**Hands on Day -2**

**INTCDE21ID008**

**C N S Varun**

**916214**

**Hands On-1**

**Unit Testing Code:**

using NUnit.Framework;

using CollectionsLib;

using System.Linq;

using System;

namespace D2Handson\_1

{

[TestFixture]

public class Tests

{

EmployeeManager e;

[SetUp]

public void Setup()

{

e = new EmployeeManager();

var l1 = e.GetEmployees();

}

[TestCase]

public void Test1()

{

var containnull = e.GetEmployees().Contains(null);

Assert.AreEqual(containnull, false);

Console.WriteLine(containnull);

}

[TestCase]

public void Test2()

{

var l1 = e.GetEmployees();

int e100 = l1.Where(e => e.EmpId == 100).Count();

Assert.That(e100, Is.EqualTo(1));

Console.WriteLine(e100);

}

[TestCase]

public void Test3()

{

var l1 = e.GetEmployees();

var unique = l1.Distinct().Count();

Assert.That(unique, Is.EqualTo(l1.Count));

Console.WriteLine(unique);

}

[TestCase]

public void Test4()

{

var ge = e.GetEmployees();

var gev = e.GetEmployeesWhoJoinedInPreviousYears();

Assert.That(ge, Is.EquivalentTo(gev));

foreach (var i in gev)

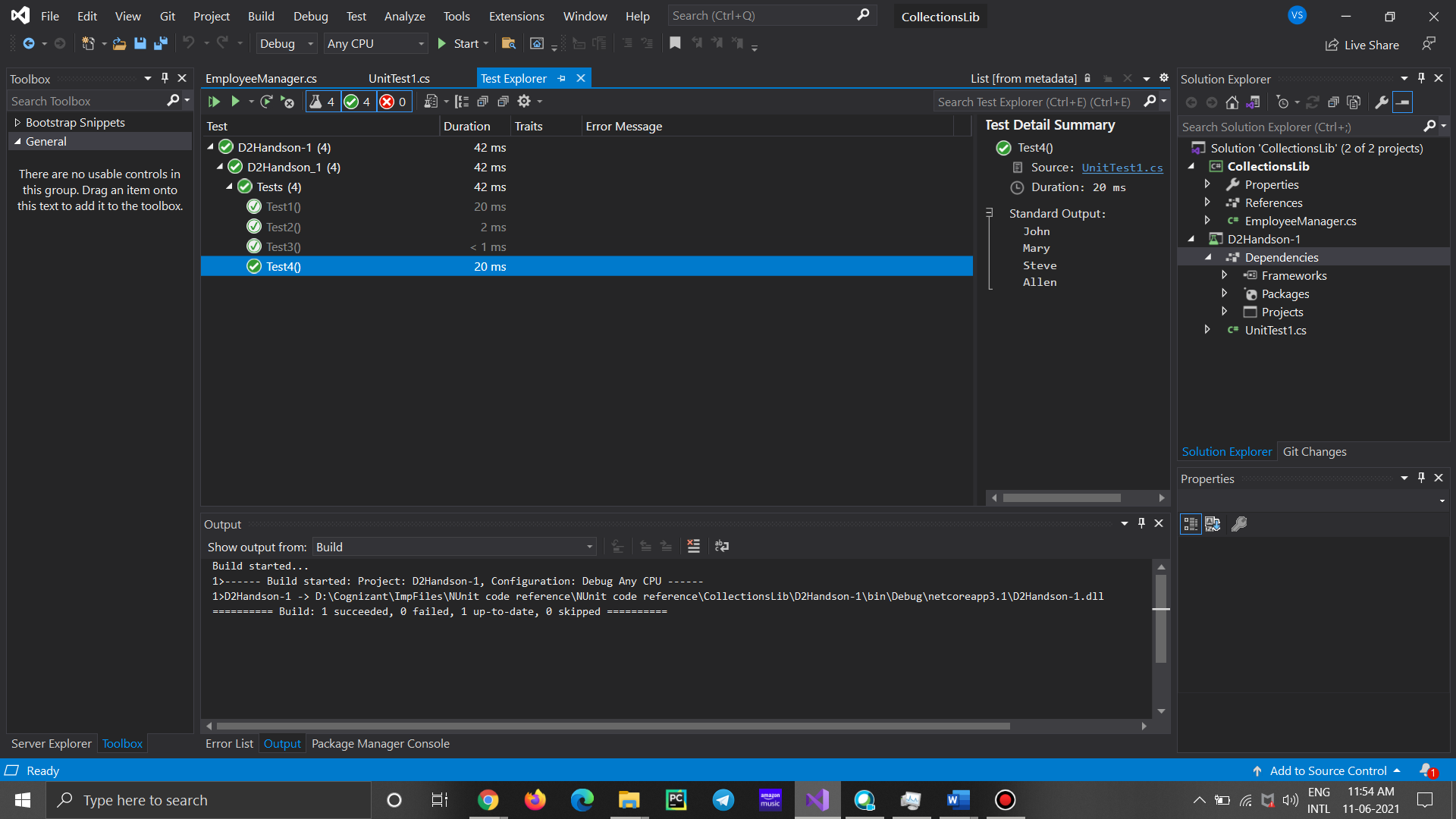
Console.WriteLine(i.EmpName);

}

}

}

**Test Explorer**



**Handson-2**

**Unit Testing Code:**

using NUnit.Framework;

using SeasonsLib;

namespace TestProject1

{

public class Test

{

[Test]

[TestCase("May","Summer")]

public void Test1(string a, string b)

{

SeasonTeller st = new SeasonTeller();

Assert.That(b, Is.EqualTo(st.DisplaySeasonBy(a)));

}

[TestCase("July", "Summer")]

public void Test2(string a, string b)

{

SeasonTeller st = new SeasonTeller();

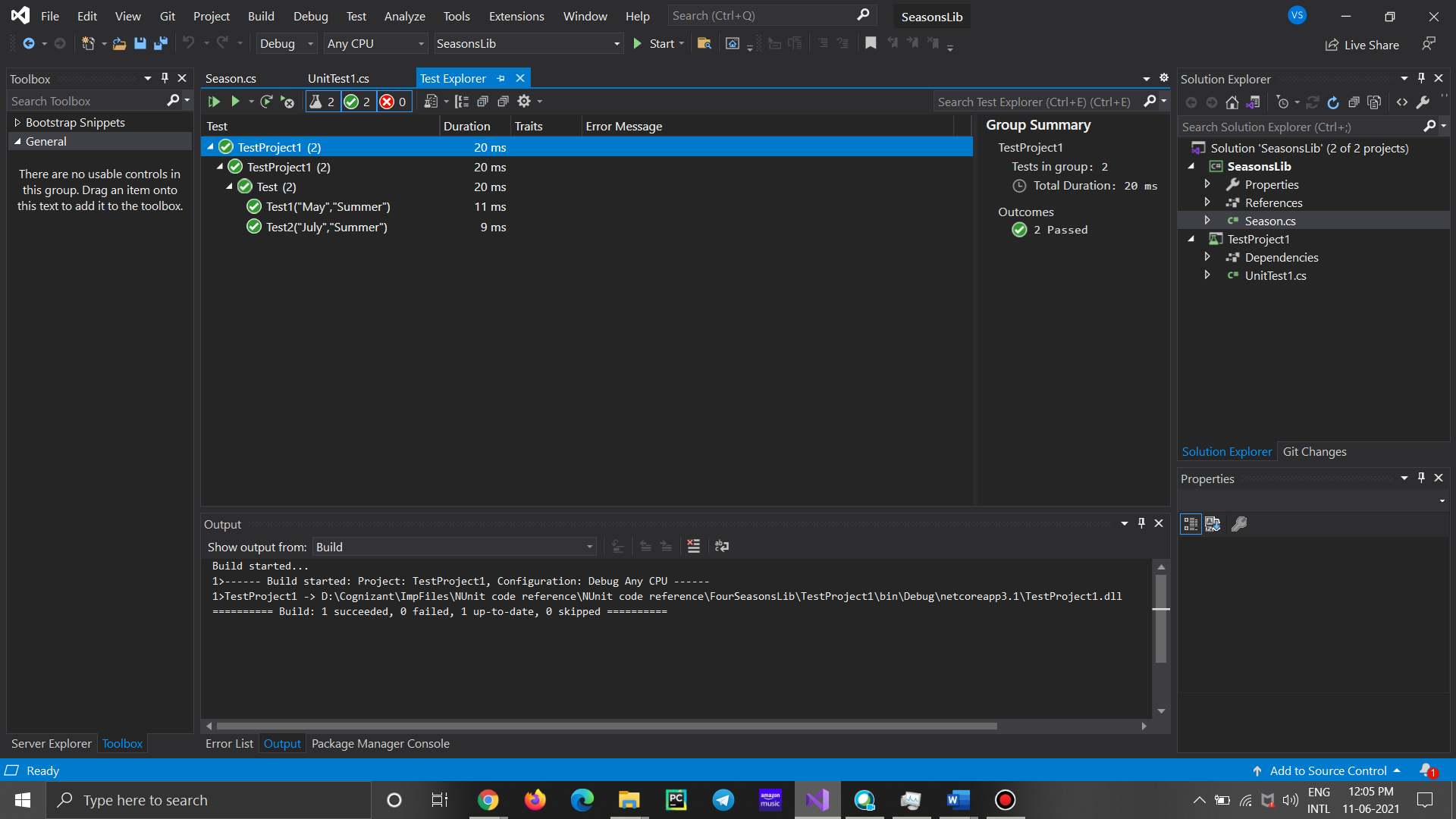
Assert.AreNotEqual(b, Is.EqualTo(st.DisplaySeasonBy(a)));

}

}

}

**Test Explorer**



**Handson-3**

**Unit Testing Code:**

using NUnit.Framework;

using LeapYearCalculatorLib;

namespace D2Handon3

{

public class SUT

{

LeapYearCalculator l;

[SetUp]

public void Setup()

{

l = new LeapYearCalculator();

}

[Test]

[TestCase(10000, -1)]

[TestCase(2004, 1)]

[TestCase(2000, 1)]

[TestCase(1995,0)]

public void Check(int ay,int e)

{

int result = l.IsLeapYear(ay);

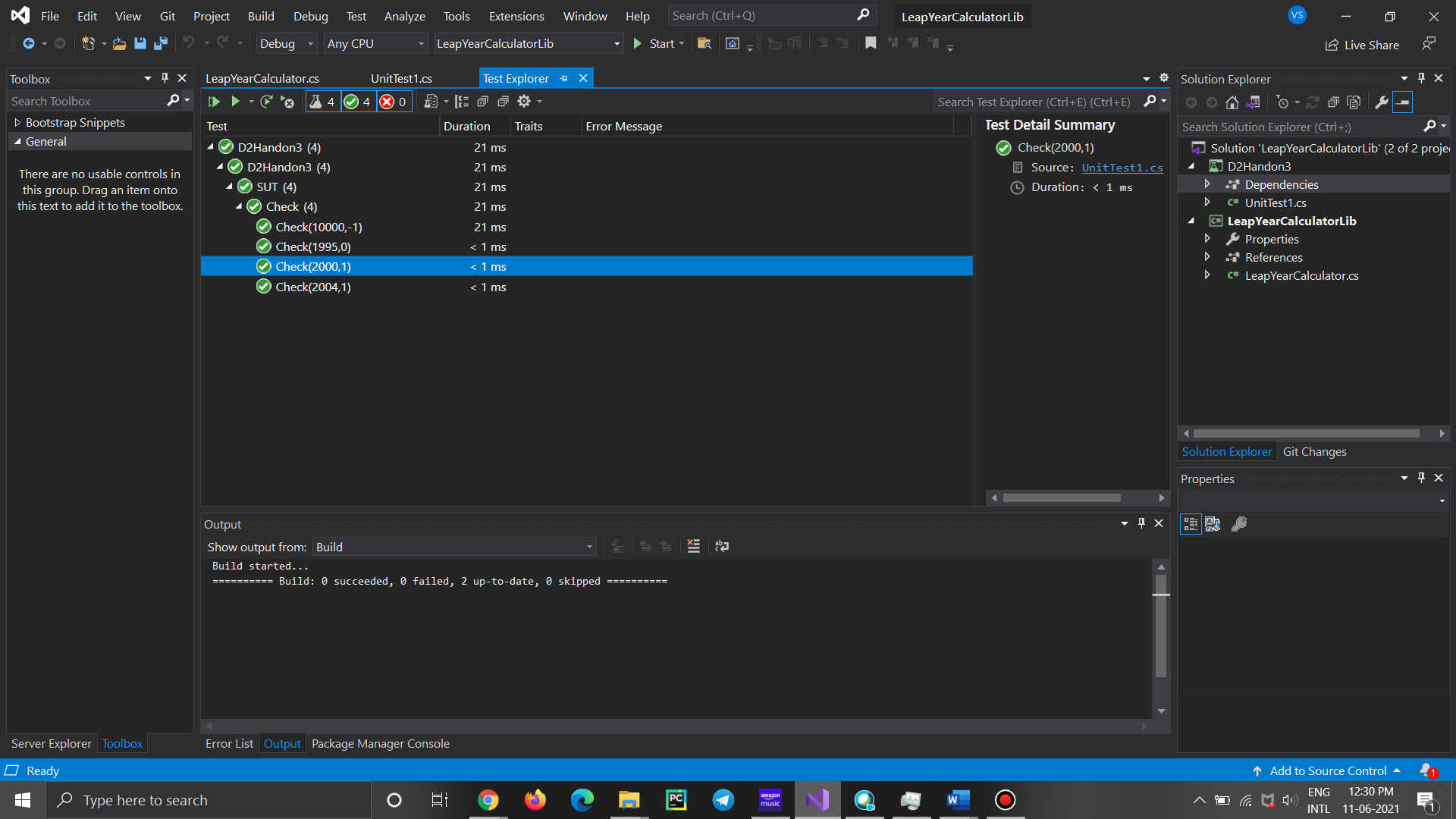
Assert.AreEqual(result,e);

}

}

}

**Test Explorer:**



**Handson-4**

**Unit Testing Code**

using NUnit.Framework;

using UserManagerLib;

using System;

namespace D2Handson4

{

[TestFixture]

public class SUT

{

User ff;

[SetUp]

public void Setup()

{

ff = new User();

}

[Test]

[TestCase("0010010010")]

[TestCase("ABCANCNVV1")]

[TestCase("12121")]

public void Test1(string a)

{

try

{

ff.CreateUser(new User { PANCardNo = a });

}

catch (NullReferenceException e)

{

Assert.Fail(e.Message);

Console.WriteLine(e.Message);

}

catch (FormatException e)

{

Assert.Fail(e.Message);

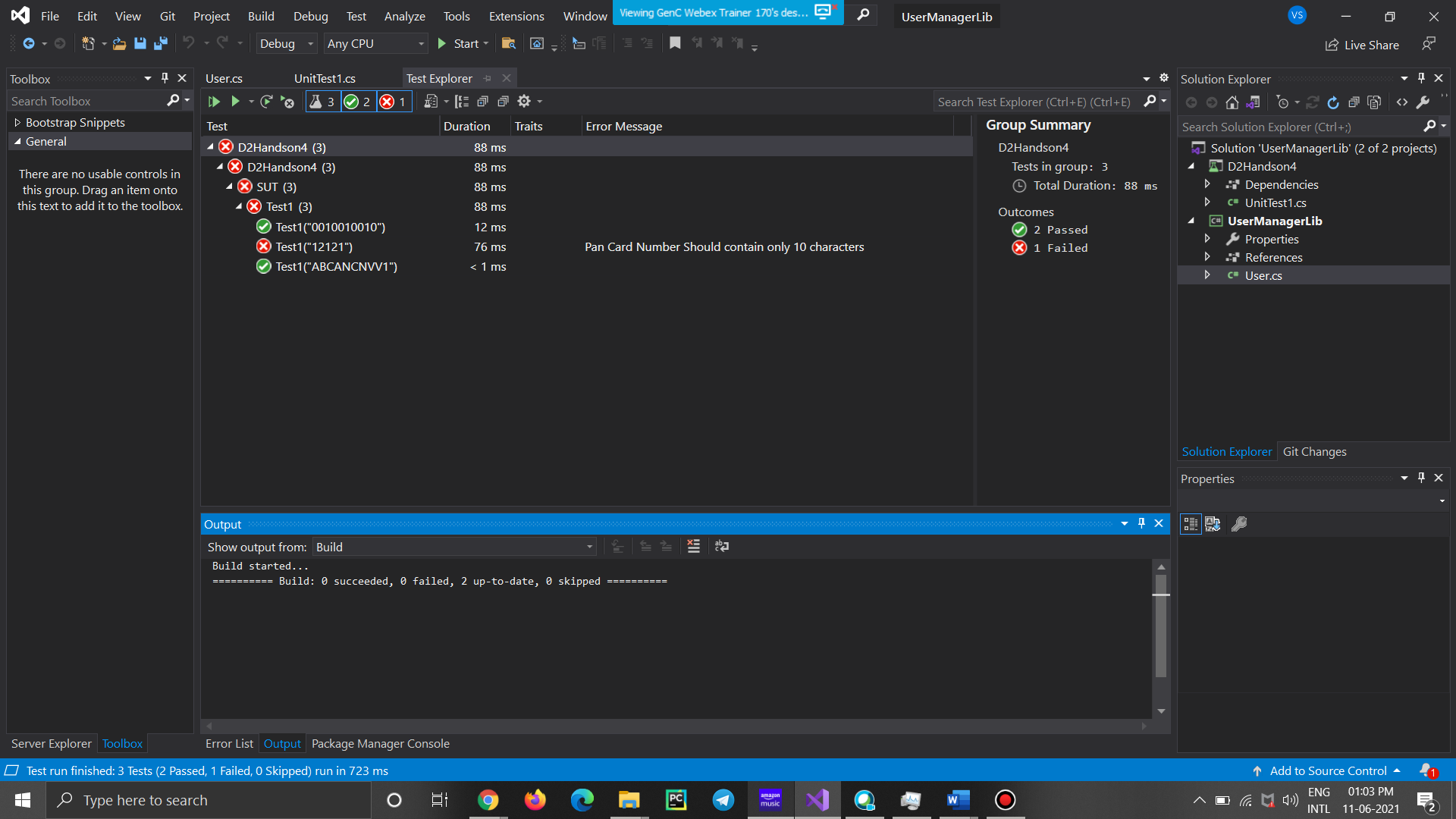
}

}

}

}

**Test Explorer**



**Handson-5**

**Unit Testing Code**

using NUnit.Framework;

using ConverterLib;

using System;

using CurrencyConverterApp;

using Moq;

namespace D2Handson5

{

[TestFixture]

public class SUTest

{

Mock<IDollarToEuroExchangeRateFeed> e;

[SetUp]

public void Setup()

{

e = new Mock<IDollarToEuroExchangeRateFeed>();

}

[Test]

public void Test1()

{

double u = 1.5;

double a = 1000;

e.Setup(t => t.GetActualUSDollarValue()).Returns(u);

Converter cv = new Converter(e.Object);

var euro = cv.USDToEuro(a);

Assert.AreEqual(euro, 1500);

}

}

}

**Test Explorer**

