1.

C# .NET NUnit Setup & Assertion Guide

1. Project Setup (Step-by-Step)

Option A: Using .NET CLI (Terminal / VS Code)

Open your terminal in your project folder and run these commands:

```
Bash

# 1. Create a solution folder
mkdir MyNUnitSolution
cd MyNUnitSolution

# 2. Create the Class Library (The code you want to test)
dotnet new classlib -n MyLibrary

# 3. Create the NUnit Test Project
dotnet new nunit -n MyLibrary.Tests

# 4. Create a Solution file and add both projects
dotnet new sln
dotnet sln add MyLibrary/MyLibrary.csproj
dotnet sln add MyLibrary.Tests/MyLibrary.Tests.csproj

# 5. Link the Test project to your Library
dotnet add MyLibrary.Tests/MyLibrary.Tests.csproj reference MyLibrary/MyLibrary.csproj
```

Option B: Using Visual Studio 2022 (GUI)

1. **New Project:** File > New > Project > **Blank Solution**.
2. **Add Library:** Right-click Solution > Add > New Project > **Class Library** (Name: `MyLibrary`).
3. **Add Test Project:** Right-click Solution > Add > New Project > Search for **"NUnit Test Project"** (Name: `MyLibrary.Tests`).
4. **Reference:** Right-click the `MyLibrary.Tests` project > **Add** > **Project Reference** > Check the box for `MyLibrary`.

---

2. Sample Class to Test

Create a file named `Calculator.cs` in your **MyLibrary** project:

```
C#

namespace MyLibrary;

public class Calculator
{
    public int Add(int a, int b) => a + b;

    public double Divide(double a, double b)
    {
        if (b == 0) throw new DivideByZeroException("Cannot divide by zero.");
        return a / b;
    }

    public string GetGreeting(string name) => $"Hello, {name}!";
}
```

---

3. Test Class with Assertion Options

Create a file named `CalculatorTests.cs` in your **MyLibrary.Tests** project. This demonstrates the "Constraint Model" (`Assert.That`).

```
C#

using NUnit.Framework;
using MyLibrary;

namespace MyLibrary.Tests;

[TestFixture]
public class CalculatorTests
{
    private Calculator _calc;

    [SetUp] // This runs before every test method
    public void Setup()
    {
        _calc = new Calculator();
    }
```

```
    [Test]
    public void All_Assertion_Examples()
    {
        // 1. Equality Assertions
        Assert.That(_calc.Add(2, 2), Is.EqualTo(4));
        Assert.That(_calc.Add(2, 2), Is.Not.EqualTo(5));

        // 2. String Assertions
        string msg = _calc.GetGreeting("Alice");
        Assert.That(msg, Does.Contain("Alice"));
        Assert.That(msg, Does.StartWith("Hello"));
        Assert.That(msg, Is.Not.Empty);

        // 3. Numeric Ranges & Precision
        Assert.That(_calc.Divide(10, 3), Is.EqualTo(3.33).Within(0.01));
        Assert.That(10, Is.GreaterThan(5));
        Assert.That(10, Is.InRange(1, 100));

        // 4. Boolean & Nulls
        Assert.That(true, Is.True);
        Assert.That(null, Is.Null);

        // 5. Collection Assertions
        var list = new List<int> { 1, 2, 3 };
        Assert.That(list, Has.Exactly(3).Items);
        Assert.That(list, Contains.Item(2));
        Assert.That(list, Is.Unique);

        // 6. Exception Assertions
        Assert.Throws<DivideByZeroException>(() => _calc.Divide(10, 0));
    }

    // 7. Parameterized Tests (Running the same test with different data)
    [TestCase(1, 2, 3)]
    [TestCase(-1, 1, 0)]
    [TestCase(100, 200, 300)]
    public void Add_MultipleInputs_ReturnsExpected(int a, int b, int expected)
    {
        Assert.That(_calc.Add(a, b), Is.EqualTo(expected));
    }
}
```

---

4. How to Run Your Tests

- **Terminal/VS Code:** Type `dotnet test`
- **Visual Studio:** Open **Test Explorer** (Test > Test Explorer) and click **Run All**.
- **NuGet Commands:** (If needed manually)
    - `dotnet add package NUnit`
    - `dotnet add package NUnit3TestAdapter`
    - `dotnet add package Microsoft.NET.Test.Sdk`