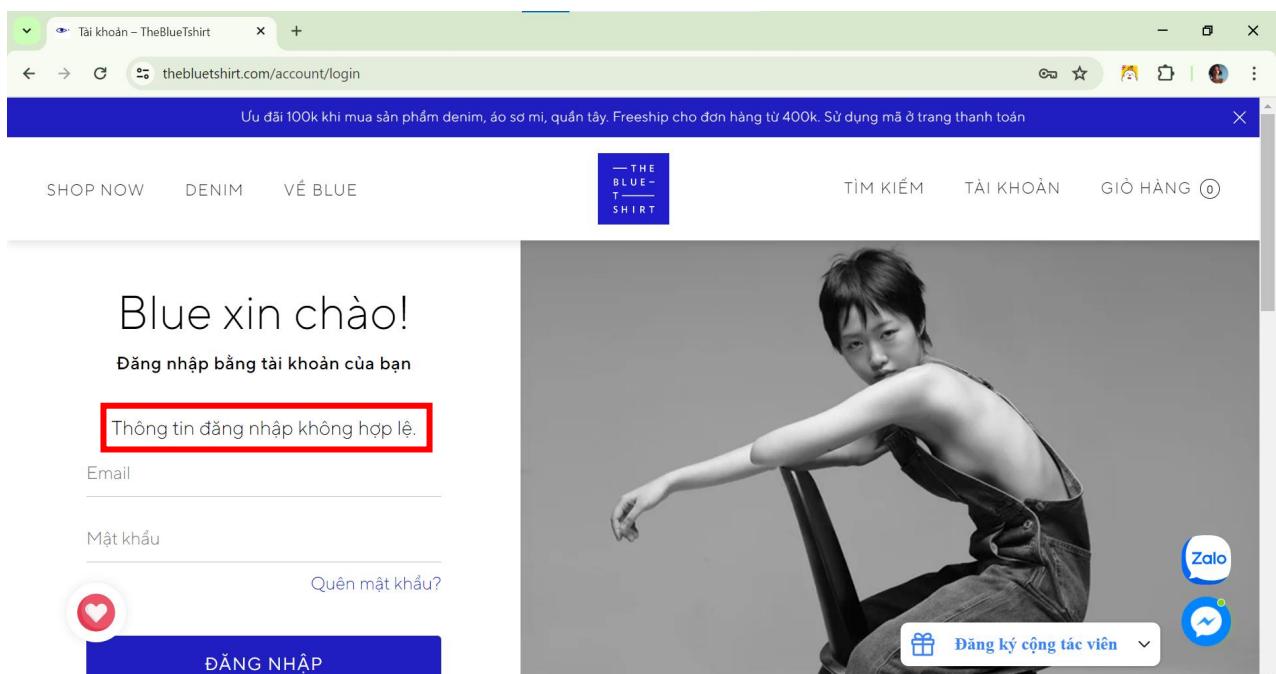


Hình 2.6.9: Test case pass

6.5. Test case 3: TestDNSaiMatKhau_45_Nhan()

- Test case này kiểm tra xem hành động đăng nhập có thất bại hay không, với trường hợp Mật khẩu không hợp lệ, Email hợp lệ.

- Giao diện trang web khi Mật khẩu không hợp lệ



Hình 2.6.10: Hiển thị cảnh báo

- Kết quả kì vọng: cảnh báo “Thông tin đăng nhập không hợp lệ.”
- Kết quả thực tế: cảnh báo xuất hiện sau khi bấm nút “ĐĂNG NHẬP”
- Nếu 2 kết quả giống nhau thì test case pass, ngược lại là fail

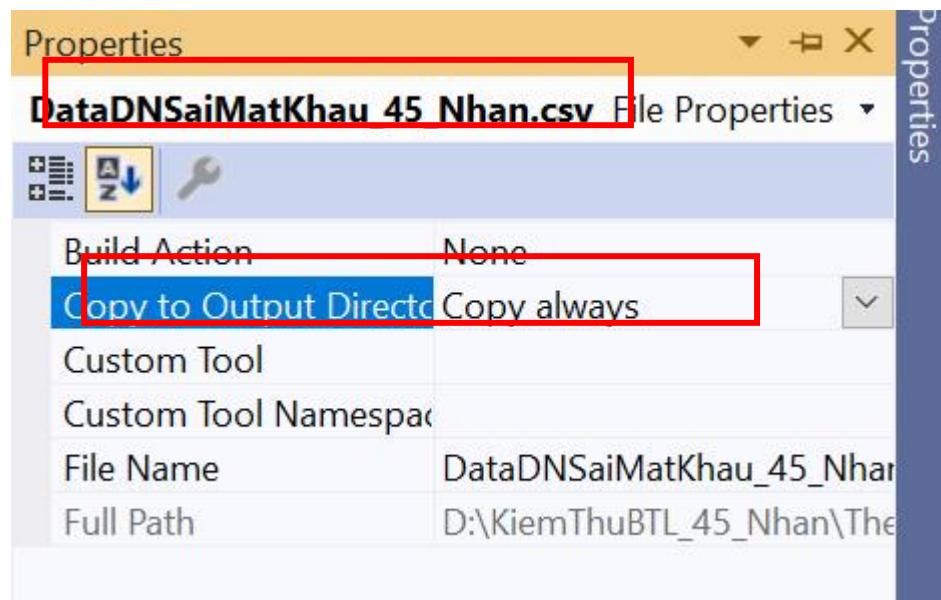
6.6. Các bước thực thi

Bước 1: Tạo 1 tập tin DataDNSaiMatKhau_45_Nhan.csv, nhập dữ liệu đầu vào

DataDNSaiMatKhau_45_Nhan.csv		UnitTestDangNhap_45_Nhan
1	Email, PassSai	
2	"ebl.tryhnt261227@gmail.com", "1434354"	

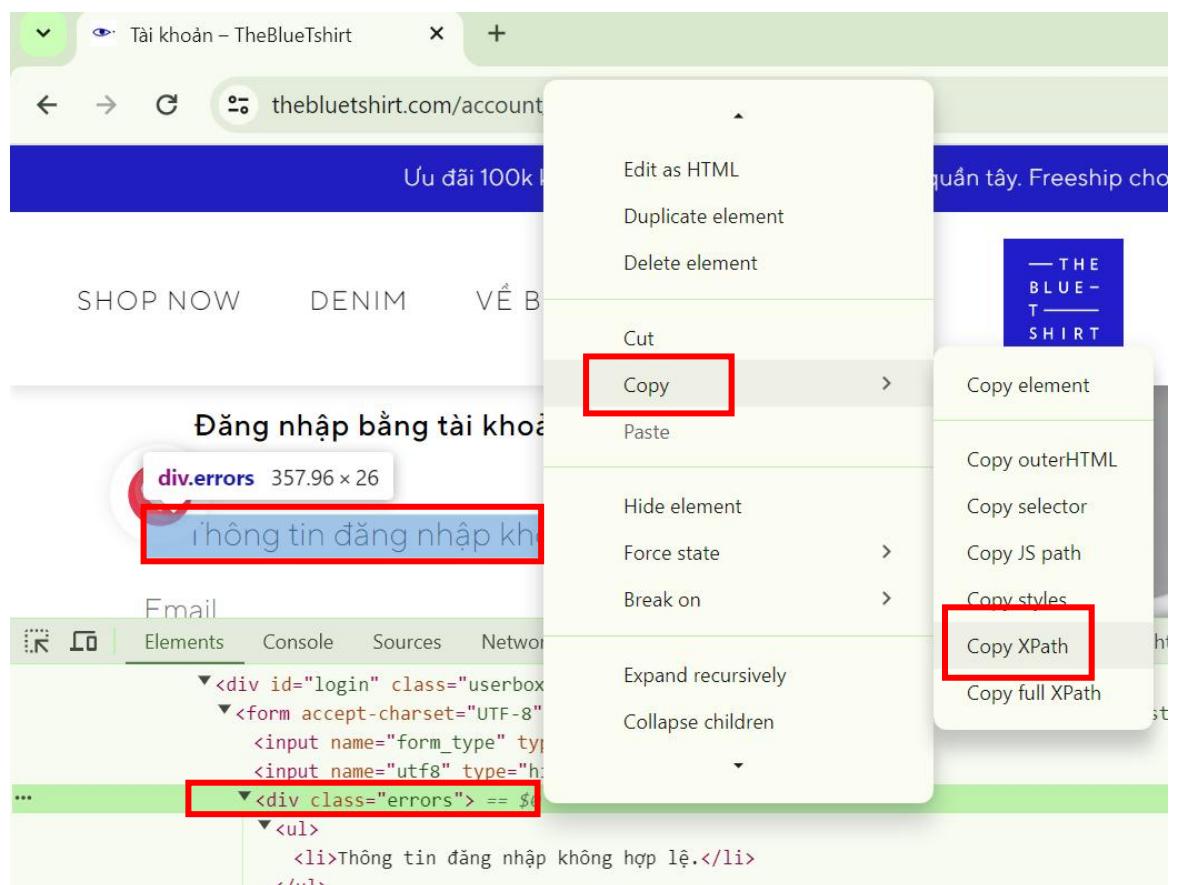
Hình 2.6.11: Dữ liệu tập tin csv

Bước 2: Chính thuộc tính “Copy to Output Driectory” thành “Copy always”



Hình 2.6.12: Thay đổi thuộc tính “Copy to Output Driectory”

Bước 3: Bắt element Xpath của cảnh báo



Hình 2.6.13: Element của cảnh báo

Bước 4: Viết code cho test case

- Code:

```
// Test case 3 email đúng pass sai STT_Ten: 45_Nhan

// Khai báo DataSource STT_Ten: 45_Nhan

[DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",
    @".\DataLogin_45_Nhan\DataDNSaiMatKhau_45_Nhan.csv",
    "DataDNSaiMatKhau_45_Nhan#csv", DataAccessMethod.Sequential)]

[TestMethod]

public void TestDNSaiMatKhau_45_Nhan()

{
    string e_45_Nhan = TestContext.DataRow[0].ToString();

    string p_45_Nhan = TestContext.DataRow[1].ToString();

    DangNhap_45_Nhan(e_45_Nhan, p_45_Nhan);

    // Cảnh báo kì vọng STT_Ten: 45_Nhan

    string ex_45_Nhan = "Thông tin đăng nhập không hợp lệ.';

    // Lấy đường cảnh báo trang web sau khi nhấn "ĐĂNG NHẬP" STT_Ten: 45_Nhan

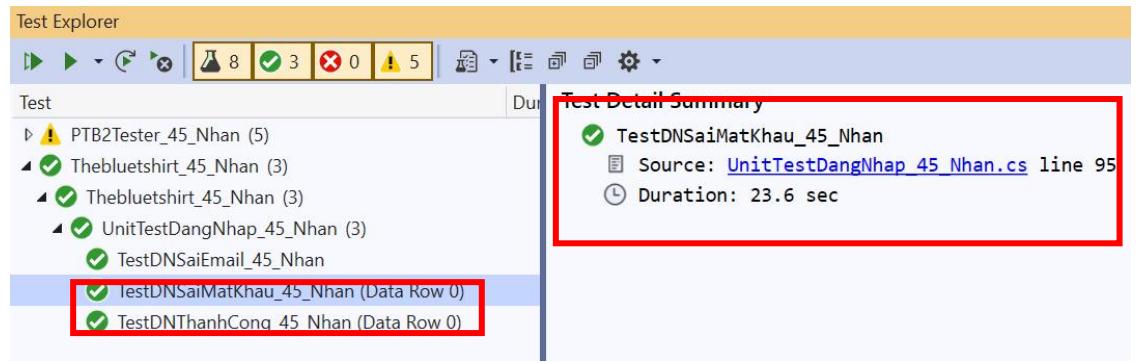
    string ac_45_Nhan =
driver_45_Nhan.FindElement(By.XPath("//*[@id=\"customer_login\"]/div[1]")).Text;

    Assert.AreEqual(ex_45_Nhan, ac_45_Nhan);

    driver_45_Nhan.Quit();

}
```

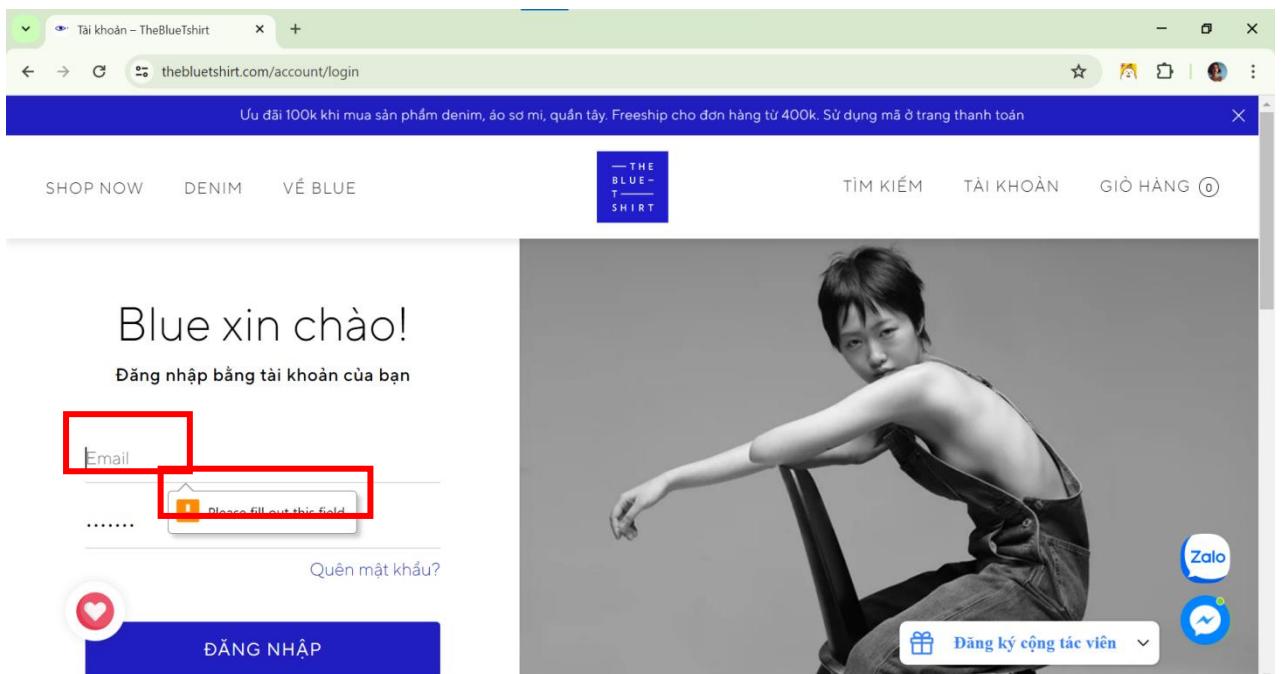
Bước 5: Kết quả chạy test case:



Hình 2.6.14: Test case pass

6.7. Test case 4: TestDNEmailRong_45_Nhan()

- Test case này kiểm tra xem hành động đăng nhập có thất bại hay không, với trường hợp Email rỗng
- Giao diện trang web khi Email rỗng



Hình 2.6.15: Cảnh báo thanh input Email

- Kết quả kì vọng: cảnh báo do thanh input Email có thuộc tính bắt buộc nhập
- Kết quả thực tế: cảnh báo xuất hiện sau khi bấm nút “ĐĂNG NHẬP”

- Nếu 2 kết quả giống nhau thì test case pass, ngược lại là fail

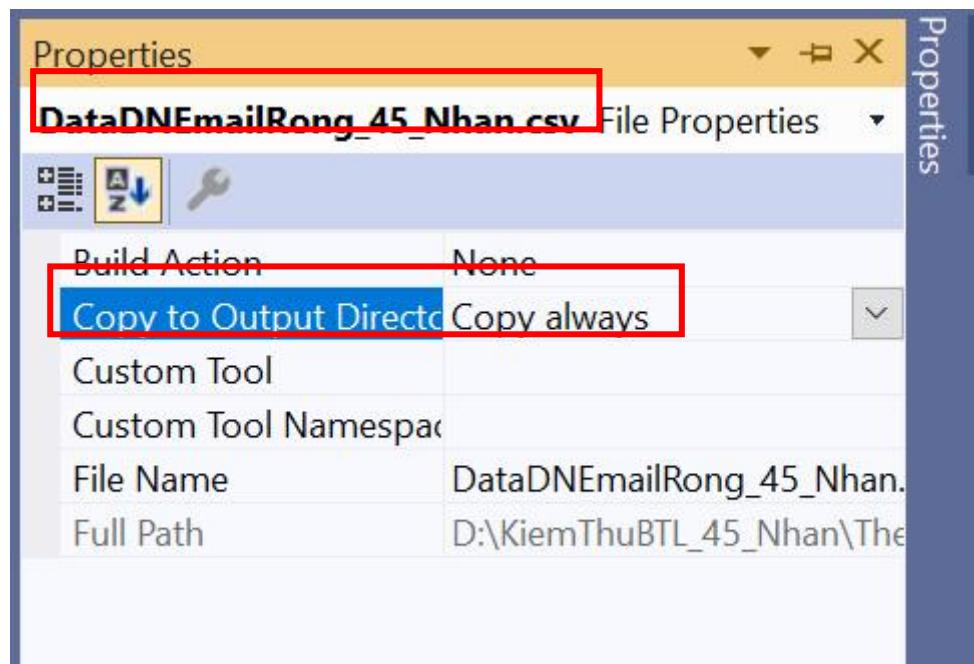
6.8. Các bước thực thi

Bước 1: Tạo 1 tập tin DataDNEmailRong_45_Nhan.csv, nhập dữ liệu đầu vào

DataDNEmailRong_45_Nhan.csv	
1	EmailRong,Pass
2	"","12345"
3	"","343545"

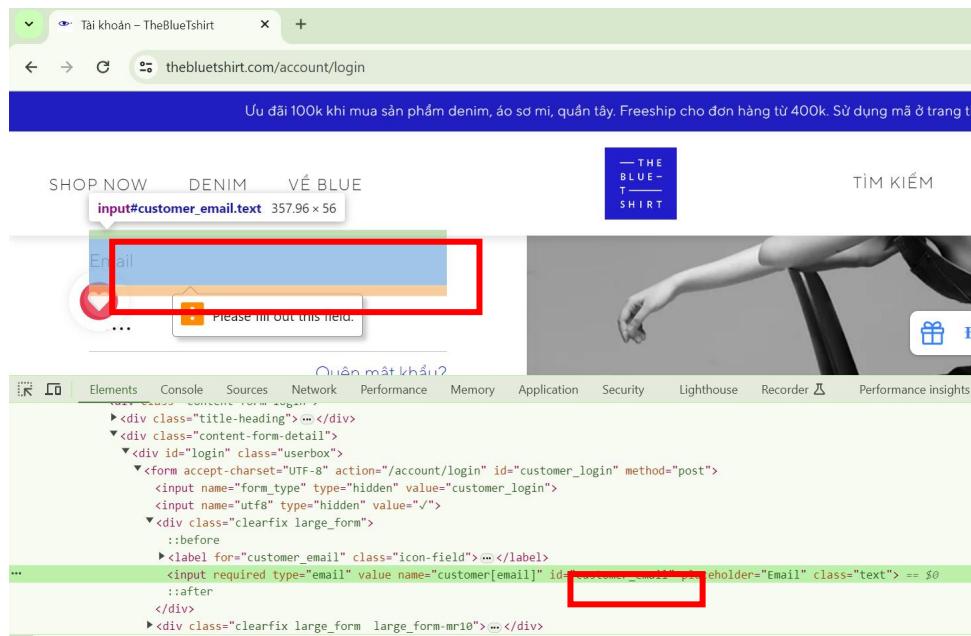
Hình 2.6.16: Dữ liệu tập tin csv

Bước 2: Chính thuộc tính “Copy to Output Directory” thành “Copy always”



Hình 2.6.17: Thay đổi thuộc tính “Copy to Output Directory”

Bước 3: Bắt element Id của cảnh báo thanh input Email



Hình 2.6.18: Element của cảnh báo input Email

Bước 4: Viết code cho test case

- Code:

```
// Test case 4 email rỗng pass đúng/sai/rỗng STT_Ten: 45_Nhan

// Khai báo DataSource STT_Ten: 45_Nhan

[DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",
    @"\.\DataLogin_45_Nhan\DataTableRong_45_Nhan.csv",
    "DataTableRong_45_Nhan#csv", DataAccessMethod.Sequential)]

[TestMethod]

public void TestDNEmailRong_45_Nhan()

{
    string e_45_Nhan = TestContext.DataRow[0].ToString();

    string p_45_Nhan = TestContext.DataRow[1].ToString();

    DangNhap_45_Nhan(e_45_Nhan, p_45_Nhan);

    // Nếu có thuộc tính bắt buộc nhập STT_Ten: 45_Nhan

    string ex_45_Nhan = "true";

    // Kiểm tra thuộc tính của thanh input Email có bắt buộc nhập hay không STT_Ten:
45_Nhan
```

```

        string ac_45_Nhan =
driver_45_Nhan.FindElement(By.Id("customer_email")).GetAttribute("required");

        // Nếu ac_45_Nhan không null -> trả về giá trị True
        Assert.IsNotNull(ac_45_Nhan);

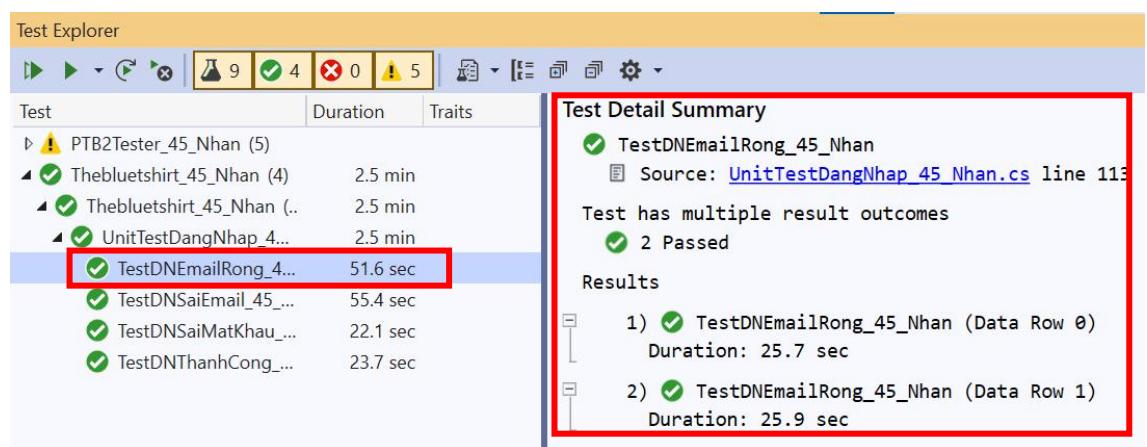
        Assert.AreEqual(ex_45_Nhan, ac_45_Nhan.ToLower());

        driver_45_Nhan.Quit();

    }

```

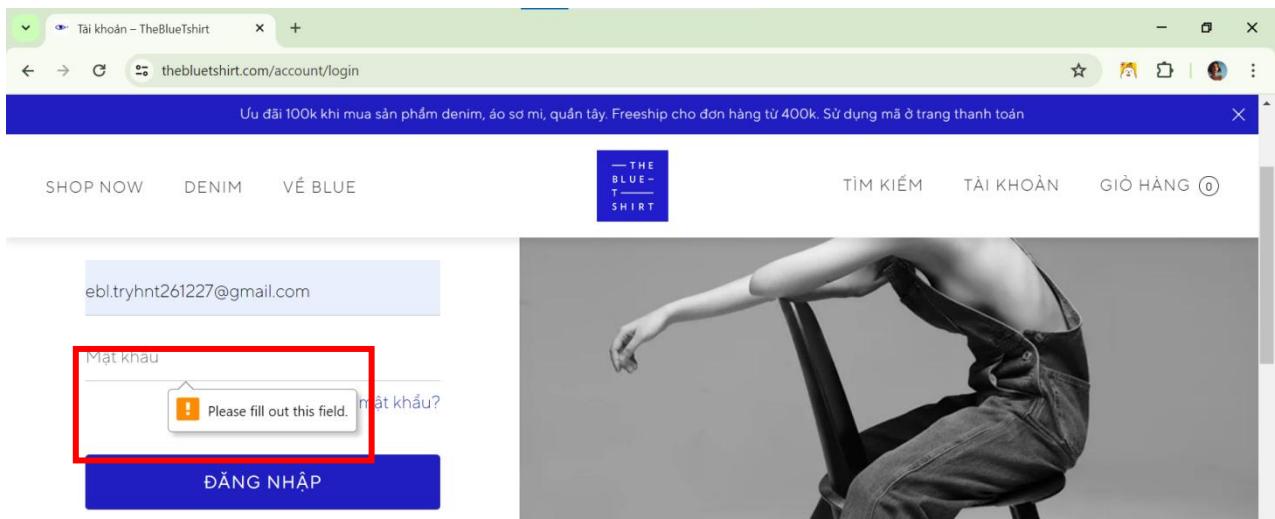
Bước 5: Kết quả chạy test case



Hình 2.6.19: Test case pass

6.9. Test case 5: TestDNMatKhauRong_45_Nhan()

- Test case này kiểm tra xem hành động đăng nhập có thất bại hay không, với trường hợp Mật khẩu rỗng
- Giao diện trang web khi Mật khẩu rỗng



Hình 2.6.20: Cảnh báo thanh input Mật khẩu

- Kết quả kì vọng: cảnh báo do thanh input Mật khẩu có thuộc tính bắt buộc nhập
- Kết quả thực tế: cảnh báo xuất hiện sau khi bấm nút “ĐĂNG NHẬP”
- Nếu 2 kết quả giống nhau thì test case pass, ngược lại là fail

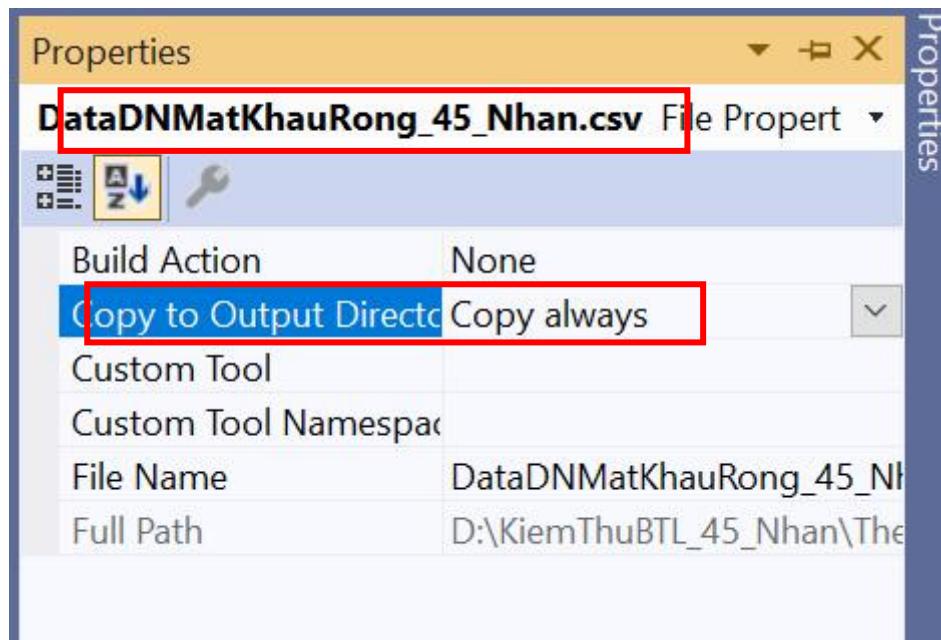
6.10. Các bước thực thi

Bước 1: Tạo 1 tập tin DataDNMatKhauRong_45_Nhan.csv, nhập dữ liệu đầu vào

DataDNMatKhauRong_45_Nhan.csv	
1	Email,PassRong
2	"ebl.tryhnt261227@gmail.com", ""
3	"jdsfhher@gmail.com", ""
4	"", ""

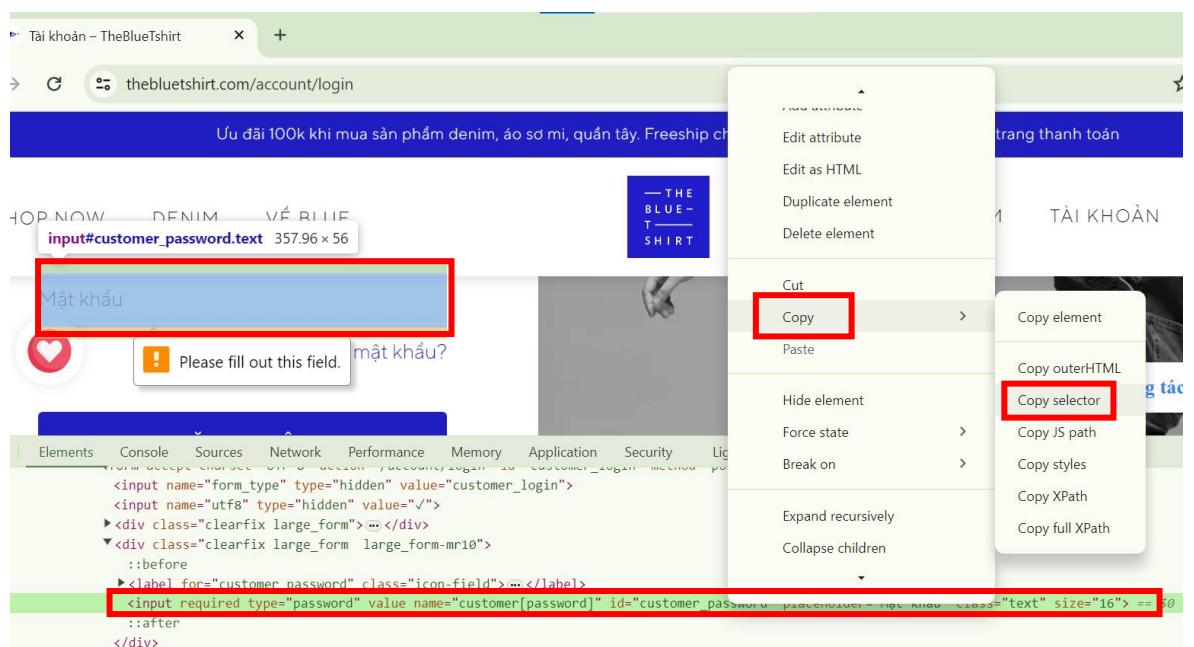
Hình 2.6.21: Dữ liệu tập tin csv

Bước 2: Chính thuộc tính “Copy to Output Directory” thành “Copy always”



Hình 2.6.22: Thay đổi thuộc tính “Copy to Output Driectory”

Bước 3: Bắt element CssSelector của cảnh báo thanh input Mật khẩu



Hình 2.6.23: Element của input Mật khẩu

Bước 4: Viết code cho test case

- Code:

```
// Test case 5 email đúng/sai/rỗng pass rỗng STT_Ten: 45_Nhan
```

```

// Khai báo DataSource STT_Ten: 45_Nhan

[DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",
    @".\DataLogin_45_Nhan\DataDNMatKhauRong_45_Nhan.csv",
    "DataDNMatKhauRong_45_Nhan#csv", DataAccessMethod.Sequential)]

[TestMethod]

public void TestDNMatKhauRong_45_Nhan()
{
    string e_45_Nhan = TestContext.DataRow[0].ToString();

    string p_45_Nhan = TestContext.DataRow[1].ToString();
    DangNhap_45_Nhan(e_45_Nhan, p_45_Nhan);

    // Nếu có thuộc tính bắt buộc nhập STT_Ten: 45_Nhan
    string ex_45_Nhan = "true";

    // Kiểm tra thuộc tính của thanh input Email có bắt buộc nhập hay không STT_Ten:
    45_Nhan

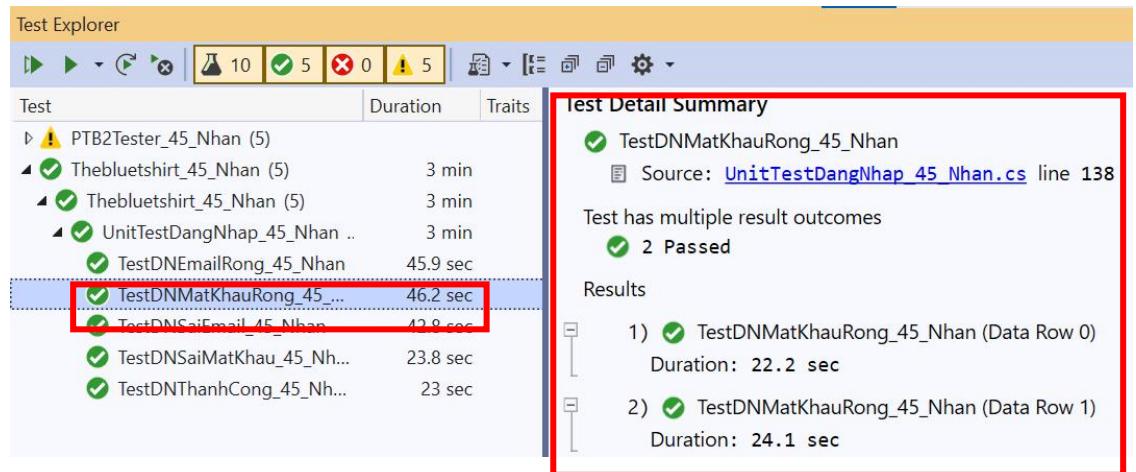
    string ac_45_Nhan =
driver_45_Nhan.FindElement(By.CssSelector("#customer_password")).GetAttribute("require
d");

    // Nếu ac_45_Nhan không null -> trả về giá trị True
    Assert.IsNotNull(ac_45_Nhan);
    Assert.AreEqual(ex_45_Nhan, ac_45_Nhan.ToLower());
    driver_45_Nhan.Quit();

}

```

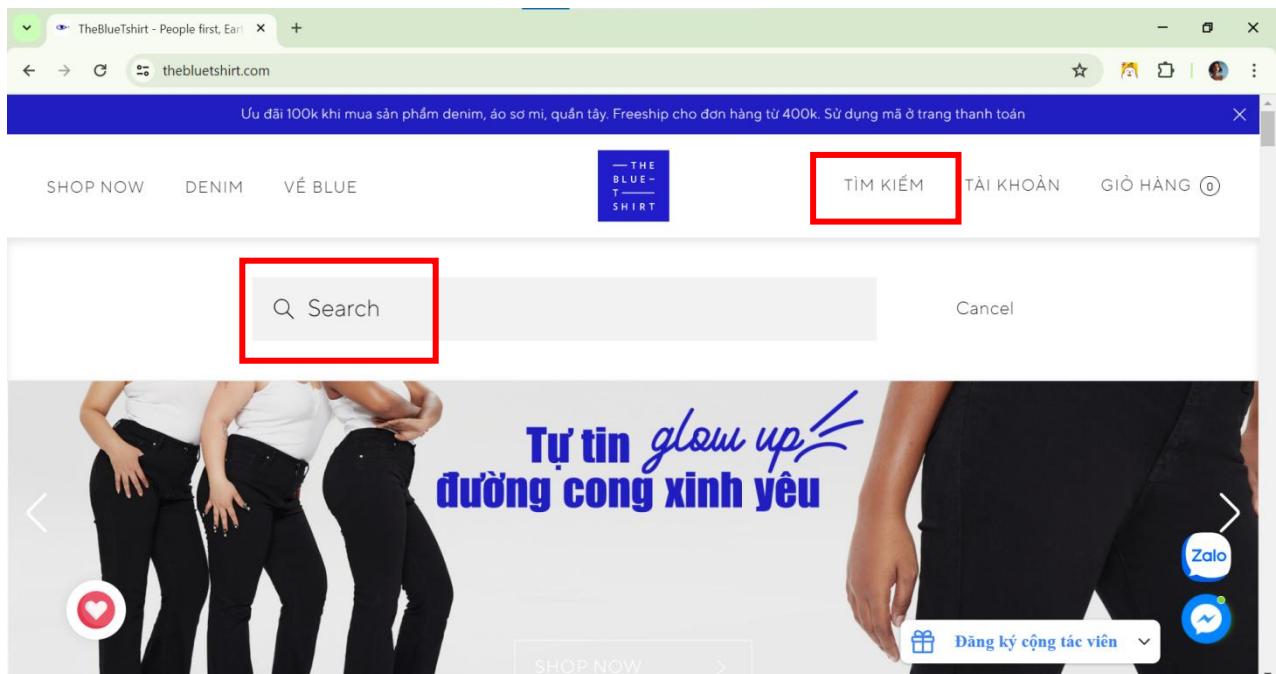
Bước 5: Kết quả chạy test case



Hình 2.6.24: Test case pass

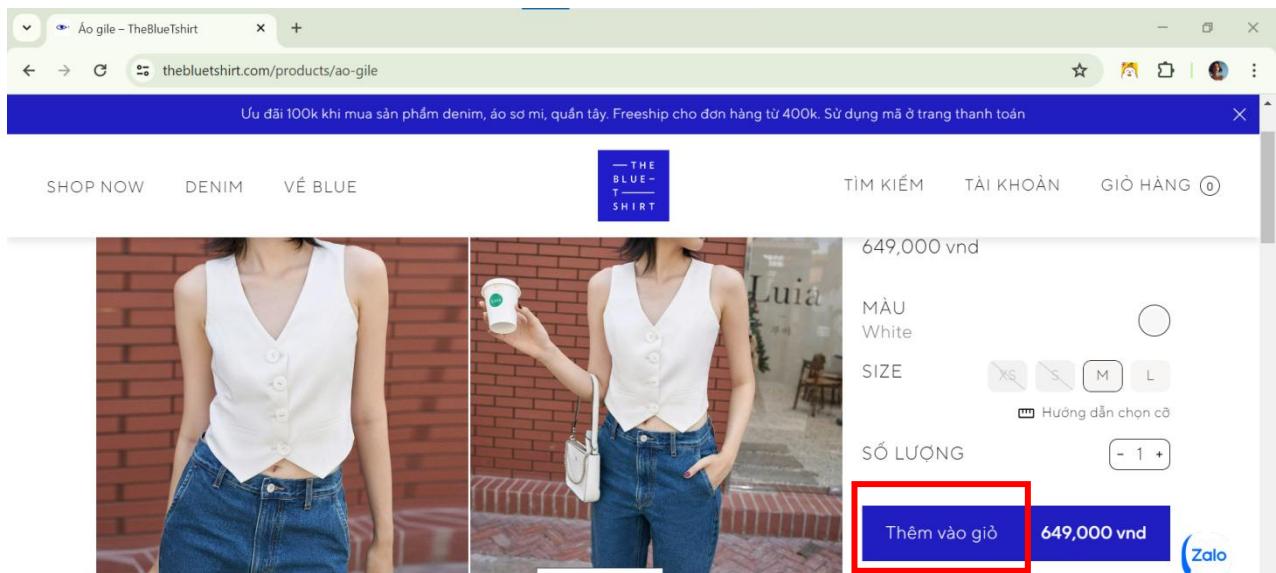
7. Mô tả chức năng tìm kiếm và thêm vào giỏ hàng

- Chức năng tìm kiếm là quy trình cho phép người dùng gõ từ khóa cần tìm kiếm vào hệ thống 1 trang web. Chức năng thêm sản phẩm vào giỏ hàng là quy trình cho phép người dùng thêm sản phẩm vào giỏ, sau đó có thể lựa chọn mua hoặc không. Và trang web cũng cho phép người dùng thực hiện 2 chức năng này liên tiếp nhau, nên em gộp 2 chức năng thành 1 chức năng là: tìm kiếm và thêm vào giỏ hàng
- Nhấn vào “TÌM KIẾM” để tìm kiếm từ khóa



Hình 2.7.1: Giao diện chức năng tìm kiếm

- Nhấn “Thêm vào giỏ” để thêm sản phẩm vào giỏ hàng



Hình 2.7.2: Giao diện chức năng thêm vào giỏ hàng

8. Xây dựng test case

- Thông qua mô tả chức năng, em xây dựng 3 test case và ghi chú như sau:

Test case 1: TestTKCoSPThemGH_45_Nhan()

Test case 2: TestTKKhongSP_45_Nhan()

Test case 3: TestTKRong_45_Nhan()

Các điều kiện	Từ khóa tìm kiếm	T	F	B
	Thông điệp lỗi			M1
Các hành động	Tìm kiếm có sản phẩm	T	F	F
	Thêm vào giỏ hàng	T	F	F

Hình 2.8.1: Các test case cho chức năng tìm kiếm và thêm vào giỏ hàng

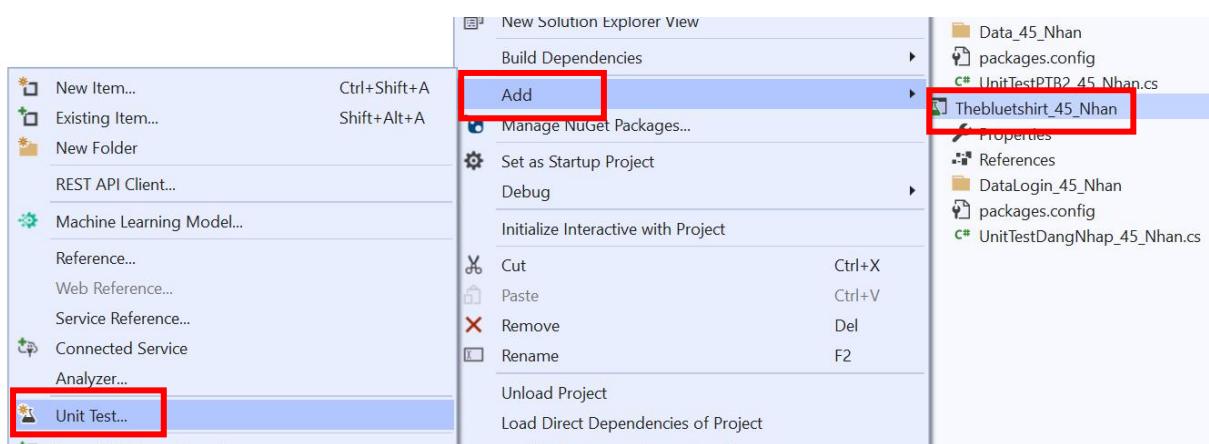
Xác định các điều kiện: từ khóa tìm kiếm

Mỗi đầu vào trên nhận 1 trong 3 giá trị: không nhập từ khóa (B), từ khóa có trong cơ sở dữ liệu (T), từ khóa không có trong cơ sở dữ liệu (F)
M1: Vui lòng nhập từ khóa tìm kiếm và thử lại.

Hình 2.8.2: Ghi chú test case

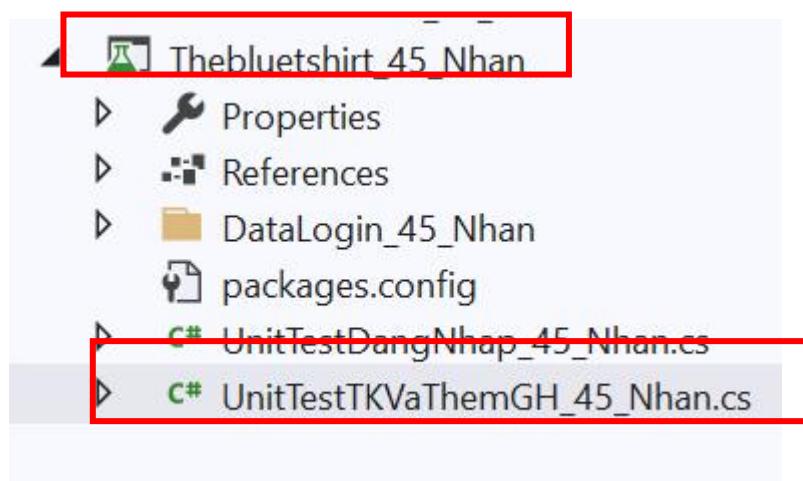
9. Tạo 1 tập tin UnitTestTKVaThemGH_45_Nhan.cs để kiểm thử các test case của chức năng tìm kiếm và thêm vào giỏ hàng

Bước 1: Click chuột phải project Thebluetshirt_45_Nhan -> Add -> Unit Test



Hình 2.9.1: Tạo tập tin unit test mới

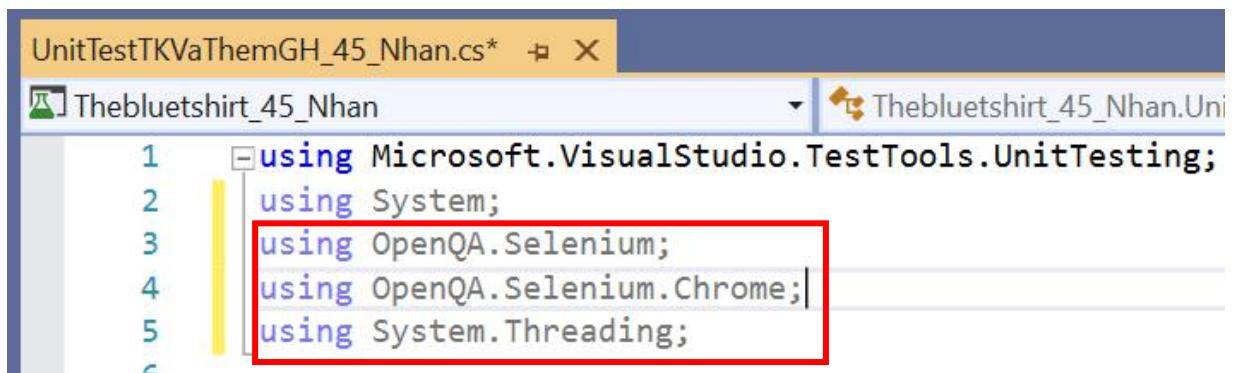
Bước 2: Click chuột phải vào unit test vừa tạo -> Rename thành
UnitTestTKVaThemGH_45_Nhan.cs



Hình 2.9.2: Đổi tên thành công

Bước 3: Tại vùng sử dụng thư viện, thêm các thư viện như sau:

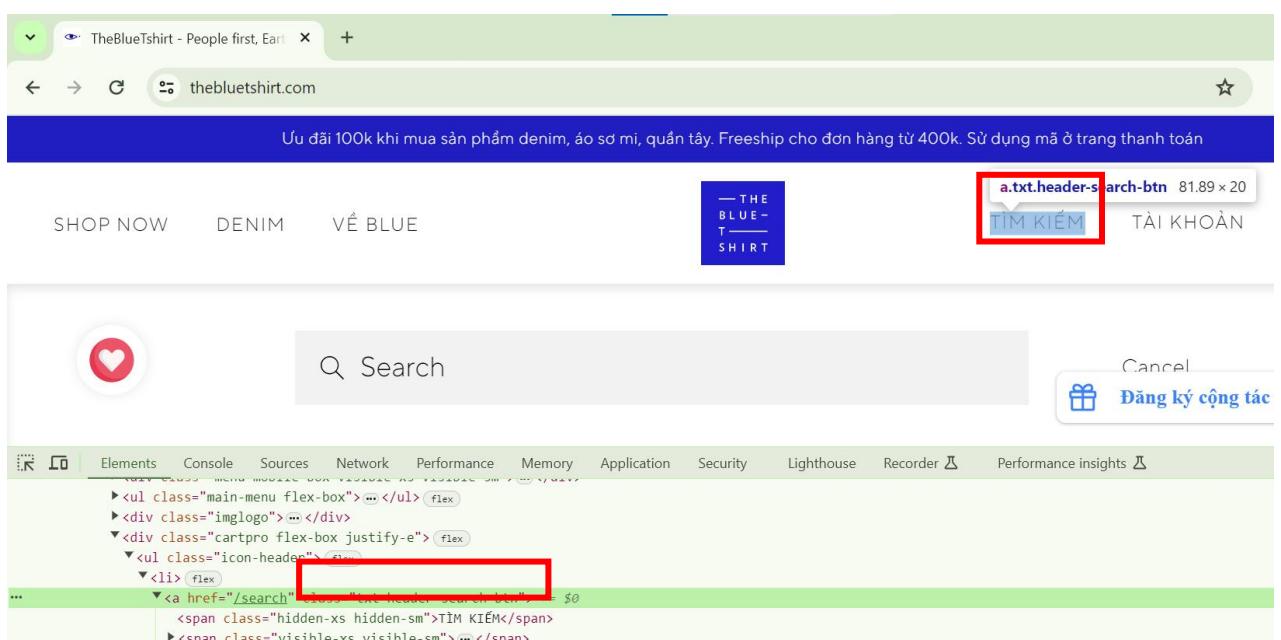
```
using Microsoft.VisualStudio.TestTools.UnitTesting;  
  
using System;  
  
using OpenQA.Selenium;  
  
using OpenQA.Selenium.Chrome;  
  
using System.Threading;
```



Hình 2.9.3: Thêm thư viện

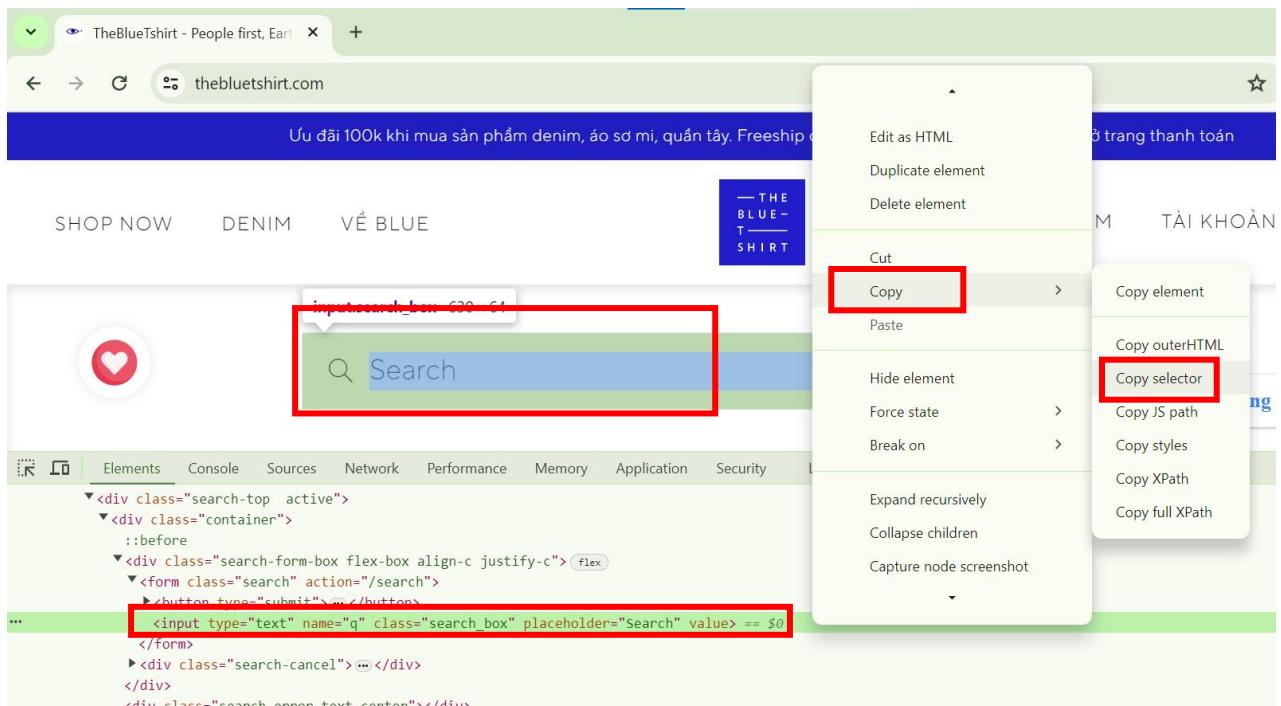
10. Lấy các elements

Bước 1: Lấy element ClassName của nút “TÌM KIẾM”



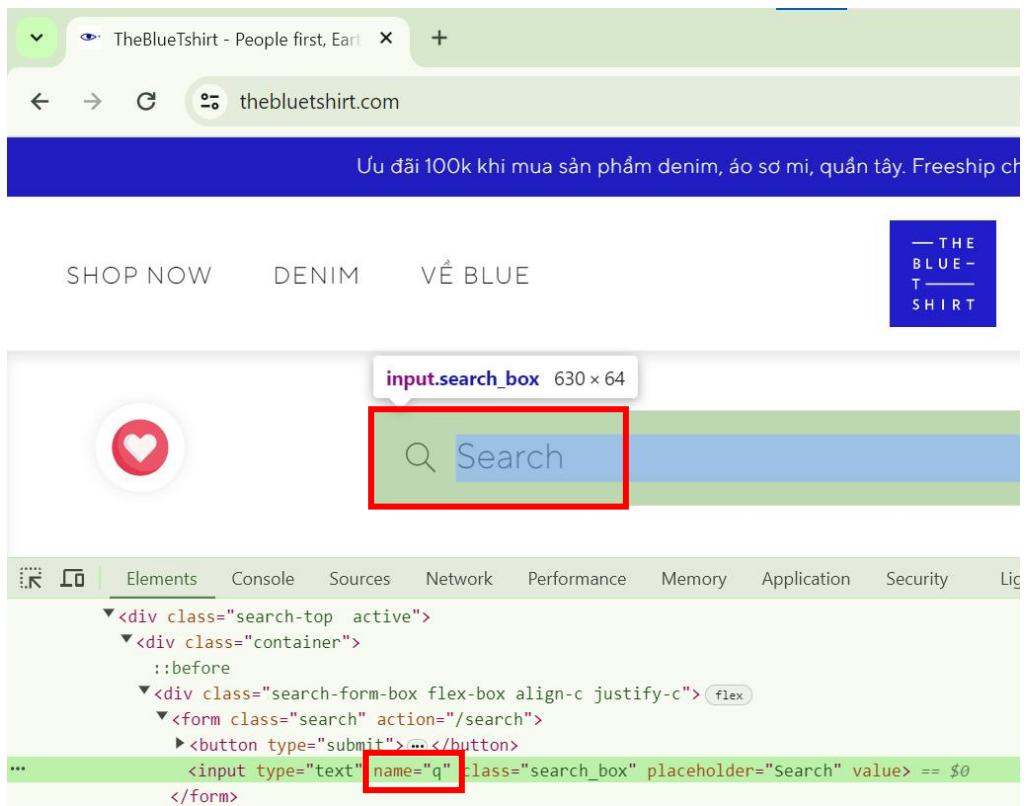
Hình 2.10.1: Element của nút “TÌM KIẾM”

Bước 2: Lấy element CssSelector của thanh input Search



Hình 2.10.2: Element của thanh input Search

Bước 3: Lấy element Name của input Search



Hình 2.10.3: Element của input Search

11. Chuẩn bị trước khi test các test case

- Thiết lập các biến và phương thức cho class kiểm thử chức năng tìm kiếm và thêm vào giỏ hàng (UnitTestTKVaThemGH_45_Nhan)
 - Code:
- ```

// Biến cục bộ

IWebDriver driver_45_Nhan = new ChromeDriver();

// Phương thức này dùng để khởi chạy trang chủ và xóa đi thanh div che phủ các nút
// và nhấp vào TÌM KIẾM STT_Ten: 45_Nhan

public void SetUpTK_45_Nhan()

{
 // Chạy trang chủ STT_Ten: 45_Nhan

 driver_45_Nhan.Navigate().GoToUrl("https://thebluetshirt.com/");

 // Web nghỉ 3s STT_Ten: 45_Nhan

 Thread.Sleep(3000);
}

```

```

// Xóa đi thanh div (nhấn vào dấu X) che phủ các nút STT_Ten: 45_Nhan
driver_45_Nhan.FindElement(By.CssSelector("#popup-contact > div > div >
" +
 "div.popup-right.flex-box.flex-c.justify-s > button")).Click();

Thread.Sleep(3000);

// Click nút "TÌM KIẾM" STT_Ten: 45_Nhan
driver_45_Nhan.FindElement(By.ClassName("header-search-btn")).Click();

Thread.Sleep(3000);

}

// Phương thức này dùng để tìm kiếm sản phẩm với biến STT_Ten: 45_Nhan
public void TimKiem_45_Nhan(string key_45_Nhan)
{
 SetUpTK_45_Nhan();

 // Nhập từ khóa tìm kiếm STT_Ten: 45_Nhan
 driver_45_Nhan.FindElement(By.CssSelector("body > div.mm-page > header >
div.nav-top > " +
 "div.search-top.active > div > " +
 "div.search-form-box.flex-box.align-c.justify-c > form >
input")).SendKeys(key_45_Nhan);

 // Trên thanh search nhấn nút enter của bàn phím STT_Ten: 45_Nhan
 driver_45_Nhan.FindElement(By.Name("q")).SendKeys(Keys.Enter);

 Thread.Sleep(3000);

}

// Phương thức này dùng để thêm sản phẩm vào giỏ hàng STT_Ten: 45_Nhan
public void ThemSP_45_Nhan()
{
 // Nhấn vào thanh div chứa sản phẩm STT_Ten: 45_Nhan
}

```

```

driver_45_Nhan.FindElement(By.XPath("//*[@id=\"search\"]/div[2]/div[2]/div/div/div/div[1]/div")).Click();

Thread.Sleep(3000);

// Thực hiện 5 lần nhấn nút xuống trên bàn phím STT_Ten: 45_Nhan

for (int i_45_Nhan =0;i_45_Nhan<5;i_45_Nhan++)

{

 driver_45_Nhan.FindElement(By.TagName("body")).SendKeys(Keys.ArrowDown);

}

Thread.Sleep(3000);

// Click nút thêm vào giỏ STT_Ten: 45_Nhan

driver_45_Nhan.FindElement(By.Id("add-to-cart")).Click();

Thread.Sleep(3000);

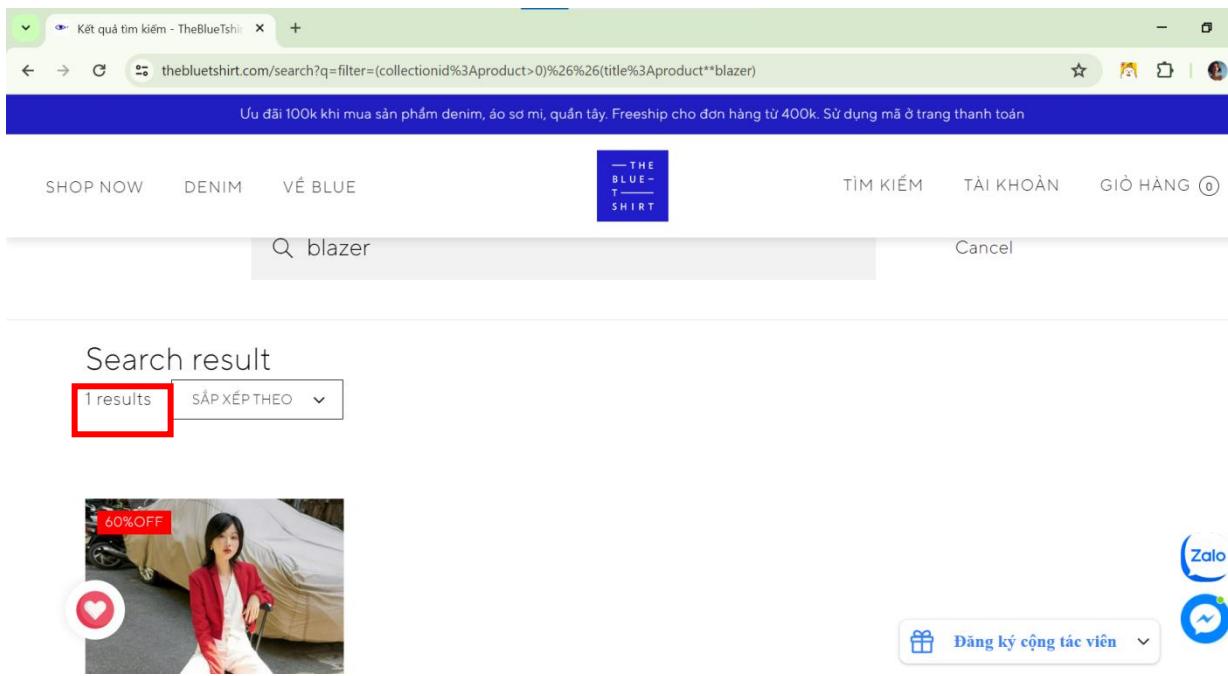
}

```

## **12. Viết các test case cho chức năng tìm kiếm và thêm vào giỏ hàng**

### **12.1. Test case 1: TestTKCoSPThemGH\_45\_Nhan()**

- Test case này cho phép tìm kiếm có sản phẩm sau đó thêm sản phẩm vào giỏ hàng
- Giao diện trang web sau khi tìm kiếm có sản phẩm

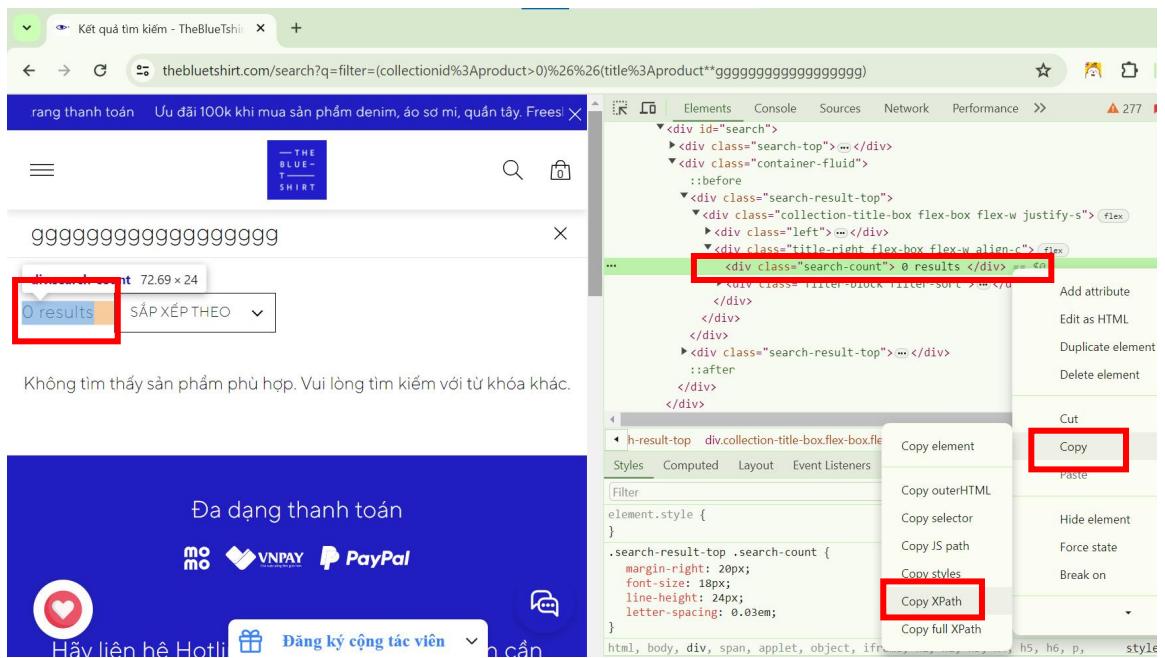


Hình 2.12.1: Giao diện trang web sau khi tìm kiếm có sản phẩm

- Kết quả kì vọng: số results lớn hơn 0
- Kết quả thực tế: số results sau khi nhấn “Enter”
- Nếu số results  $> 0$  thì kết quả trả về true. Nếu kết quả là true thì thêm vào giỏ hàng, test case pass, ngược lại là fail

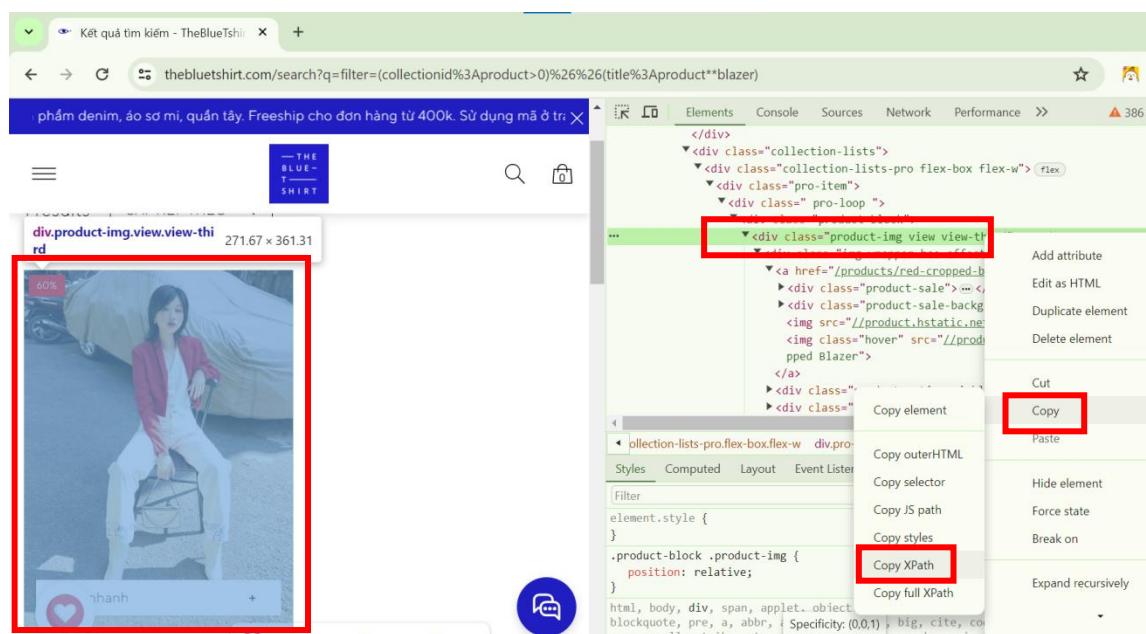
## 12.2. Các bước thực thi test case

Bước 1: Bắt element Xpath của kết quả tìm kiếm



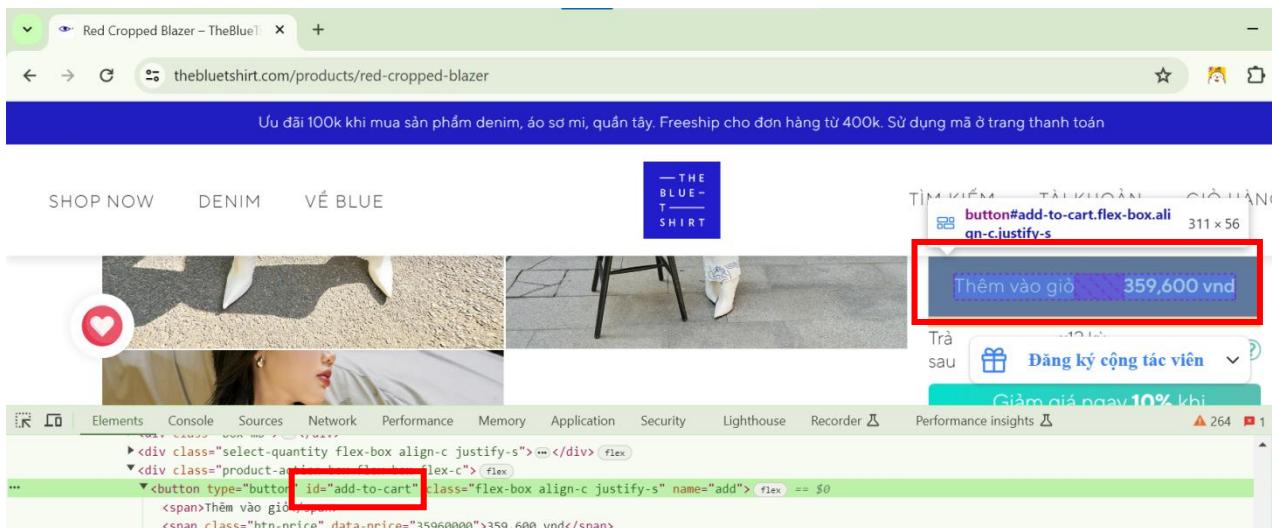
Hình 2.12.2: Element Xpath của kết quả tìm kiếm

### Bước 2: Bắt element Xpath của sản phẩm tìm thấy đầu tiên



Hình 2.12.3: Element Xpath của sản phẩm tìm thấy đầu tiên

### Bước 3: Bắt element Id của button thêm vào giỏ



Hình 2.12.3: Element Id của button thêm vào giỏ

#### Bước 4: Viết code cho test case

```
// Test case 1 STT_Ten: 45_Nhan

[TestMethod]

public void TestTKCoSPTThemGH_45_Nhan()

{
 string kw_45_Nhan = "blazer";

 TimKiem_45_Nhan(kw_45_Nhan);

 // Kết quả tìm kiếm STT_Ten: 45_Nhan

 string kq_45_Nhan =
driver_45_Nhan.FindElement(By.XPath("//*[@id=\"search\"]//div[2]//div[1]//div[2]//div[1]")).Text;

 // Lấy chữ đầu tiên chuyển thành số STT_Ten: 45_Nhan

 int ac_45_Nhan = int.Parse(kq_45_Nhan.Substring(0, kq_45_Nhan.IndexOf(' ')));

 // Nếu số lớn hơn 0 thì trả về true STT_Ten: 45_Nhan

 Assert.IsTrue(ac_45_Nhan > 0);

 // Nếu tìm kiếm thành công -> thêm sản phẩm vào giỏ hàng STT_Ten: 45_Nhan

 if (true)

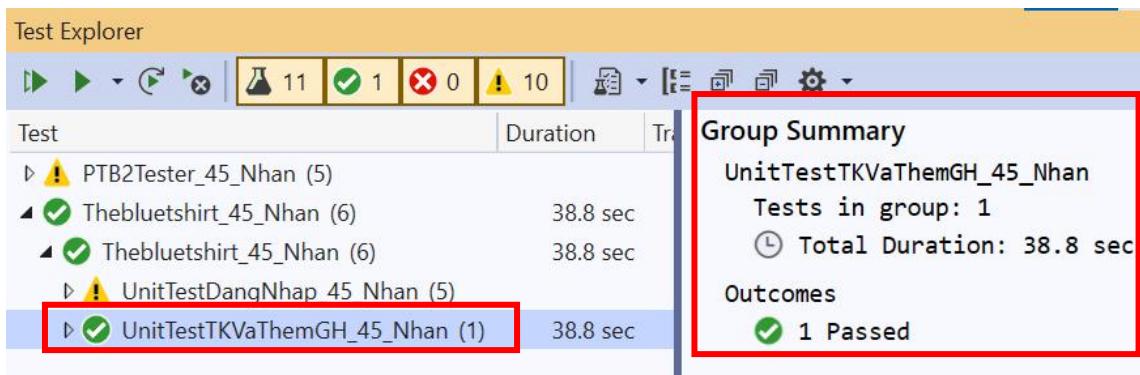
 {
 ThemSP_45_Nhan();
 }
}
```

```
}

driver_45_Nhan.Quit();

}
}
```

#### Bước 5: Kết quả chạy test case



Hình 2.12.4: Test case pass

### 12.3. Test case 2: TestTKKhongSP 45 Nhan()

- Test case này thực hiện tìm kiếm không có sản phẩm và cũng không thêm vào giỏ hàng
  - Giao diện trang web sau khi tìm kiếm không có sản phẩm

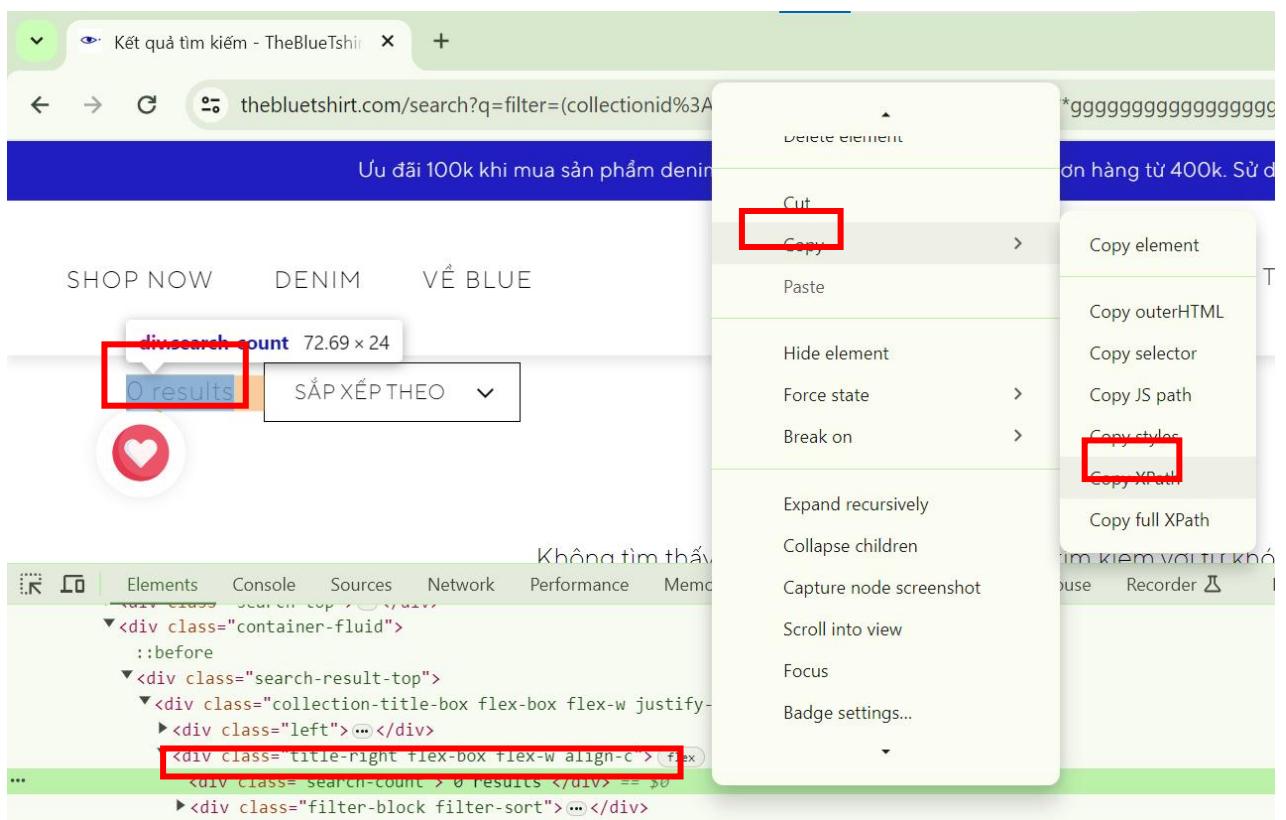


Hình 2.12.5: Giao diện trang web sau khi tìm kiếm không có sản phẩm

- Kết quả kì vọng: số results bằng 0
- Kết quả thực tế: số results sau khi nhấn “Enter”
- Nếu số results = 0 thì kết quả trả về true, test case pass ngược lại fail

#### 12.4. Các bước thực thi test case

Bước 1: Bắt element Xpath của kết quả tìm kiếm



Hình 2.12.6: Element Xpath của kết quả tìm kiếm

Bước 2: Viết code cho test case

```
// Test case 2 STT_Ten: 45_Nhan

[TestMethod]

public void TestTKKhongSP_45_Nhan()

{
 string kw_45_Nhan = "lovito";

 TimKiem_45_Nhan(kw_45_Nhan);
}
```

```

 // Lấy dòng chữ sau khi nhấn enter STT_Ten: 45_Nhan

 string kq_45_Nhan =
driver_45_Nhan.FindElement(By.XPath("//*[@id=\"search\"]/div[2]/div[1]/div/div[2]/div[1]")).Text;

 // Lấy chữ đầu tiên chuyển thành số STT_Ten: 45_Nhan

 int ac_45_Nhan = int.Parse(kq_45_Nhan.Substring(0, kq_45_Nhan.IndexOf(' ')));

 // Nếu số = 0 thì trả về true STT_Ten: 45_Nhan

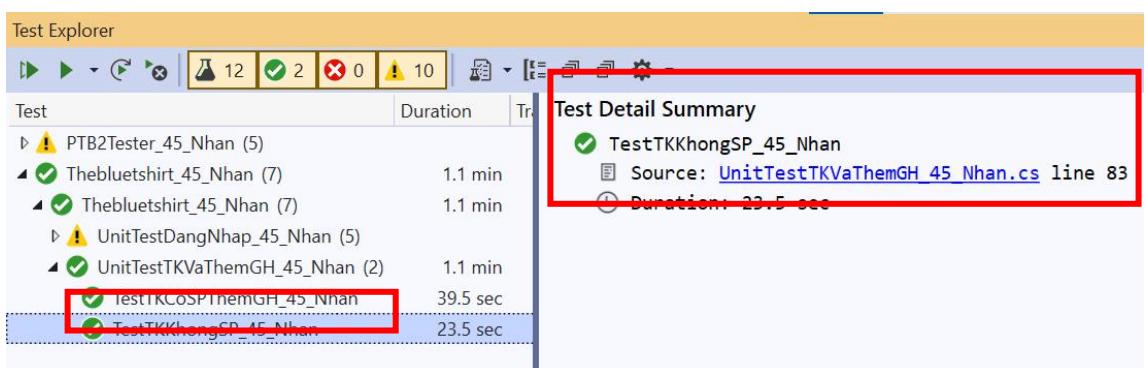
 Assert.IsTrue(ac_45_Nhan == 0);

 driver_45_Nhan.Quit();

 }

```

### Bước 3: Kết quả chạy test case



Hình 2.12.7: Test case pass

### 12.5. Test case 3: TestTKRong\_45\_Nhan()

- Test case này thực hiện không nhập từ khóa khi tìm kiếm
- Giao diện trang web sau khi không nhập từ khóa tìm kiếm



*Hình 2.12.8: Giao diện trang web sau khi không nhập từ khóa tìm kiếm*

- Kết quả kì vọng: hiển thị cảnh báo "Vui lòng nhập từ khóa tìm kiếm và thử lại."
- Kết quả thực tế: cảnh báo sau khi nhấn “Enter”
- Nếu 2 kết quả giống nhau thì test case pass, ngược lại là fail

## 12.6. Các bước thực thi test case

Bước 1: Bắt element ClassName của cảnh báo



*Hình 2.12.9: Element của cảnh báo*

## Bước 2: Viết code cho test case

```
// Test case 3 STT_Ten: 45_Nhan

[TestMethod]

public void TestTKRong_45_Nhan()

{
 string kw_45_Nhan = "";

 TimKiem_45_Nhan(kw_45_Nhan);

 // Cảnh báo mong muốn STT_Ten: 45_Nhan

 string ex_45_Nhan = "Vui lòng nhập từ khóa tìm kiếm và thử lại.";

 // Lấy dòng chữ cảnh báo STT_Ten: 45_Nhan

 string ac_45_Nhan =
driver_45_Nhan.FindElement(By.ClassName("search-error")).Text;

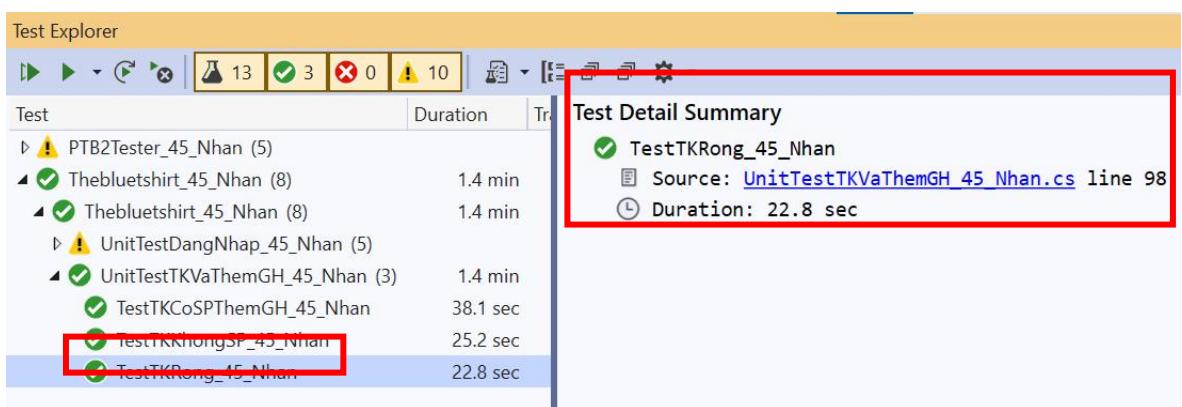
 Assert.AreEqual(ex_45_Nhan, ac_45_Nhan);

 driver_45_Nhan.Quit();

}

}
```

## Bước 3: Kết quả chạy test case



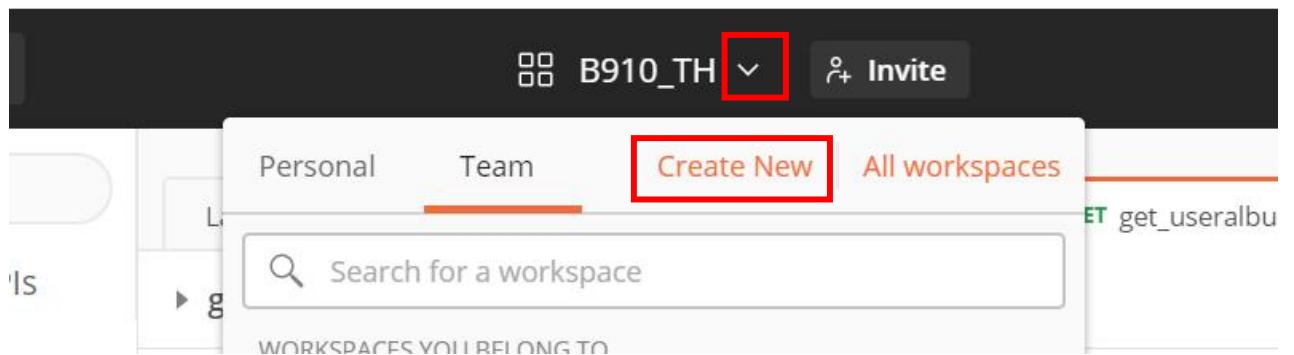
Hình 2.12.10: Test case pass

## Chương 3: API

### 1. Test API bằng Postman

#### 1.1. Chuẩn bị

Bước 1: Cài đặt postman, mở app Postman, tạo 1 Workspace mới



Hình 3.1.1: Chọn Create New

Bước 2: Đặt tên Workspace, có thể viết description, chọn type, nhấn create workspaces

**CREATE NEW WORKSPACE**

Name  
TestAPI\_bc\_45\_Nhan

Description  
Test api cho trang https://jsonplaceholder.typicode.com/guide/

Type  
**Team** Personal

Invite people to join this workspace

+ Enter an email address Add

You Admin

Inviting users to this workspace will add them to your Postman team, if they're not already members.

Cancel **Create Workspace**

*Hình 3.1.2: Tạo thông tin Workspace*

Postman

+ New Import Runner

TestAPI\_bc\_45\_Nhan Invite

Filter Collections APIs

History + New Collection Trash

You don't have any collections

Collections let you group related requests, making them easier to access and run.

+ Create a collection

Start something new

- Create a request
- Create a collection
- Create an environment
- Create an API
- View More

Recent workspaces

- B910\_TH
- My Workspace

Good afternoon, buinhan!

Use Launchpad to start something new, pick up where you left off, or explore some resources to help you master Postman.

Work smarter with Postman

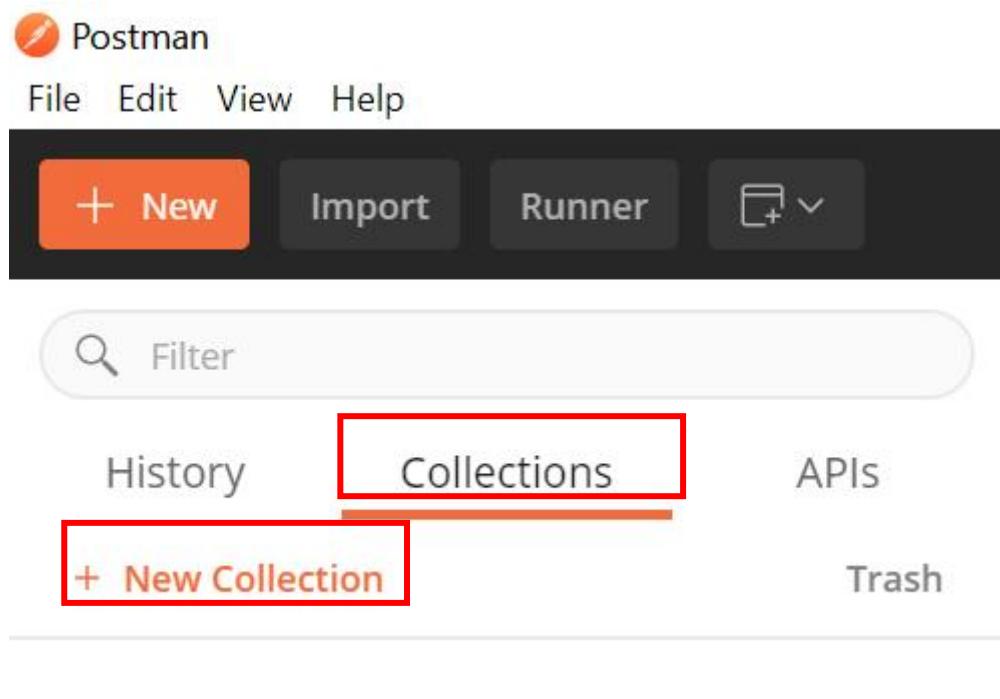
Learn how Postman can help you at every stage of the API development lifecycle with these in-app tutorials.

- Designing and mocking APIs 1 lesson
- Debugging and manual testing 4 lessons
- Collaboration 1 lesson

Find and Replace Console Bootcamp Build Browse

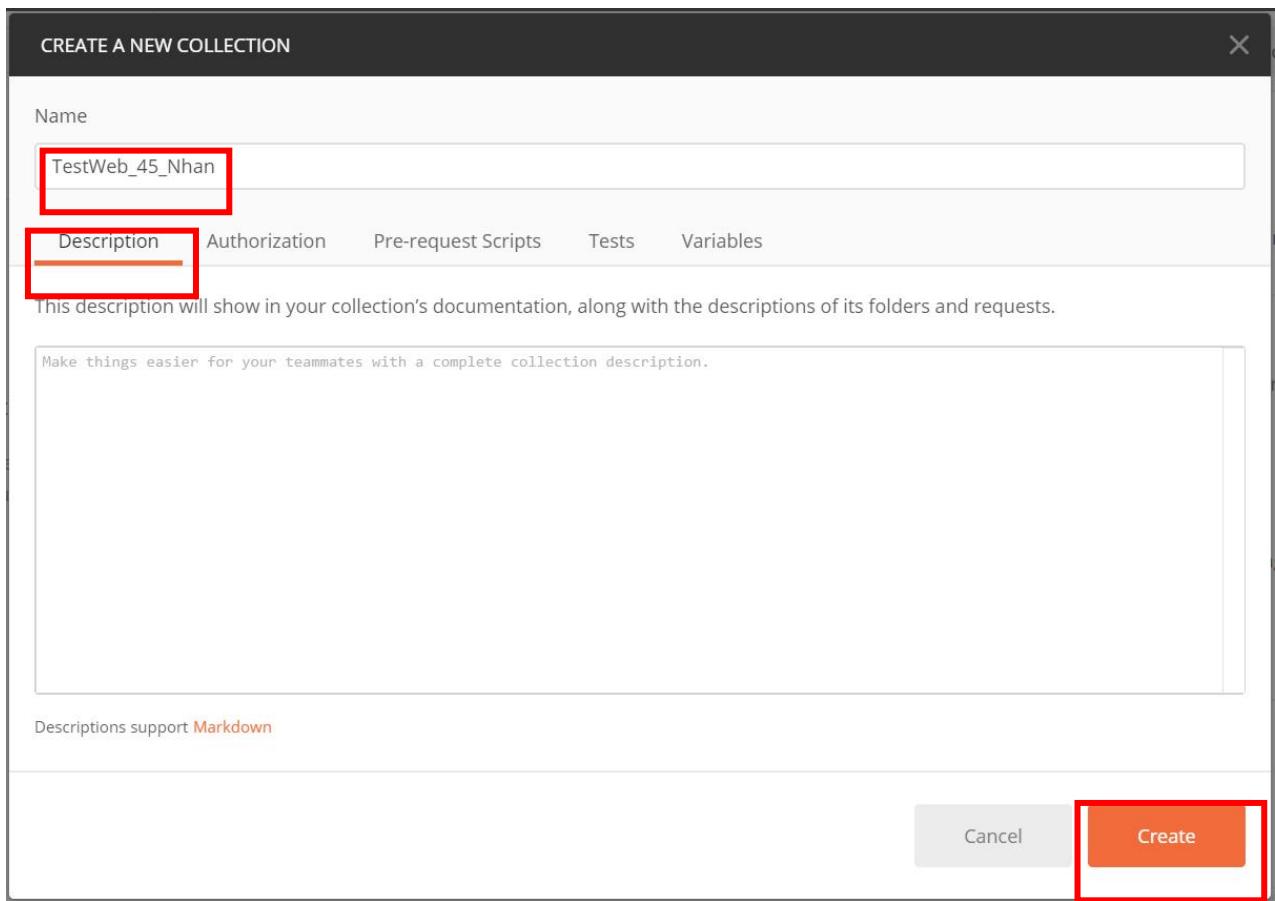
*Hình 3.1.3: Tạo thành công*

Bước 3: Tạo 1 collections mới



*Hình 3.1.4: New Collection*

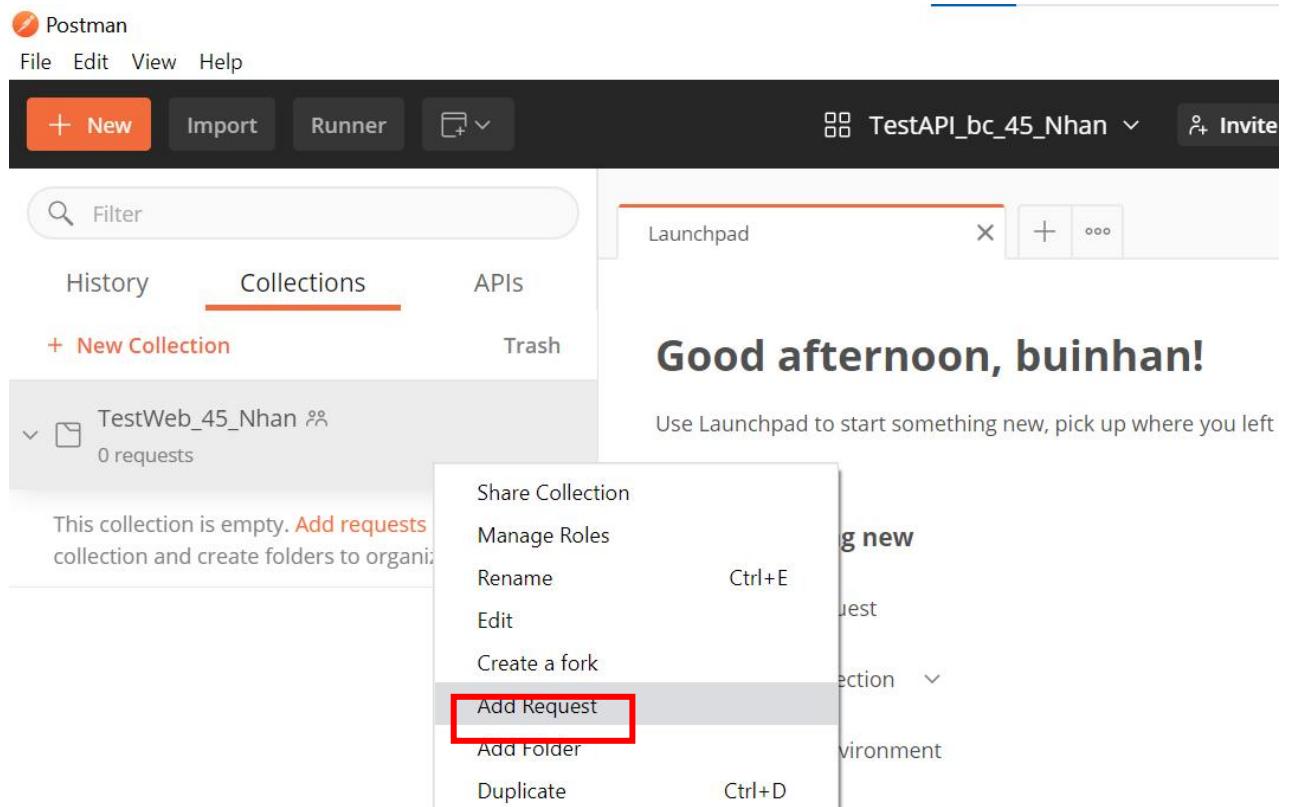
Bước 4: Đặt tên, có thể viết description -> Create



*Hình 3.1.5: Tạo thông tin collection*

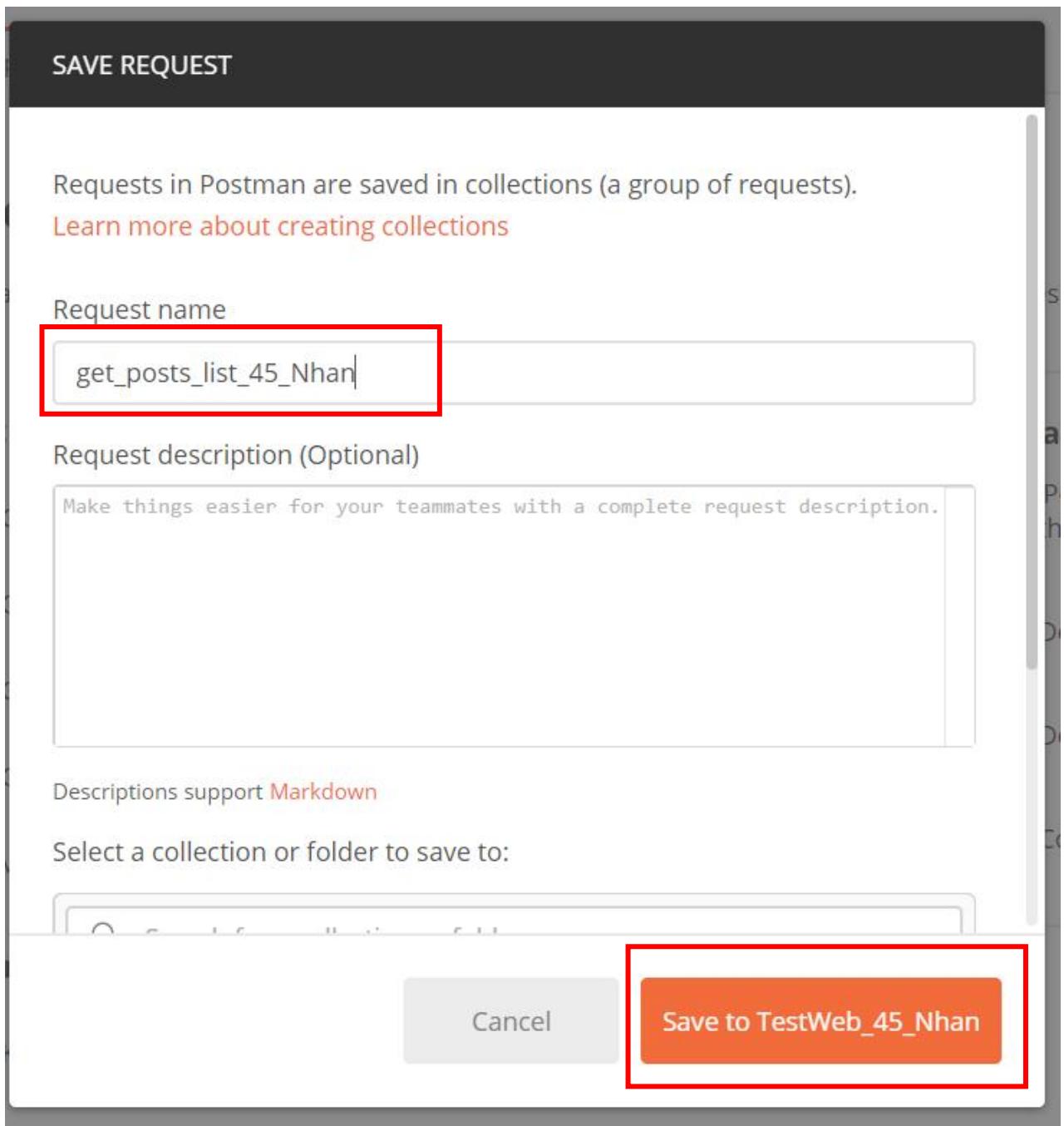
## 1.2. Phương thức GET

Bước 1: Click phải vào collection vừa tạo -> Add Request



Hình 3.1.6: Add Request

Bước 2: Đặt tên request -> Save vào collection



Hình 3.1.7: Lưu thông tin request

Bước 3: Copy đường dẫn vào thanh URL để lấy list posts -> Save -> Send, xem kết quả ở phần “Body”

The screenshot shows the Postman application interface. In the top navigation bar, there are buttons for 'File', 'Edit', 'View', 'Help', and 'Import'. The title bar says 'TestAPI\_bc\_45\_Nhan'. Below the title bar, there's a search bar with 'Filter' placeholder text. On the left sidebar, there are tabs for 'History', 'Collections', 'APIs', and '+ New Collection'. Under 'Collections', there is a section for 'TestWeb\_45\_Nhan' with a 'Request' item. A red box highlights the 'Request' item. The main workspace shows a 'Launchpad' with a 'get\_posts\_list\_45\_Nhan' entry. The 'get\_posts\_list\_45\_Nhan' entry has a 'GET' method and a URL 'https://jsonplaceholder.typicode.com/posts'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is selected and highlighted with a red box. The 'Body' tab has sub-options: 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'. The 'JSON' option is selected and highlighted with a red box. The response body is displayed in a code editor-like area, showing a JSON array of posts. This area is also highlighted with a red box. At the bottom right of the response area, there are buttons for 'Bootcamp', 'Build', 'Browse', and 'Help'. The status bar at the bottom shows 'Status: 200 OK Time: 384 ms Size: 27.97 KB Save Response'.

Hình 3.1.8: Request GET

Bước 4: Chọn tab Tests -> viết testscript cho phương thức GET

```
// Kiểm tra thời gian phản hồi dưới 300 mili giây STT_Tên: 45_Nhan
pm.test("TC1_getlist_reponseTime_45_Nhan",function() {
 pm.expect(pm.response.responseTime).to.be.below(300);
});

// Kiểm tra status code của get request là 200 STT_Tên: 45_Nhan
pm.test("TC2_getlist_status_get_45_Nhan",function() {
 pm.response.to.have.status(200);
});

// Kiểm tra header có thuộc tính Content-type và giá trị là "application/json; charset=utf-8" hay không STT_Tên: 45_Nhan
pm.test("TC3_getlist_header_45_Nhan", function() {
 pm.response.to.have.header("Content-type", "application/json; charset=utf-8")
});

// Kiểm tra kiểu dữ liệu STT_Tên: 45_Nhan
pm.test("TC4_getlist_array_45_Nhan", function() {
 var data_45_Nhan = pm.response.json();
});
```

```

pm.expect(data_45_Nhan).to.be.an("array");
});

// Kiểm tra chiều dài của mảng STT_Tên: 45_Nhan

pm.test("TC5_getlist_items_45_Nhan", function(){

 var data_45_Nhan = pm.response.json();

 pm.expect(data_45_Nhan).to.have.lengthOf.above(100);

});

// Kiểm tra phần từ 0 của mảng có chứa chuỗi sunt hay không STT_Tên: 45_Nhan

pm.test("TC6_getlist_checkvalue_45_Nhan", function(){

 var data_45_Nhan = pm.response.json();

 pm.expect(data_45_Nhan[0].title).to.include("sunt");

});

```

### Bước 5: Nhấn Save -> Send -> Xem ở tab Test Results

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Help', and other options like 'Import', 'Runner'. The main area shows a collection named 'TestWeb\_45\_Nhan' containing four requests: 'get\_posts\_list\_45\_Nhan', 'post\_posts\_45\_Nhan', 'put\_posts\_45\_Nhan', and 'delete\_posts\_45\_Nhan'. The 'get\_posts\_list\_45\_Nhan' request is currently selected. On the right, there's a 'Test Results' section with a status of '4/6'. This section is highlighted with a red box. It lists six test cases: TC1\_getlist\_reponseTime\_45\_Nhan (FAIL), TC2\_getlist\_status\_get\_45\_Nhan (PASS), TC3\_getlist\_header\_45\_Nhan (PASS), TC4\_getlist\_array\_45\_Nhan (PASS), TC5\_getlist\_items\_45\_Nhan (FAIL), and TC6\_getlist\_checkvalue\_45\_Nhan (PASS). The 'Send' button, located at the top right of the request details, is also highlighted with a red box.

Hình 3.1.9: Kết quả testscript

### 1.3. Phương thức POST

Bước 1: Tạo 1 request dạng POST

Bước 2: Copy đường dẫn vào URL, mở tab Body -> raw-> dán code từ trang web vào -> Save -> Send

The screenshot shows the Postman application interface. In the left sidebar, there is a collection named 'TestWeb\_45\_Nhan' containing two requests: a GET request for 'get\_posts\_list\_45\_Nhan' and a POST request for 'post\_posts\_45\_Nhan'. The POST request is selected. The URL is set to 'https://jsonplaceholder.typicode.com/posts'. The 'Body' tab is selected, and the raw JSON content is:

```
{
 "title": "feel",
 "body": "best",
 "userId": 1,
}
```

The 'Send' button is highlighted with a red box. The response panel at the bottom shows the status: 201 Created, Time: 820 ms, Size: 1.2 KB, and the response body:

```
1 {
2 "id": 101
3 }
```

Hình 3.1.10: Request POST

Bước 3: Chọn tab Tests -> viết testscript cho phương thức POST

```
// Kiểm tra status của post là thành công STT_Tên: 45_Nhan

pm.test("TC1_post_status_45_Nhan", function() {

 pm.expect(pm.response.code)
 .to.be.oneOf([200, 201, 202]);

});

// Kiểm tra chuỗi trạng thái trả về có phải là "Created" hay không STT_Tên: 45_Nhan

pm.test("TC2_post_status_string_45_Nhan", function () {

 pm.response.to.have.status("Created");

});

// Kiểm tra header có thuộc tính Server không STT_Tên: 45_Nhan
```

```

pm.test("TC3_post_header_45_Nhan", function() {
 pm.response.to.have.header("Server")
});

// Kiểm tra response body chứa 1 chuỗi nào đó STT_Tên: 45_Nhan

pm.test("TC4_post_body_45_Nhan", function() {
 pm.expect(pm.response.text()).to.include("feel")
});

```

#### Bước 4: Nhấn Save -> Send -> Xem ở tab Test Results

The screenshot shows the Postman interface with a collection named "TestAPI\_bc\_45\_Nhan". A POST request titled "post\_posts\_45\_Nhan" is selected. The "Tests" tab is active, displaying the following test script:

```

10 // Kiểm tra header có thuộc tính Server không STT_Tên: 45_Nhan
11 pm.test("TC3_post_header_45_Nhan", function() {
12 pm.response.to.have.header("Server")
13 });
14 // Kiểm tra response body chứa 1 chuỗi nào đó STT_Tên: 45_Nhan

```

The "Test Results" section shows the execution status of the tests:

- TC1\_post\_status\_45\_Nhan: PASS
- TC2\_post\_status\_string\_45\_Nhan: PASS
- TC3\_post\_header\_45\_Nhan: PASS
- TC4\_post\_body\_45\_Nhan: FAIL | Assertion Error: expected '{\n"id":101\n}' to include 'feel'

Hình 3.1.11: Kết quả testscript

#### 1.4. Phương thức PUT

##### Bước 1: Tạo 1 resquest dạng PUT

Bước 2: Copy đường dẫn vào URL, mở tab Body -> raw-> dán code từ trang web vào -> Save -> Send

The screenshot shows the Postman application interface. In the top navigation bar, there are buttons for 'File', 'Edit', 'View', 'Help', and 'Import', followed by 'Runner' and a dropdown menu. The title bar says 'TestAPI\_bc\_45\_Nhan' and has an 'Invite' button. The left sidebar shows 'History', 'Collections' (which is selected), 'APIs', and '+ New Collection'. Under 'Collections', there is a folder named 'TestWeb\_45\_Nhan' containing 3 requests: 'GET get\_posts\_list\_45\_Nhan' and 'POST post\_posts\_45\_Nhan'. Below these is a 'PUT put\_posts\_45\_Nhan' request. The main workspace shows a 'PUT' request for 'put\_posts\_45\_Nhan' with a 'Body' tab selected. The body contains the following JSON payload:

```

1: {
2: id: 1,
3: title: 'foo',
4: body: 'bar',
5: userId: 1,
6: }

```

Below the body, the response is shown in a JSON format:

```

1: {
2: "id": 1
3: }

```

At the bottom right of the workspace, there are status indicators: 'Status: 200 OK', 'Time: 2.29 s', 'Size: 1.08 KB', and 'Save Response'.

Hình 3.1.12: Request PUT

Bước 3: Chọn tab Tests -> viết testscript cho phương thức PUT

```

// Kiểm tra thời phản hồi trên 50 mili giây STT_Tên: 45_Nhan
pm.test("TC1_put_responsetime_45_Nhan", function () {
 pm.expect(pm.response.responseTime).to.be.above(50);
});

//Kiểm tra status code put là OK STT_Tên: 45_Nhan
pm.test("TC2_put_statuscode_45_Nhan",function() {
 pm.response.to.have.status("OK");
});

// Kiểm tra dữ liệu trả về STT_Tên: 45_Nhan
pm.test("TC3_put_value_return_45_Nhan", function () {
 var v_45_Nhan = pm.response.json();
 pm.expect(v_45_Nhan.id).to.eql(1);
});

// Kiểm tra thuộc tính của header STT_Tên: 45_Nhan
pm.test("TC4_put_header", function() {
 pm.response.to.have.header("Nelq")
})

```

)

#### Bước 4: Nhấn Save -> Send -> Xem ở tab Test Results

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Help', and various icons. Below it, a toolbar has buttons for '+ New', 'Import', 'Runner', and a search/filter field. The main area shows a collection named 'TestWeb\_45\_Nhan' with four requests: 'get\_posts\_list\_45\_Nhan' (GET), 'post\_posts\_45\_Nhan' (POST), 'put\_posts\_45\_Nhan' (PUT), and 'delete\_posts\_45\_Nhan' (DEL). The 'put\_posts\_45\_Nhan' request is selected. The 'Tests' tab is active, displaying a JavaScript test script:

```
10 pm.test("TC3_put_value_return_45_Nhan", function () {
11 var v_45_Nhan = pm.response.json();
12 pm.expect(v_45_Nhan.id).to.eq(1);
13 });
14 // Kiểm tra thuộc tính của header STT là: 45 Nhan
```

Below the script, the 'Body' tab is selected, showing the response body: `{ "id": 1, "title": "Nhan", "body": "Bài viết về Nhan", "userId": 1 }`. The 'Headers' section shows a single entry: 'Content-Type: application/json'. The status bar at the bottom indicates a 'Status: 200 OK'.

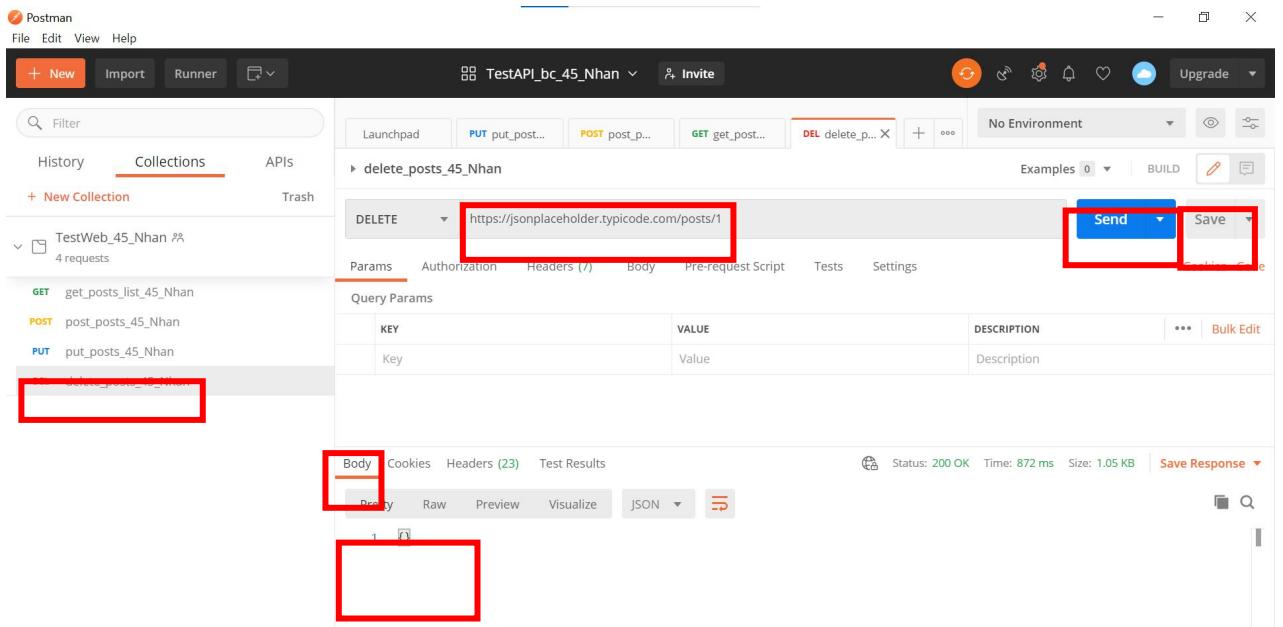
At the bottom right of the interface, there are 'Send' and 'Save' buttons, both of which are highlighted with red boxes. The 'Send' button is blue, and the 'Save' button is grey. Below these buttons, the 'Test Results' section is highlighted with a large red box. It shows three green 'PASS' status indicators for tests TC1, TC2, and TC3, followed by a 'FAIL' indicator for TC4, which is described as failing because the expected value '45' did not match the actual value '1'.

Hình 3.1.13: Kết quả testscript

#### 1.5. Phương thức DELETE

Bước 1: Tạo 1 request dạng DELETE

Bước 2: Copy đường dẫn vào URL -> Save -> Send



Hình 3.1.14: Request DELETE

Bước 3: Chọn tab Tests -> viết testscript cho phương thức DELETE

```
// Kiểm tra mảng rỗng STT_Tên: 45_Nhan
pm.test("TC1_delete_Empty_45_Nhan", function () {
 pm.expect({}).to.be.empty;
});

// Kiểm tra kết quả không chứa keys STT_Tên: 45_Nhan
pm.test("TC2_delete_results_45_Nhan", function () {
 var data_45_Nhan = pm.response.json();
 pm.expect(data_45_Nhan).to.not.any.keys("id");
});

// Kiểm tra kiểu dữ liệu STT_Tên: 45_Nhan
pm.test("TC3_delete_type_45_Nhan", function () {
 var data_45_Nhan = pm.response.json();
 pm.expect(data_45_Nhan).to.be.an('undefined');
});
```

Bước 4: Nhấn Save -> Send -> Xem ở tab Test Results

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Help', and various icons. Below the navigation is a toolbar with buttons for '+ New', 'Import', 'Runner', and 'Invite'. The main area shows a collection named 'TestWeb\_45\_Nhan' containing four requests: 'get\_posts\_list\_45\_Nhan', 'post\_posts\_45\_Nhan', 'put\_posts\_45\_Nhan', and 'delete\_posts\_45\_Nhan'. The 'delete\_posts\_45\_Nhan' request is selected. The 'Tests' tab is active, displaying a JavaScript test script:

```

1 // Kiểm tra mảng rỗng STT_Tên: 45_Nhan
2 pm.test("TC1_delete_Empty_45_Nhan", function () {
3 pm.expect({}).to.be.empty();
4 });
5 // Kiểm tra kết quả không chứa keys STT_Tên: 45_Nhan
6 pm.test("TC2_delete_results_45_Nhan", function () {

```

Below the test script, the 'Test Results' section shows three entries:

- PASS TC1\_delete\_Empty\_45\_Nhan
- PASS TC2\_delete\_results\_45\_Nhan
- FAIL TC3\_delete\_type\_45\_Nhan | Assertion Error: expected {} to be an undefined

At the bottom right of the results table, it says 'Status: 200 OK Time: 376 ms Size: 1.07 KB Save Response'.

Hình 3.1.15: Kết quả testscript

## 2. Tạo API với Nodejs

### 2.1. Lấy link api của riêng mình bằng nodejs

Bước 1: Truy cập vào link <https://nodejs.org/en/download>, tải file cài đặt, nhấn next -> đến cuối -> finish

Bước 2: Mở cmd, vào thư mục đã cài đặt để kiểm tra version

```

C:\Users\HP>node -v
v20.13.1

C:\Users\HP>npm -v
10.5.2

```

### *Hình 3.2.1: Kiểm tra version*

Bước 3: Cài đặt gói (npm init) NODE package manager(NPM): Tạo folder D:\installnodejs\_45\_Nhan\jsonserver\_45\_Nhan

```
cmd: npm init
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

D:\installnodejs_45_Nhan\jsonserver_45_Nhan>npm -v
10.5.2

D:\installnodejs_45_Nhan\jsonserver_45_Nhan>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (jsonserver_45_nhan) ■
```

### *Hình 3.2.2: Cài đặt package*

Bước 4: Enter đến cuối nhập “yes” -> enter để tạo project

```
npm init
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (jsonserver_45_nhan)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\installnodejs_45_Nhan\jsonserver_45_Nhan\package.json:

{
 "name": "jsonserver_45_nhan",
 "version": "1.0.0",
 "main": "index.js",
 "scripts": {
 "test": "echo \\"$Error: no test specified\\" && exit 1"
 },
 "author": "",
 "license": "ISC",
 "description": ""
}

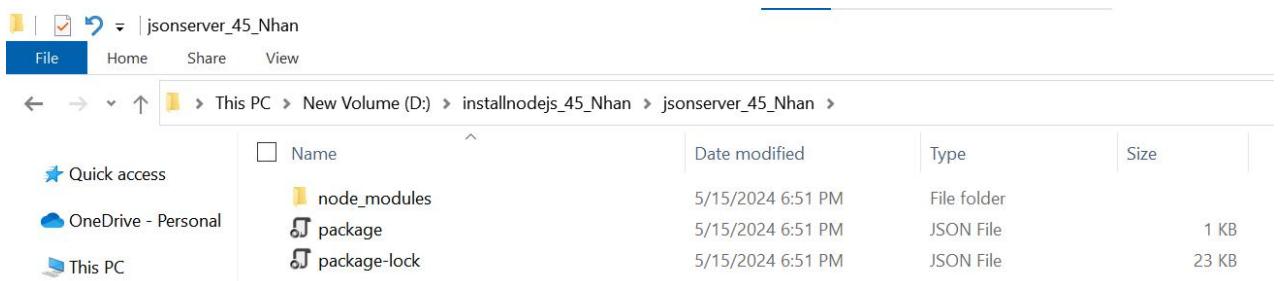
Is this OK? (yes) yes
```

Hình 3.2.3: Tạo project

Bước 5: Sau đó gõ npm install i json-server

```
C:\Windows\System32\cmd.exe
D:\installnodejs_45_Nhan\jsonserver_45_Nhan>npm install i json-server
added 55 packages, and audited 56 packages in 14s
15 packages are looking for funding
 run `npm fund` for details
found 0 vulnerabilities
D:\installnodejs_45_Nhan\jsonserver_45_Nhan>
```

*Hình 3.2.4: Tạo thành công*



*Hình 3.2.5: Thư mục*

Bước 6: Tạo file ualbums\_45\_Nhan.json thành API trên server ảo của mình

- Code: {

```
"ualbums_45_Nhan": [
 {
 "userId": 1,
 "id": 1,
 "title": "Friends"
 },
 {
 "userId": 1,
 "id": 2,
```

```
 "title": "Fruits"
 },
 {
 "userId": 1,
 "id": 3,
 "title": "Foods"
 },
 {
 "userId": 1,
 "id": 4,
 "title": "Animals"
 },
 {
 "userId": 1,
 "id": 5,
 "title": "Family"
 },
 {
 "userId": 1,
 "id": 6,
 "title": "Natures"
 },
 {
 "userId": 1,
 "id": 7,
 "title": "Flowers"
 },
 {
 "userId": 1,
```

```

 "id": 8,
 "title": "Houses"
 },
 {
 "userId": 1,
 "id": 9,
 "title": "Wedding Dress"
 },
 {
 "userId": 1,
 "id": 10,
 "title": "Studios"
 }
]
}

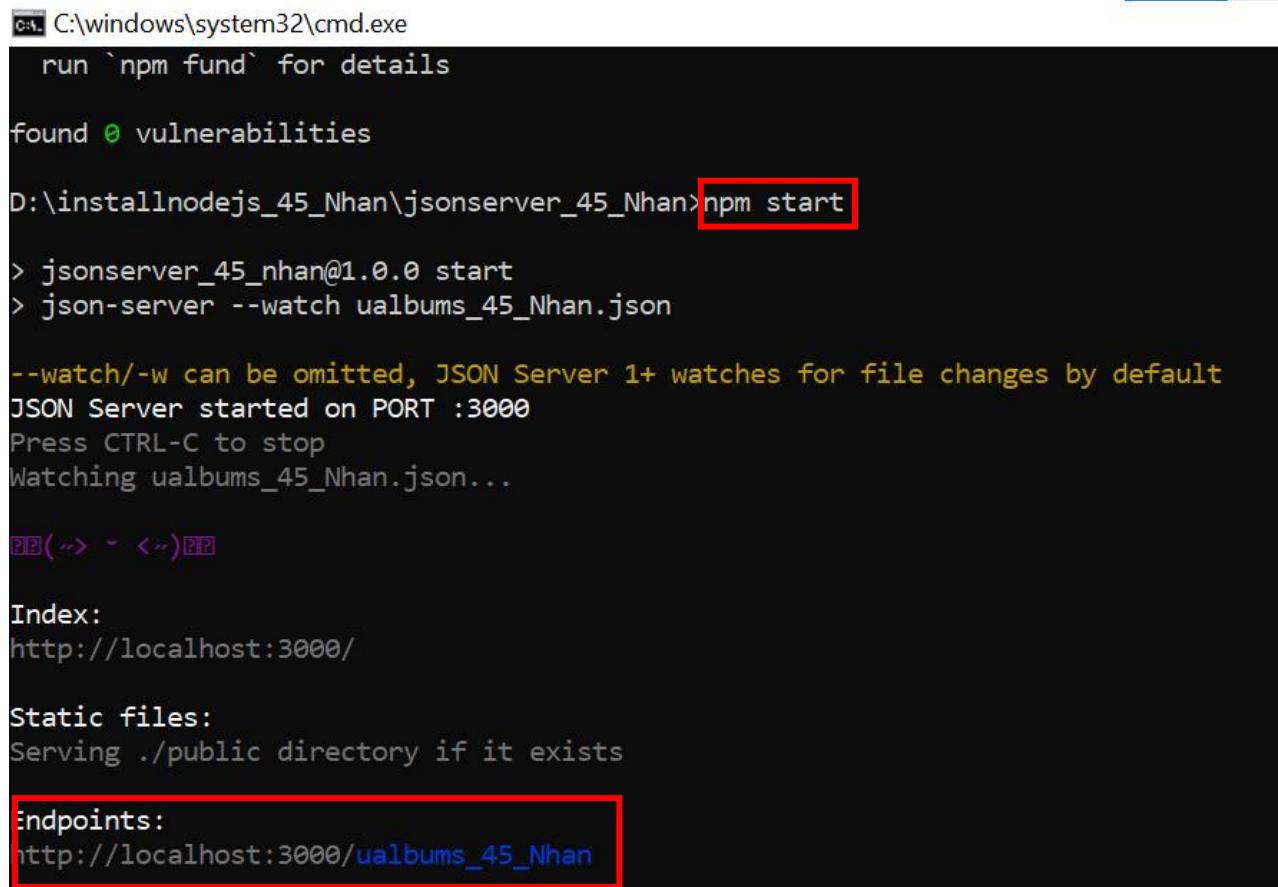
```

Bước 7: Mở file package.json thêm dòng start vào

```
{
 "name": "jsonserver_45_nhan",
 "version": "1.0.0",
 "main": "index.js",
 "scripts": {
 "start": "json-server --watch ualbums_45_Nhan.json",
 "test": "echo \"Error: no test specified\" && exit 1"
 },
 "author": "",
 "license": "ISC",
 "description": "",
 "dependencies": {
 "i": "^0.3.7",
 }
}
```

```
"json-server": "^1.0.0-beta.0"
}
}
```

#### Bước 8: Mở cmd, chạy lệnh



```
C:\Windows\system32\cmd.exe
run `npm fund` for details
Found 0 vulnerabilities
D:\installnodejs_45_Nhan\jsonserver_45_Nhan>npm start
> jsonserver_45_nhan@1.0.0 start
> json-server --watch ualbums_45_Nhan.json
--watch/-w can be omitted, JSON Server 1+ watches for file changes by default
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching ualbums_45_Nhan.json...
Index:
http://localhost:3000/
Static files:
Serving ./public directory if it exists
Endpoints:
http://localhost:3000/ualbums_45_Nhan
```

Hình 3.2.6: Server ảo

#### Bước 9: Thu được API, chạy web kiểm tra

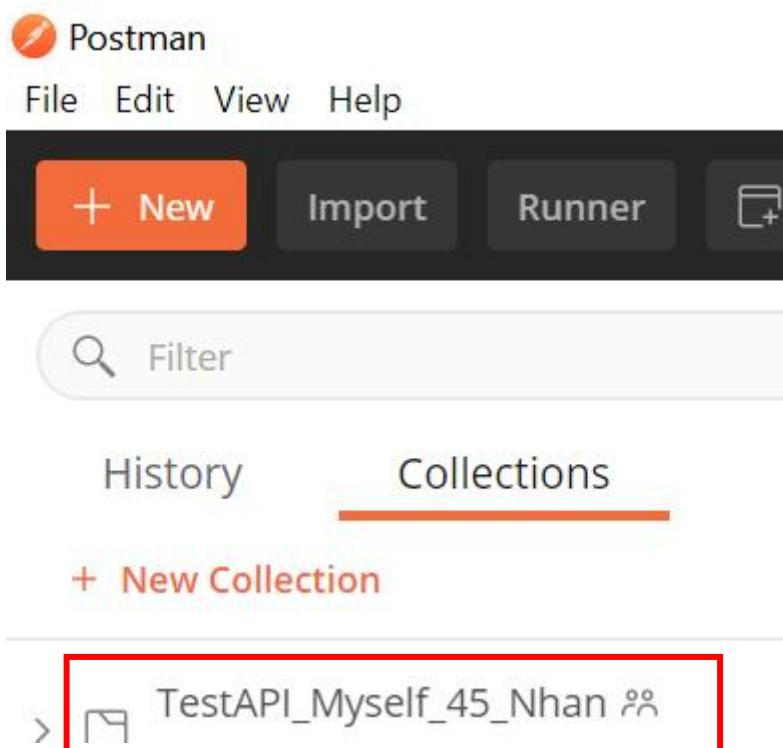
[http://localhost:3000/ualbums\\_45\\_Nhan](http://localhost:3000/ualbums_45_Nhan)

```
pretty-print ✓
[
 {
 "userId": 1,
 "id": "1",
 "title": "Family"
 },
 {
 "userId": 1,
 "id": "2",
 "title": "Colors"
 },
 {
 "userId": 1,
 "id": "3",
 "title": "Dresses"
 },
 {
 "userId": 1,
 "id": "4",
 "title": "Foods"
 },
 {
 "userId": 1,
 "id": "5",
 "title": "Fruits"
 },
 {
 "userId": 1,
 "id": "6",
 "title": "Friends"
 },
 {
 "userId": 1,
 "id": "7",
 "title": "Natures"
 },
 {
 "userId": 1,
 "id": "8",
 "title": "Sports"
 },
 {
 "userId": 1,
 "id": "9",
 "title": "Tech"
 },
 {
 "userId": 1,
 "id": "10",
 "title": "Travel"
 }
]
```

Hình 3.2.7: Kết quả Chrome chạy

## 2.2. Test trên api vừa tạo

- Vừa test postman vừa giữ cmd để web hoạt động
- Trên workspace TestAPI\_bc\_45\_Nhan, tạo 1 collection



Hình 3.2.8: Tạo collection thành công

- GET

Bước 1: Add Request GET, copy đường dẫn vào thanh URL -> Save -> Send, xem kết quả ở phần “Body”

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Help', and a 'New' button. Below it is a toolbar with 'Import', 'Runner', and other icons. The main area shows a collection named 'TestAPI\_bc\_45\_Nhan' with a single item: 'get\_albums\_list\_45\_Nhan'. This item is a GET request to 'http://localhost:3000/ualbums\_45\_Nhan'. The 'Body' tab is selected, displaying a JSON response with two albums:

```

1 [
2 {
3 "userId": 1,
4 "id": "1",
5 "title": "Family"
6 },
7 {
8 "userId": 1,
9 "id": "2",
10 "title": "Hobbies"
11 }
12]

```

Below the body, the status is shown as 'Status: 200 OK' with a green icon, and the response time is 'Time: 32 ms'. There are buttons for 'Send' and 'Save' at the top right of the request card.

Hình 3.2.9: Request GET

Bước 2: Chọn tab Tests -> viết testscript cho phương thức GET

```

// Kiểm tra thời gian phản hồi dưới 300 mili giây STT_Tên: 45_Nhan
pm.test("TC1_albums_reponseTime_45_Nhan",function() {
 pm.expect(pm.response.responseTime).to.be.below(20);
});

// Kiểm tra status code của get request là 200 STT_Tên: 45_Nhan
pm.test("TC2_albums_status_get_45_Nhan",function() {
 pm.response.to.have.status("OK");
});

// Kiểm tra header có thuộc tính Content-type và giá trị là "application/json; charset=utf-8" hay không STT_Tên: 45_Nhan
pm.test("TC3_albums_header_45_Nhan", function() {

```

```

pm.response.to.have.header("Content-type", "application/json")

});

// Kiểm tra kiểu dữ liệu STT_Tên: 45_Nhan

pm.test("TC4_galbums_array_45_Nhan", function() {

 var data_45_Nhan = pm.response.json();

 pm.expect(data_45_Nhan).to.be.an("array");

});

// Kiểm tra phần từ 0 của mảng có chứa chuỗi sunt hay không STT_Tên: 45_Nhan

pm.test("TC5_albums_checkvalue_45_Nhan", function(){

 var data_45_Nhan = pm.response.json();

 pm.expect(data_45_Nhan[0].title).to.include("Family");

});

// Kiểm tra chiều dài của mảng STT_Tên: 45_Nhan

pm.test("TC6_albums_items_45_Nhan", function(){

 var data_45_Nhan = pm.response.json();

 pm.expect(data_45_Nhan).to.have.lengthOf.above(10);

});

```

Bước 3: Nhấn Save -> Send -> Xem ở tab Test Results

The screenshot shows the Postman interface with a collection named 'TestAPI\_Myself\_45\_Nhan'. A specific GET request for '/albums\_45\_Nhan' is selected. The 'Test Results' tab is active, showing six test cases all marked as 'PASS'. The 'Send' and 'Save' buttons are highlighted with a red box.

Hình 3.2.10: Kết quả testscript

## ● POST

Bước 1: Add Request POST, copy đường dẫn vào URL, mở tab Body -> raw-> dán code từ trang web vào -> Save -> Send

The screenshot shows the Postman interface with a collection named 'TestAPI\_Myself\_45\_Nhan'. A POST request for '/post\_albums\_45\_Nhan' is selected. The 'Body' tab is active and set to 'raw', containing the following JSON code:

```

1 {
2 "userId": 1,
3 "title": "T shirts"
4 }

```

The 'Send' and 'Save' buttons are highlighted with a red box. Below the request, the response body is displayed in a red box:

```

1 {
2 "id": "e46c",
3 "userId": 1,
4 "title": "T shirts"
5 }

```

Hình 3.2.11: Request POST

Bước 2: Chọn tab Tests -> viết testscript cho phương thức POST

```

// Kiểm tra status của post là thành công STT_Tên: 45_Nhan
pm.test("TC1_album_status_45_Nhan", function() {
 pm.expect(pm.response.code)
 .to.be.oneOf([200, 201, 202]);
});

// Kiểm tra chuỗi trạng thái trả về có phải là "Created" hay không STT_Tên: 45_Nhan
pm.test("TC2_albums_status_string_45_Nhan", function () {
 pm.response.to.have.status("Created");
});

// Kiểm tra header có thuộc tính Server không STT_Tên: 45_Nhan
pm.test("TC3_post_header_45_Nhan", function() {
 pm.response.to.have.header("Content-Length")
});

// Kiểm tra response body chứa 1 chuỗi nào đó STT_Tên: 45_Nhan
pm.test("TC4_post_body_45_Nhan", function() {
 pm.expect(pm.response.text()).to.include("ets")//id
});

```

Bước 3: Nhấn Save -> Send -> Xem ở tab Test Results

Postman interface showing a successful test run. The 'Tests' tab is active, displaying the results of four test cases. The 'Send' button is highlighted with a red box.

```

1 // Kiểm tra status của post là thành công STT_tên: 45_Nhan
2 pm.test("TC1_album_status_45_Nhan",function() {
3 pm.expect(pm.response.code)
4 .to.be.oneOf([200, 201, 202]);
5 });

```

Test results:

- PASS TC1\_album\_status\_45\_Nhan
- PASS TC2\_albums\_status\_string\_45\_Nhan
- PASS TC3\_post\_header\_45\_Nhan
- FAIL TC4\_post\_body\_45\_Nhan | AssertionException: expected '{\n "id": "a769",\n "userId": 1,\n "title": "T Shirts"\n}' to include 'ets'

Hình 3.2.12: Kết quả testscript

## ● PUT

Bước 1: Add Request PUT, copy đường dẫn vào URL, mở tab Body -> raw-> dán code từ trang web vào -> Save -> Send

Postman interface showing a successful PUT request. The 'Body' tab is active, displaying the raw JSON payload. The 'Send' button is highlighted with a red box.

```

{
 "title": "Candys",
 "userId": 1
}

```

Test results:

- PASS put\_album\_45\_Nhan | Status: 200 OK

### *Hình 3.2.13: Request PUT*

Bước 2: Chọn tab Tests -> viết testscript cho phương thức PUT

```
// Kiểm tra thời phản hồi trên 50 mili giây STT_Tên: 45_Nhan
pm.test("TC1_albums_responsetime_45_Nhan", function () {
 pm.expect(pm.response.responseTime).to.be.above(5);
});

//Kiểm tra status code put là OK STT_Tên: 45_Nhan
pm.test("TC2_albums_statuscode_45_Nhan", function() {
 pm.response.to.have.status("OK");
});

// Kiểm tra dữ liệu trả về STT_Tên: 45_Nhan
pm.test("TC3_albums_value_return_45_Nhan", function () {
 var v_45_Nhan = pm.response.json();
 pm.expect(v_45_Nhan.title).to.eql("Cats");
});

// Kiểm tra thuộc tính của header STT_Tên: 45_Nhan
pm.test("TC4_albums_header", function() {
 pm.response.to.have.header("Date")
})
```

Bước 3: Nhấn Save -> Send -> Xem ở tab Test Results

The screenshot shows the Postman interface with a collection named "TestAPI\_Myself\_45\_Nhan". A PUT request titled "put\_album\_45\_Nhan" is selected. The "Tests" tab is active, containing a JavaScript test script:

```

9 // Kiểm tra dữ liệu trả về STT_Tên: 45_Nhan
10 pm.test("TC3_albums_value_return_45_Nhan", function () {
11 var v_45_Nhan = pm.response.json();
12 pm.expect(v_45_Nhan.title).to.eql("Cats");

```

The "Test Results" section shows four tests: TC1\_albums\_responsetime\_45\_Nhan (PASS), TC2\_albums\_statuscode\_45\_Nhan (PASS), TC3\_albums\_value\_return\_45\_Nhan (FAIL), and TC4\_albums\_header (PASS).

Hình 3.2.14: Kết quả testscript

## ● DELETE

Bước 1: Add Request POST, copy đường dẫn vào URL -> Save -> Send

The screenshot shows the Postman interface with a collection named "TestAPI\_Myself\_45\_Nhan". A DELETE request titled "delete\_albums\_45\_Nhan" is selected. The "Send" button is highlighted. The "Body" tab is active, showing a JSON response:

```

1 {
2 "id": "a769",
3 "userId": 1,
4 "title": "T shirts"
5

```

Hình 3.2.15: Request DELETE

## Bước 2: Chọn tab Tests -> viết testscript cho phương thức DELETE

```
// Kiểm tra mảng rỗng STT_Tên: 45_Nhan

pm.test("TC1_albums_Empty_45_Nhan", function () {
 pm.expect({}).to.be.empty;
});

// Kiểm tra kiểu dữ liệu STT_Tên: 45_Nhan

pm.test("TC2_albums_type_45_Nhan", function () {
 var data_45_Nhan = pm.response.json();
 pm.expect(data_45_Nhan).to.be.an('object');
});

// Kiểm tra status code STT_Tên: 45_Nhan

pm.test("TC3_albums_statuscode_45_Nhan", function () {
 var data_45_Nhan = pm.response.json();
 pm.response.to.have.status("Not Found");
});
```

## Bước 3: Nhấn Save -> Send -> Xem ở tab Test Results

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Help', and other options like 'Import', 'Runner', and 'Invite'. The main area shows a collection named 'TestAPI\_bc\_45\_Nhan' with several requests listed under it. On the right side, there's a 'Tests' tab which is currently active. A red box highlights the 'Send' button and the 'Tests' tab. Another red box highlights the 'Test Results' tab at the bottom of the interface. The results section shows three test cases: 'TC1\_albums\_Empty\_45\_Nhan' (Status: PASS), 'TC2\_albums\_type\_45\_Nhan' (Status: FAIL), and 'TC3\_albums\_statuscode\_45\_Nhan' (Status: FAIL). The entire screenshot is framed by a large red border.

Hình 3.2.16: Kết quả testscript