**TDD** stands for **Test-Driven Development**, and it's a **software development approach** where you write tests **before** writing the actual code.

**The TDD Cycle: “Red → Green → Refactor”**

**1. Red**

Write a test for a small piece of functionality — it should fail because the code isn’t written yet.

**2. Green**

Write the minimum code required to make the test pass.

**3. Refactor**

Clean up the code while keeping the test green.

Repeat this cycle for every new feature or change.

**🔁 Example: TDD in Action (C#)**

Let’s say you want to write a method to add two numbers.

**Step 1: Write the test (Red)**

[TestClass]

public class CalculatorTests

{

[TestMethod]

public void Add\_TwoNumbers\_ReturnsSum()

{

var calculator = new Calculator();

int result = calculator.Add(2, 3);

Assert.AreEqual(5, result); // Fails, Add() doesn't exist yet

}

}

**Step 2: Write the code (Green)**

public class Calculator

{

public int Add(int a, int b)

{

return a + b;

}

}

Test passes ✅

**Step 3: Refactor (if needed)**

Code is already clean? Move on.  
Otherwise, optimize or generalize the logic.

**✅ Benefits of TDD**

* Catches bugs early 🔍
* Ensures your code is testable and modular
* Improves design and structure
* Acts as documentation for future devs
* Gives confidence during refactoring

**⚠️ Common Myths**

❌ TDD takes too much time  
✅ It saves time in the long run by reducing bugs and rework.

❌ TDD is only for testing  
✅ TDD is about **designing software** through testing.

**🔧 Tools for TDD in .NET (C#)**

* **xUnit**
* **NUnit**
* **MSTest** (comes with Visual Studio)
* Mocking: **Moq**, **NSubstitute**

**🧱 Step-by-Step: Create a TDD Project in Visual Studio**

**✅ 1. Create the Main Project**

1. Open **Visual Studio**.
2. Click on **Create a new project**.
3. Choose **ASP.NET Core Web API** (or **Class Library** if you’re not building an API yet).
4. Name it something like MyTddApp.
5. Click **Create**.

**✅ 2. Add a Unit Test Project**

1. **Right-click** on the solution → **Add** → **New Project**.
2. Search for xUnit Test Project (or NUnit Test Project / MSTest Test Project).
3. Name it something like MyTddApp.Tests.
4. Click **Create**.

**✅ 3. Reference the Main Project in the Test Project**

1. Right-click on the MyTddApp.Tests project → **Add** → **Project Reference**.
2. Check MyTddApp → Click **OK**.

Now your test project can access your main project classes.

**✅ 4. Install Required NuGet Packages (if not already there)**

Go to **Tools → NuGet Package Manager → Manage NuGet Packages for Solution**.

* For xUnit:
  + xunit
  + xunit.runner.visualstudio
  + Microsoft.NET.Test.Sdk
* For NUnit:
  + NUnit
  + NUnit3TestAdapter

✔️ Install them in your test project.

**✅ 5. Write Your First Test (Red)**

Inside MyTddApp.Tests project:

using Xunit;

using MyTddApp;

public class CalculatorTests

{

[Fact]

public void Add\_TwoNumbers\_ReturnsSum()

{

var calc = new Calculator();

var result = calc.Add(2, 3);

Assert.Equal(5, result); // Test will fail until Add() is implemented

}

}

**✅ 6. Now Write the Actual Code (Green)**

In MyTddApp project:

public class Calculator

{

public int Add(int a, int b)

{

return a + b;

}

}

**✅ 7. Run Tests**

* Go to **Test → Run All Tests**
* Or open **Test Explorer** (Ctrl + E, T)
* You should see your test **pass** ✅

**🧠 TDD Cycle Reminder**

**Write a failing test → Write just enough code to pass → Refactor → Repeat**