# Objectives

* This lab will help you become skilled at using one of the flexible mock frameworks called Moq and how it can be used to isolate units of code from their dependencies.
* Demonstrate on how to create a testable code with Moq.

Create a unit test project using NUnit for the given **ConverterLib** project. Click [here](https://cognizantonline.sharepoint.com/:u:/r/sites/GTP-Solutions/Gencsharepath/Shared%20Documents/Internship2020/FSE/DotNet/02%20-%20NUnit,%20C%23%204.5,%20ASP.Net%20Core/Handson/ConverterLib.zip?csf=1&web=1&e=O4XTfh) to download the source project.

One of the functionalities called **USDToEuro** which is defined in the **Converter** class should be your primary target while unit testing. It takes the US dollar as an input and convert it to Euro with the help of an [external service](https://cognizantonline.sharepoint.com/:u:/r/sites/GTP-Solutions/Gencsharepath/Shared%20Documents/Internship2020/FSE/DotNet/02%20-%20NUnit,%20C%23%204.5,%20ASP.Net%20Core/Handson/CurrencyConverterApp.zip?csf=1&web=1&e=nQyygR), **IDollarToEuroExchangeRateFeed**. Since your application requires this functionality and the same can’t be tested while unit testing because you may not have a grip on the logic behind that service. On top of that, this particular functionality might have tested before it’s made available.

Use Moq framework in order to bypass the functionality which is defined in the IDollarToEuroExchangeRateFeed service.

Write test methods for the given functionalities to make sure that it returns the expected result under various circumstances.

**Recommendations:**

Test Project Name:*<ClassLib\_Project>.Tests*

Test Class Name: *<SUT>Tests*

Test Method Name:  *UnitUnderTest\_Scenario\_ExpectedOutcome*

**Note:**

* *Enforce the Single Assertion Rule*
* *Use Assert.That()*

**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.