**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

A blue book with a ring around it

Description automatically generated

**BÁO CÁO MÔN HỌC**

**ĐỀ TÀI UNIT TEST: QUẢN LÝ KHO HÀNG**

**ĐỀ TÀI WEB DRIVER: MAGENTO WEB SITE**

**ĐỀ TÀI AIPs POSTMAN: QUẢN LÝ APIs DỮ LIỆU NGƯỜI DÙNG**

**GVHD: TH.S NGUYỄN THỊ NGỌC THANH**

**SVTH: KIỀU MINH TUẤN**

**MSSV: 2251012149**

**KHOA: CÔNG NGHỆ THÔNG TIN**

**LỚP: DH22CS02**

**TP. HỒ CHÍ MINH, NĂM 2025**

**MỤC LỤC**

[A. UNIT TEST 5](#_Toc194731977)

[I - TỔNG QUAN 5](#_Toc194731978)

[1. Tổng quan về sản phẩm quản lý kho hàng 5](#_Toc194731979)

[2. Cài đặt NUNIT 5](#_Toc194731980)

[2.1. Tạo Project Unit Test có hỗ trợ .NET Framework 5](#_Toc194731981)

[2.2. Import Project quản lý kho hàng vào Unit Test Project 5](#_Toc194731982)

[2.3. Import thư viện System.Data vào Unit Test Project để tiến hành đọc Data File 5](#_Toc194731983)

[2.4. Using Project quản lý kho hàng vào code Unit Test 6](#_Toc194731984)

[2.5. Khởi tạo các DataFile có dạng csv và đặt tên bản quyền là 45\_Tuan 6](#_Toc194731985)

[2.6. Thiết lập Output Directory cho từng Data File là Copy always 6](#_Toc194731986)

[II - CÁC CHỨC NĂNG CHÍNH 7](#_Toc194731987)

[1. Chức năng đăng nhập 7](#_Toc194731988)

[1.1. Đặc tả 7](#_Toc194731989)

[1.2. Giao diện đăng nhập 7](#_Toc194731990)

[1.3. Test case đăng nhập thành công 7](#_Toc194731991)

[1.4. Test case đăng nhập username chưa được đăng ký 8](#_Toc194731992)

[1.5. Test case đăng nhập sai password 9](#_Toc194731993)

[2. Chức năng đăng ký 10](#_Toc194731994)

[2.1. Đặc tả 10](#_Toc194731995)

[2.2. Giao diện đăng ký 10](#_Toc194731996)

[2.3. Test case đăng ký thành công 10](#_Toc194731997)

[2.4. Test case đăng ký username đã tồn tại 11](#_Toc194731998)

[2.5. Test case đăng ký username và password không hợp lệ 12](#_Toc194731999)

[3. Chức năng thêm sản phẩm 13](#_Toc194732000)

[3.1. Đặc tả 13](#_Toc194732001)

[3.2. Giao diện thêm sản phẩm 13](#_Toc194732002)

[3.3. Test case thêm sản phẩm thành công 13](#_Toc194732003)

[3.4. Test case thêm sản phẩm mã không hợp lệ 15](#_Toc194732004)

[3.5. Test case thêm sản phẩm giá bán và số lượng không hợp lệ 16](#_Toc194732005)

[4. Chức năng xóa sản phẩm 17](#_Toc194732006)

[4.1. Đặc tả 17](#_Toc194732007)

[4.2. Giao diện xóa sản phẩm 17](#_Toc194732008)

[4.3. Test case xóa sản phẩm thành công 17](#_Toc194732009)

[4.4. Test case xóa sản phẩm không thành công 18](#_Toc194732010)

[B. WEB DRIVER 19](#_Toc194732011)

[I - TỔNG QUAN 19](#_Toc194732012)

[1. Tổng quan về website Magento 19](#_Toc194732013)

[2. Cài đặt Selenium 19](#_Toc194732014)

[2.2. Chọn Tool -> Nuget Package Manager -> Manage Nuget Packages for Solution… 20](#_Toc194732015)

[2.3. Cài đặt Selenium.WebDriver sau đó thêm Project WebDriver\_45\_Tuan vào 21](#_Toc194732016)

[2.4. Cài đặt Selenium.WebDriver.ChromeDriver sau đó thêm Project WebDriver\_45\_Tuan vào 21](#_Toc194732017)

[2.5. Thêm thư viện cần thiết vào Project 21](#_Toc194732018)

[II - CÁC CHỨC NĂNG CHÍNH 22](#_Toc194732019)

[1. Chức năng đăng nhập 22](#_Toc194732020)

[1.1. Đặc tả 22](#_Toc194732021)

[1.2. Giao diên đăng nhập 22](#_Toc194732022)

[1.3. Giao diện thông tin tài khoản 22](#_Toc194732023)

[1.4. Test case đăng nhập thành công 23](#_Toc194732024)

[2. Chức năng thêm sản phẩm 23](#_Toc194732025)

[2.1. Đặc tả 23](#_Toc194732026)

[2.2. Giao diện thêm sản phẩm vào giỏ hàng 23](#_Toc194732027)

[2.3. Giao diện giỏ hàng 24](#_Toc194732028)

[2.4. Test case thêm sản phẩm vào giỏ hàng thành công 24](#_Toc194732029)

[3. Chức năng chỉnh sửa giỏ hàng 25](#_Toc194732030)

[3.1. Đặc tả 25](#_Toc194732031)

[3.2. Giao diện chỉnh sửa giỏ hàng 25](#_Toc194732032)

[3.3. Test case chỉnh sửa giỏ hàng thành công 25](#_Toc194732033)

[4. Chức năng thanh toán 26](#_Toc194732034)

[4.1. Đặc tả 26](#_Toc194732035)

[4.2. Giao diện thanh toán 26](#_Toc194732036)

[4.3. Test case thanh toán thành công 26](#_Toc194732037)

[C. APIs POSTMAN 27](#_Toc194732038)

[I - TỔNG QUAN 27](#_Toc194732039)

[1. Tổng quan về quản lý APIs dữ liệu người dùng 27](#_Toc194732040)

[2. Cài đặt Posman, Json 27](#_Toc194732041)

[2.1. Khởi tạo File Json 27](#_Toc194732042)

[2.2. Cài đặt Postman 33](#_Toc194732043)

[II - CÁC PHƯƠNG THỨC CHÍNH 34](#_Toc194732044)

[1. PHƯƠNG THỨC GET 34](#_Toc194732045)

[1.1. Đặc tả 34](#_Toc194732046)

[1.2. Giao diện 34](#_Toc194732047)

[1.3. Test Scrip 35](#_Toc194732048)

[2. PHƯƠNG THỨC POST 39](#_Toc194732049)

[2.1. Đặc tả 39](#_Toc194732050)

[2.2. Giao diện 39](#_Toc194732051)

[2.3. Test Scrip 39](#_Toc194732052)

[3. PHƯƠNG THỨC PUT/PATCH 42](#_Toc194732053)

[3.1. Đặc tả 42](#_Toc194732054)

[3.2. Giao diện 42](#_Toc194732055)

[3.3. Test Scrip 43](#_Toc194732056)

[4. PHƯƠNG THỨC DELETE 46](#_Toc194732057)

[4.1. Đặc tả 46](#_Toc194732058)

[4.2. Giao diện 46](#_Toc194732059)

[4.3. Test Scrip 46](#_Toc194732060)

# A. UNIT TEST

# I - TỔNG QUAN

## 1. Tổng quan về sản phẩm quản lý kho hàng

Trong thời đại kinh doanh phát triển nhanh chóng, việc quản lý kho hàng không chỉ đơn thuần là theo dõi số lượng sản phẩm mà còn ảnh hưởng trực tiếp đến hiệu quả hoạt động của doanh nghiệp. Vì thế, sản phẩm quản lý kho hàng là một công cụ giúp dễ dàng theo dõi, quản lý sản phẩm, …v.v. Bên cạnh đó, với giao diện thân thiện, dễ sử dụng, hệ thống này hỗ trợ thực hiện các thao tác như đăng nhập/đăng ký, thêm và xóa sản phẩm.

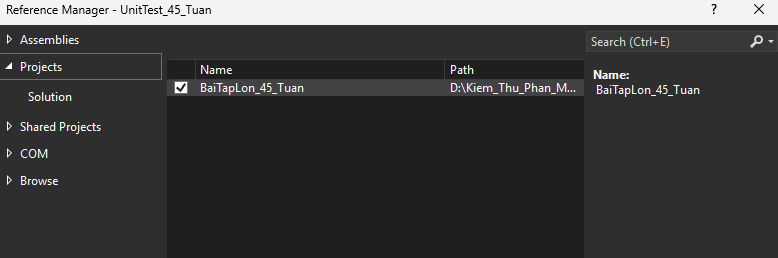
## 2. Cài đặt NUNIT

### 2.1. Tạo Project Unit Test có hỗ trợ .NET Framework

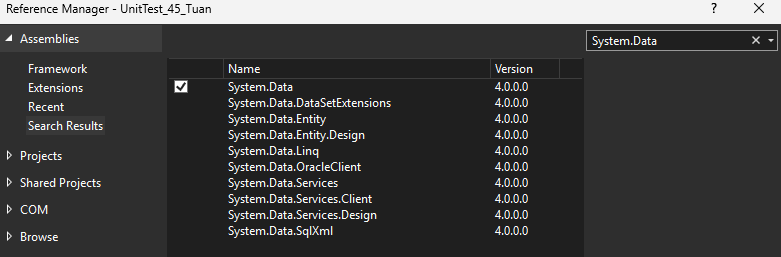
A screenshot of a computer

AI-generated content may be incorrect.

### 2.2. Import Project quản lý kho hàng vào Unit Test Project



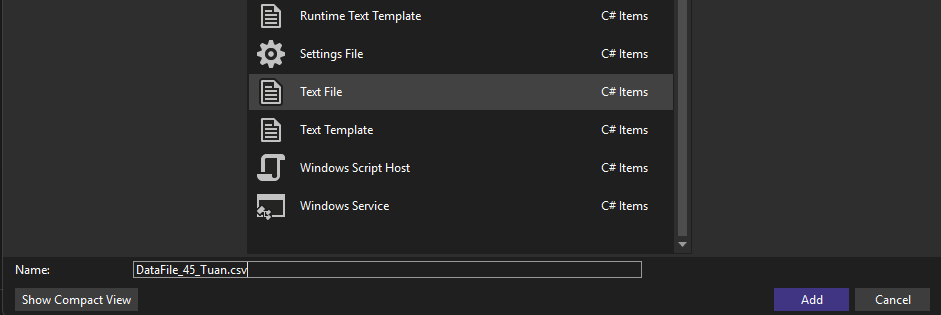
### 2.3. Import thư viện System.Data vào Unit Test Project để tiến hành đọc Data File

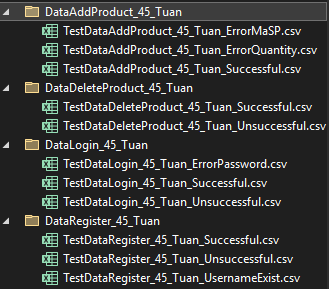
****

### 2.4. Using Project quản lý kho hàng vào code Unit Test

|  |
| --- |
| using System;  using BaiTapLon\_45\_Tuan;  using Microsoft.VisualStudio.TestTools.UnitTesting; |

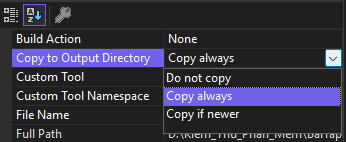
### 2.5. Khởi tạo các DataFile có dạng csv và đặt tên bản quyền là 45\_Tuan





*Các Data File*

### 2.6. Thiết lập Output Directory cho từng Data File là Copy always



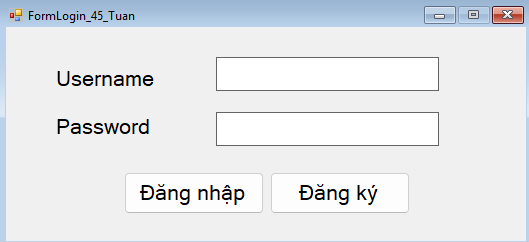
# II - CÁC CHỨC NĂNG CHÍNH

## 1. Chức năng đăng nhập

### 1.1. Đặc tả

Người dùng nhập đúng username và password có độ dài tối thiểu là 6 ký tự gồm chữ hoa, chữ thường, số và ký tự đặc biệt đã được đăng ký trước đó để truy cập vào được hệ thống.

### 1.2. Giao diện đăng nhập

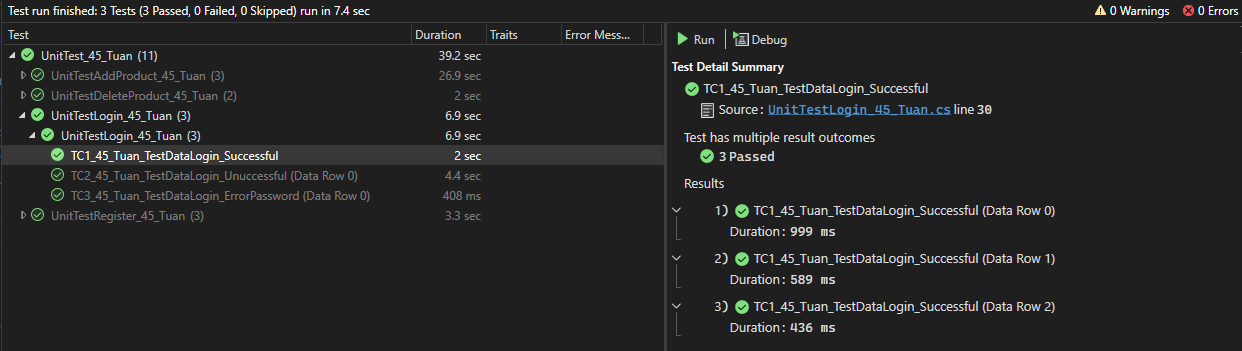
**

### 1.3. Test case đăng nhập thành công

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataLogin\_45\_Tuan\TestDataLogin\_45\_Tuan\_Successful.csv", "TestDataLogin\_45\_Tuan\_Successful#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC1: Test data đăng nhập thành công; 2 cột và 3 hàng; 3 PASS  public void TC1\_45\_Tuan\_TestDataLogin\_Successful()  {  string username\_45\_Tuan = TestContext.DataRow[0].ToString(); //Lấy username\_45\_Tuan  string password\_45\_Tuan = TestContext.DataRow[1].ToString(); //Lấy password\_45\_Tuan  acc\_45\_Tuan = new Account\_45\_Tuan(username\_45\_Tuan, password\_45\_Tuan); //Tạo đối tượng  bool login\_45\_Tuan = acc\_45\_Tuan.checkLogin\_45\_Tuan();  Assert.IsTrue(login\_45\_Tuan); //Nếu trả ra True thì đăng nhập thành công; PASS  } |

|  |  |
| --- | --- |
| username\_45\_Tuan | password\_45\_Tuan |
| Kieu@123 | Testpass@123 |
| Minh@123 | Testpass@123 |
| Tuan@123 | Testpass@123 |

*TestDataLogin\_45\_Tuan\_Successful.csv*



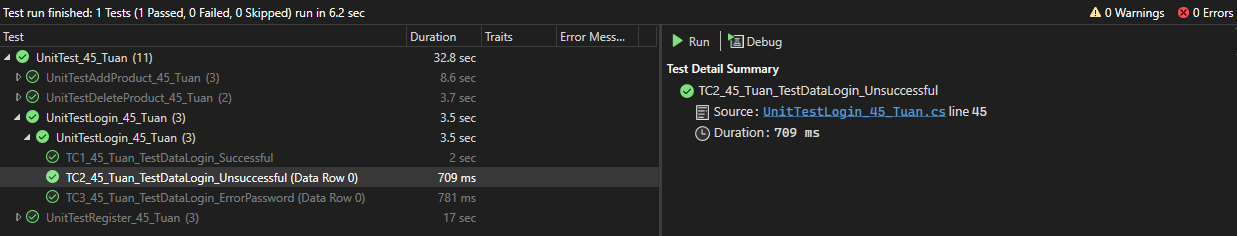
*Run test case hiển thị kết quả*

### 1.4. Test case đăng nhập username chưa được đăng ký

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataLogin\_45\_Tuan\TestDataLogin\_45\_Tuan\_Unsuccessful.csv", "TestDataLogin\_45\_Tuan\_Unsuccessful#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC2: Test data đăng nhập không thành công; 2 cột và 1 hàng; 1 PASS  public void TC2\_45\_Tuan\_TestDataLogin\_Unsuccessful()  {  string username\_45\_Tuan = TestContext.DataRow[0].ToString(); //Lấy username\_45\_Tuan  string password\_45\_Tuan = TestContext.DataRow[1].ToString(); //Lấy password\_45\_Tuan  acc\_45\_Tuan = new Account\_45\_Tuan(username\_45\_Tuan, password\_45\_Tuan); //Tạo đối tượng  bool login\_45\_Tuan = acc\_45\_Tuan.checkLogin\_45\_Tuan();  Assert.IsFalse(login\_45\_Tuan); //Nếu trả ra False thì đăng nhập thất bại; PASS  } |

|  |  |
| --- | --- |
| username\_45\_Tuan | password\_45\_Tuan |
| Kieu@2025 | Testpass@123 |

*TestDataLogin\_45\_Tuan\_Unsuccessful.csv*



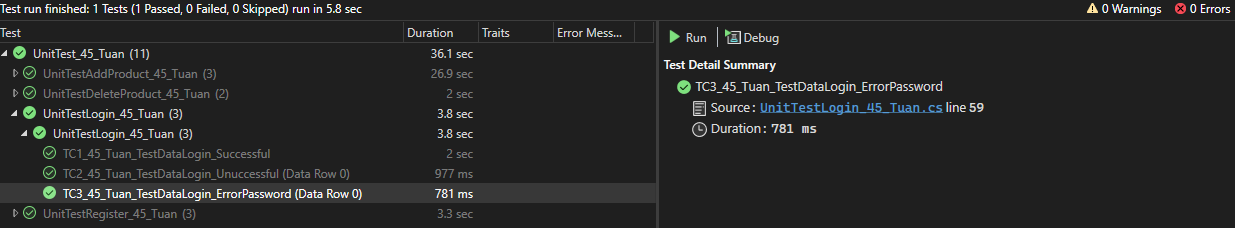
*Run test case hiển thị kết quả*

### 1.5. Test case đăng nhập sai password

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataLogin\_45\_Tuan\TestDataLogin\_45\_Tuan\_ErrorPassword.csv", "TestDataLogin\_45\_Tuan\_ErrorPassword#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC3: Test data đăng nhập sai mật khẩu; 2 cột và 1 hàng; 1 PASS  public void TC3\_45\_Tuan\_TestDataLogin\_ErrorPassword()  {  string username\_45\_Tuan = TestContext.DataRow[0].ToString(); //Lấy username\_45\_Tuan  string password\_45\_Tuan = TestContext.DataRow[1].ToString(); //Lấy password\_45\_Tuan  acc\_45\_Tuan = new Account\_45\_Tuan(username\_45\_Tuan, password\_45\_Tuan); //Tạo đối tượng  bool login\_45\_Tuan = acc\_45\_Tuan.checkLogin\_45\_Tuan();  Assert.IsFalse(login\_45\_Tuan); //Nếu trả ra False thì đăng nhập sai mật khẩu; PASS  } |

|  |  |
| --- | --- |
| username\_45\_Tuan | password\_45\_Tuan |
| Tuan@123 | 123456789 |

*TestDataLogin\_45\_Tuan\_ErrorPassword.csv*



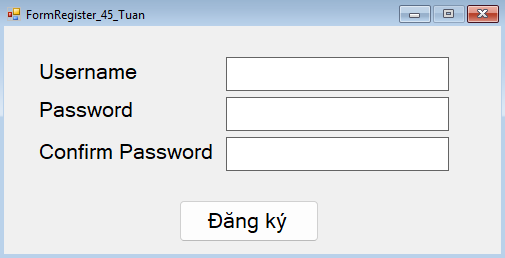
*Run test case hiển thị kết quả*

## 2. Chức năng đăng ký

### 2.1. Đặc tả

Người dùng nhập username, password và confỉm password có độ dài tối thiểu là 6 ký tự gồm chữ hoa, chữ thường, số và ký tự đặc biệt để được xác nhận tài khoản đăng nhập.

### 2.2. Giao diện đăng ký

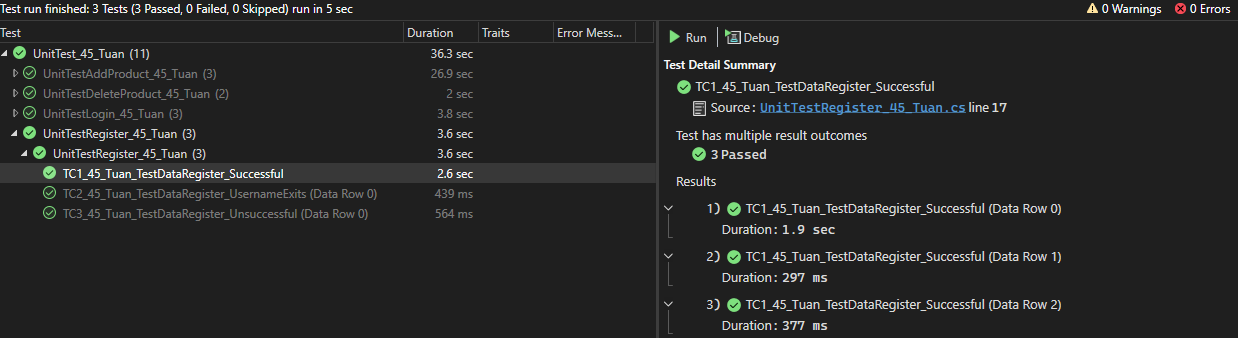


### 2.3. Test case đăng ký thành công

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataRegister\_45\_Tuan\TestDataRegister\_45\_Tuan\_Successful.csv", "TestDataRegister\_45\_Tuan\_Successful#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC1: Test data đăng ký thành công; 2 cột và 3 hàng; 3 PASS  public void TC1\_45\_Tuan\_TestDataRegister\_Successful()  {  string username\_45\_Tuan = TestContext.DataRow[0].ToString(); //Lấy username\_45\_Tuan  string password\_45\_Tuan = TestContext.DataRow[1].ToString(); //Lấy password\_45\_Tuan  acc\_45\_Tuan = new Account\_45\_Tuan(username\_45\_Tuan, password\_45\_Tuan); //Tạo đối tượng  bool register\_45\_Tuan = acc\_45\_Tuan.checkRegister\_45\_Tuan();  Assert.IsTrue(register\_45\_Tuan); //Nếu trả ra True thì đăng ký thành công; PASS  } |

|  |  |
| --- | --- |
| username\_45\_Tuan | password\_45\_Tuan |
| Kieu@123 | Testpass@123 |
| Minh@123 | Testpass@123 |
| Tuan@123 | Testpass@123 |

*TestDataRegister\_45\_Tuan\_Successful.csv*



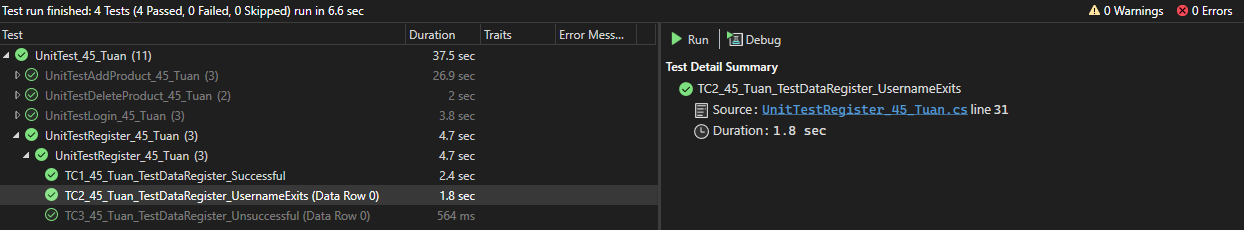
*Run test case hiển thị kết quả*

### 2.4. Test case đăng ký username đã tồn tại

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataRegister\_45\_Tuan\TestDataRegister\_45\_Tuan\_UsernameExist.csv", "TestDataRegister\_45\_Tuan\_UsernameExist#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC2: Test data đăng ký username đã tồn tại; 2 cột và 1 hàng; 1 PASS  public void TC2\_45\_Tuan\_TestDataRegister\_UsernameExist()  {  string username\_45\_Tuan = TestContext.DataRow[0].ToString(); //Lấy username\_45\_Tuan  string password\_45\_Tuan = TestContext.DataRow[1].ToString(); //Lấy password\_45\_Tuan  acc\_45\_Tuan = new Account\_45\_Tuan(username\_45\_Tuan, password\_45\_Tuan); //Tạo đối tượng  bool register\_45\_Tuan = acc\_45\_Tuan.checkRegister\_45\_Tuan();  Assert.IsFalse(register\_45\_Tuan); //Nếu trả ra False thì đăng ký thất bại; PASS  } |

|  |  |
| --- | --- |
| username\_45\_Tuan | password\_45\_Tuan |
| Kieu@123 | Testpass@123 |

*TestDataRegister\_45\_Tuan\_UsernameExist.csv*



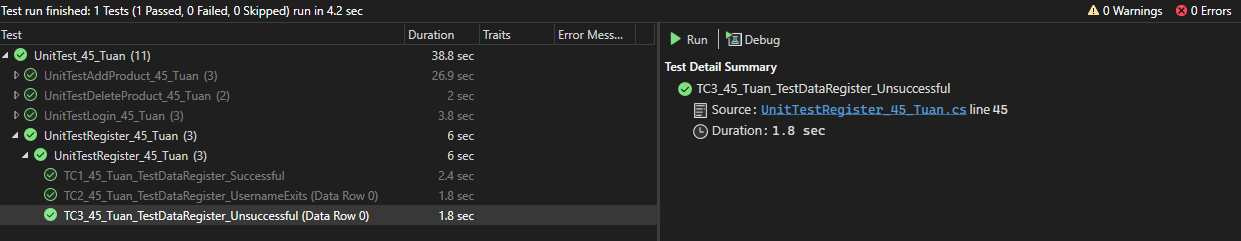
*Run test case hiển thị kết quả*

### 2.5. Test case đăng ký username và password không hợp lệ

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataRegister\_45\_Tuan\TestDataRegister\_45\_Tuan\_Unsuccessful.csv", "TestDataRegister\_45\_Tuan\_Unsuccessful#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC3: Test data đăng ký username và password không hợp lệ; 2 cột và 1 hàng; 1 PASS  public void TC3\_45\_Tuan\_TestDataRegister\_Unsuccessful()  {  string username\_45\_Tuan = TestContext.DataRow[0].ToString(); //Lấy username\_45\_Tuan  string password\_45\_Tuan = TestContext.DataRow[1].ToString(); //Lấy password\_45\_Tuan  acc\_45\_Tuan = new Account\_45\_Tuan(username\_45\_Tuan, password\_45\_Tuan); //Tạo đối tượng  bool register\_45\_Tuan = acc\_45\_Tuan.checkRegister\_45\_Tuan();  Assert.IsFalse(register\_45\_Tuan); //Nếu trả ra Fail thì đăng ký thất bại; PASS  } |

|  |  |
| --- | --- |
| username\_45\_Tuan | password\_45\_Tuan |
| kieuminhtuan | 123456 |

*TestDataRegister\_45\_Tuan\_Unsuccessful.csv*



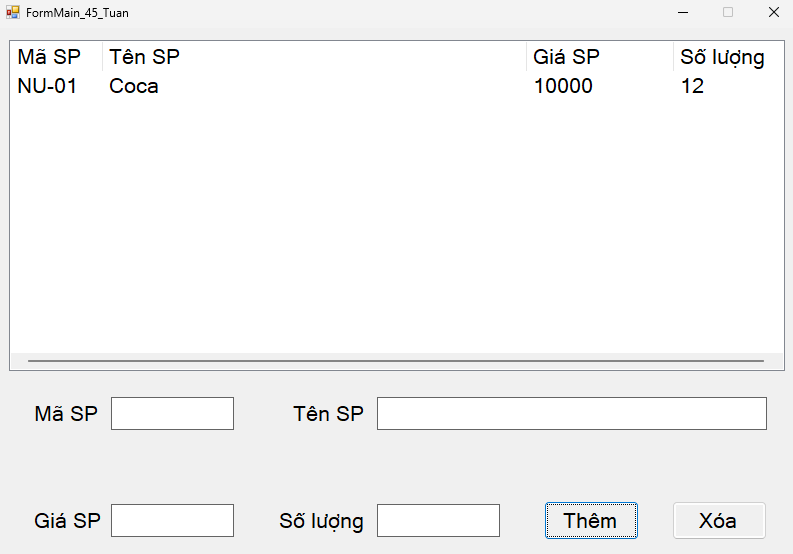
*Run test case hiển thị kết quả*

## 3. Chức năng thêm sản phẩm

### 3.1. Đặc tả

Người dùng có thể thêm một sản phẩm bằng cách nhập mã sản phẩm theo định dạng “AA-00”, tên, giá bán và số lượng của sản phẩm. Nếu hợp lệ sản phẩm sẽ được thêm vào danh sách và được hiển thị lên màn hình.

### 3.2. Giao diện thêm sản phẩm

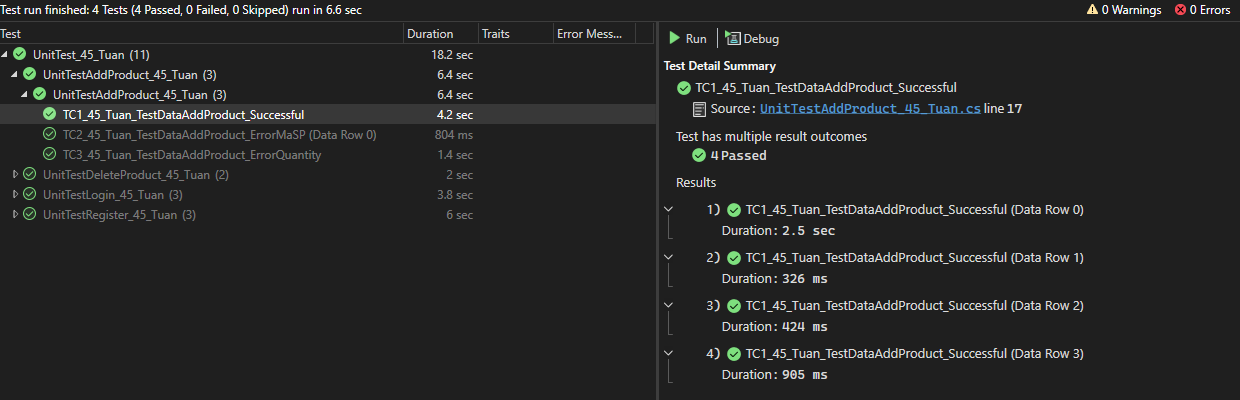


### 3.3. Test case thêm sản phẩm thành công

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataAddProduct\_45\_Tuan\TestDataAddProduct\_45\_Tuan\_Successful.csv", "TestDataAddProduct\_45\_Tuan\_Successful#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC1: Test data thêm SP thành công; 4 cột và 4 hàng; 4 PASS  public void TC1\_45\_Tuan\_TestDataAddProduct\_Successful()  {  string maSp\_45\_Tuan = TestContext.DataRow[0].ToString(); //Lấy cột maSP  string tenSP\_45\_Tuan = TestContext.DataRow[1].ToString(); //Lấy cột tenSP  double giaSP\_45\_Tuan = double.Parse(TestContext.DataRow[2].ToString()); //Lấy cột giaSP  int soLuong\_45\_Tuan = int.Parse(TestContext.DataRow[3].ToString()); //Lấy cột soLuong  prod\_45\_Tuan = new Product\_45\_Tuan(maSp\_45\_Tuan, tenSP\_45\_Tuan, giaSP\_45\_Tuan, soLuong\_45\_Tuan);  bool register\_45\_Tuan = prod\_45\_Tuan.checkAddProduct\_45\_Tuan();  Assert.IsTrue(register\_45\_Tuan); //Nếu trả ra True thì thêm SP thành công; PASS  } |

|  |  |  |  |
| --- | --- | --- | --- |
| maSP\_45\_Tuan | tenSP\_45\_Tuan | giaSP\_45\_Tuan | soLuong\_45\_Tuan |
| NU-01 | Pepsi | 10000 | 50 |
| NU-02 | Coca | 12000 | 25 |
| TA-01 | Kokomi Noodle | 5000 | 40 |
| TC-01 | Banana | 3000 | 15 |

*TestDataAddProduct\_45\_Tuan\_Successful.csv*

**

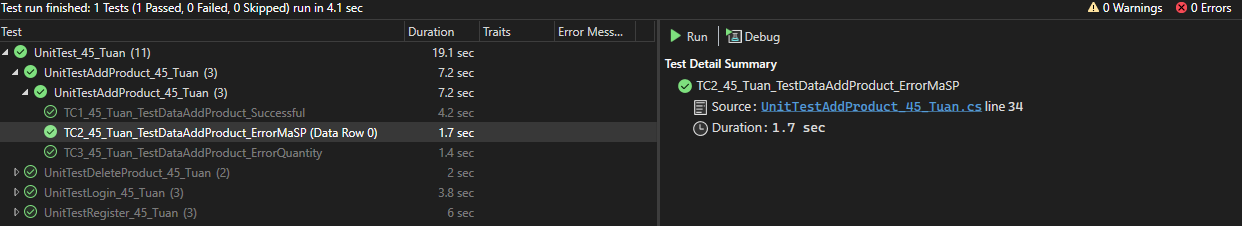
*Run test case hiển thị kết quả*

### 3.4. Test case thêm sản phẩm mã không hợp lệ

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataAddProduct\_45\_Tuan\TestDataAddProduct\_45\_Tuan\_ErrorMaSP.csv", "TestDataAddProduct\_45\_Tuan\_ErrorMaSP#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC2: Test data thêm SP thất bại vì maSP không hợp lệ; 4 cột và 1 hàng; 1 PASS  public void TC2\_45\_Tuan\_TestDataAddProduct\_ErrorMaSP()  {  string maSp\_45\_Tuan = TestContext.DataRow[0].ToString(); //Lấy cột maSP  string tenSP\_45\_Tuan = TestContext.DataRow[1].ToString(); //Lấy cột tenSP  double giaSP\_45\_Tuan = double.Parse(TestContext.DataRow[2].ToString()); //Lấy cột giaSP  int soLuong\_45\_Tuan = int.Parse(TestContext.DataRow[3].ToString()); //Lấy cột soLuong  prod\_45\_Tuan = new Product\_45\_Tuan(maSp\_45\_Tuan, tenSP\_45\_Tuan, giaSP\_45\_Tuan, soLuong\_45\_Tuan);  bool register\_45\_Tuan = prod\_45\_Tuan.checkAddProduct\_45\_Tuan();  Assert.IsFalse(register\_45\_Tuan); //Nếu trả ra False thì thêm SP thất bại; PASS  } |

|  |  |  |  |
| --- | --- | --- | --- |
| maSP\_45\_Tuan | tenSP\_45\_Tuan | giaSP\_45\_Tuan | soLuong\_45\_Tuan |
| NUAA | Pepsi | 10000 | 50 |

*TestDataAddProduct\_45\_Tuan\_ErrorMaSP.csv*

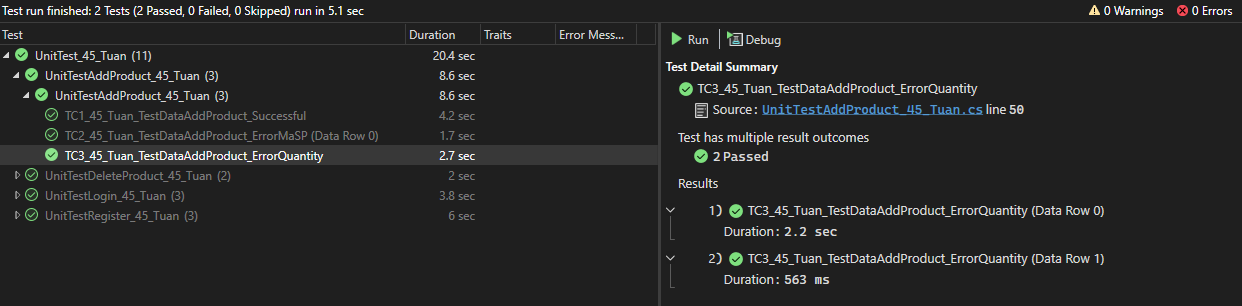
*Run test case hiển thị kết quả*

### 3.5. Test case thêm sản phẩm giá bán và số lượng không hợp lệ

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataAddProduct\_45\_Tuan\TestDataAddProduct\_45\_Tuan\_ErrorQuantity.csv", "TestDataAddProduct\_45\_Tuan\_ErrorQuantity#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC3: Test data thêm SP thất bại vì đơn giá và số lượng không hợp lệ; 4 cột và 2 hàng; 2 PASS  public void TC3\_45\_Tuan\_TestDataAddProduct\_ErrorQuantity()  {  string maSp\_45\_Tuan = TestContext.DataRow[0].ToString(); //Lấy cột maSP  string tenSP\_45\_Tuan = TestContext.DataRow[1].ToString(); //Lấy cột tenSP  double giaSP\_45\_Tuan = double.Parse(TestContext.DataRow[2].ToString()); //Lấy cột giaSP  int soLuong\_45\_Tuan = int.Parse(TestContext.DataRow[3].ToString()); //Lấy cột soLuong  prod\_45\_Tuan = new Product\_45\_Tuan(maSp\_45\_Tuan, tenSP\_45\_Tuan, giaSP\_45\_Tuan, soLuong\_45\_Tuan);  bool register\_45\_Tuan = prod\_45\_Tuan.checkAddProduct\_45\_Tuan();  Assert.IsFalse(register\_45\_Tuan); //Nếu trả ra False thì thêm SP thất bại; PASS  } |

|  |  |  |  |
| --- | --- | --- | --- |
| maSP\_45\_Tuan | tenSP\_45\_Tuan | giaSP\_45\_Tuan | soLuong\_45\_Tuan |
| NU-01 | Pepsi | -10000 | 50 |
| NU-02 | Coca | 12000 | -2 |

*TestDataAddProduct\_45\_Tuan\_ErrorQuantity.csv*

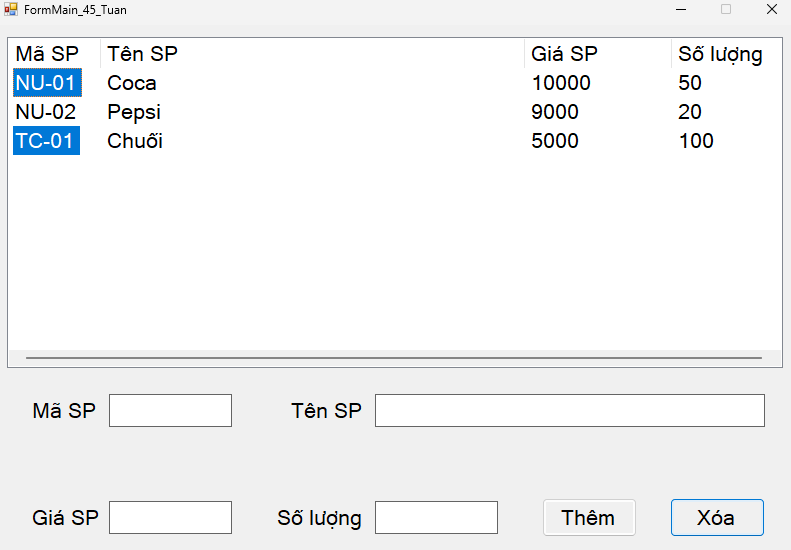
*Run test case hiển thị kết quả*

## 4. Chức năng xóa sản phẩm

### 4.1. Đặc tả

Người dùng có thể chọn sản phẩm đang hiển thị trên màn hình để xóa. Đồng thời sản phẩm được chọn sẽ dựa vào mã sản phẩm để xóa sản phẩm đó.

### 4.2. Giao diện xóa sản phẩm

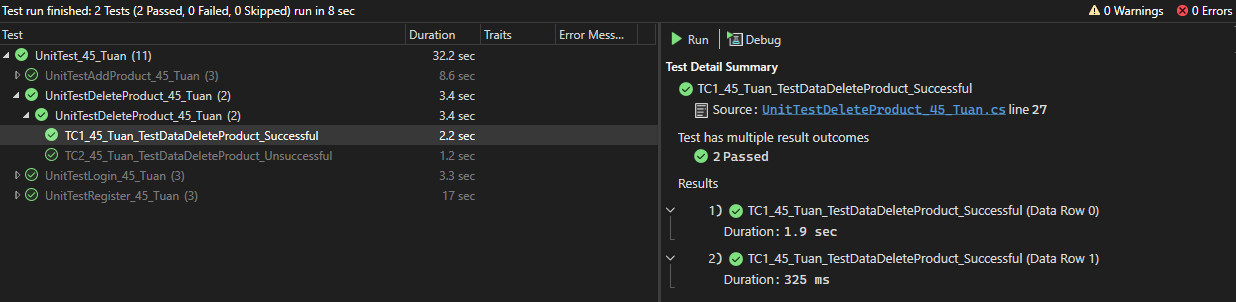


### 4.3. Test case xóa sản phẩm thành công

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataDeleteProduct\_45\_Tuan\TestDataDeleteProduct\_45\_Tuan\_Successful.csv", "TestDataDeleteProduct\_45\_Tuan\_Successful#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC1: Test data xóa SP thành công; 1 cột và 2 hàng; 2 PASS  public void TC1\_45\_Tuan\_TestDataDeleteProduct\_Successful()  {  string maSp\_45\_Tuan = TestContext.DataRow[0].ToString();//Lấy cột maSP  Assert.IsTrue(prod\_45\_Tuan.deleteProduct\_45\_Tuan(maSp\_45\_Tuan)); //Nếu trả ra True thì xóa SP thành công; PASS  } |

|  |
| --- |
| MaSP\_45\_Tuan |
| NU-01 |
| NU-02 |

*TestDataDeleteProduct\_45\_Tuan\_Successful.csv*

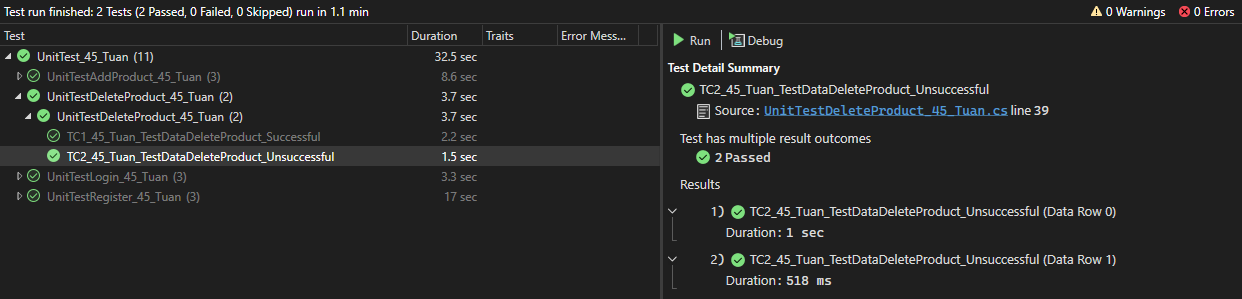
*Run test case hiển thị kết quả*

### 4.4. Test case xóa sản phẩm không thành công

|  |
| --- |
| [DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV",  @".\DataDeleteProduct\_45\_Tuan\TestDataDeleteProduct\_45\_Tuan\_Unsuccessful.csv", "TestDataDeleteProduct\_45\_Tuan\_Unsuccessful#csv", DataAccessMethod.Sequential)]  [TestMethod]  //TC2: Test data xóa SP thất bại; 1 cột và 2 hàng; 2 PASS  public void TC2\_45\_Tuan\_TestDataDeleteProduct\_Unsuccessful()  {  string maSp\_45\_Tuan = TestContext.DataRow[0].ToString();//Lấy cột maSP  Assert.IsFalse(prod\_45\_Tuan.deleteProduct\_45\_Tuan(maSp\_45\_Tuan)); //Nếu trả ra False thì xóa SP thất bại; PASS  } |

|  |
| --- |
| MaSP\_45\_Tuan |
| NU-01 |
| NU-02 |

*TestDataDeleteProduct\_45\_Tuan\_Unsuccessful.csv*

*Run test case hiển thị kết quả*

# B. WEB DRIVER

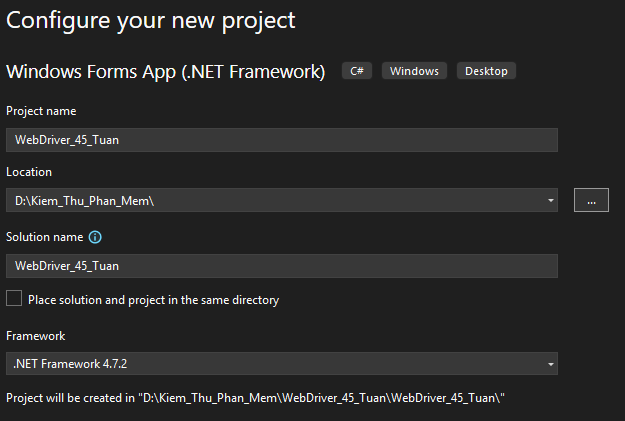
# I - TỔNG QUAN

## 1. Tổng quan về website Magento

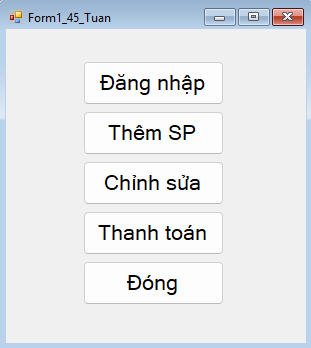
​ Trang web Magento là một trang web demo cho Hyvä, một bộ giao diện người dùng (theme) dành cho nền tảng thương mại điện tử Magento 2. Hyvä được thiết kế nhằm thay thế giao diện mặc định của Magento, tập trung vào hiệu suất cao, đơn giản hóa và trải nghiệm người dùng tối ưu.

## 2. Cài đặt Selenium

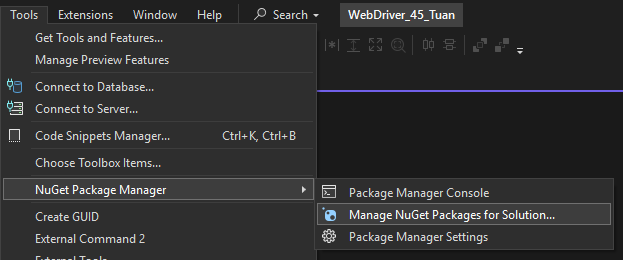
2.1. Tạo Project mới có hỗ trợ .NET Framework đặt tên là WebDriver\_45\_Tuan



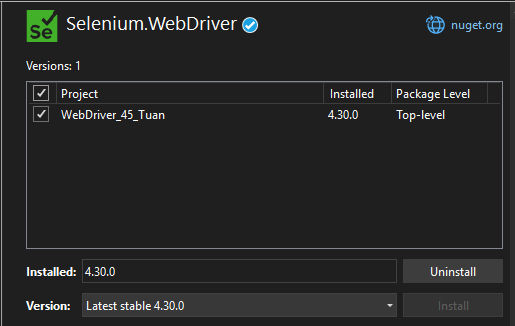
### 2.1. Giao diện thao tác với web



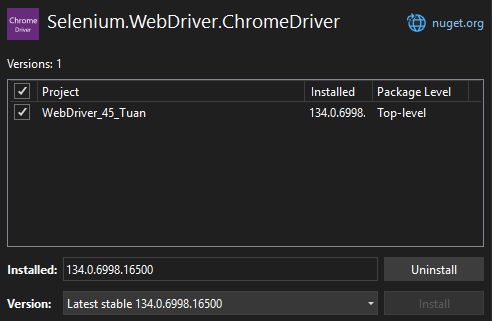
### 2.2. Chọn Tool -> Nuget Package Manager -> Manage Nuget Packages for Solution…

**

### 2.3. Cài đặt Selenium.WebDriver sau đó thêm Project WebDriver\_45\_Tuan vào



### 2.4. Cài đặt Selenium.WebDriver.ChromeDriver sau đó thêm Project WebDriver\_45\_Tuan vào



### 2.5. Thêm thư viện cần thiết vào Project

|  |
| --- |
| using OpenQA.Selenium;  using OpenQA.Selenium.Chrome; |

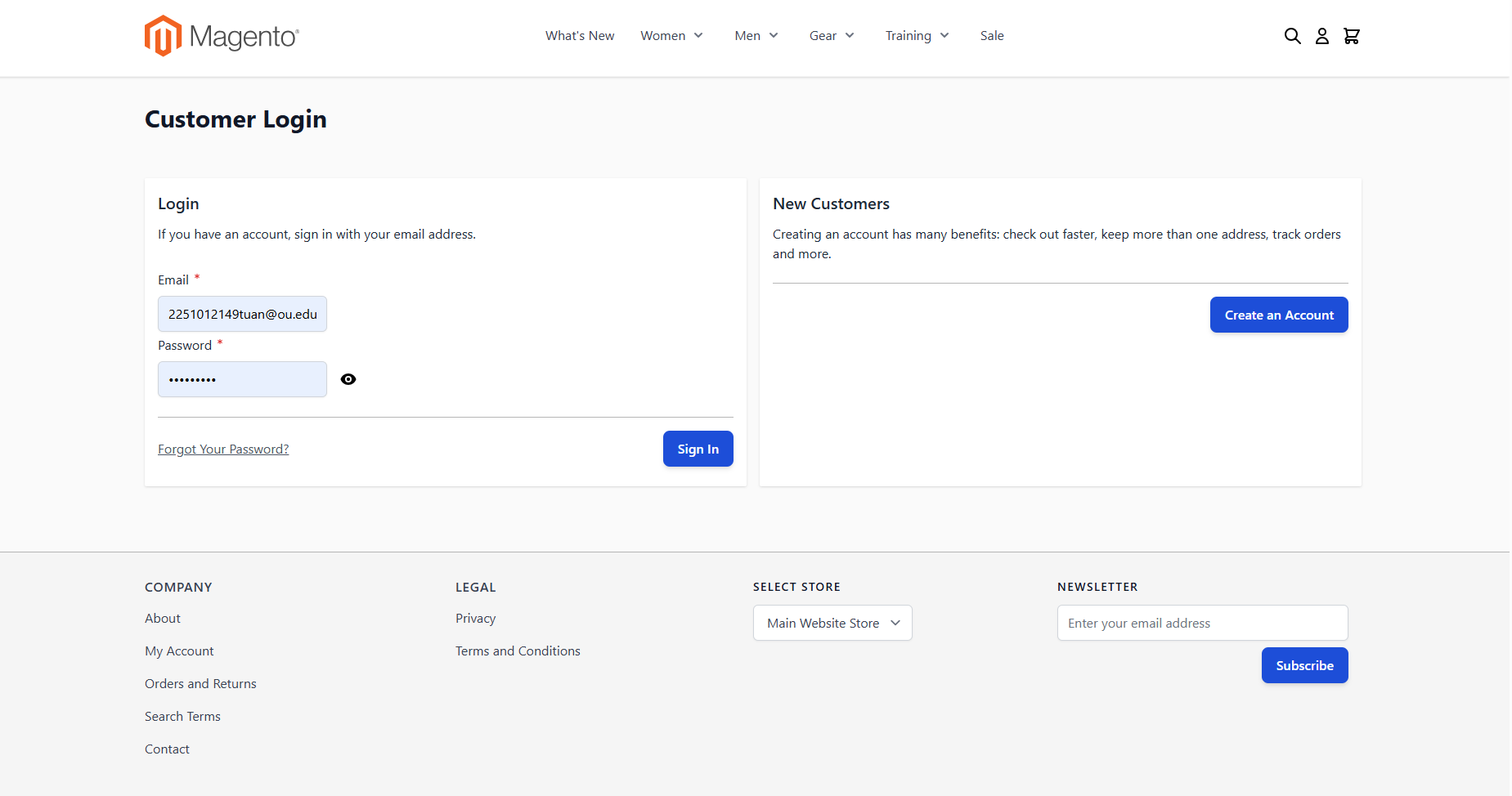
# II - CÁC CHỨC NĂNG CHÍNH

## 1. Chức năng đăng nhập

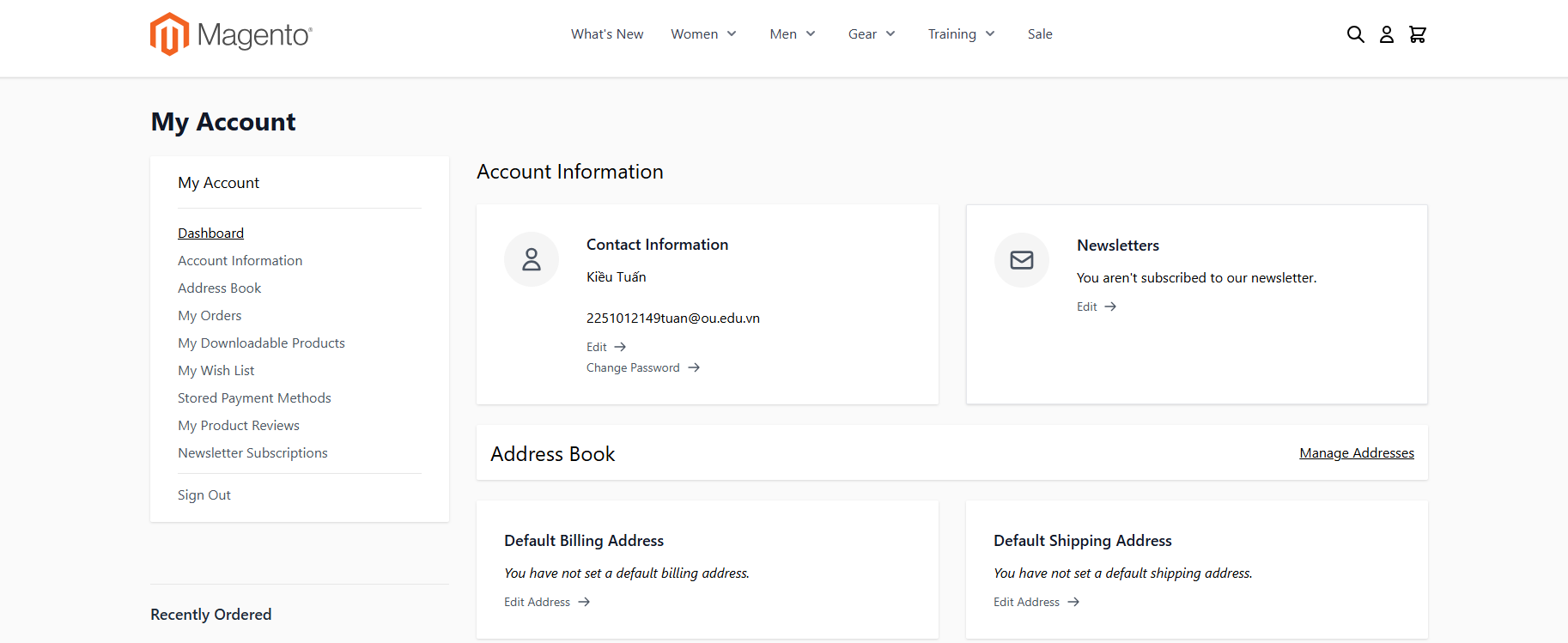
### 1.1. Đặc tả

Khi người dùng đăng nhập thành công Email và Password đã được đăng ký trước đó thì sẽ được điều hướng đến trang thông tin tài khoản của người dùng.

### 1.2. Giao diên đăng nhập



### 1.3. Giao diện thông tin tài khoản



### 1.4. Test case đăng nhập thành công

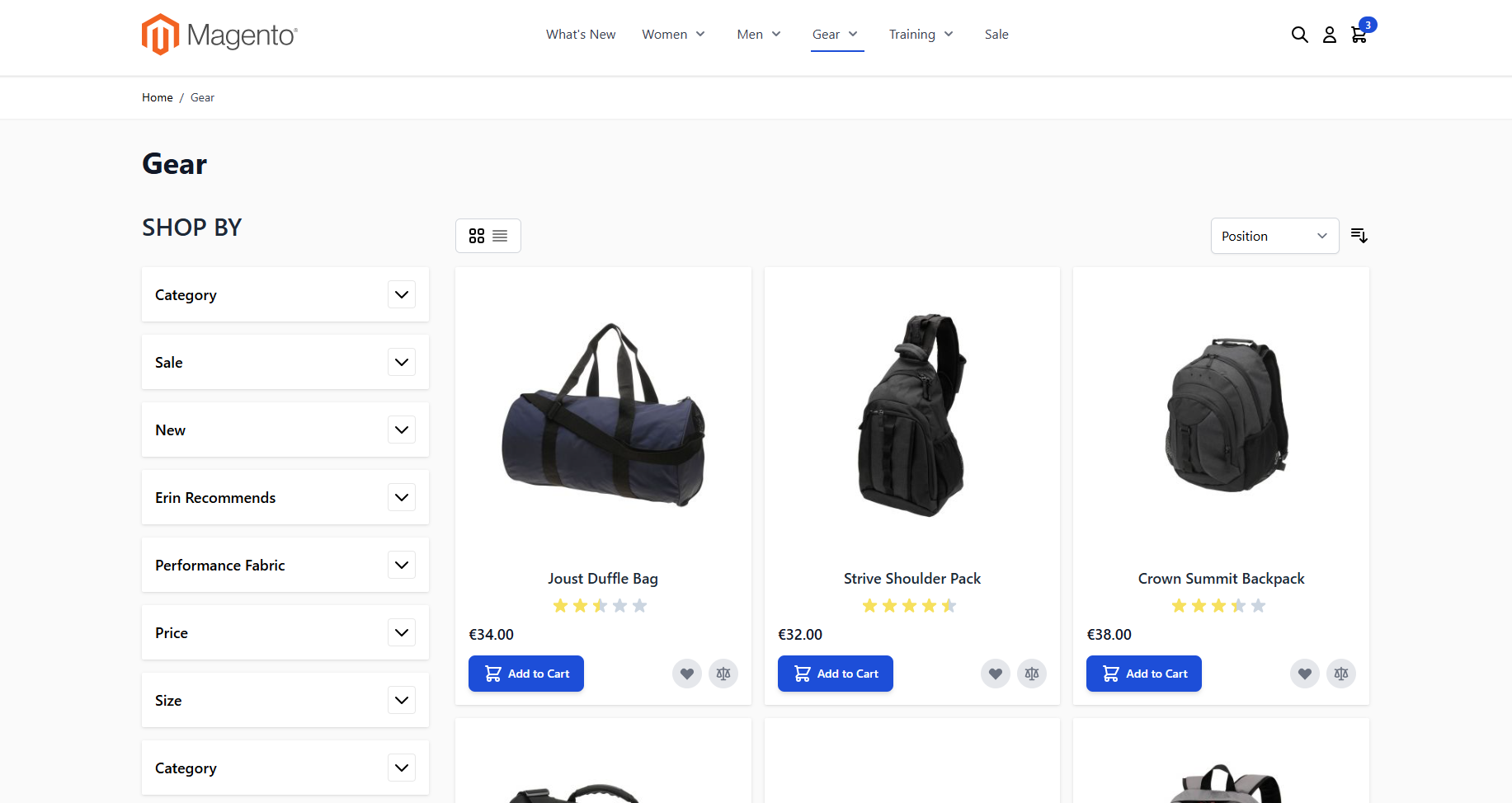
|  |
| --- |
| // TC1: đăng nhập thành công  private void btLogin\_45\_Tuan\_Click\_1(object sender, EventArgs e)  {  ChromeDriverService chrome\_45\_Tuan = ChromeDriverService.CreateDefaultService();  chrome\_45\_Tuan.HideCommandPromptWindow = true;  driver\_45\_Tuan = new ChromeDriver(chrome\_45\_Tuan);  driver\_45\_Tuan.Navigate().GoToUrl("https://demo.hyva.io/default/customer/account/login/");  IWebElement username\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Name("login[username]")); //Lấy element username  username\_45\_Tuan.SendKeys("2251012149tuan@ou.edu.vn");  IWebElement password\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Name("login[password]")); //Lấy element password  password\_45\_Tuan.SendKeys("Admin@123");  IWebElement login\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Name("send")); //Lấy element send  login\_45\_Tuan.Click();  } |

## 2. Chức năng thêm sản phẩm

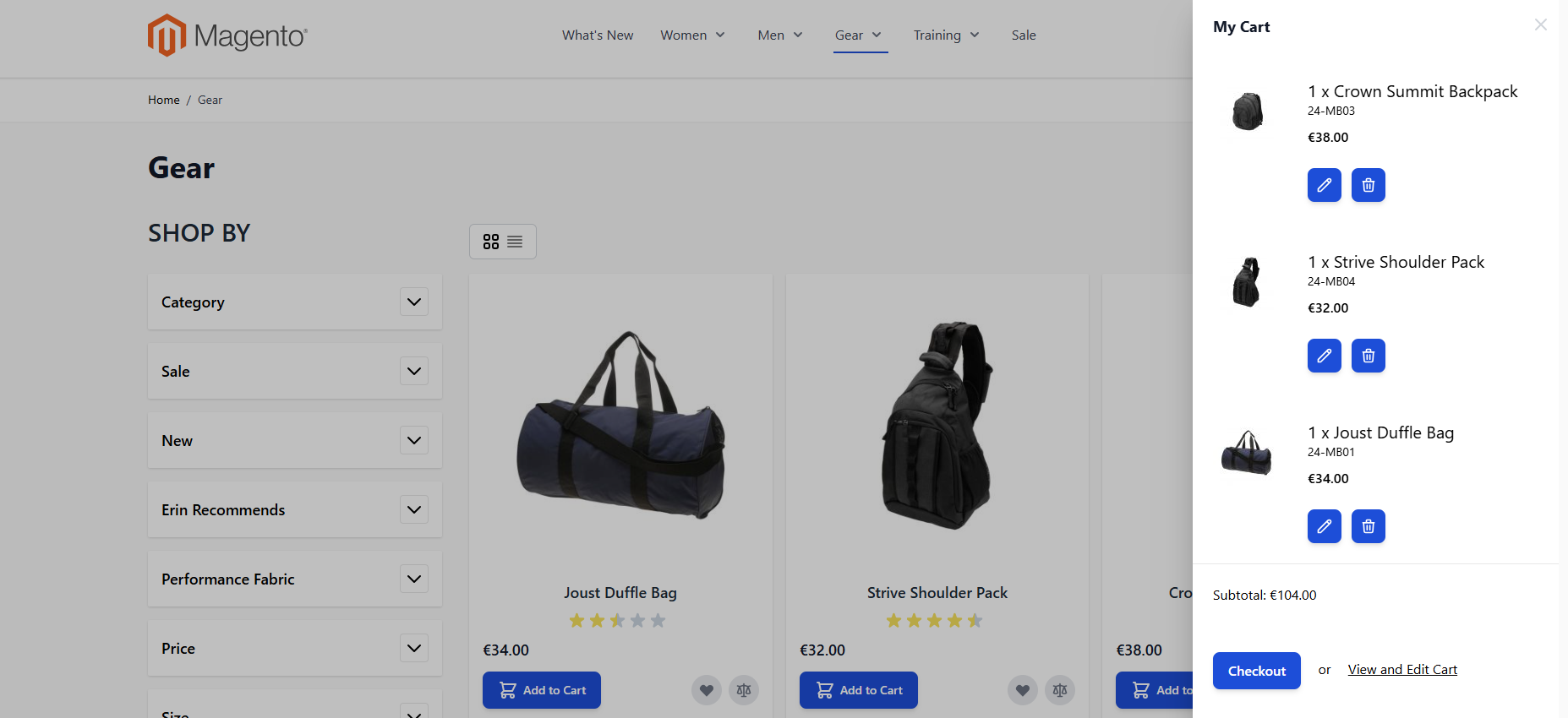
### 2.1. Đặc tả

Người dùng lần lượt ấn “Add to Cart” cho từng sản phẩm muốn mua, các sản phẩm sẽ được thêm vào giỏ hàng với số lượng mặc định là 1.

### 2.2. Giao diện thêm sản phẩm vào giỏ hàng



### 2.3. Giao diện giỏ hàng



### 2.4. Test case thêm sản phẩm vào giỏ hàng thành công

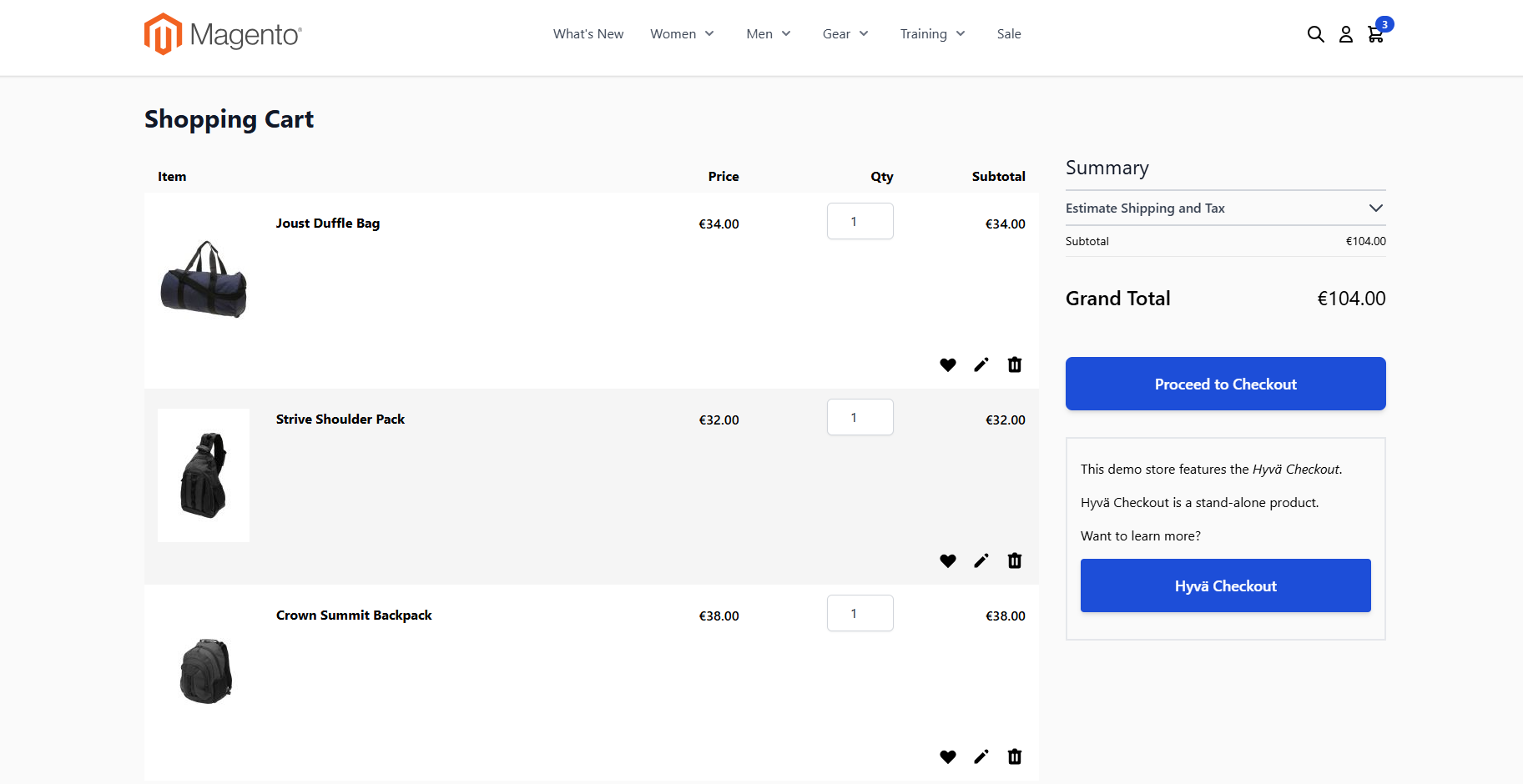
|  |
| --- |
| // TC2: Thêm sản phẩm vào giỏ hàng thành công  private void btOrder\_45\_Tuan\_Click(object sender, EventArgs e)  {  IWebElement bags\_45\_Tuan = driver\_45\_Tuan.FindElement(By.CssSelector("#header > div.container.flex.flex-wrap.lg\\:flex-nowrap.items-center.justify-between.w-full.px-6.py-3.mx-auto.mt-0 > div.z-20.order-2.sm\\:order-1.lg\\:order-2.navigation.hidden.lg\\:flex > div > nav > ul > li:nth-child(4) > span > a"));  bags\_45\_Tuan.Click();  Thread.Sleep(1000);  IWebElement addToCart\_45\_Tuan = driver\_45\_Tuan.FindElement(By.CssSelector("#product-list > div.products.wrapper.mode-grid.products-grid > ul > li:nth-child(1) > form > div > div.mt-auto.pt-3.flex.flex-wrap.items-center.gap-y-4.gap-x-2 > button"));  addToCart\_45\_Tuan.Click(); //Thêm SP 1  Thread.Sleep(2000);  IWebElement addToCart2\_45\_Tuan = driver\_45\_Tuan.FindElement(By.CssSelector("#product-list > div.products.wrapper.mode-grid.products-grid > ul > li:nth-child(2) > form > div > div.mt-auto.pt-3.flex.flex-wrap.items-center.gap-y-4.gap-x-2 > button"));  addToCart2\_45\_Tuan.Click(); //Thêm SP 2  Thread.Sleep(2000);  IWebElement addToCart3\_45\_Tuan = driver\_45\_Tuan.FindElement(By.CssSelector("#product-list > div.products.wrapper.mode-grid.products-grid > ul > li:nth-child(3) > form > div > div.mt-auto.pt-3.flex.flex-wrap.items-center.gap-y-4.gap-x-2 > button"));  addToCart3\_45\_Tuan.Click(); //Thêm SP 3  } |

## 3. Chức năng chỉnh sửa giỏ hàng

### 3.1. Đặc tả

Người dùng có thể tăng giảm số lượng sản phẩm cho phù hợp với nhu cầu của mình trước khi tiến hành thanh toán.

### 3.2. Giao diện chỉnh sửa giỏ hàng



### 3.3. Test case chỉnh sửa giỏ hàng thành công

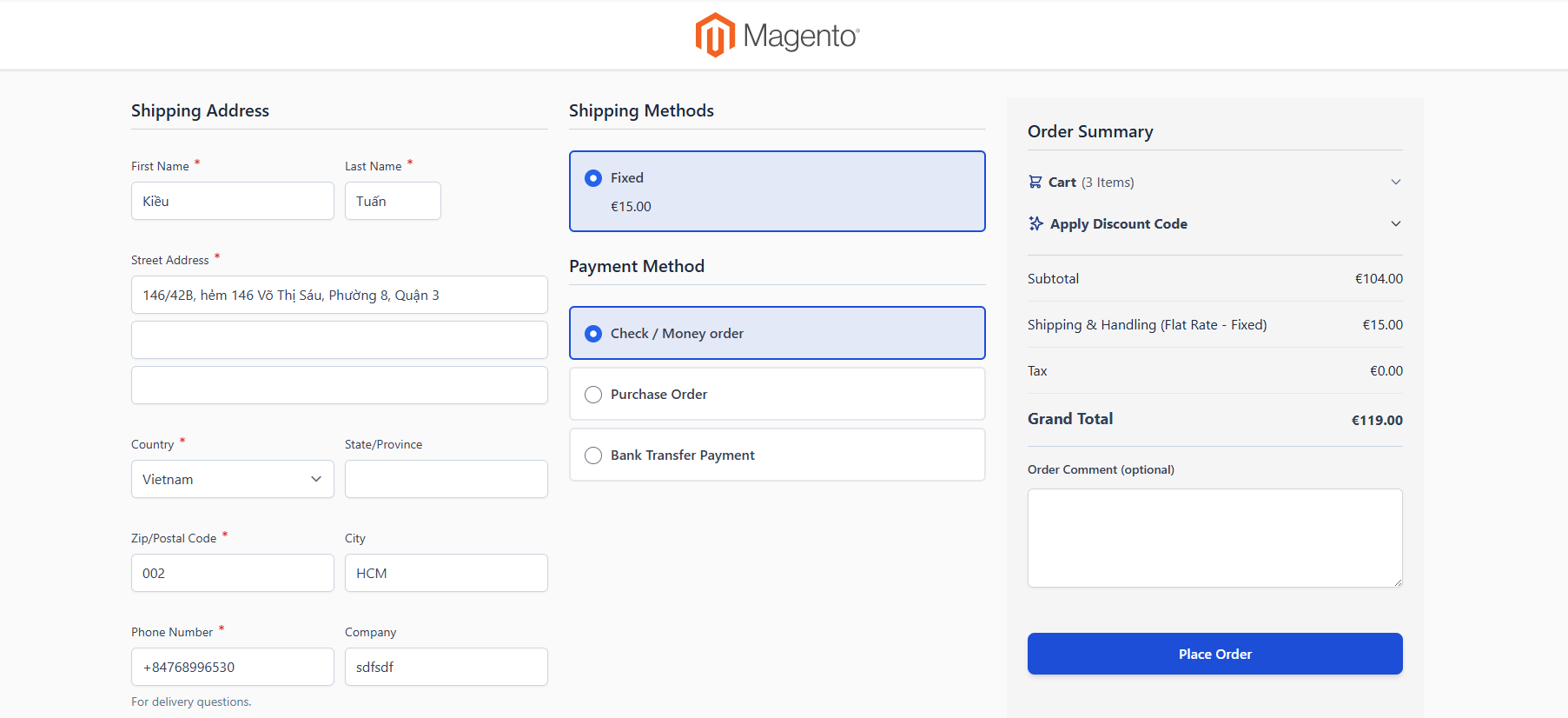
|  |
| --- |
| // TC3: Chỉnh sửa giỏ hàng  private void btEdit\_45\_Tuan\_Click(object sender, EventArgs e)  {  IWebElement cart\_45\_Tuan = driver\_45\_Tuan.FindElement(By.CssSelector("#menu-cart-icon"));  cart\_45\_Tuan.Click();  Thread.Sleep(1000);  IWebElement editDetail\_45\_Tuan = driver\_45\_Tuan.FindElement(By.CssSelector("#cart-drawer > div > div.relative.w-screen.max-w-md.shadow-2xl > div > div:nth-child(6) > div > div:nth-child(2) > a.underline"));  editDetail\_45\_Tuan.Click();  IWebElement bag\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Id("cart-93045-qty"));  bag\_45\_Tuan.Clear();  bag\_45\_Tuan.SendKeys("2"); //Chỉnh sửa số lượng SP 1  IWebElement pack\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Id("cart-93046-qty"));  pack\_45\_Tuan.Clear();  pack\_45\_Tuan.SendKeys("3"); //Chỉnh sửa số lượng SP 2  IWebElement backpack\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Id("cart-93047-qty"));  backpack\_45\_Tuan.Clear();  backpack\_45\_Tuan.SendKeys("4"); //Chỉnh sửa số lượng SP 3  IWebElement checkout\_45\_Tuan = driver\_45\_Tuan.FindElement(By.CssSelector("#checkout-link-button"));  checkout\_45\_Tuan.Click();  } |

## 4. Chức năng thanh toán

### 4.1. Đặc tả

Sau khi người dùng ấn nút “Checkout” thì sẽ được điểu hướng đến trang thanh toán, lúc này người dùng cùng cung cấp đầy đủ thông tin như first name, last name, phương thức thanh toán, …v.v. Sau khi cung cấp đầy đủ thông tin thì người dùng sẽ tiến hành nhấn nút “Place Order” để thanh toán.

### 4.2. Giao diện thanh toán



### 4.3. Test case thanh toán thành công

|  |
| --- |
| // TC4: Thanh toán thành công  private void btPay\_45\_Tuan\_Click(object sender, EventArgs e)  {  // Điền thông tin thanh toán  IWebElement firstname\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Name("firstname"));  firstname\_45\_Tuan.Clear();  firstname\_45\_Tuan.SendKeys("Kieu");  IWebElement lastname\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Name("lastname"));  lastname\_45\_Tuan.Clear();  lastname\_45\_Tuan.SendKeys("Tuan");  IWebElement street\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Name("street[0]"));  street\_45\_Tuan.Clear();  street\_45\_Tuan.SendKeys("146/23B Vo Thi Sau, Quan 3, TP Ho Chi Minh");  IWebElement code\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Name("postcode"));  code\_45\_Tuan.Clear();  code\_45\_Tuan.SendKeys("002");  IWebElement city\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Name("city"));  city\_45\_Tuan.Clear();  city\_45\_Tuan.SendKeys("HCM");  IWebElement phone\_45\_Tuan = driver\_45\_Tuan.FindElement(By.Name("telephone"));  phone\_45\_Tuan.Clear();  phone\_45\_Tuan.SendKeys("0768996667");  IWebElement order\_45\_Tuan = driver\_45\_Tuan.FindElement(By.XPath("/html/body/div[2]/main/div[4]/div/div[1]/div/div/div[3]/nav/div/button"));  order\_45\_Tuan.Click(); // click nút thanh toán  } |

# C. APIs POSTMAN

# I - TỔNG QUAN

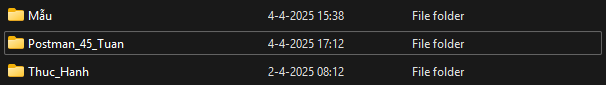
## 1. Tổng quan về quản lý APIs dữ liệu người dùng

Postman là một công cụ mạnh mẽ hỗ trợ kiểm thử và phát triển API. Khi quản lý dữ liệu người dùng, Postman đóng vai trò quan trọng trong việc tạo, đọc, cập nhật và xóa thông tin của người dùng thông qua APIs.

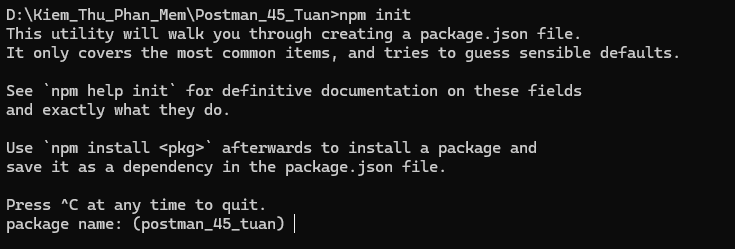
## 2. Cài đặt Posman, Json

### 2.1. Khởi tạo File Json

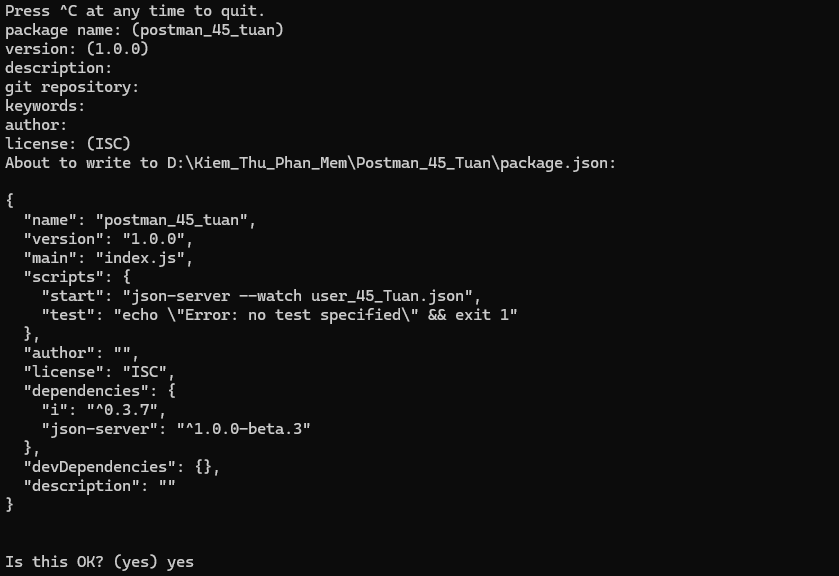
Tạo Folder mới đặt tên là Postman\_45\_Tuan



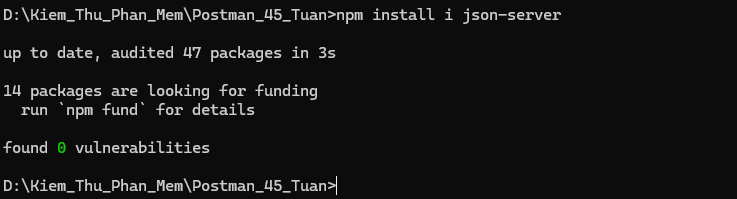
Mở màn hình CMD điều hướng địa chỉ tới Folder vừa tạo và nhập npm init



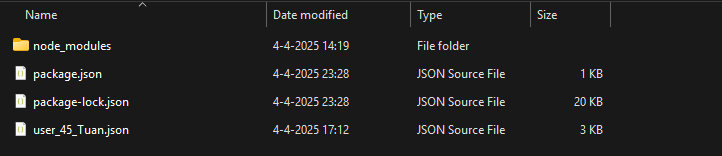
Nhập Enter đến cuối sau đó nhập yes



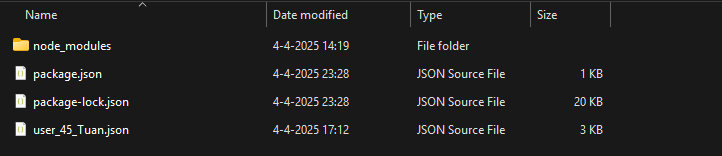
Sau đó nhập npm install i json-server để hoàn tất cài đặt



Kết quả sau khi khởi tạo



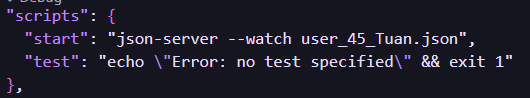
Khởi tạo File json mới đặt tên là user\_45\_Tuan.json



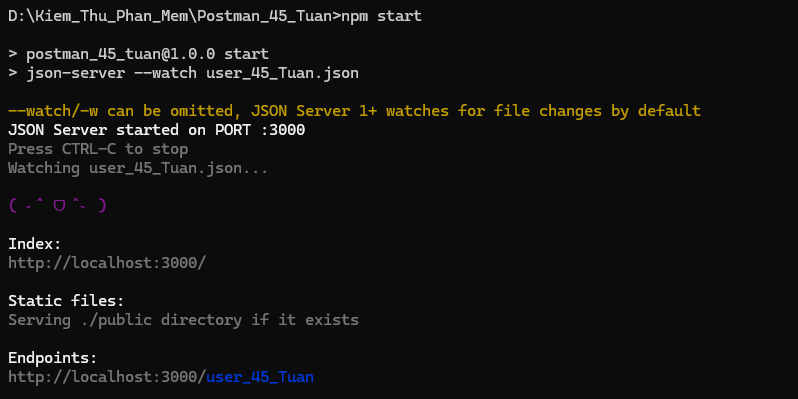
Thêm dữ liệu vào File

|  |
| --- |
| {  "user\_45\_Tuan": [  {  "id": "1",  "email": "trangnguyen@gmail.com",  "first\_name": "Nguyen",  "last\_name": "Trang",  "phone": "0768996543",  "subject\_scores": [  {  "math": 9,  "literature": 8,  "english": 9  }  ]  },  {  "id": "2",  "email": "minhhoang@gmail.com",  "first\_name": "Hoang",  "last\_name": "Minh",  "phone": "0987654321",  "subject\_scores": [  {  "math": 8,  "literature": 7,  "english": 8  }  ]  },  {  "id": "3",  "email": "thanhle@gmail.com",  "first\_name": "Le",  "last\_name": "Thanh",  "phone": "0345678912",  "subject\_scores": [  {  "math": 7,  "literature": 9,  "english": 8  }  ]  },  {  "id": "4",  "email": "haianh@gmail.com",  "first\_name": "Anh",  "last\_name": "Hai",  "phone": "0912345678",  "subject\_scores": [  {  "math": 9,  "literature": 6,  "english": 7  }  ]  },  {  "id": "5",  "email": "lamphong@gmail.com",  "first\_name": "Phong",  "last\_name": "Lam",  "phone": "0321654987",  "subject\_scores": [  {  "math": 8,  "literature": 8,  "english": 9  }  ]  },  {  "id": "6",  "email": "thuyduong@gmail.com",  "first\_name": "Duong",  "last\_name": "Thuy",  "phone": "0798456123",  "subject\_scores": [  {  "math": 7,  "literature": 8,  "english": 7  }  ]  },  {  "id": "7",  "email": "quangtrung@gmail.com",  "first\_name": "Trung",  "last\_name": "Quang",  "phone": "0887456321",  "subject\_scores": [  {  "math": 9,  "literature": 7,  "english": 8  }  ]  }  ]  } |

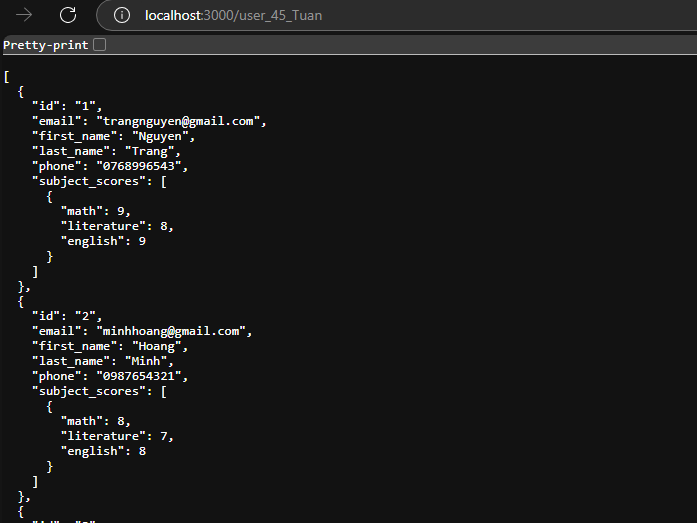
Mở File package.json và thêm dòng start như bên dưới



Mở lại màn hình CMD nhập npm start để khởi chạy API

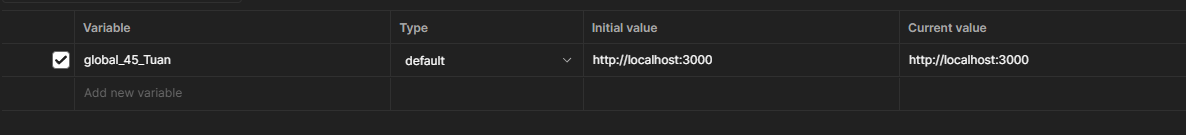


Nhập địa chỉ http://[localhost:3000/user\_45\_Tuan](http://localhost:3000/user_45_Tuan) vào thanh tìm kiếm để kiểm tra

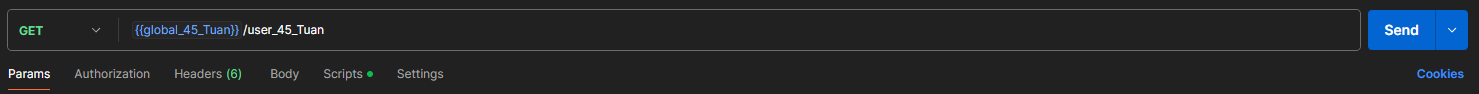


### 2.2. Cài đặt Postman

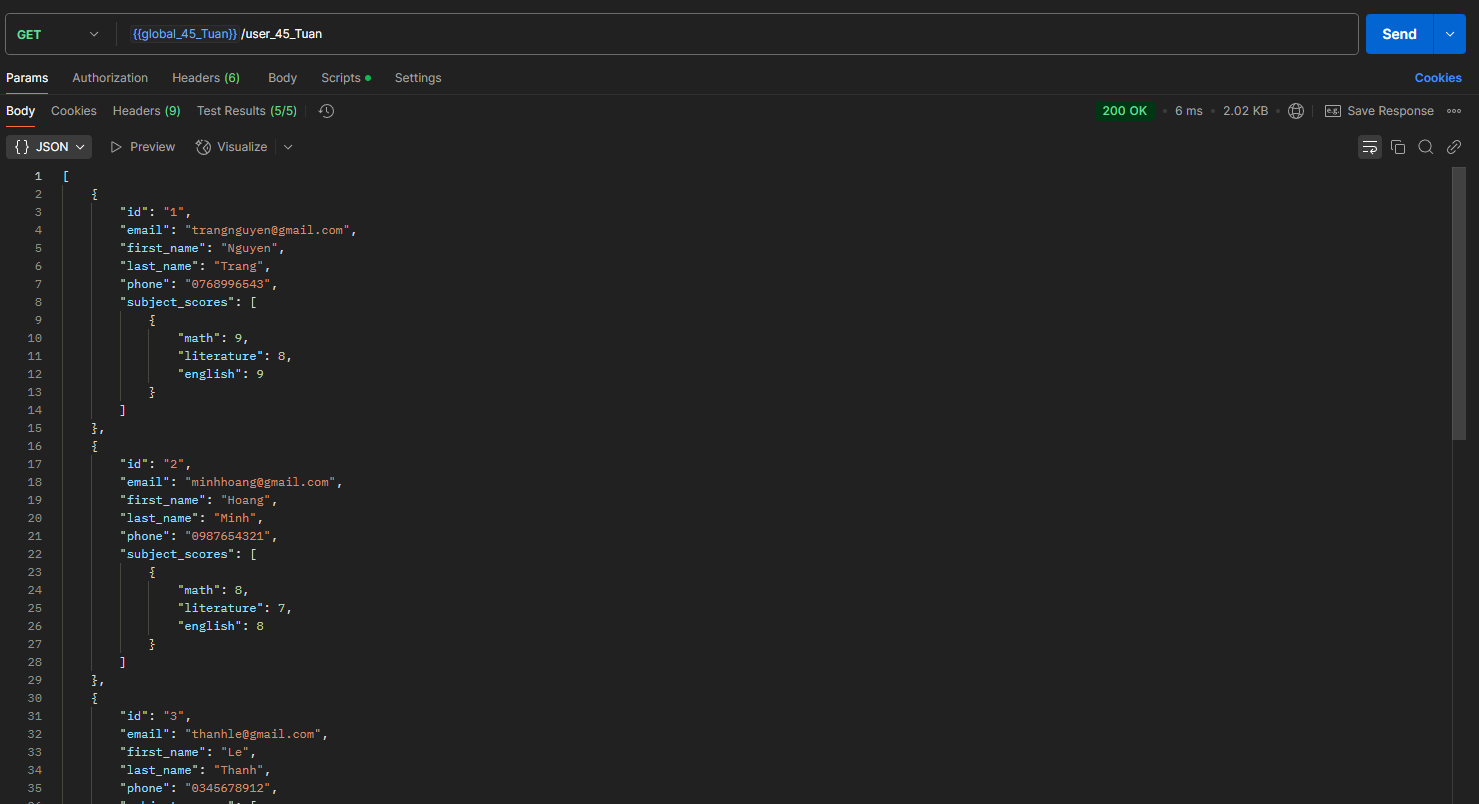
Thiết lập biến môi trường cho địa chỉ Endpoint và đặt tên là global\_45\_Tuan



Sử dụng biến môi trường mới tạo để kiểm tra lấy dữ liệu từ APIs



Kết quả dữ liệu được lấy về



# II - CÁC PHƯƠNG THỨC CHÍNH

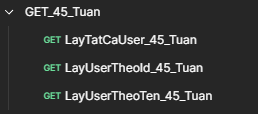
## 1. PHƯƠNG THỨC GET

### 1.1. Đặc tả

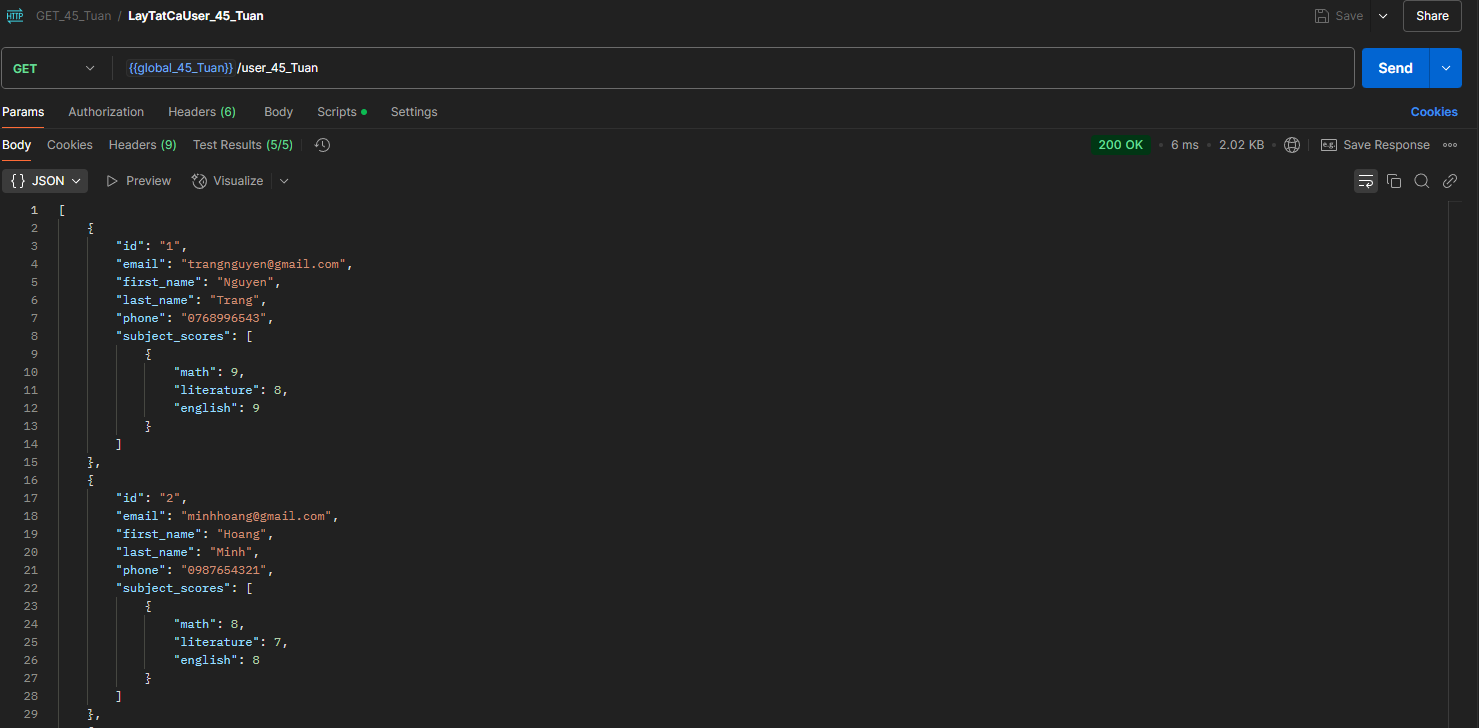
Phương thức GET hỗ trợ lấy dữ liệu từ máy chủ về để người dùng tiến hành kiểm tra.

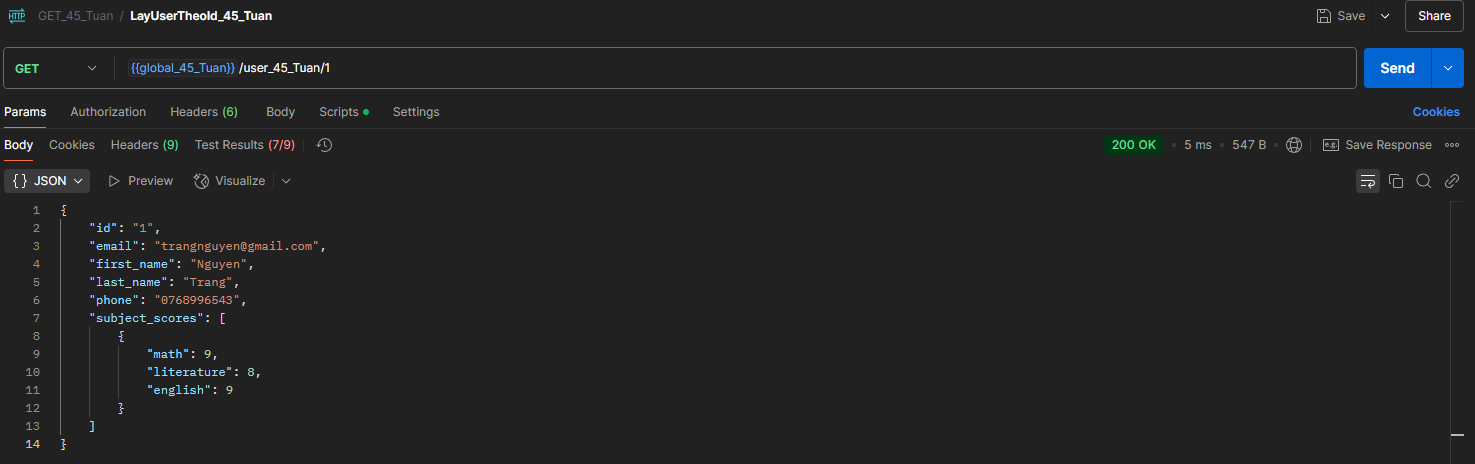
### 1.2. Giao diện

Khởi tạo Collection mới và đặt tên là GET\_45\_Tuan

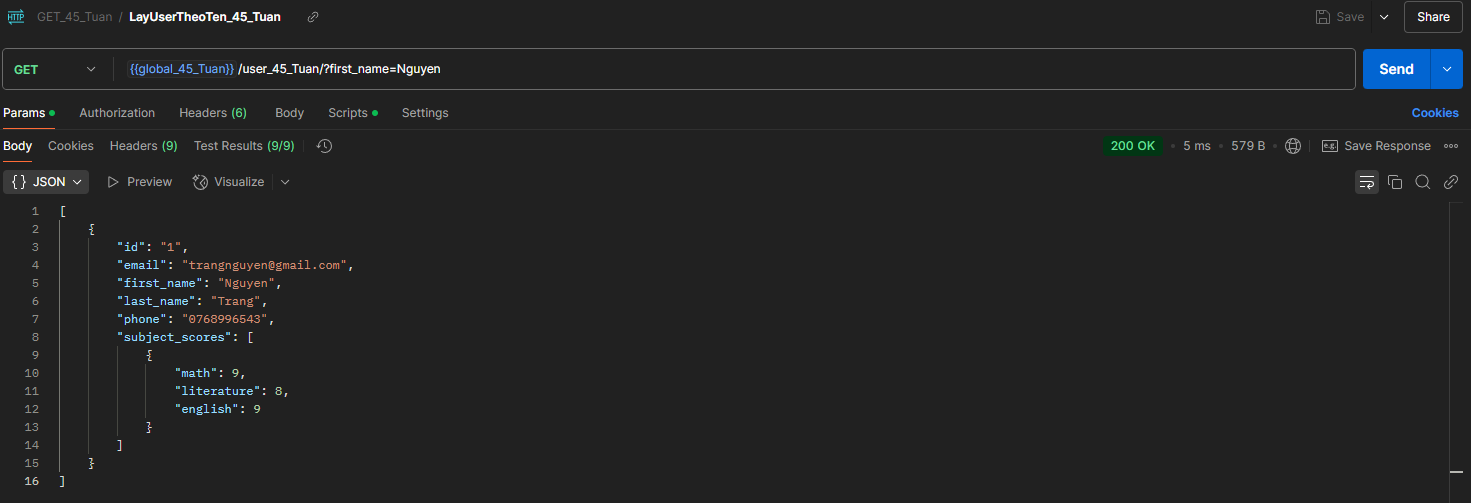


GET lấy tất cả user {{global\_45\_Tuan}}/user\_45\_Tuan



GET lấy user theo Id {{global\_45\_Tuan}}/user\_45\_Tuan/1

GET lấy user theo tên {{global\_45\_Tuan}}/user\_45\_Tuan/?first\_name=Nguyen



### 1.3. Test Scrip

Test Scrip cho phương thức GET lấy tất cả user

//TC1\_status\_45\_Tuan: Kiểm tra trạng thái phản hồi

pm.test("TC1\_status\_45\_Tuan", *function*(){

    pm.response.to.have.status(200);

});

//TC2\_response\_45\_Tuan: Kiểm tra thời gian phản hồi dưới 200ms

pm.test("TC2\_response\_45\_Tuan", *function*(){

    pm.expect(pm.response.responseTime).to.be.below(200);

});

//TC3\_Content-Type\_45\_Tuan: Kiểm tra header có thuộc tính Content-Type

pm.test("TC3\_Content-Type\_45\_Tuan", *function*(){

    pm.response.to.have.header("Content-Type")

})

//TC4\_Json\_45\_Tuan: Kiểm tra xem có JSON không

pm.test("TC4\_Json\_45\_Tuan", *function*(){

    pm.response.to.have.jsonBody()

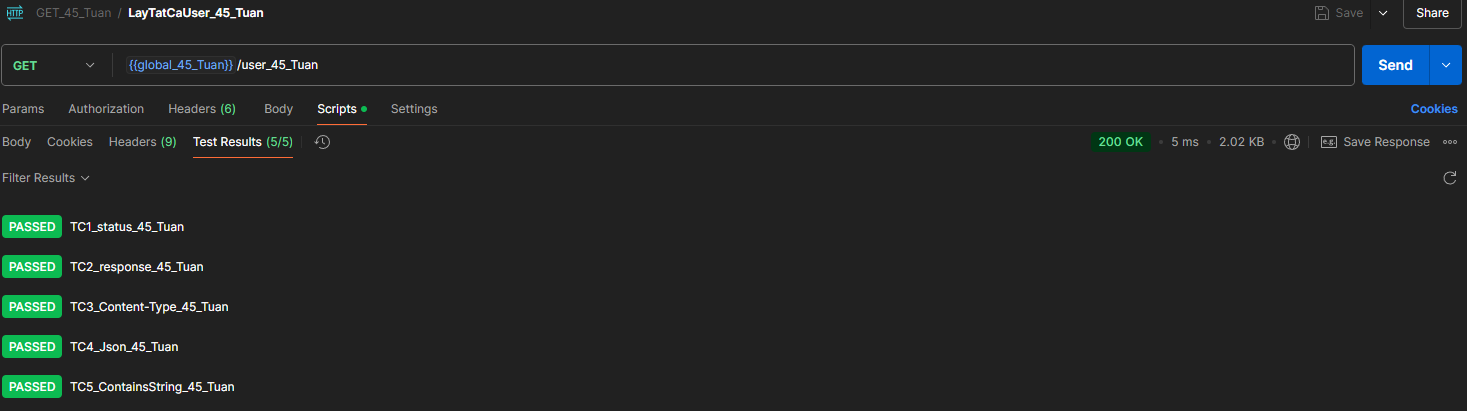
});

//TC5\_ContainsString\_45\_Tuan: response có chứa chuỗi "trangnguyen@gmail.com" không

pm.test("TC5\_ContainsString\_45\_Tuan", *function*(){

    pm.expect(pm.response.text()).to.include("trangnguyen@gmail.com")

});



*Kết quả hiển thị*

Test Scrip cho phương thức GET lấy user theo Id

//TC1\_status\_45\_Tuan: Kiểm tra trạng thái phản hồi

pm.test("TC1\_status\_45\_Tuan", *function*(){

    pm.response.to.have.status(200);

});

//TC2\_response\_45\_Tuan: Kiểm tra thời gian phản hồi dưới 200ms

pm.test("TC2\_response\_45\_Tuan", *function*(){

    pm.expect(pm.response.responseTime).to.be.below(200);

});

//TC3\_Content-Type\_45\_Tuan: Kiểm tra header có thuộc tính Content-Type

pm.test("TC3\_Content-Type\_45\_Tuan", *function*(){

    pm.response.to.have.header("Content-Type")

})

//TC4\_Json\_45\_Tuan: Kiểm tra xem có JSON không

pm.test("TC4\_Json\_45\_Tuan", *function*(){

    pm.response.to.have.jsonBody()

});

//TC5\_ContainsString\_45\_Tuan: response có chứa chuỗi "trangnguyen@gmail.com" không

pm.test("TC5\_ContainsString\_45\_Tuan", *function*(){

    pm.expect(pm.response.text()).to.include("trangnguyen@gmail.com")

});

//Lấy dữ liệu trả về

*let* jsonData\_45\_Tuan = pm.response.json();

//TC6\_Type\_45\_Tuan: Kiểm tra kiểu dữ liệu trả về của first\_name là string

pm.test("TC6\_Type\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan.first\_name).to.be.a("string")

});

//TC7\_ID\_45\_Tuan: Kiểm trả giá trị trả về của id

pm.test("TC7\_ID\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan.id).to.equal("1")

});

//TC8\_Math\_45\_Tuan: Kiểm trả kiểu dữ liệu trả về của math là number

pm.test("TC8\_Math\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan.subject\_scores[0].math).to.be.a("number")

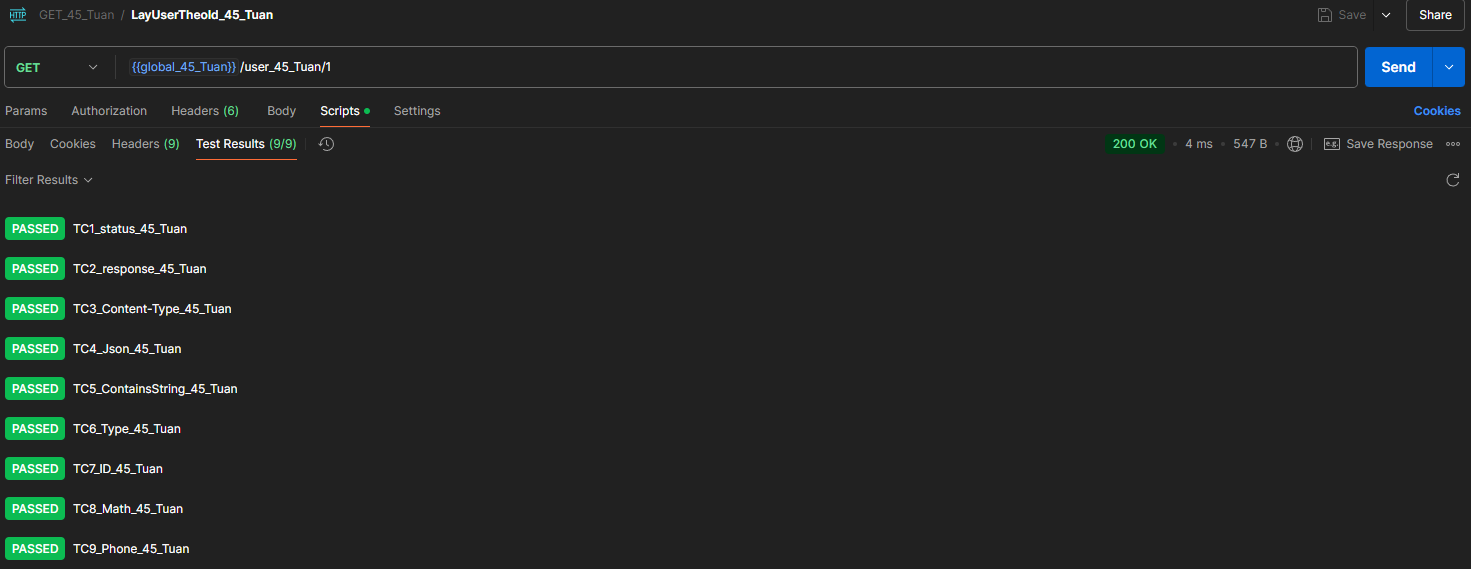
});

//TC9\_Phone\_45\_Tuan: Kiểm tra giá trị trả về của phone

pm.test("TC9\_Phone\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan.phone).to.equal("0768996543")

});



*Kết quả hiển thị*

Test Scrip cho phương thức GET lấy user theo tên

//TC1\_status\_45\_Tuan: Kiểm tra trạng thái phản hồi

pm.test("TC1\_status\_45\_Tuan", *function*(){

    pm.response.to.have.status(200);

});

//TC2\_response\_45\_Tuan: Kiểm tra thời gian phản hồi dưới 200ms

pm.test("TC2\_response\_45\_Tuan", *function*(){

    pm.expect(pm.response.responseTime).to.be.below(200);

});

//TC3\_Content-Type\_45\_Tuan: Kiểm tra header có thuộc tính Content-Type

pm.test("TC3\_Content-Type\_45\_Tuan", *function*(){

    pm.response.to.have.header("Content-Type")

})

//TC4\_Json\_45\_Tuan: Kiểm tra xem có JSON không

pm.test("TC4\_Json\_45\_Tuan", *function*(){

    pm.response.to.have.jsonBody()

});

//TC5\_ContainsString\_45\_Tuan: response có chứa chuỗi "trangnguyen@gmail.com" không

pm.test("TC5\_ContainsString\_45\_Tuan", *function*(){

    pm.expect(pm.response.text()).to.include("trangnguyen@gmail.com")

});

//Lấy dữ liệu trả về

*let* jsonData\_45\_Tuan = pm.response.json();

//TC6\_Type\_45\_Tuan: Kiểm tra kiểu dữ liệu trả về của first\_name là string

pm.test("TC6\_Type\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan[0].first\_name).to.be.a("string")

});

//TC7\_ID\_45\_Tuan: Kiểm trả giá trị trả về của id

pm.test("TC7\_ID\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan[0].id).to.equal("1")

});

//TC8\_Math\_45\_Tuan: Kiểm trả kiểu dữ liệu trả về của math là number

pm.test("TC8\_Math\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan[0].subject\_scores[0].math).to.be.a("number")

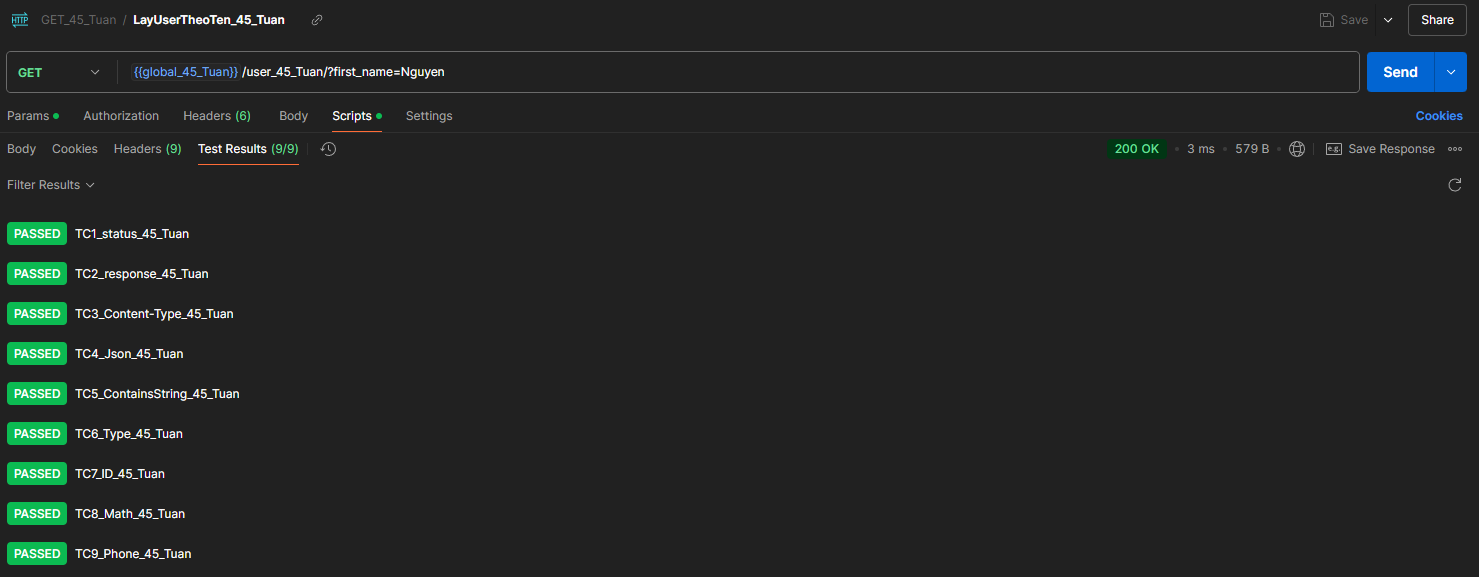
});

//TC9\_Phone\_45\_Tuan: Kiểm tra giá trị trả về của phone

pm.test("TC9\_Phone\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan[0].phone).to.equal("0768996543")

});



*Kết quả hiển thị*

## 2. PHƯƠNG THỨC POST

### 2.1. Đặc tả

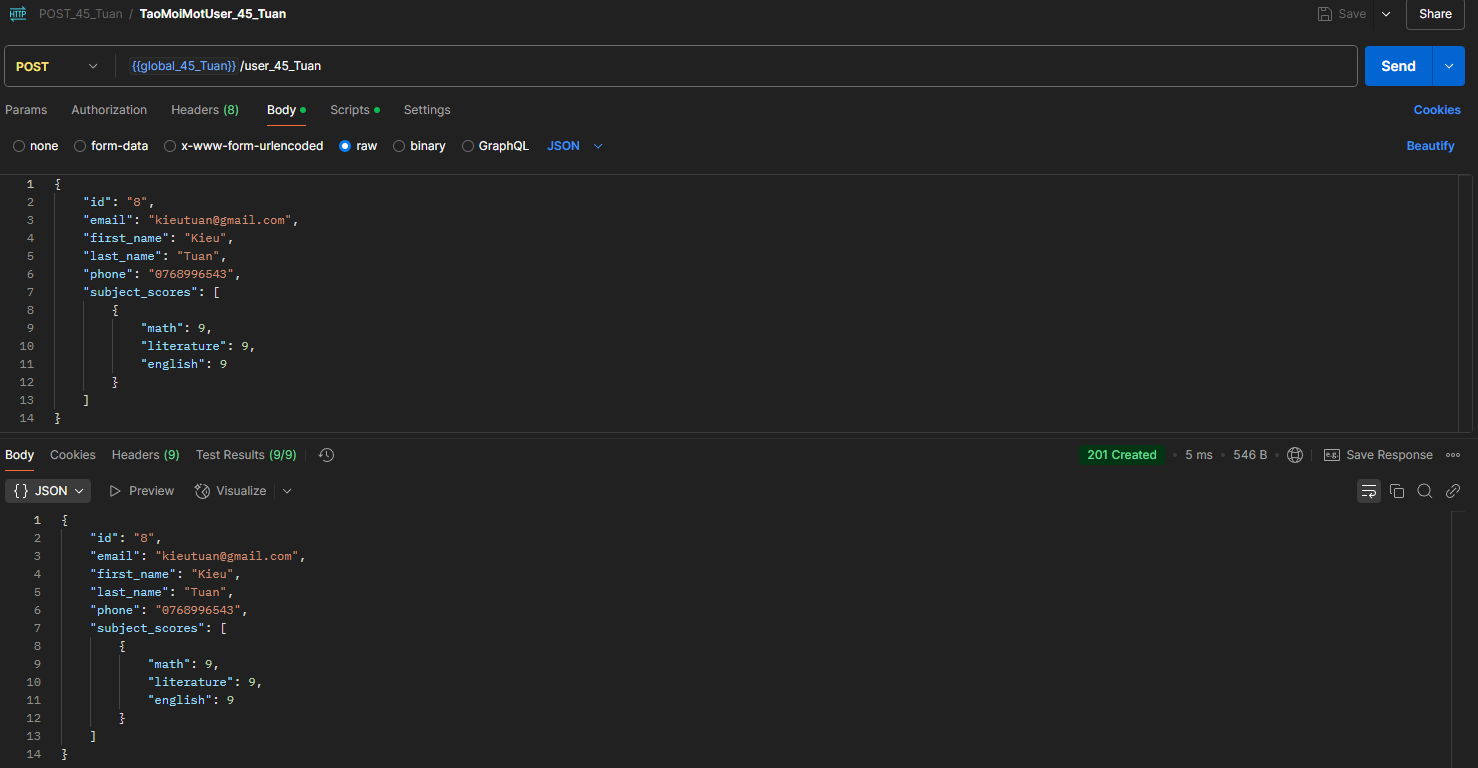
Phương thức POST hỗ trợ gửi dữ liệu lên máy chủ để tiến hành tạo mới tài nguyên.

### 2.2. Giao diện

Khởi tạo Collection mới và đặt tên là POST\_45\_Tuan



POST tạo mới một user {{global\_45\_Tuan}}/user\_45\_Tuan



### 2.3. Test Scrip

Test Scrip cho phương thức POST tạo mới một user

//Test dữ liệu được gửi lên API

*let* requestData\_45\_Tuan = JSON.parse(pm.request.body.raw);

//TC1\_NotNull\_45\_Tuan: dữ liệu không trống

pm.test("TC1\_NotNull\_45\_Tuan", *function*(){

    pm.expect(requestData\_45\_Tuan).to.not.be.empty

});

//TC2\_Conten\_45\_Tuan: Gửi lên API có đủ các trường dữ liệu

pm.test("TC2\_Conten\_45\_Tuan", *function*(){

    pm.expect(requestData\_45\_Tuan).to.have.all.keys("id", "email", "first\_name", "last\_name", "phone", "subject\_scores")

});

//TC3\_Request\_45\_Tuan: Dữ liệu gửi lên APIs

pm.test("TC3\_Request\_45\_Tuan", *function*(){

    pm.expect(requestData\_45\_Tuan.id).to.be.a("string")

    pm.expect(requestData\_45\_Tuan.email).to.be.a("string")

    pm.expect(requestData\_45\_Tuan.first\_name).to.be.a("string")

    pm.expect(requestData\_45\_Tuan.last\_name).to.be.a("string")

    pm.expect(requestData\_45\_Tuan.phone).to.be.a("string")

    pm.expect(requestData\_45\_Tuan.subject\_scores).to.be.a("array")

});

//TC4\_Math\_45\_Tuan: Kiểm tra math không âm

pm.test("TC4\_Math\_45\_Tuan", *function*(){

    pm.expect(requestData\_45\_Tuan.subject\_scores[0].math).to.be.above(0)

});

//TC5\_Status\_45\_Tuan: Kiểm tra trạng thái phản hồi

pm.test("TC5\_Status\_45\_Tuan", *function*(){

    pm.response.to.have.status(201)

});

//TC6\_Response\_45\_Tuan: Kiểm tra thời gian phản hồi dưới 200ms

pm.test("TC6\_Response\_45\_Tuan", *function*(){

    pm.expect(pm.response.responseTime).to.be.below(200)

});

//TC7\_Content\_45\_Tuan: Kiểm tra header có thuộc tính Content-Type

pm.test("TC7\_Content\_45\_Tuan", *function*(){

    pm.response.to.have.header("Content-Type")

})

//TC8\_Json\_45\_Tuan: Kiểm tra xem có JSON không

pm.test("TC8\_Json\_45\_Tuan", *function*(){

    pm.response.to.have.jsonBody()

});

//Test dữ liệu được trả về

*let* jsonData\_45\_Tuan = pm.response.json();

//TC7\_DataResponse\_45\_Tuan: Giá trị trả về giống giá trị gửi lên API

pm.test("TC7\_DataResponse\_45\_Tuan", *function*(){

*let* requestData\_45\_Tuan = JSON.parse(pm.request.body.raw);

    pm.expect(jsonData\_45\_Tuan.id).to.equal(requestData\_45\_Tuan.id)

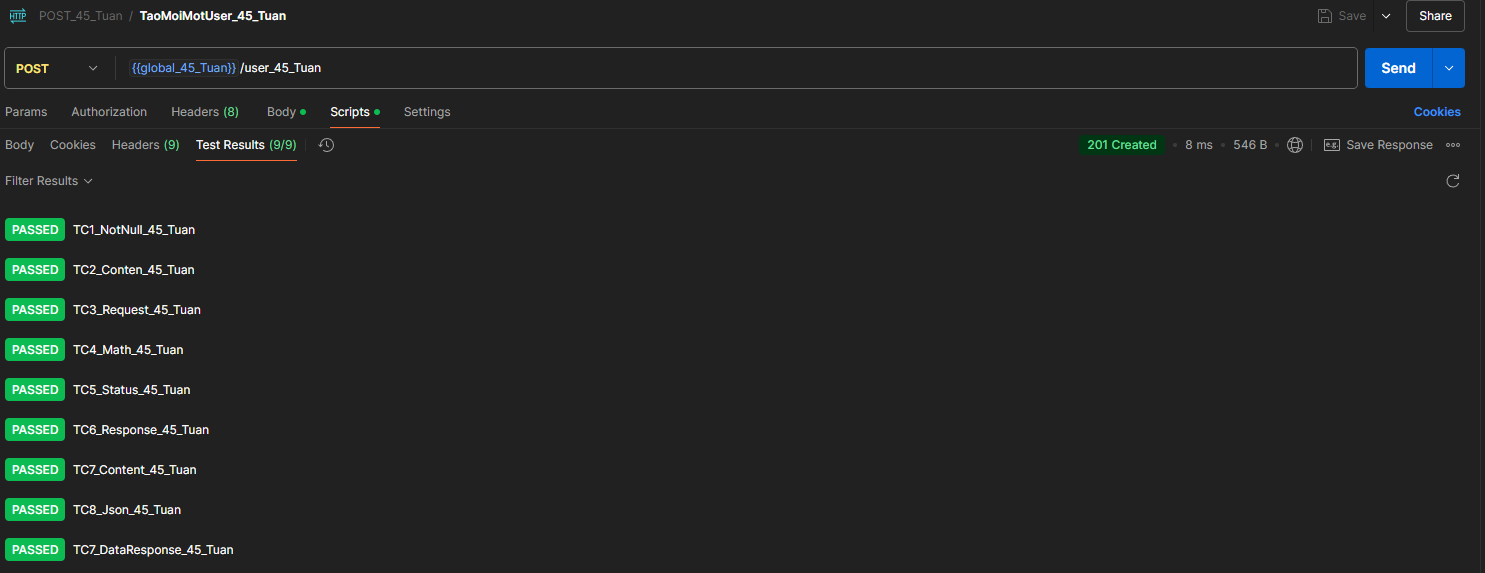
    pm.expect(jsonData\_45\_Tuan.email).to.equal(requestData\_45\_Tuan.email)

    pm.expect(jsonData\_45\_Tuan.first\_name).to.equal(requestData\_45\_Tuan.first\_name)

    pm.expect(jsonData\_45\_Tuan.last\_name).to.equal(requestData\_45\_Tuan.last\_name)

    pm.expect(jsonData\_45\_Tuan.phone).to.equal(requestData\_45\_Tuan.phone)

});



*Kết quả hiển thị*

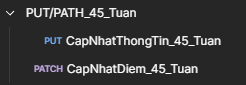
## 3. PHƯƠNG THỨC PUT/PATCH

### 3.1. Đặc tả

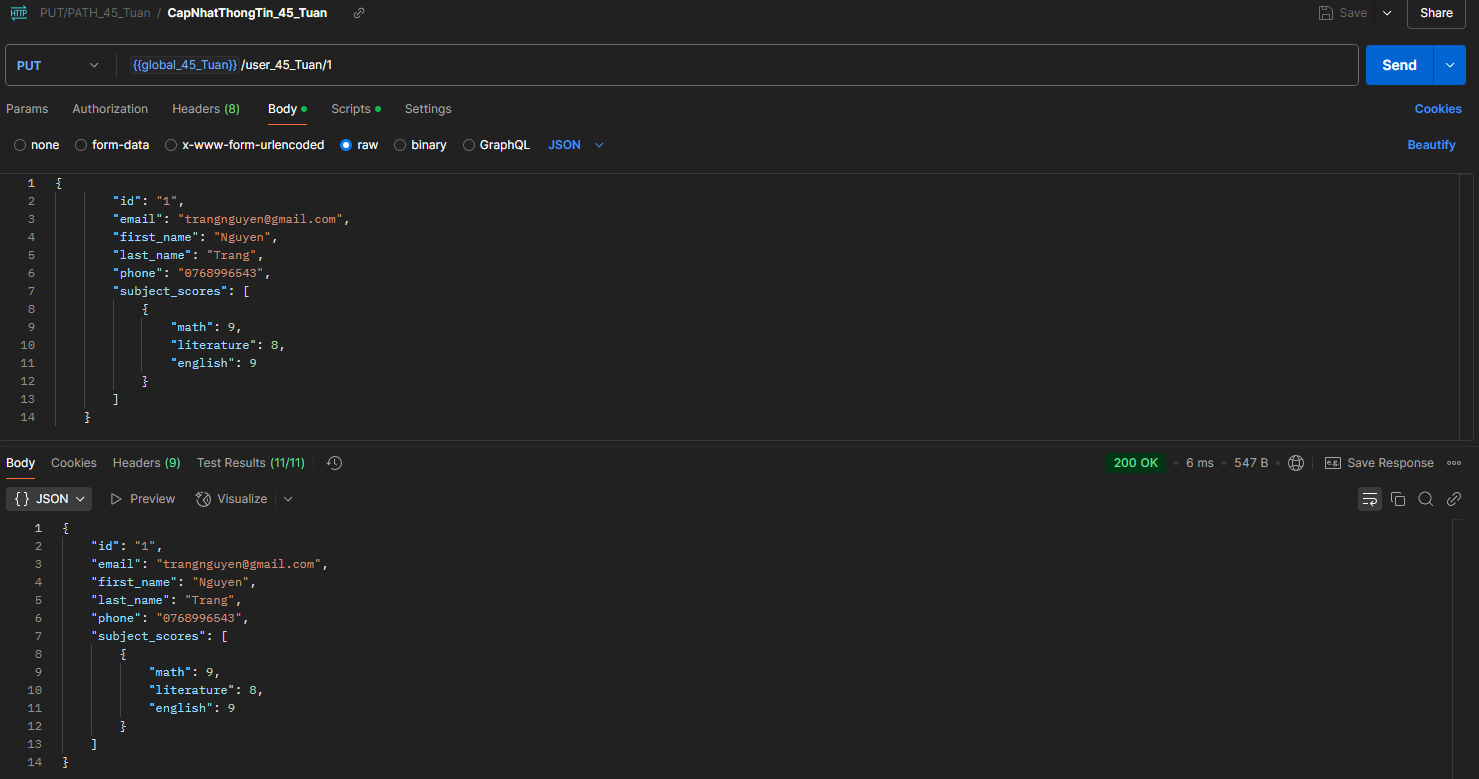
Phương thức PUT/PATCH hỗ trợ cập nhật một phần hoặc toàn bộ tài nguyên trên máy chủ.

### 3.2. Giao diện

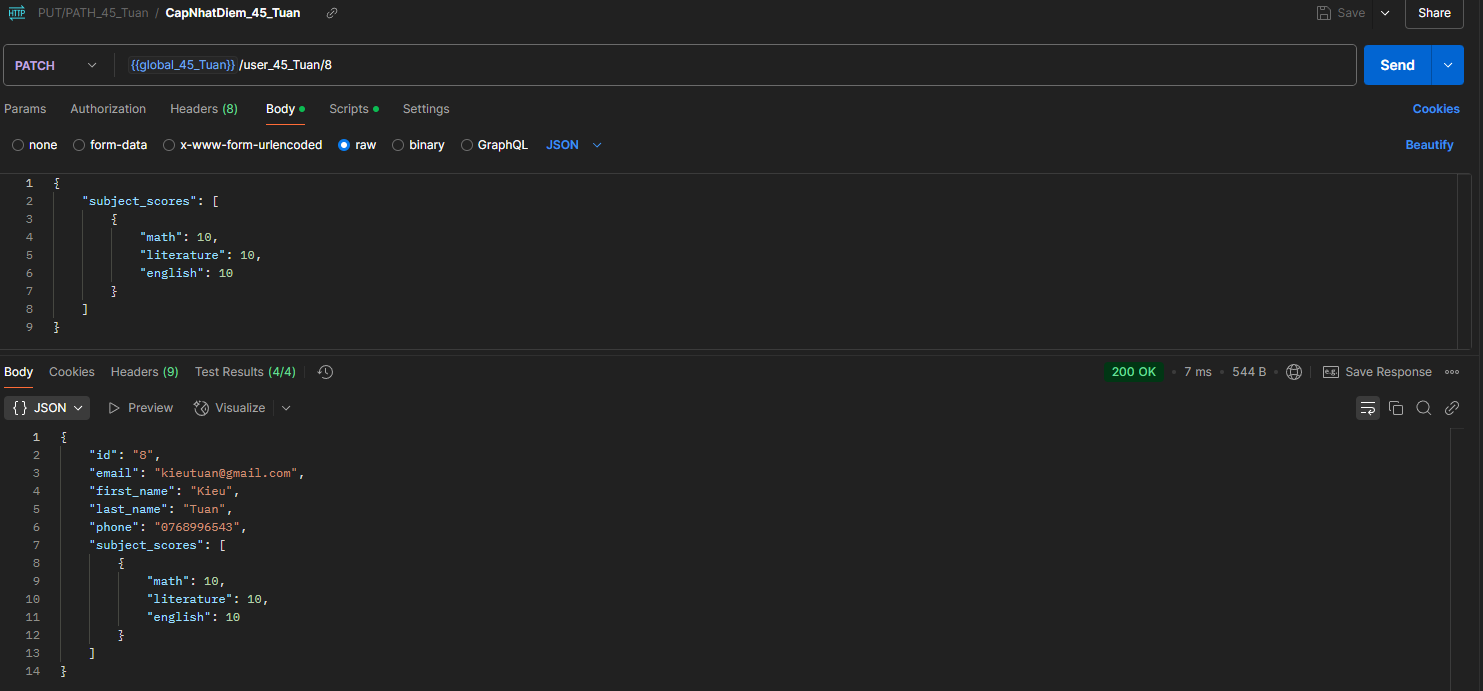
Khởi tạo Collection mới và đặt tên là PUT/PATCH\_45\_Tuan



PUT cập nhật toàn bộ thông tin của user {{global\_45\_Tuan}}/user\_45\_Tuan/1



PATCH cập nhật điểm của một user {{global\_45\_Tuan}}/user\_45\_Tuan/8



### 3.3. Test Scrip

Test Scrip cho phương thức PUT cập nhật toàn bộ thông tin của user

//Test dữ liệu được gửi lên API

*let* requestData\_45\_Tuan = JSON.parse(pm.request.body.raw);

//TC1\_first\_name\_45\_Tuan: Kiểm tra có first\_name không

pm.test("TC1\_first\_name\_45\_Tuan", *function*(){

    pm.expect(requestData\_45\_Tuan).to.have.property("first\_name")

});

//TC2\_RequestType\_45\_Tuan: Kiểm tra kiểu truyền có phải string

pm.test("TC2\_RequestType\_45\_Tuan", *function*(){

    pm.expect(requestData\_45\_Tuan.email).to.be.a("string")

});

//TC3\_RequestNotEmpty\_45\_Tuan: Kiểm tra giá trị truyền không rỗng

pm.test("TC3\_RequestNotEmpty\_45\_Tuan", *function*(){

    pm.expect(pm.request.body.raw).to.not.be.empty

});

//TC4\_Suitable\_45\_Tuan: Kiểm tra English truyền không âm

pm.test("TC4\_Suitable\_45\_Tuan", *function*(){

    pm.expect(requestData\_45\_Tuan.subject\_scores[0].english).to.be.above(0)

});

//TC5\_status\_45\_Tuan: Kiểm tra trạng thái phản hồi

pm.test("TC5\_status\_45\_Tuan", *function*(){

    pm.response.to.have.status(200)

});

//TC6\_Response\_45\_Tuan: Kiểm tra thời gian phản hồi dưới 200ms

pm.test("TC6\_Response\_45\_Tuan", *function*(){

    pm.expect(pm.response.responseTime).to.be.below(200)

});

//TC7\_Content-Type\_45\_Tuan: Kiểm tra header có thuộc tính Content-Type

pm.test("TC7\_Content-Type\_45\_Tuan", *function*(){

    pm.response.to.have.header("Content-Type")

})

//TC8\_Json\_45\_Tuan: Kiểm tra xem có JSON không

pm.test("TC8\_Json\_45\_Tuan", *function*(){

    pm.response.to.have.jsonBody()

});

//Test dữ liệu được trả về

*let* jsonData\_45\_Tuan = pm.response.json();

//TC9\_first\_name\_45\_Tuan: Kiểm tra giá trị của first\_name

pm.test("TC9\_first\_name\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan.first\_name).to.equal("Nguyen")

});

//TC10\_ResponseType\_45\_Tuan: Kiểm tra kiểu dữ liệu trả về là string

pm.test("TC10\_ResponseType\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan.id).to.be.a("string")

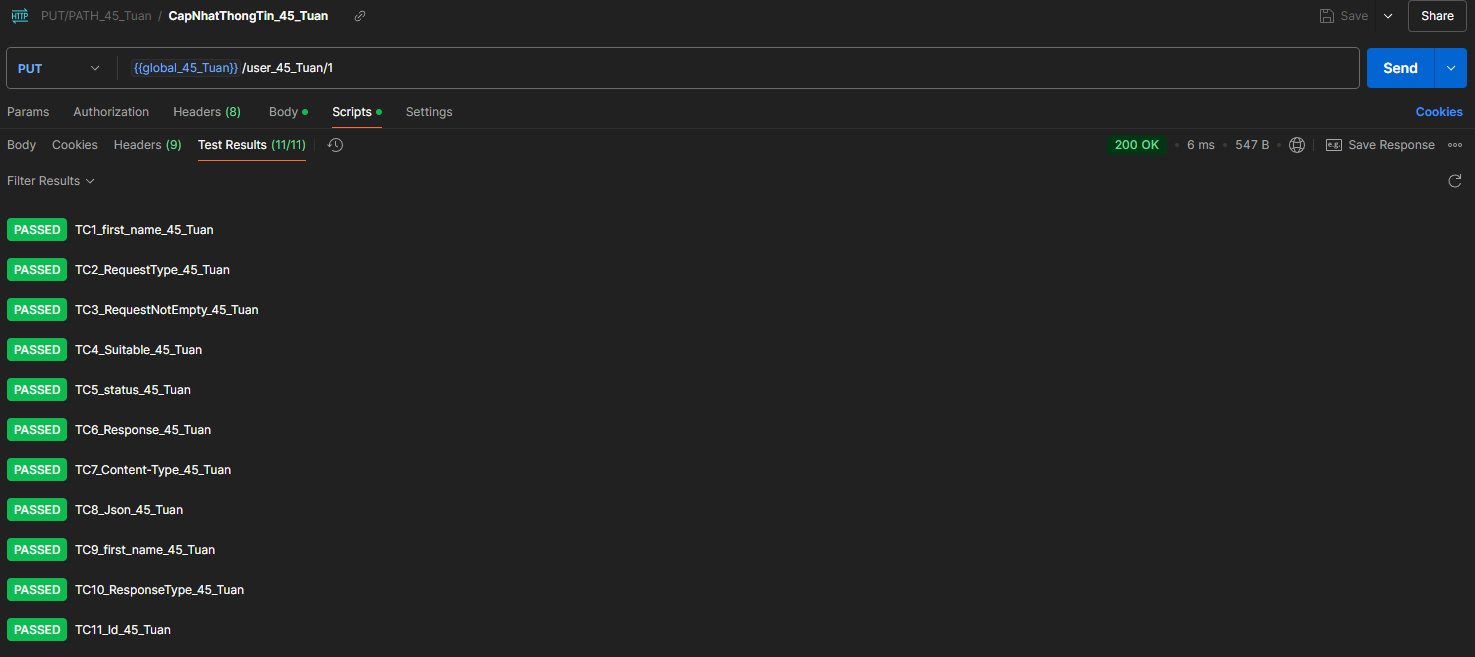
});

//TC11\_Id\_45\_Tuan: Kiểm tra id trả về không thay đổi

pm.test("TC11\_Id\_45\_Tuan", *function*(){

    pm.expect(jsonData\_45\_Tuan.id).to.equal("1")

});



*Kết quả hiển thị*

Test Scrip cho phương thức PATCH cập nhật điểm của một user

// Test dữ liệu được gửi lên API

*let* requestData\_45\_Tuan = JSON.parse(pm.request.body.raw);

// TC3\_RequestNotEmpty\_45\_Tuan: Kiểm tra giá trị truyền không rỗng

pm.test("TC3\_RequestNotEmpty\_45\_Tuan", *function*(){

    pm.expect(pm.request.body.raw).to.not.be.empty

});

// TC4\_Suitable\_45\_Tuan: Kiểm tra math truyền không âm

pm.test("TC4\_Suitable\_45\_Tuan", *function*(){

    pm.expect(requestData\_45\_Tuan.subject\_scores[0].math).to.be.above(0)

});

// TC5\_Suitable\_45\_Tuan: Kiểm tra literature truyền không âm

pm.test("TC5\_Suitable\_45\_Tuan", *function*(){

    pm.expect(requestData\_45\_Tuan.subject\_scores[0].literature).to.be.above(0)

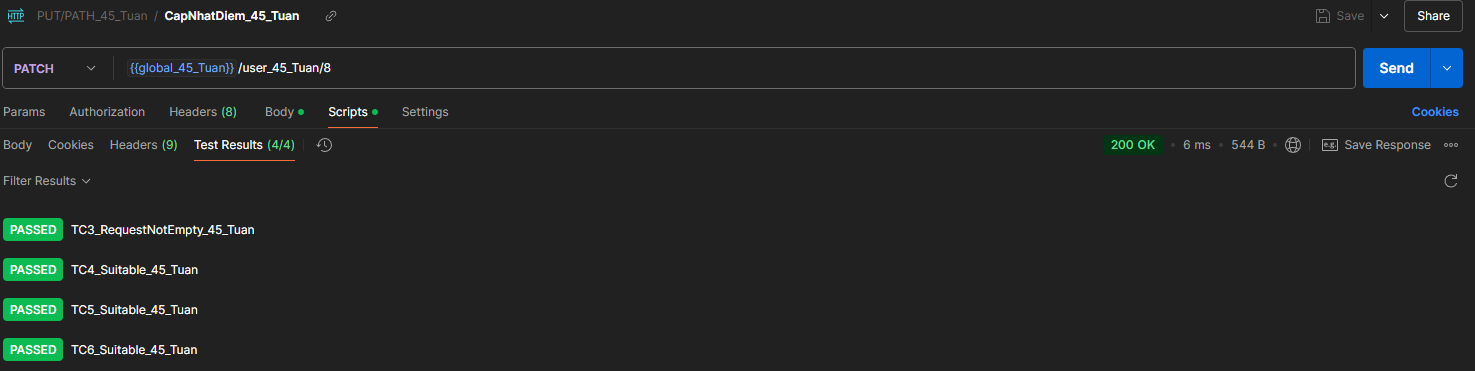
});

// TC6\_Suitable\_45\_Tuan: Kiểm tra english truyền không âm

pm.test("TC6\_Suitable\_45\_Tuan", *function*(){

    pm.expect(requestData\_45\_Tuan.subject\_scores[0].english).to.be.above(0)

});



*Kết quả hiển thị*

## 4. PHƯƠNG THỨC DELETE

### 4.1. Đặc tả

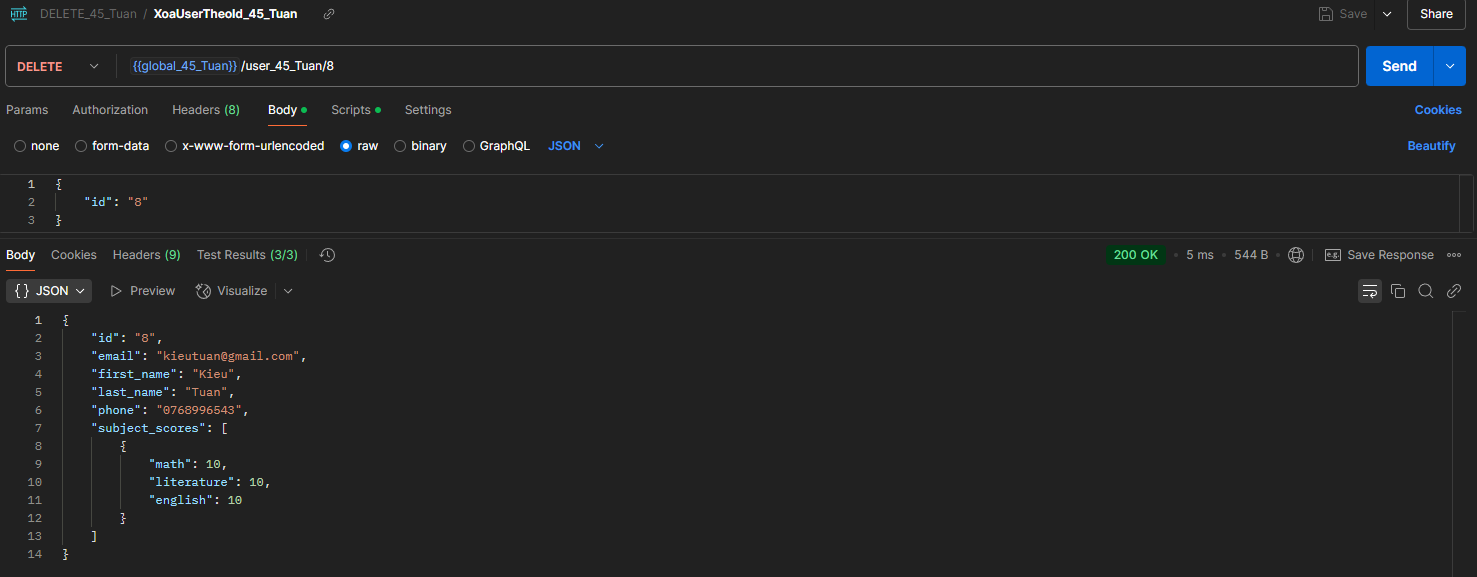
Phương thức DELETE hỗ trợ xóa một tài nguyên trên máy chủ.

### 4.2. Giao diện

Khởi tạo Collection mới và đặt tên là DELETE\_45\_Tuan



DELETE xóa user dựa theo Id {{global\_45\_Tuan}}/user\_45\_Tuan/8



### 4.3. Test Scrip

Test Scrip cho phương thức DELETE xóa user dựa theo Id

//TC1\_status\_45\_Tuan: Kiểm tra trạng thái phản hồi

pm.test("TC1\_status\_45\_Tuan", *function*(){

    pm.response.to.have.status(200);

});

//TC2\_Response\_45\_Tuan: Kiểm tra thời gian phản hồi dưới 200ms

pm.test("TC2\_Response\_45\_Tuan", *function*(){

    pm.expect(pm.response.responseTime).to.be.below(200);

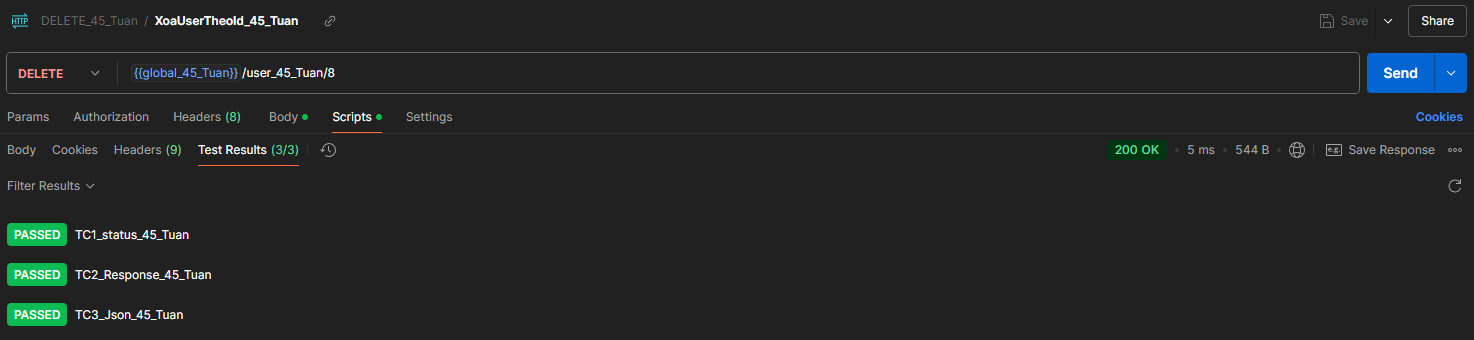
});

//TC3\_Json\_45\_Tuan: Kiểm tra xem có JSON không

pm.test("TC3\_Json\_45\_Tuan", *function*(){

    pm.response.to.have.jsonBody();

});



*Kết quả hiển thị*