Week 2 - NUnit Hands-On

Name: Pranjal Yadav

Roll Number: 22052918

# Objective Overview

- Understand Unit Testing and its difference from Functional Testing.

- Explore types of testing: Unit, Functional, Automated, Performance.

- Learn benefits of Automated Testing.

- Understand loosely coupled & testable design.

- Implement a testable calculator logic.

- Learn NUnit attributes: [TestFixture], [SetUp], [TearDown], [Test], [TestCase], [Ignore].

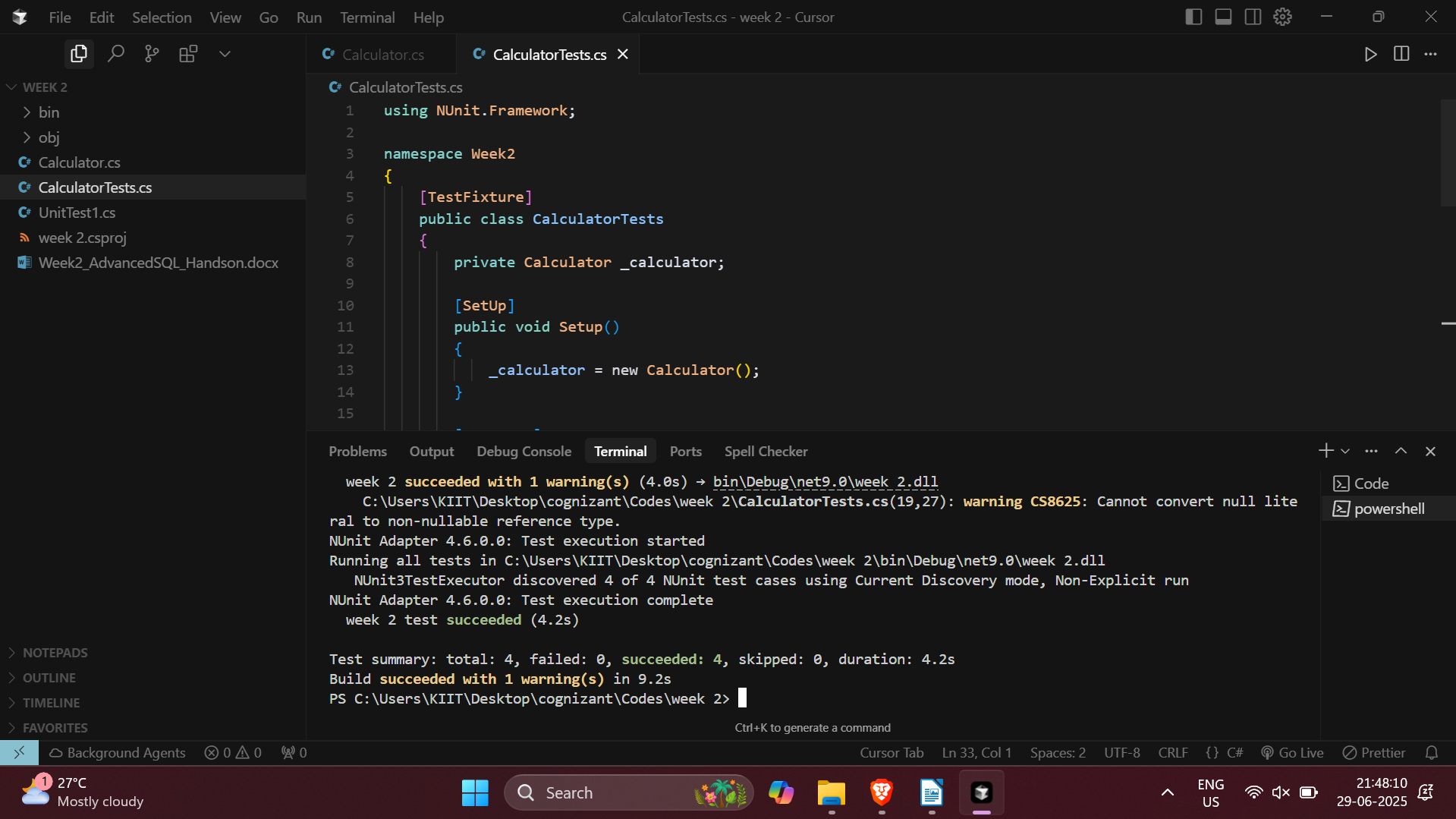
# Calculator Logic Code (Calculator.cs)

namespace Week2  
{  
 public class Calculator  
 {  
 public int Add(int a, int b) => a + b;  
 public int Subtract(int a, int b) => a - b;  
 public int Multiply(int a, int b) => a \* b;  
 public int Divide(int a, int b) => b != 0 ? a / b : 0;  
 }  
}

# Unit Test Code (CalculatorTests.cs)

using NUnit.Framework;  
  
namespace Week2  
{  
 [TestFixture]  
 public class CalculatorTests  
 {  
 private Calculator \_calculator;  
  
 [SetUp]  
 public void Setup()  
 {  
 \_calculator = new Calculator();  
 }  
  
 [TearDown]  
 public void Teardown()  
 {  
 \_calculator = null;  
 }  
  
 [TestCase(3, 5, 8)]  
 [TestCase(10, 20, 30)]  
 [TestCase(-5, 5, 0)]  
 public void Add\_WhenCalled\_ReturnsExpectedResult(int a, int b, int expected)  
 {  
 var result = \_calculator.Add(a, b);  
 Assert.That(result, Is.EqualTo(expected));  
 }  
  
 [Test]  
 public void Subtract\_ReturnsCorrectResult()  
 {  
 Assert.That(\_calculator.Subtract(10, 5), Is.EqualTo(5));  
 }  
  
 [Test]  
 public void Multiply\_ReturnsCorrectResult()  
 {  
 Assert.That(\_calculator.Multiply(3, 4), Is.EqualTo(12));  
 }  
  
 [Test]  
 public void Divide\_ReturnsCorrectResult()  
 {  
 Assert.That(\_calculator.Divide(10, 2), Is.EqualTo(5));  
 }  
 }  
}

# Test Execution Screenshot



All NUnit test cases executed successfully.

# Week 2 - Moq Hands-On

## IMailSender.cs

namespace CustomerCommLib  
{  
 public interface IMailSender  
 {  
 bool SendMail(string toAddress, string message);  
 }  
}

## MailSender.cs

using System.Net;  
using System.Net.Mail;  
  
namespace CustomerCommLib  
{  
 public class MailSender : IMailSender  
 {  
 public bool SendMail(string toAddress, string message)  
 {  
 MailMessage mail = new MailMessage();  
 SmtpClient smtpServer = new SmtpClient("smtp.gmail.com");  
  
 mail.From = new MailAddress("your\_email\_address@gmail.com");  
 mail.To.Add(toAddress);  
 mail.Subject = "Test Mail";  
 mail.Body = message;  
  
 smtpServer.Port = 587;  
 smtpServer.Credentials = new NetworkCredential("username", "password");  
 smtpServer.EnableSsl = true;  
  
 smtpServer.Send(mail);  
 return true;  
 }  
 }  
}

## CustomerComm.cs

namespace CustomerCommLib  
{  
 public class CustomerComm  
 {  
 private IMailSender \_mailSender;  
  
 public CustomerComm(IMailSender mailSender)  
 {  
 \_mailSender = mailSender;  
 }  
  
 public bool SendMailToCustomer()  
 {  
 return \_mailSender.SendMail("cust123@abc.com", "Some Message");  
 }  
 }  
}

## CustomerCommTests.cs

using Moq;  
using NUnit.Framework;  
using CustomerCommLib;  
  
namespace MoqHandsOn  
{  
 [TestFixture]  
 public class CustomerCommTests  
 {  
 private Mock<IMailSender> \_mockMailSender;  
 private CustomerComm \_customerComm;  
  
 [OneTimeSetUp]  
 public void Init()  
 {  
 \_mockMailSender = new Mock<IMailSender>();  
 \_mockMailSender.Setup(m => m.SendMail(It.IsAny<string>(), It.IsAny<string>())).Returns(true);  
 \_customerComm = new CustomerComm(\_mockMailSender.Object);  
 }  
  
 [Test]  
 public void SendMailToCustomer\_ShouldReturnTrue()  
 {  
 var result = \_customerComm.SendMailToCustomer();  
 Assert.That(result, Is.True);  
 }  
 }  
}

## Test Execution Screenshot

