**NUnit and Mocking Framework - Day 2**

**Handson 1:**

Steps to be followed:

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the UtilLib project to the test project.

4) Additionally add the reference of both NUnit and NUnit3TestAdapter in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

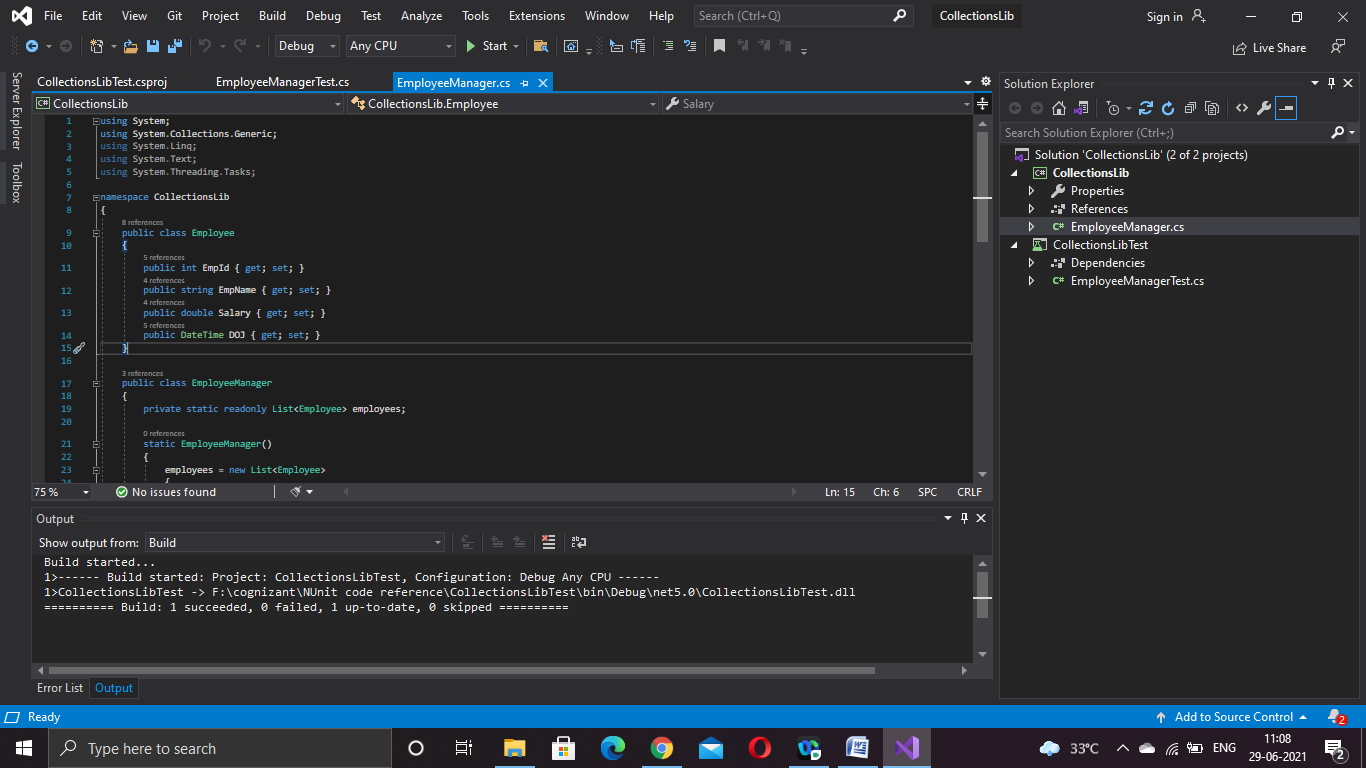
6) Run your tests.

7) Break the test by modifying the source project functionality.

8) Rerun the test.

9) Observe the test result.

**Program:**

****

**EmployeeManagerTest.cs:**

using System.Linq;

using NUnit.Framework;

using CollectionsLib;

namespace CollectionsLibTest

{

public class Tests

{

[TestFixture]

public class UnitTest1

{

[Test]

public void Test()

{

EmployeeManager employeeManager = new EmployeeManager();

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

Assert.AreEqual(containnull, false);

var a = employeeManager.GetEmployees();

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

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

var dis = a.Distinct().Count();

Assert.That(dis, Is.EqualTo(a.Count));

var x = employeeManager.GetEmployees();

var y = employeeManager.GetEmployeesWhoJoinedInPreviousYears();

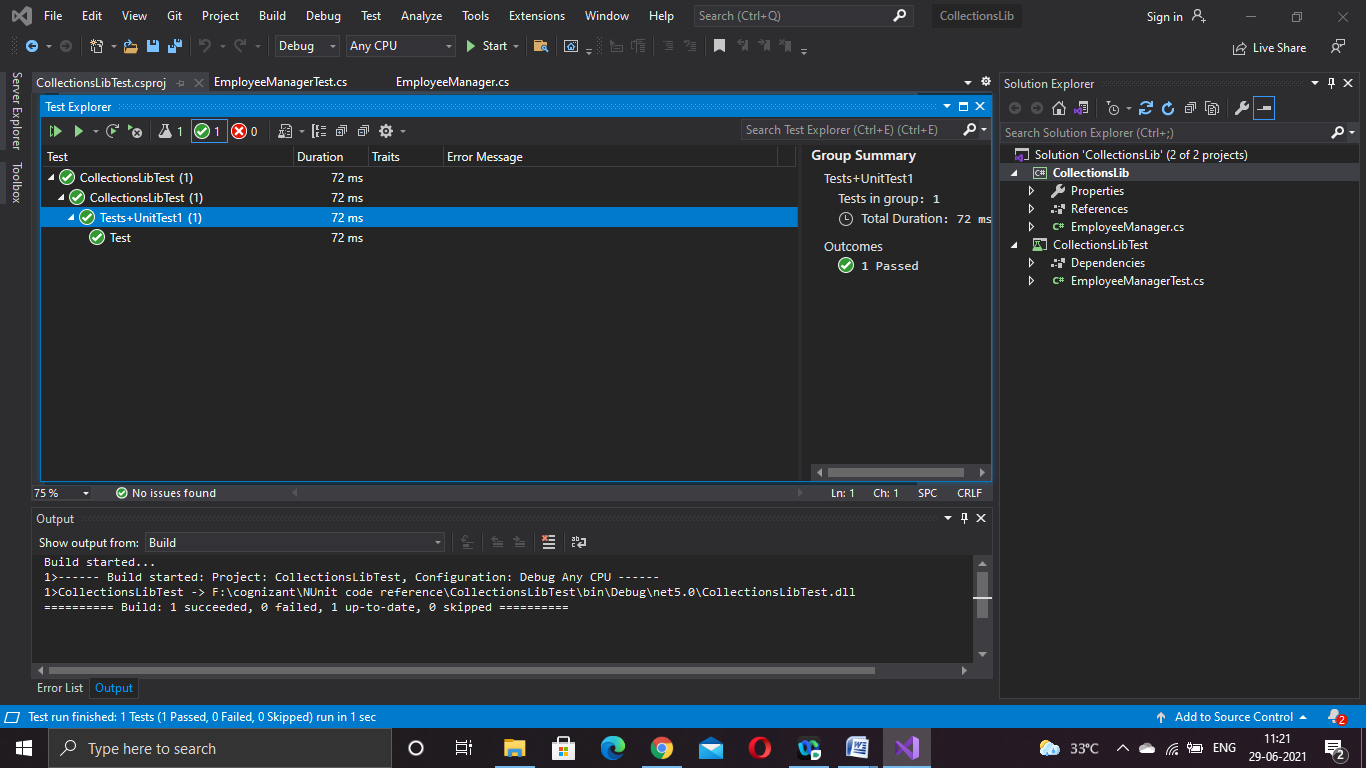
Assert.That(x, Is.EquivalentTo(y));

}

}

}

**Output:**

****

**Handson 2:**

**Steps to be followed:**

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the ConverterLib project to the test project.

4) Additionally add the reference of NUnit, NUnit3TestAdapter and Moq in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

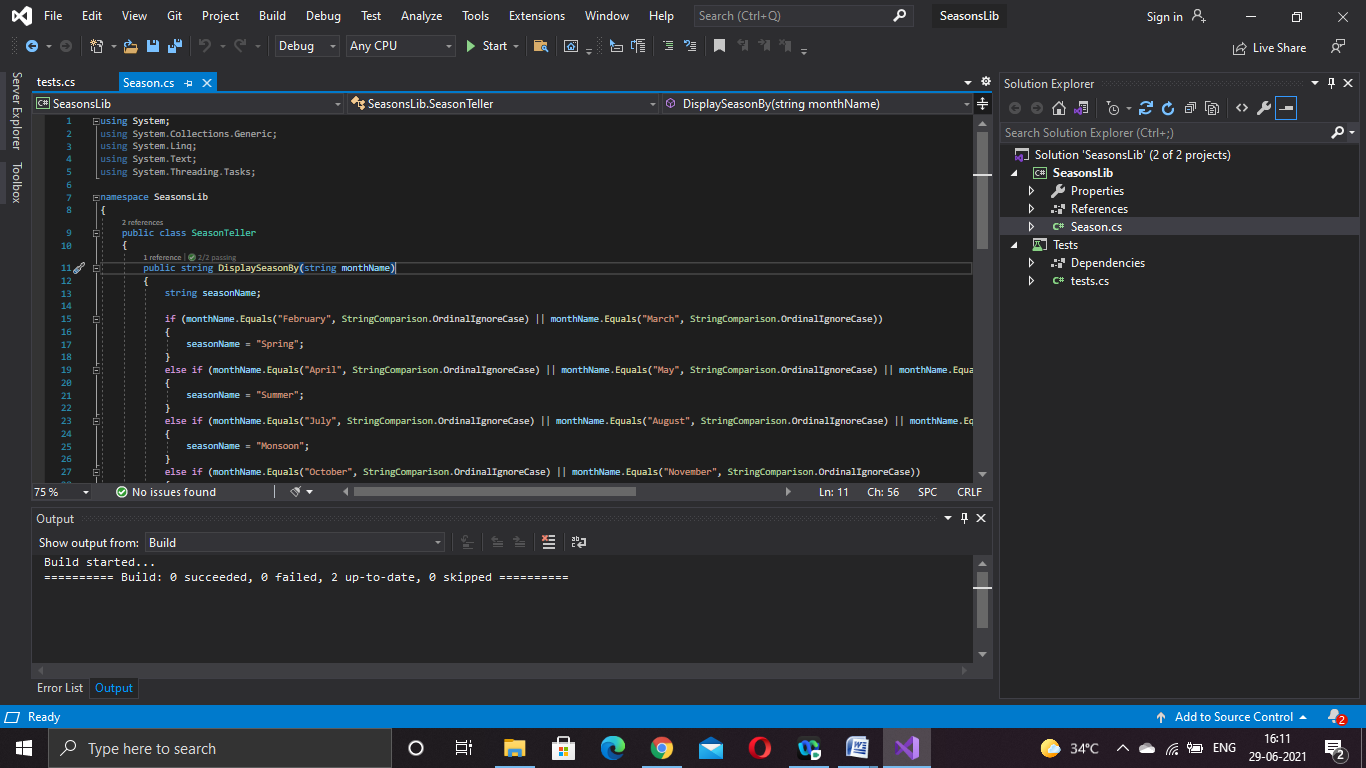
6) Run your tests.

7) Break the test by modifying the source project functionality.

8) Rerun the test.

9) Observe the test result.

**Program:**

****

**Tests.cs:**

using NUnit.Framework;

using SeasonsLib;

namespace Tests

{

public class Tests

{

[Test]

[TestCaseSource(nameof(str))]

public void UnitUnderTest\_Scenario\_ExpectedOutcome(string x, string y)

{

SeasonTeller s = new SeasonTeller();

Assert.That(y, Is.EqualTo(s.DisplaySeasonBy(x)));

}

static object[] str =

{

new object[] { "February", "Spring" },

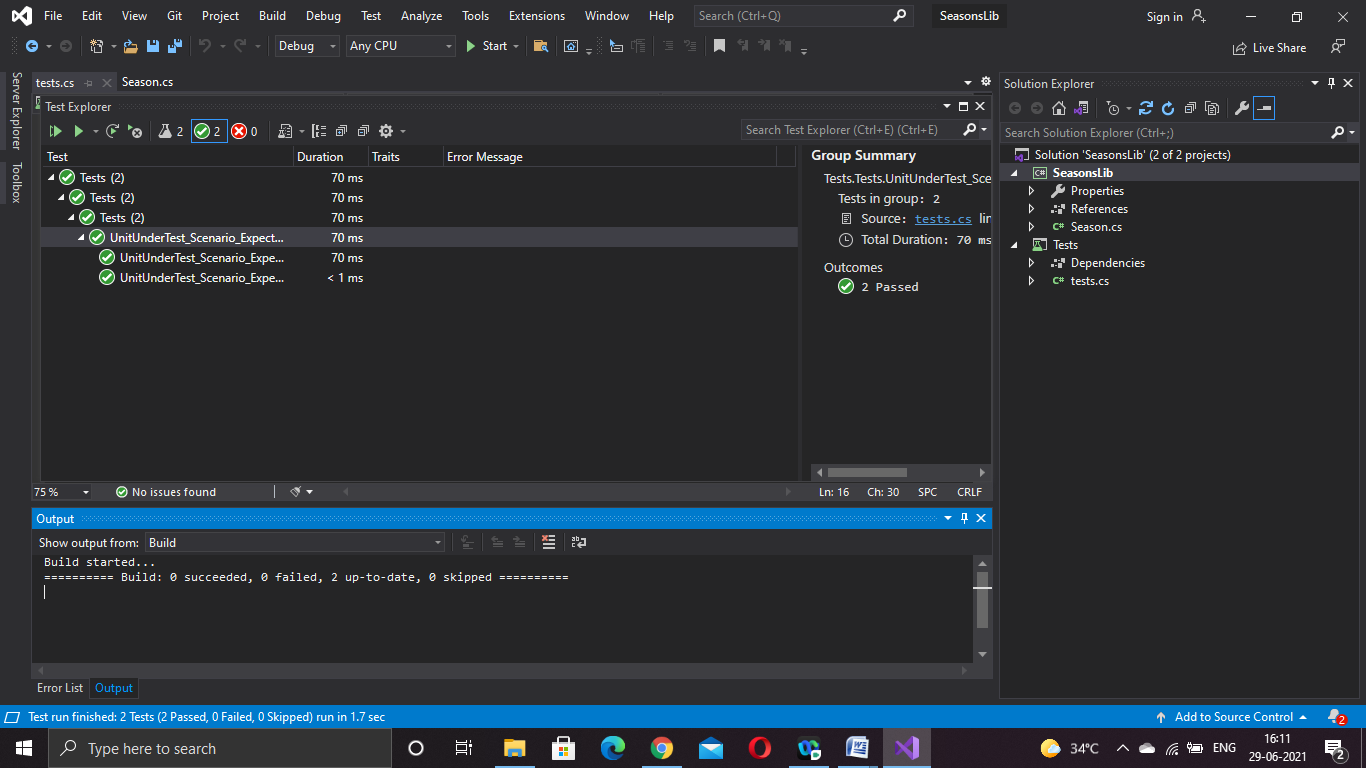
new object[] {"July","Monsoon"}

};

}

}

**Output:**

****

**Handson 3:**

**Steps to perform:**

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the UtilLib project to the test project.

4) Additionally add the reference of both NUnit and NUnit3TestAdapter in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

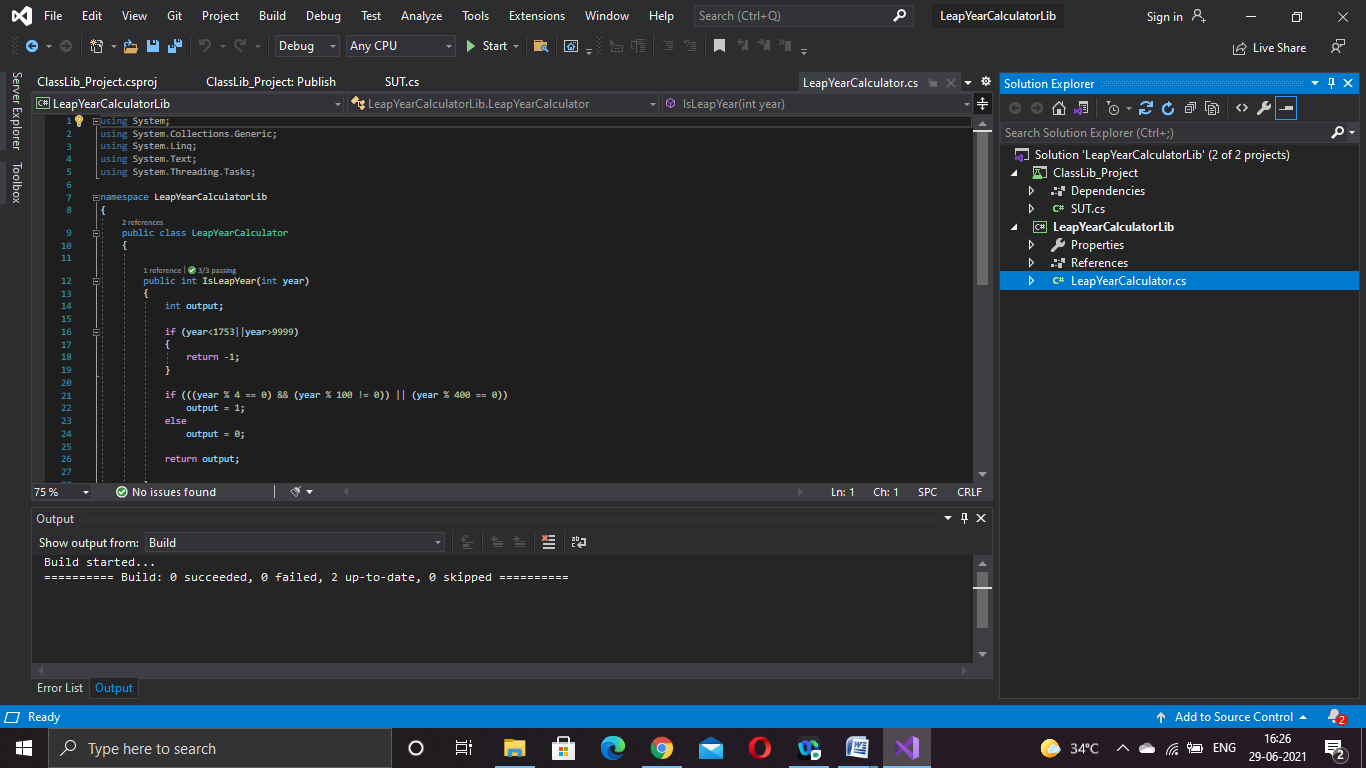
6) Run your tests.

7) Break the test by modifying the source project functionality.

8) Rerun the test.

9) Observe the test result.

**Program:**

****

**SUT.cs**

using LeapYearCalculatorLib;

using NUnit.Framework;

namespace ClassLib\_Project

{

public class Tests

{

[Test]

[TestCase(1752, -1)]

[TestCase(2000, 1)]

[TestCase(1998, 0)]

public void UnitUnderTest\_Scenario\_ExpectedOutcome(int year, int expectedresult)

{

LeapYearCalculator leapYearCalculator = new LeapYearCalculator();

int res = leapYearCalculator.IsLeapYear(year);

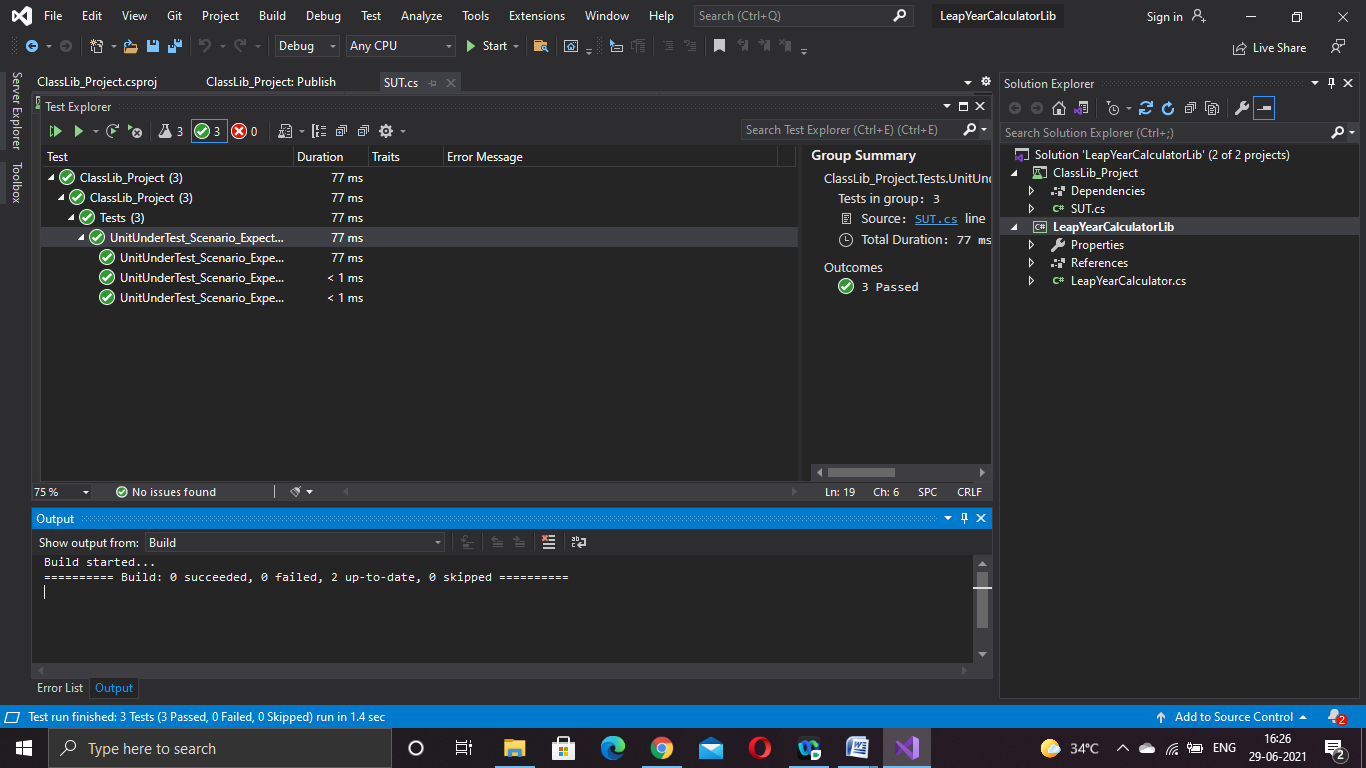
Assert.That(res, Is.EqualTo(expectedresult));

}

}

}

**Output:**

****

**Handson 4:**

**Steps to perform**

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the UtilLib project to the test project.

4) Additionally add the reference of both NUnit and NUnit3TestAdapter in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

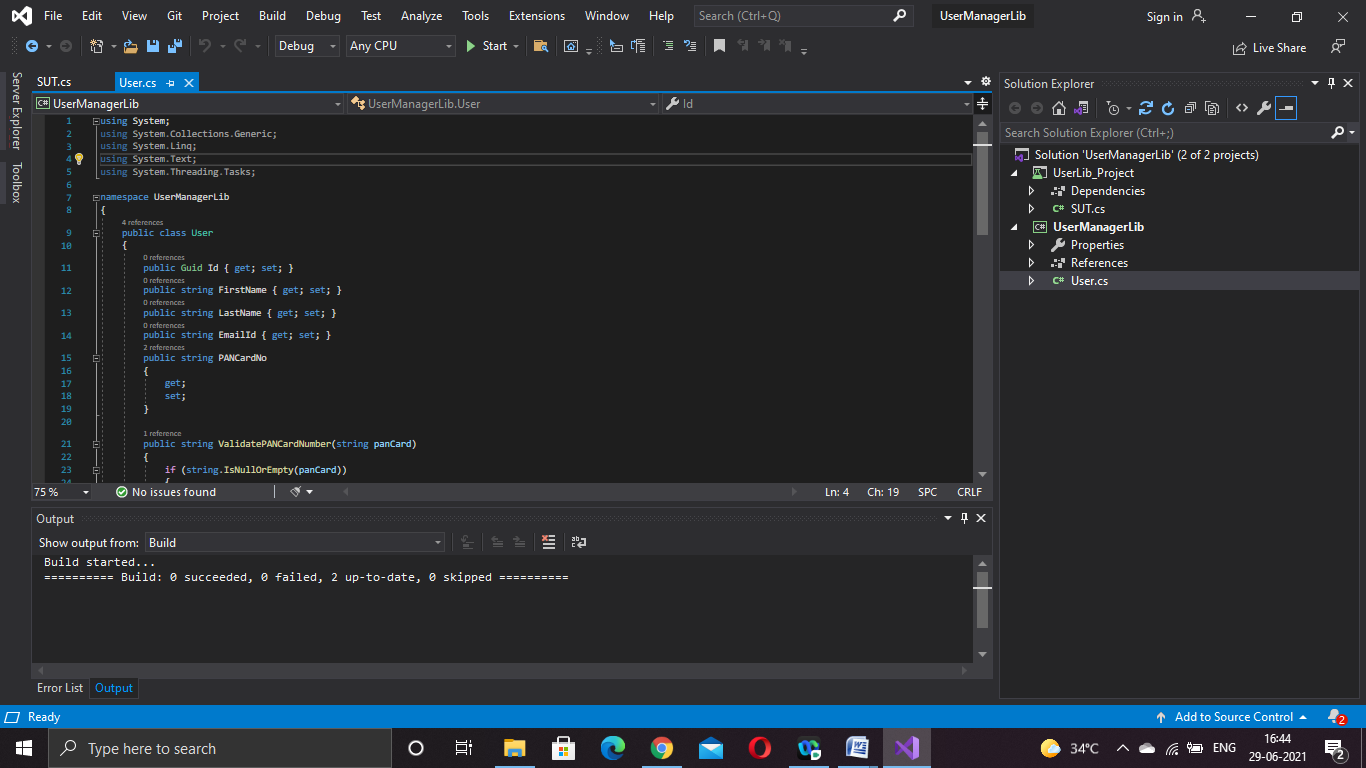
6) Run your tests.

7) Break the test by modifying the source project functionality.

8) Rerun the test.

9) Observe the test result.

**Program:**

****

**SUT.cs:**

using NUnit.Framework;

using System;

using UserManagerLib;

namespace UserLib\_Project

{

public class Tests

{

[Test]

[TestCase("1234567890")]

[TestCase("ABCDEFGHIJKLMNOPQRSTUVWXYZ")]

public void validpancard(string a)

{

User u = new User();

try

{

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

}

catch (NullReferenceException e)

{

Assert.Fail(e.Message);

}

catch (FormatException e)

{

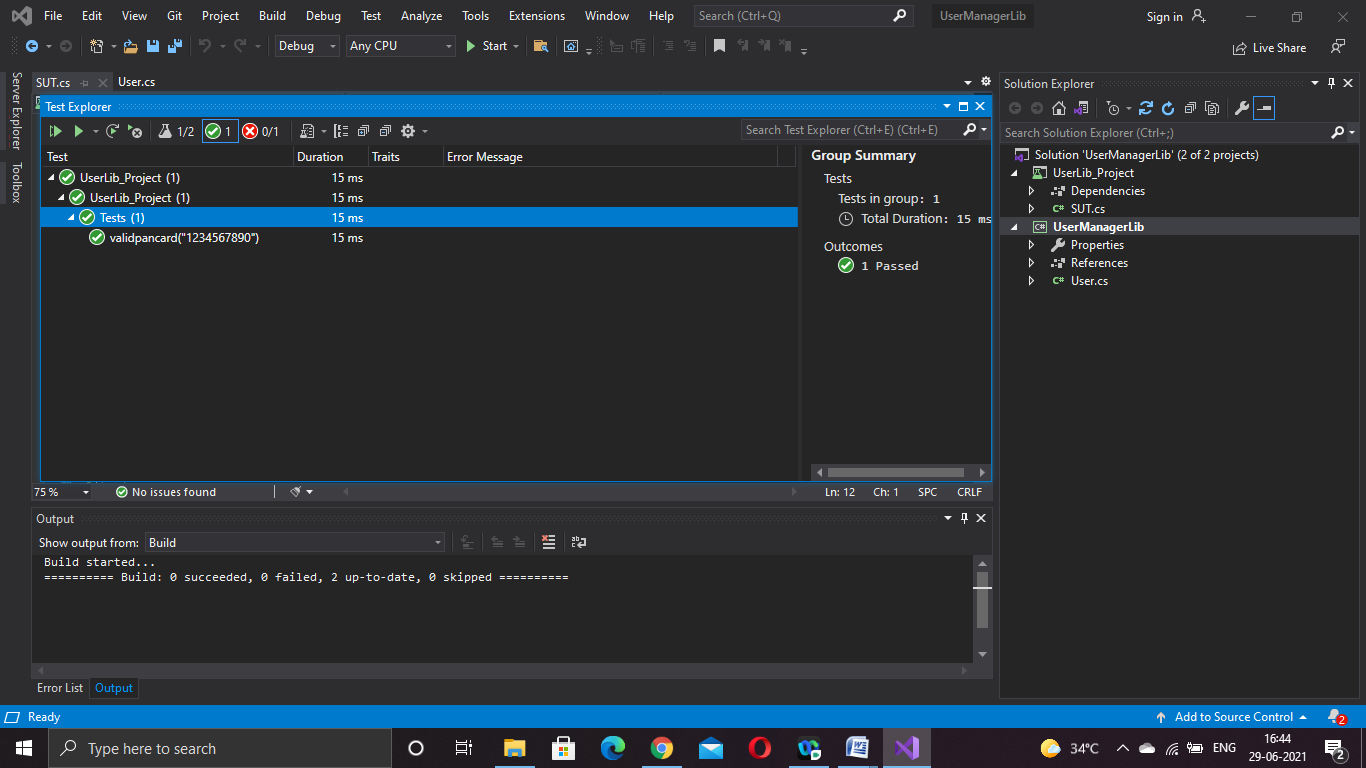
Assert.Fail(e.Message);

}

}

}

**Output:**

****

**Handson 5:**

**Steps to perform:**

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the ConverterLib project to the test project.

4) Additionally, add the reference of NUnit, NUnit3TestAdapter and Moq in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

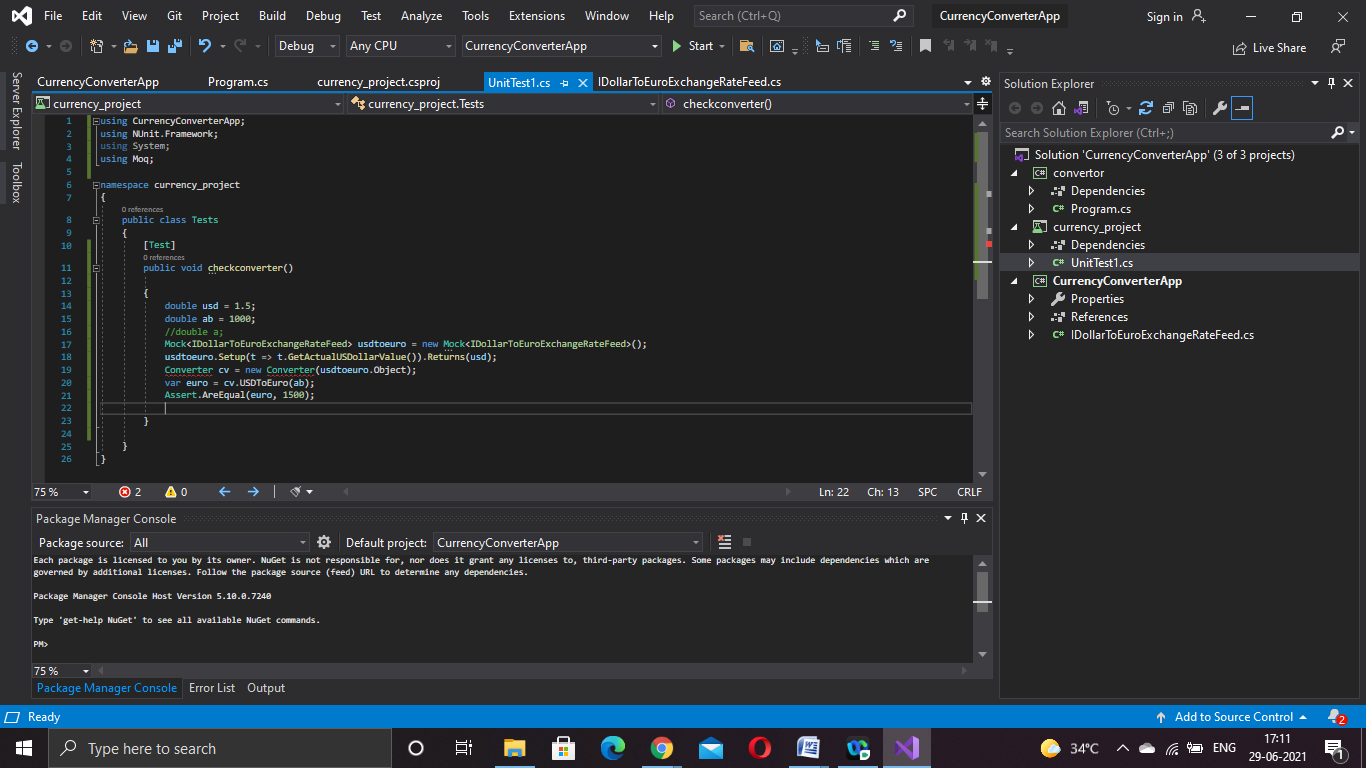
6) Run your tests.

7) Break the test by modifying the source project functionality.

8) Rerun the test.

9) Observe the test result.

**Program:**

****

**Unittest.cs:**

using CurrencyConverterApp;

using NUnit.Framework;

using System;

using Moq;

namespace currency\_project

{

public class Tests

{

[Test]

public void checkconverter()

{

double usd = 1.5;

double ab = 1000;

//double a;

Mock<IDollarToEuroExchangeRateFeed> usdtoeuro = new Mock<IDollarToEuroExchangeRateFeed>();

usdtoeuro.Setup(t => t.GetActualUSDollarValue()).Returns(usd);

Converter cv = new Converter(usdtoeuro.Object);

var euro = cv.USDToEuro(ab);

Assert.AreEqual(euro, 1500);

}

}

}

**Output:**

