Software Unit Testing, part 2

C# - Generic Testing, xUnit

1 Introduction

In part 1 we discussed the need for Software Unit Testing and how this is different from End-to-End testing. We briefly discussed Test Coverage and why we need it. There was a discussion of how automated software testing was done in the early days, and an explanation of how automated software testing would be performed using JavaScript with the modern Web testing framework Cypress. This article will discuss how to do software unit testing in the C# language using .NET and the modern xUnit test framework.

2 C#

Microsoft® first released Visual Studio™ for Windows in 1997. [45] An update was released in 2002 with the ability to do Windows Forms applications. [47] It represented a departure from the traditional way of programming in that the programming editor was visual, and the programmer was given a library of icons representing objects such as buttons, combo boxes, menus, text boxes, toolbars and webpages. [21] Icons could be placed in a program's screens by dragging the object to the appropriate location and dropping it at that location. [22] [24] It was still necessary to write code that would define the interactions between the objects as well as code that performed calculations, but the result was a higher level of productivity. Windows Forms is still in use today. The drag and drop interface of the editor makes it attractive for rapidly prototyping an application.

As part of Visual Studio™, Microsoft® also included a complete programming framework which they called **.NET**. [1] [21] [22] [24] Programs or apps were written as ***managed code***, which was under the control of the ***Common Language Runtime*** (CLR). [2] [3] Rather than compiling software code into a format that could run on the system's processor chip(s), all program code would compile into what was called **Intermediate Language** (IL). [2] Upon program execution, the CLR performs what is known as ***Just In Time compilation*** (JIT). [2] The output of this second last-minute compilation is in a form that the processor chip(s) can understand. The CLR is able to perform many safety checks which reduce the possibility of unintended program behavior.

Managed code became progressively more popular as time passed. Since the .NET Framework was based on ECMA standards [3] [43], it was ported to other computing platforms in addition to its original implementation for Windows®. The C# language, which Microsoft® introduced in 2002 [47], proved to be popular since it was also based on ECMA standards [44], and was ported to other environments as well.

What programming platforms are available to the developer working in C# and .NET? ***Mono*** [38] supports .NET and C# on ***BSD***, ***Linux***, ***macOS*** and ***Microsoft® Windows*** on a variety of microprocessors . [37] ***Xamarin*** can build .NET apps in C# for ***Android***™, ***iOS***, ***macOS***, ***tvOS***, ***watchOS*** and ***Windows®*** [40], and can share code on ***Android***™, ***iOS***, ***Linux***, ***macOS*** and ***Windows®***. [42]

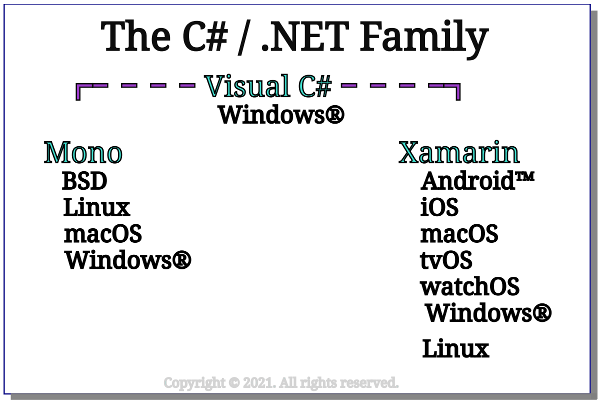


Figure 1: Programming platforms available using C# and .NET. [37] [38] [40] [42]

3 .NET CLI

The .NET framework also included a ***command line interface (CLI)***. The .NET command-line interface (CLI) is a cross-platform toolchain for developing, building, running, and publishing .NET applications. The .NET CLI is included with the .NET SDK. [123] After installing the SDK, CLI commands are run by opening a terminal and entering the commands at the terminal prompt. [122] [124] [134]

Once the .NET SDK is installed, further help on using the CLI may be obtained by typing

dotnet --help

4 Test Scenario

For our testing example let's use the children's game "FizzBuzz", which is sometimes used as a coding test. FizzBuzz works like this:

Output the numbers from 1 to 100.

1. If the number is not divisible by either 3 or 5, just display the number.
2. If the number is divisible by 3, display the word "FIZZ" instead of the number.
3. If the number is divisible by 5, display the word "BUZZ" instead of the number.
4. If the number is divisible by both 3 and 5, display the word "FIZZBUZZ" instead of the number.

So, our code must meet the four requirements listed above. In order to do so, the code must make several decisions based on the input value. We would therefore need a minimum of 4 test cases to do testing based on the requirements. This function has the following code paths:

path 1 - the input IS NOT divisible by EITHER 3 or 5, return the input value
path 2 - the input IS divisible by BOTH 3 and 5, return "FIZZBUZZ"
path 3 - the input IS divisible by 3 but IS NOT divisible by 5, return "FIZZ"
path 4 - the input IS NOT divisible by 3 but IS divisible by 5, return "BUZZ"
path 5 - if an error occurs, return "*** ERROR ***"

If we test to meet the requirements, we would expect to see 100% path coverage of the main section of the code with the four test cases needed to meet the requirements. We would add a fifth test case to verify that errors are handled correctly.

5 Unit Testing - Generic Application

Let's write a function called ***fizzBuzz*** that we want to use in an application. We want to test this function to make sure it works correctly. .NET supports multiple languages, but we'll write this one in C#. Our function would look like this:

```
static String fizzBuzz ( int counter ) {
       //  Input a number between 1 and 100.
       //  If the number is divisible by 3, output "FIZZ".
       //  If the number is divisible by 5, output "BUZZ".
       //  If the number is divisible by both 3 and 5, output "FIZZBUZZ".
       //  For all other numbers, output the original number.

         try
         {
              String resultStr = "";                         //  initialize the variable

               resultStr = counter.ToString();      // default - just return the counter

               if(counter % 3 == 0) {     //  divisible by 3
                  resultStr = "FIZZ";
               }

               if(counter % 5 == 0) {     //  divisible by 5
                  resultStr = "BUZZ";
               }

               if( (counter % 3 == 0) && (counter % 5 == 0) ) {  //  divisible by both 3 and  5
                  resultStr =  "FIZZBUZZ";
               }
```

Continues on the next page.

```
            return resultStr;

            //  end      try
        }
        catch (Exception excp)
        {
            //  caught an exception

            Exception savedExcp = excp;
            //  prevent the warning that excp
            //  is never used

            return "*** ERROR ***";

            //  end      caught an exception
        }

    //  end      static String fizzBuzz
}
```

Figure 2: C# code for the fizzBuzz function.

We need to create a test application for the function fizzBuzz. Let's call the test application **testFizzBuzz** . We'll create and compile our test application using the .NET CLI. [122] [123] [124] [134]

Create a directory for the project.

```
mkdir testFizzBuzz
```

Change directory to the project directory.

```
cd testFizzBuzz
```

Create the initial scaffolding for the project.

```
dotnet new sln
```

Create the initial scaffolding for a library.

```
dotnet new classlib -o library
```

Inside the directory "library" that we created, we see a file "Class1.cs". Class1.cs looks like this:

```
// library/Class1.cs

namespace library;

public class Class1
{

}
```

Add the library to the project.

```
dotnet sln add library/library.csproj
```

Make sure everything is up to date.

```
dotnet restore
```

Make sure that the project builds correctly.

```
dotnet build
```

Create the initial scaffolding for xUnit testing.

```
dotnet new xunit -o xUnit-library
```

Add the xUnit test library to the app.

```
dotnet sln add xUnit-library/xUnit-library.csproj
```

Add a reference to the library into the xUnit test library.

```
dotnet add xUnit-library/xUnit-library.csproj reference library/library.csproj
```

****
.NET can run on multiple operating systems, but xUnit may not be able to run on all of them. [128] [129] [132] In order to cover those situations, lets create a console application where we can also run tests.
****

```
dotnet new console -o console-app
```

Inside the directory "console-app" that we created, we see a file "Program.cs". Program.cs looks like this:

```
// console-app/Program.cs

// See https://aka.ms/new-console-template for more information
Console.WriteLine("Hello, World!");
```

Add the console application to the project.

```
dotnet sln add console-app/console-app.csproj
```

Add a reference to the library into the console application.

```
dotnet add console-app/console-app.csproj reference library/library.csproj
```

Make sure that the project builds correctly.

```
dotnet build
```

Run the console application to verify that it's working.

```
dotnet run --project console-app/console-app.csproj
```

We also need to write a procedure inside the test application that calls fizzBuzz with an input value and retrieves the result produced by fizzBuzz. .NET supports multiple operating systems. The .NET Framework does not currently support dotnet test with xUnit.net on non-Windows environments. [128] In order to run the tests under those environments, we will need to download and install Mono on our machine. [129] [132] If we do not use Mono, we will have to build a generic console app and include the functions to be tested within that application or within the library. We will also need to write a procedure to perform the test cases.

1. We could put the function fizzBuzz AND our test procedure inside the console application that we previously created.

2. We could put the function fizzBuzz in the library which we previously created, and call it from the test procedure inside the console application that we previously created.

6 Unit Testing - Local Functions

Let's start with the function and the test procedure local to the console application we previously created. Let's call the test procedure **localTestCase** so we know we're not calling a library procedure. This procedure would compare the actual result that was returned by fizzBuzz with the result that we were expecting. If they matched, localTestCase will tell us that the test case passed. If they didn't match, we would be told that the test case failed. The localTestCase procedure would generate a one line report of the status of that particular test case. The report would look similar to this:

```
Test case 1   fizzBuzz with input of 1    expecting 1    returned 1    PASSED
Test case 2   fizzBuzz with input of 16   expecting 17   returned 16   FAILED
```

Our localTestCase procedure would look like this:

```
static void localTestCase (
               int testCaseNumber,
               int inputValue,
               String expectedResultStr ) {

      String resultStr = "";

      resultStr = fizzBuzz (inputValue);

      Console.Write ("Local test case ");
      Console.Write (testCaseNumber.ToString());
      Console.Write (" ");
      Console.Write ("fizzBuzz with input of ");
      Console.Write (inputValue.ToString());
      Console.Write (" ");
      Console.Write ("expecting ");
      Console.Write (expectedResultStr);
      Console.Write (" ");
      Console.Write ("returned ");
      Console.Write (resultStr);
      Console.Write (" ");

      if(expectedResultStr == resultStr) {
            Console.WriteLine ("PASSED");
      }
      else
      {
            Console.WriteLine ("FAILED");
      }

     //  end       static void localTestCase
}
```

Figure 3: C# code for the localTestCase procedure.

In the Main section of our test application we set up several test cases, as follows:

```
//  localTestCase (int testCaseNumber,  int inputValue, String expectedResultStr);

localTestCase(1, 1, "1");
localTestCase(2, 2, "2");
localTestCase(3, 3, "FIZZ");
localTestCase(4, 4, "4");
localTestCase(5, 5, "BUZZ");
localTestCase(6, 15, "FIZZBUZZ");
localTestCase(7, -9, "FIZZ");
localTestCase(8, 16, "17");  // *** should fail ***
```

Figure 4: C# code for the local fizzBuzz function test cases.

Test cases 1 through 5 would test the numbers 1 through 5. This would cover 3 of the requirements: NOT divisible by 3 OR 5, divisible by 3 ONLY and divisible by 5 ONLY. Test case 6 would cover the requirement where the input (15) is divisible by BOTH 3 AND 5. Test case 7 is designed to show that fizzBuzz works correctly when the input is a negative number. These 7 test cases should cover the 4 original requirements. Since we have covered all the requirements, we should have also covered all the paths in the main section of this function. Test case 8 shows that we correctly handle situations where the value that was returned does not match the expected value. After compiling our test application, we can run it to see the results:

```
dotnet run --project console-app/console-app.csproj
```

```
Local test case 1    fizzBuzz with input of 1     expecting 1          returned 1          PASSED
Local test case 2    fizzBuzz with input of 2     expecting 2          returned 2          PASSED
Local test case 3    fizzBuzz with input of 3     expecting FIZZ       returned FIZZ       PASSED
Local test case 4    fizzBuzz with input of 4     expecting 4          returned 4          PASSED
Local test case 5    fizzBuzz with input of 5     expecting BUZZ       returned BUZZ       PASSED
Local test case 6    fizzBuzz with input of 15    expecting FIZZBUZZ   returned FIZZBUZZ   PASSED
Local test case 7    fizzBuzz with input of -9    expecting FIZZ       returned FIZZ       PASSED
Local test case 8    fizzBuzz with input of 16    expecting 17         returned 16         FAILED
```

7 Unit Testing - Library Functions

If we put the fizzBuzz function inside our library, the non-local testCase procedure inside the console application would look like this:

```
static void testCase (
               int testCaseNumber,
               int inputValue,
               String expectedResultStr ) {

      String resultStr = "";

      resultStr = library.Class1.fizzBuzz (inputValue);

      Console.Write ("Library test case ");
      Console.Write (testCaseNumber.ToString());
      Console.Write (" ");
      Console.Write ("fizzBuzz with input of ");
      Console.Write (inputValue.ToString());
      Console.Write (" ");
      Console.Write ("expecting ");
      Console.Write (expectedResultStr);
      Console.Write (" ");
      Console.Write ("returned ");
      Console.Write (resultStr);
      Console.Write (" ");

      if(expectedResultStr == resultStr) {
            Console.WriteLine ("PASSED");
      }
      else
      {
            Console.WriteLine ("FAILED");
      }

     //  end       static void testCase
}
```

Figure 5: C# code for the testCase procedure.

In the Main section of our test application we set up several test cases, as follows:

```
        //  testCase (int testCaseNumber,  int inputValue, String expectedResultStr);

        testCase(1, 1, "1");
        testCase(2, 2, "2");
        testCase(3, 3, "FIZZ");
        testCase(4, 4, "4");
        testCase(5, 5, "BUZZ");
        testCase(6,15, "FIZZBUZZ");
        testCase(7, -9, "FIZZ");
        testCase(8, 16, "17");        // *** should fail ***
```

Figure 6: C# code for the non-local fizzBuzz function test cases.

Make sure that the project builds correctly.

```
dotnet build
```

After compiling our test application, we can run it to see the results:

```
dotnet run --project console-app/console-app.csproj
```

```
testFizzBuzz
For help, type testFizzBuzz -h.

Library test case 1    fizzBuzz with input of 1     expecting 1             returned 1              PASSED
Library test case 2    fizzBuzz with input of 2     expecting 2             returned 2              PASSED
Library test case 3    fizzBuzz with input of 3     expecting FIZZ          returned FIZZ           PASSED
Library test case 4    fizzBuzz with input of 4     expecting 4             returned 4              PASSED
Library test case 5    fizzBuzz with input of 5     expecting BUZZ          returned BUZZ           PASSED
Library test case 6    fizzBuzz with input of 15    expecting FIZZBUZZ      returned FIZZBUZZ       PASSED
Library test case 7    fizzBuzz with input of -9    expecting FIZZ          returned FIZZ           PASSED
Library test case 8    fizzBuzz with input of 16    expecting 17            returned 16             FAILED


Local test case 1    fizzBuzz with input of 1     expecting 1             returned 1              PASSED
Local test case 2    fizzBuzz with input of 2     expecting 2             returned 2              PASSED
Local test case 3    fizzBuzz with input of 3     expecting FIZZ          returned FIZZ           PASSED
Local test case 4    fizzBuzz with input of 4     expecting 4             returned 4              PASSED
Local test case 5    fizzBuzz with input of 5     expecting BUZZ          returned BUZZ           PASSED
Local test case 6    fizzBuzz with input of 15    expecting FIZZBUZZ      returned FIZZBUZZ       PASSED
Local test case 7    fizzBuzz with input of -9    expecting FIZZ          returned FIZZ           PASSED
Local test case 8    fizzBuzz with input of 16    expecting 17            returned 16             FAILED
End of program.
```

8 Unit Testing - xUnit

Let's test the fizzBuzz function using xUnit. [126] [127] The .NET Framework does not currently support dotnet test with xUnit.net on non-Windows environments. [128] In order to run the tests under those environments, we will need to download and install Mono on our machine. [129] [132].

Inside the test library "xUnit-library" there is a file named "UnitTest1.cs". The file looks like this:

```
// xUnit_library/UnitTest1.cs

namespace xUnit_library;

public class UnitTest1
{
   [Fact]
   public void Test1()
   {

   }
}
```

Let's add some tests to UnitTest1.cs:

```
[Fact]
public void Test1()
{
   Assert.Equal("1", library.Class1.fizzBuzz(1));
}

[Fact]
public void Test2()
{
   Assert.Equal("2", library.Class1.fizzBuzz(2));
}

[Fact]
public void Test3()
{
   Assert.Equal("FIZZ", library.Class1.fizzBuzz(3));
}

[Fact]
public void Test4()
{
   Assert.Equal("4", library.Class1.fizzBuzz(4));
}

[Fact]
public void Test5()
{
   Assert.Equal("BUZZ", library.Class1.fizzBuzz(5));
}

[Fact]
public void Test6()
{
   Assert.Equal("FIZZ", library.Class1.fizzBuzz(-9));
}

[Fact]
public void Test7()
{
   Assert.Equal("FIZZBUZZ", library.Class1.fizzBuzz(15));
}

[Fact]
public void Test8()
{
   Assert.Equal("16", library.Class1.fizzBuzz(16));
}

[Fact]
public void Test9()
{
   Assert.NotEqual("16", library.Class1.fizzBuzz(16));
}
```

Make sure that the project builds correctly.

```
dotnet build
```

We can now run the tests. [125] [127]

```
dotnet test --logger "console;verbosity=detailed" xUnit-library/xUnit-library.csproj
Determining projects to restore...
All projects are up-to-date for restore.
...
library -> ... library.dll
xUnit-library -> ... xUnit-library.dll

Test run for ... xUnit-library.dll (.NETCoreApp,Version=v8.0)
Microsoft (R) Test Execution Command Line Tool Version 17.10.0 (x64)
Copyright (c) Microsoft Corporation. All rights reserved.

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.
... xUnit-library.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v2.5.3.1+6b60a9e56a (64-bit .NET 8.0.7)
[xUnit.net 00:00:00.15] Discovering: xUnit-library
[xUnit.net 00:00:00.20] Discovered: xUnit-library
[xUnit.net 00:00:00.21] Starting: xUnit-library
[xUnit.net 00:00:00.31] xUnit_library.UnitTest1.Test9 [FAIL]
[xUnit.net 00:00:00.31] Assert.NotEqual() Failure: Strings are equal
[xUnit.net 00:00:00.31] Expected: Not "16"
[xUnit.net 00:00:00.31] Actual: "16"
[xUnit.net 00:00:00.32] Stack Trace:
[xUnit.net 00:00:00.32] ... \testFizzBuzz\xUnit-library\UnitTest1.cs(65,0): at xUnit_library.UnitTest1.Test9()
[xUnit.net 00:00:00.32] at System.RuntimeMethodHandle.InvokeMethod(Object target, Void** arguments, Signature sig,
Boolean isConstructor)
[xUnit.net 00:00:00.32] at System.Reflection.MethodBaseInvoker.InvokeWithNoArgs(Object obj, BindingFlags invokeAttr)
[xUnit.net 00:00:00.32] Finished: xUnit-library
Passed xUnit_library.UnitTest1.Test3 [5 ms]
Failed xUnit_library.UnitTest1.Test9 [7 ms]
Error Message:
Assert.NotEqual() Failure: Strings are equal
Expected: Not "16"
Actual: "16"
Stack Trace:
at xUnit_library.UnitTest1.Test9() in ... \testFizzBuzz\xUnit-library\UnitTest1.cs:line 65
at System.RuntimeMethodHandle.InvokeMethod(Object target, Void** arguments, Signature sig, Boolean isConstructor)
at System.Reflection.MethodBaseInvoker.InvokeWithNoArgs(Object obj, BindingFlags invokeAttr)
Passed xUnit_library.UnitTest1.Test8 [< 1 ms]
Passed xUnit_library.UnitTest1.Test2 [< 1 ms]
Passed xUnit_library.UnitTest1.Test1 [< 1 ms]
Passed xUnit_library.UnitTest1.Test4 [< 1 ms]
Passed xUnit_library.UnitTest1.Test5 [< 1 ms]
Passed xUnit_library.UnitTest1.Test6 [< 1 ms]
Passed xUnit_library.UnitTest1.Test7 [< 1 ms]

Test Run Failed.
Total tests: 9
Passed: 8
Failed: 1
Total time: 1.2151 Seconds
```

9 Test Coverage using xUnit

How do we know if we have tested the software completely?

If we are using a .NET SDK earlier than 8, we may need to add some software to the project in order to generate coverage reports. [145] [146] [151]

```
dotnet new tool-manifest
```

```
dotnet add package coverlet.collector
```

```
dotnet add package coverlet.msbuild
```

```
dotnet tool install coverlet.console
```

```
dotnet tool install -g dotnet-reportgenerator-globaltool
```

If we are using .NET SDK 8, we may need to install the following software: [146] [147]

```
dotnet tool install --global dotnet-coverage
```

Make sure that everything is up to date.

```
dotnet restore
```

Make sure that the project builds correctly.

```
dotnet build
```

Run the tests and compute the code coverage. [146] [152]

```
dotnet test --results-directory "test-results" --collect:"Code Coverage;Format=cobertura"
```

Cobertura [155] is a popular code coverage tool for Java projects. Cobertura supports different types of code coverage metrics, including line coverage, branch coverage, and method coverage. [154] [156] Using the Cobertura file format allows for flexibility in the use of reporting tools.

If there are multiple test runs, we can merge them into a single Cobertura XML file. [146] [147]

```
dotnet-coverage merge --output test-result.cobertura.xml --output-format cobertura "test-results/**/*.coverage"
```

We can generate a report from the Cobertura XML file that we created. [146] [150] [152]

```
reportgenerator -reports:test-result.cobertura.xml -targetdir:coverage-report -reporttypes:"Html_Dark"
```

The report generator converts coverage reports generated by coverlet, OpenCover, dotCover, Visual Studio, NCover, Cobertura, JaCoCo, Clover, gcov or lcov into human readable reports in various formats. The reports show which lines of your source code have been covered. ReportGenerator is available under the Apache [137] License. [148]

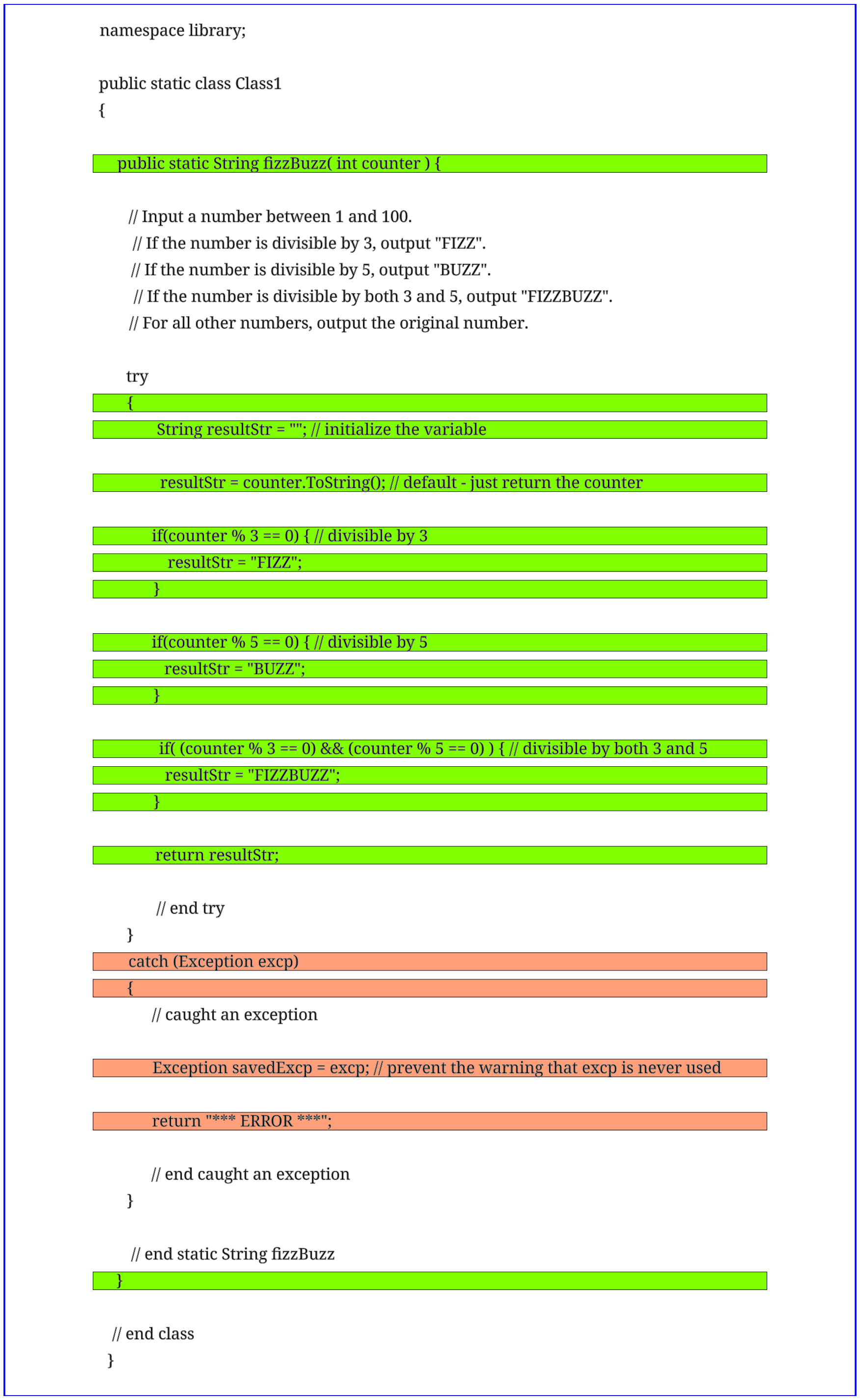
Some of the many report types available include: [157]

Html	A summary (index.html) and detailed reports for each class.
Html_Light	Html with a light theme.
Html_Dark	Html with a dark theme.
HtmlSummary	A single HTML file (summary.html) without links.
TextSummary	A single TXT file containing coverage information per class.
SvgChart	A single SVG file containing a chart with historic coverage information.
Badges	SVG files that show line and/or branch coverage information.

We wrote a coverage report to **coverage-report\index.html** . We can open the file in a browser. On the following page is a sample report similar to the output of reportgenerator.

10 Coverage Report

The coverage report created by reportgenerator says that we have covered 78.95% of the lines of code in the fizzBuzz function. In order to handle any unforeseen problems when we execute the fizzBuzz function, we put the code meeting the requirements within the try section of a try / catch construct. If an error (exception) occurs, the catch section will handle it. None of the test cases encountered an error, so the code for error handling was never executed. In order to cover 100% of the code, we would need to create a situation that produced an error.



```
namespace library;

public static class Class1
{

    public static String fizzBuzz( int counter ) {

      // Input a number between 1 and 100.
       // If the number is divisible by 3, output "FIZZ".
       // If the number is divisible by 5, output "BUZZ".
       // If the number is divisible by both 3 and 5, output "FIZZBUZZ".
      // For all other numbers, output the original number.

      try
      {
            String resultStr = ""; // initialize the variable

             resultStr = counter.ToString(); // default - just return the counter

           if(counter % 3 == 0) { // divisible by 3
               resultStr = "FIZZ";
            }

           if(counter % 5 == 0) { // divisible by 5
              resultStr = "BUZZ";
            }

             if( (counter % 3 == 0) && (counter % 5 == 0) ) { // divisible by both 3 and 5
              resultStr = "FIZZBUZZ";
            }

            return resultStr;

            // end try
      }
      catch (Exception excp)
      {
           // caught an exception

           Exception savedExcp = excp; // prevent the warning that excp is never used

           return "*** ERROR ***";

           // end caught an exception
      }

       // end static String fizzBuzz
    }

   // end class
  }
```

Figure 7: Coverage report similar to the output of the reportgenerator tool.

This page intentionally left blank.

References

[1] G. Warren *et al.*, “Overview of .NET Framework.” *Microsoft Docs*, 21-Oct-2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview. [Accessed: 21-Jan-2021].

[2] G. Warren *et al.*, “What is managed code?” *Microsoft Docs*, 20-Jun-2016. [Online].Available: https://docs.microsoft.com/en-us/dotnet/standard/managed-code. [Accessed: 21-Jan-2021].

[3] G. Warren *et al.*, “Common Language Runtime (CLR) overview.” *Microsoft Docs*, 22-Oct-2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/clr. [Accessed: 28-Jan-2021].

[4] G. Warren, “gewarren - Overview,” *GitHub*. [Online]. Available: https://github.com/gewarren. [Accessed: 22-Jan-2021].

[5] B. Wagner, “BillWagner - Overview,” *GitHub*. [Online]. Available: https://github.com/BillWagner. [Accessed: 26-Jan-2021].

[6] D. Lee, “DennisLee-DennisLee - Overview,” *GitHub*. [Online]. Available: https://github.com/DennisLee-DennisLee. [Accessed: 10-Feb-2021].

[7] nxtn, “nxtn - Overview,” *GitHub*. [Online]. Available: https://github.com/nxtn. [Accessed: 10-Feb-2021].

[8] J. Kotas, “jkotas - Overview,” *GitHub*. [Online]. Available: https://github.com/jkotas. [Accessed: 10-Feb-2021].

[9] M. Wenzel, “mairaw - Overview,” *GitHub*. [Online]. Available: https://github.com/mairaw. [Accessed: 10-Feb-2021].

[10] N. Schonning, “nschonni - Overview,” *GitHub*. [Online]. Available: https://github.com/nschonni. [Accessed: 10-Feb-2021].

[11] A. Aymeric, “nemrism - Overview,” *GitHub*. [Online]. Available: https://github.com/nemrism. [Accessed: 10-Feb-2021].

[12] C. Maddock, “ChrisMaddock - Overview,” *GitHub*. [Online]. Available: https://github.com/ChrisMaddock. [Accessed: 10-Feb-2021].

[13] xaviex, “xaviex - Overview,” *GitHub*. [Online]. Available: https://github.com/xaviex. [Accessed: 10-Feb-2021].

[14] M. Jones, “Mikejo5000 - Overview,” *GitHub*. [Online]. Available: https://github.com/Mikejo5000. [Accessed: 10-Feb-2021].

[15] L. Latham, “guardrex - Overview,” *GitHub*. [Online]. Available: https://github.com/guardrex. [Accessed: 10-Feb-2021].

[16] tompratt-AQ, “tompratt-AQ - Overview,” *GitHub*. [Online]. Available: https://github.com/tompratt-AQ. [Accessed: 10-Feb-2021].

[17] yishengjin1413, “yishengjin1413 - Overview,” *GitHub*. [Online]. Available: https://github.com/yishengjin1413. [Accessed: 10-Feb-2021].

[18] S. Hoag, “stevehoag - Overview,” *GitHub*. [Online]. Available: https://github.com/stevehoag. [Accessed: 10-Feb-2021].

[19] P. Onderka, “svick - Overview,” *GitHub*. [Online]. Available: https://github.com/svick. [Accessed: 10-Feb-2021].

[20] T. Dykstra, “tdykstra - Overview,” *GitHub*. [Online]. Available: https://github.com/tdykstra. [Accessed: 10-Feb-2021].

[21] A. De George, “Desktop Guide (Windows Forms .NET).” *Microsoft Docs*, 26-Oct-2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/. [Accessed: 19-Jan-2021].

References (continued)

[22] A. De George and M. Wenzel, “Tutorial: Create a new WinForms app (Windows Forms .NET).” *Microsoft Docs*, 26-Oct-2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/get-started/create-app-visual-studio. [Accessed: 28-Jan-2021].

[23] A. De George, “adegeo - Overview,” *GitHub*. [Online]. Available: https://github.com/adegeo. [Accessed: 22-Jan-2021].

[24] O. Altunyan, G. Hogenson, D. Coulter, Y. Victor, and T. Lee, “Create a Windows Forms app with C# - Visual Studio.” *Microsoft Docs*, 26-Sep-2019. [Online]. Available: https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio. [Accessed: 28-Jan-2021].

[25] O. Altunyan, “ornellaalt - Overview,” *GitHub*. [Online]. Available: https://github.com/ornellaalt. [Accessed: 22-Jan-2021].

[26] E. Nakamura, “eddynaka - Overview,” *GitHub*. [Online]. Available: https://github.com/eddynaka. [Accessed: 13-Feb-2021].

[27] D. Mabee, “damabe - Overview,” *GitHub*. [Online]. Available: https://github.com/damabe. [Accessed: 13-Feb-2021].

[28] A. Dev, “amaldevv - Overview,” *GitHub*. [Online]. Available: https://github.com/amaldevv. [Accessed: 13-Feb-2021].

[29] M. Blome, “mikeblome - Overview,” *GitHub*. [Online]. Available: https://github.com/mikeblome. [Accessed: 14-Feb-2021].

[30] A. Sal, “abdullahsalem - Overview,” *GitHub*. [Online]. Available: https://github.com/abdullahsalem. [Accessed: 14-Feb-2021].

[31] M. Koudelka, “v-makoud - Overview,” *GitHub*. [Online]. Available: https://github.com/v-makoud. [Accessed: 14-Feb-2021].

[32] B. Wagner, “.NET documentation.” *Microsoft Docs*, 2021. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/. [Accessed: 14-Feb-2021].

[33] G. Hogenson, “ghogen - Overview,” *GitHub*. [Online]. Available: https://github.com/ghogen. [Accessed: 15-Feb-2021].

[34] Y. Victor, “Youssef1313 - Overview,” *GitHub*. [Online]. Available: https://github.com/Youssef1313. [Accessed: 15-Feb-2021].

[35] T. Lee, “TerryGLee - Overview,” *GitHub*. [Online]. Available: https://github.com/TerryGLee. [Accessed: 15-Feb-2021].

[36] D. Coulter, “DCtheGeek - Overview,” *GitHub*. [Online]. Available: https://github.com/DCtheGeek. [Accessed: 04-Feb-2021].

[37] “About Mono | Mono.” *Mono Project*. [Online]. Available: https://www.mono-project.com/docs/about-mono/. [Accessed: 27-Feb-2021].

[38] “Home | Mono.” *Mono Project*, 2021. [Online]. Available: https://www.mono-project.com/. [Accessed: 28-Feb-2021].

[39] .NET Foundation and Contributors, “.NET Foundation.” *.NET Foundation*, 2021. [Online]. Available: https://dotnetfoundation.org/. [Accessed: 28-Feb-2021].

[40] “Xamarin | Open-source mobile app platform for .NET,” *Microsoft*. [Online]. Available: https://dotnet.microsoft.com/apps/xamarin. [Accessed: 27-Feb-2021].

[41] “What is Xamarin? | .NET,” *Microsoft*. [Online]. Available: https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin. [Accessed: 27-Feb-2021].

[42] “Cross-platform with Xamarin | .NET,” *Microsoft*. [Online]. Available: https://dotnet.microsoft.com/apps/xamarin/cross-platform. [Accessed: 27-Feb-2021].

[43] “ECMA-335,” *Ecma International*, 2012. [Online]. Available: https://www.ecma-international.org/publications-and-standards/standards/ecma-335/. [Accessed: 28-Feb-2021].

[44] “ECMA-334,” *Ecma International*, 2017. [Online]. Available: https://www.ecma-international.org/publications-and-standards/standards/ecma-334/. [Accessed: 28-Feb-2021].

References (continued)

[45] “Microsoft Announces Visual Studio 97, A Comprehensive Suite of Microsoft Visual Development Tools,” *Stories*, 28-Jan-1997. [Online]. Available: https://news.microsoft.com/1997/01/28/microsoft-announces-visual-studio-97-a-comprehensive-suite-of-microsoft-visual-development-tools/. [Accessed: 05-Mar-2021].

[46] “WineHQ - Visual Studio 2002 (7.0).” *WINE^{HQ}*. [Online]. Available: https://appdb.winehq.org/objectManager.php?sClass=version&iId=15422. [Accessed: 05-Mar-2021].

[47] P. Morlion, “The Death of WinForms Has Been Greatly Exaggerated - SubMain Blog,” *Software Quality Blog - SubMain Software*, 28-Aug-2018. [Online]. Available: https://blog.submain.com/death-winforms-greatly-exaggerated/. [Accessed: 05-Mar-2021].

[48] T. Segal, “Profit Center: Characteristics vs. a Cost Center, With Examples,” *Investopedia*, 07-Dec-2020. Available: https://www.investopedia.com/terms/p/profitcentre.asp. [Accessed: 31-May-2024].

[49] “Test Coverage Tutorial: Comprehensive Guide With Best Practices.” *lambdatest.com*. Available: https://www.lambdatest.com/learning-hub/test-coverage. [Accessed: 31-May-2024].

[50] H. Shah, “A Detailed Guide on Test Coverage,” *Simform - Product Engineering Company*, 28-Jun-2021. Available: https://www.simform.com/blog/test-coverage/. [Accessed: 31-May-2024].

[51] T. Hamilton, “Test Coverage in Software Testing,” *Guru99*, 13-Apr-2024. Available: https://www.guru99.com/test-coverage-in-software-testing.html. [Accessed: 31-May-2024].

[52] “What is DO-178B?,” *AdaCore*. Available: https://www.adacore.com/industries/avionics/what-is-do-178b. [Accessed: 31-May-2024].

[53] D. Goswami, “Why does your Avionics Software Need DO-178B/C Compliance?,” *Qualitest*, 24-Dec-2020. Available: https://medium.com/qualitest/why-does-your-avionics-software-need-do-178b-c-compliance-4d1cba8c94a0. [Accessed: 31-May-2024].

[54] V. Nanda, “Test Coverage in Software Testing.” *tutorialspoint.com*. Available: https://www.tutorialspoint.com/test-coverage-in-software-testing. [Accessed: 31-May-2024].

[55] “Manual vs Automated Testing | Cypress Testing Tools.” *cypress.io*. Available: https://learn.cypress.io/testing-foundations/manual-vs-automated-testing. [Accessed: 30-May-2024].

[56] “End-to-End Testing: Your First Test with Cypress | Cypress Documentation,” *Cypress Documentation*, 09-May-2024. Available: https://docs.cypress.io/guides/end-to-end-testing/writing-your-first-end-to-end-test. [Accessed: 30-May-2024].

[57] “From Manual Testing to Automated Testing With Cypress - Our Story | Cogworks Blog.” *Cogworks Blog*. Available: https://www.wearecogworks.com/innerworks/cogworks-blog-archive/from-manual-to-automated-testing-with-cypress-our-story. [Accessed: 30-May-2024].

[58] “Best Practices | Cypress Documentation,” *Cypress Documentation*, 11-Apr-2024. Available: https://docs.cypress.io/guides/references/best-practices. [Accessed: 30-May-2024].

[59] “Testing Frameworks for Javascript | Write, Run, Debug | Cypress.” *cypress.io*. Available: https://www.cypress.io/. [Accessed: 30-May-2024].

[60] “Comprehensive Cypress Test Automation Guide | Cypress Documentation,” *Cypress Documentation*, 11-Apr-2024. Available: https://docs.cypress.io/guides/overview/why-cypress. [Accessed: 30-May-2024].

[61] “Key Differences | Cypress Documentation,” *Cypress Documentation*, 14-Dec-2022. Available: https://docs.cypress.io/guides/overview/key-differences. [Accessed: 30-May-2024].

[62] “How Cypress Works | End to end and component testing tools.” *cypress.io*. Available: https://www.cypress.io/how-it-works. [Accessed: 30-May-2024].

[63] G. Hegde, “Getting Started with Cypress Test Automation : Tutorial,” *BrowserStack*. Available: https://browserstack.wpengine.com/guide/cypress-automation-tutorial/. [Accessed: 30-May-2024].

References (continued)

[64] G. Hegde and P. Bhat, “How to perform Component Testing using Cypress,” *BrowserStack*, 03-Oct-2022. Available: https://browserstack.wpengine.com/guide/cypress-component-testing/. [Accessed: 30-May-2024].

[65] K. Pathak, “How to Perform Component Testing using Cypress,” *Medium*, 29-Jan-2023. Available: https://kailash-pathak.medium.com/how-to-perform-component-testing-using-cypress-3bbe74af7492. [Accessed: 30-May-2024].

[66] F. Hric, “Component testing in Cypress: What is it and why it’s important,” *Automated Visual Testing | Applitools*, 15-Dec-2022. Available: https://applitools.com/blog/component-testing-in-cypress-what-is-it-and-why-its-important/. [Accessed: 30-May-2024].

[67] Technocrat, “How to write Test Case in Cypress: (with testing example),” *BrowserStack*. Available: https://browserstack.wpengine.com/guide/how-to-write-test-case-in-cypress/. [Accessed: 30-May-2024].

[68] D. Greene, “Wow! Cypress can run unit tests! 🥳,” *DEV Community*, 13-Nov-2020. Available: https://dev.to/dgreene1/wow-cypress-can-run-unit-tests-15l5. [Accessed: 29-May-2024].

[69] “Recipes | Cypress Documentation,” *Cypress Documentation*, 11-Apr-2024. Available: https://docs.cypress.io/examples/recipes. [Accessed: 29-May-2024].

[70] “Code Coverage | Cypress Documentation,” *Cypress Documentation*, 25-Mar-2024. Available: https://docs.cypress.io/guides/tooling/code-coverage. [Accessed: 29-May-2024].

[71] G. Bahmutov, “Readable Cypress.io tests,” *Better world by better software*, 01-Apr-2019. Available: https://glebbahmutov.com/blog/readable-tests/index.html. [Accessed: 29-May-2024].

[72] “cypress-example-recipes/examples/unit-testing__application-code at master · cypress-io/cypress-example-recipes,” *GitHub*. Available: https://github.com/cypress-io/cypress-example-recipes/tree/master/examples/unit-testing__application-code. [Accessed: 29-May-2024].

[73] “Cypress Testing Framework Tutorial With Examples | Sauce Labs.” *Sauce Labs*. Available: https://saucelabs.com/resources/blog/getting-started-with-cypress. [Accessed: 29-May-2024].

[74] Morrismoses, “Cypress For Test Automation: A Step-by-Step Guide,” *Medium*, 22-Apr-2024. Available: https://medium.com/@morrismoses149/cypress-for-test-automation-a-step-by-step-guide-93cb47163d05. [Accessed: 29-May-2024].

[75] S. Nadeesha, “Making a simple Test Automation Framework with Cypress.io.” *LinkedIn*. Available: https://www.linkedin.com/pulse/making-simple-test-automation-framework-cypressio-supun-nadeesha. [Accessed: 29-May-2024].

[76] “Selenium vs Cypress vs alternatives: choosing the right framework for web testing.” *zebrunner.com*. Available: https://zebrunner.com/blog-posts/what-is-cypress-and-is-it-a-real-alternative-to-selenium. [Accessed: 29-May-2024].

[77] “How to use Cypress Testing Framework?,” *Testbytes Softwares*. Available: https://www.testbytes.net/blog/cypress-testing/. [Accessed: 29-May-2024].

[78] “Reporters | Cypress Documentation,” 25-Mar-2024. *Cypress Documentation*. Available: https://docs.cypress.io/guides/tooling/reporters. [Accessed: 29-May-2024].

[79] “Mocha - the fun, simple, flexible JavaScript test framework.” *mochajs.org*. Available: https://mochajs.org/. [Accessed: 29-May-2024].

[80] “Jest vs Mocha: Which one Should You Choose?,” *GeeksforGeeks*, 27-Feb-2024. Available: https://www.geeksforgeeks.org/jest-vs-mocha-which-one-should-you-choose/. [Accessed: 29-May-2024].

[81] “Bundled Libraries | Cypress Documentation,” *Cypress Documentation*, 15-Dec-2022. Available: https://docs.cypress.io/guides/references/bundled-libraries. [Accessed: 29-May-2024].

[82] “Chai.” *chaijs.com*. Available: https://www.chaijs.com/. [Accessed: 29-May-2024].

References (continued)

[83] D. Vasudevan, “How can you use Jest or Cypress to ensure front-end code quality?” *LinkedIn*. Available: https://www.linkedin.com/advice/3/how-can-you-use-jest-cypress-ensure-front-end-code-k6bnf. [Accessed: 29-May-2024].

[84] “Cypress vs Jest | Top 15 Key Differences,” *Testsigma Blog*, 02-Apr-2024. Available: https://testsigma.com/blog/cypress-vs-jest/. [Accessed: 29-May-2024].

[85] “Testing Types | Cypress Documentation,” *Cypress Documentation*, 30-Jan-2023. Available: https://docs.cypress.io/guides/core-concepts/testing-types. [Accessed: 29-May-2024].

[86] “Karma - Spectacular Test Runner for Javascript.” Available: https://karma-runner.github.io/latest/index.html. [Accessed: 29-May-2024].

[87] K. Choksi, “How to write unit tests with Jasmine & Karma?,” *Simform Engineering*, 01-Jun-2023. Available: https://medium.com/simform-engineering/how-to-write-unit-tests-with-jasmine-karma-f1908bdeb617. [Accessed: 29-May-2024].

[88] Testim, “Karma JS Testing: What, Why, and How to Get Going Right Now,” *AI-driven E2E automation with code-like flexibility for your most resilient tests*, 11-Dec-2021. Available: https://www.testim.io/blog/karma-js-testing-what-why-and-how-to-get-going-right-now/. [Accessed: 29-May-2024].

[89] “How to Install Cypress for Test Automation,” *BrowserStack*. Available: https://browserstack.wpengine.com/guide/cypress-installation-for-test-automation/. [Accessed: 30-May-2024].

[90] “Code Coverage | Cypress Documentation,” *Cypress Documentation*, 25-Mar-2024. Available: https://docs.cypress.io/guides/tooling/code-coverage. [Accessed: 30-May-2024].

[91] “Open the App with Cypress: Step-by-Step Guide | Cypress Documentation,” *Cypress Documentation*, 19-Apr-2024. Available: https://docs.cypress.io/guides/getting-started/opening-the-app. [Accessed: 30-May-2024].

[92] “End-to-End Testing: Your First Test with Cypress | Cypress Documentation,” *Cypress Documentation*, 09-May-2024. Available: https://docs.cypress.io/guides/end-to-end-testing/writing-your-first-end-to-end-test. [Accessed: 30-May-2024].

[93] “Installing Cypress and writing your first test | Cypress Testing Tools.” Available: https://learn.cypress.io/testing-your-first-application/installing-cypress-and-writing-your-first-test. [Accessed: 30-May-2024].

[94] admin, “Cypress Code Coverage,” *testomat.io*, 07-Feb-2024. Available: https://testomat.io/blog/cypress-code-coverage/. [Accessed: 29-May-2024].

[95] “Setting up Cypress Code Coverage,” *BrowserStack*. Available: https://browserstack.wpengine.com/guide/cypress-code-coverage/. [Accessed: 29-May-2024].

[96] “Node.js — Run JavaScript Everywhere.” *nodejs.org*. Available: https://nodejs.org/en. [Accessed: 31-May-2024].

[97] “Node.js Tutorial.” *W3Schools*. Available: https://www.w3schools.com/nodejs/. [Accessed: 31-May-2024].

[98] “Node.js — Introduction to Node.js.” *nodejs.org*. Available: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs. [Accessed: 31-May-2024].

[99] M. Wankhede, “Automation testing with Cypress, Mocha, and JavaScript,” *TO THE NEW BLOG*, 17-Feb-2023. Available: https://www.tothenew.com/blog/automation-testing-with-cypress-mocha-and-javascript/. [Accessed: 06-Jun-2024].

[100] C. Adams, “Cypress Limitations you should be aware of,” *Codoid*, 12-Jan-2023. Available: https://codoid.com/automation-testing/cypress-limitations-you-should-be-aware-of/. [Accessed: 06-Jun-2024].

[101] “Mocha vs Cypress.io comparison of testing frameworks.” *Knapsack Pro*. Available: https://knapsackpro.com/testing_frameworks/difference_between/mochajs/vs/cypress-io. [Accessed: 06-Jun-2024].

[102] K. Halder, “Automated Testing with Cypress,” *The Startup*, 03-May-2020. Available: https://medium.com/swlh/automated-testing-with-cypress-17bf74bfd97d. [Accessed: 06-Jun-2024].

References (continued)

[103] “Comparing cypress vs. jasmine vs. jest vs. karma vs. mocha,” *NPMCompare*. Available: https://npmcompare.com/compare/cypress,jasmine,jest,karma,mocha. [Accessed: 06-Jun-2024].

[104] “Comparing cucumber vs. cypress vs. jasmine vs. mocha vs. nightwatch,” *NPMCompare*. Available: https://npmcompare.com/compare/cucumber,cypress,jasmine,mocha,nightwatch. [Accessed: 06-Jun-2024].

[105] R. Kumari, “Cypress vs Cucumber - Which One to Choose?,” *Testsigma Blog*, 03-Feb-2024. Available: https://testsigma.com/blog/cypress-vs-cucumber/. [Accessed: 06-Jun-2024].

[106] “Cucumber vs Cypress.io comparison of testing frameworks.” *Knapsack Pro*. Available: https://knapsackpro.com/testing_frameworks/difference_between/cucumber/vs/cypress-io. [Accessed: 06-Jun-2024].

[107] “Selenium,” *Selenium*. Available: https://www.selenium.dev/. [Accessed: 09-Jun-2024].

[108] “WebDriver,” *Selenium*, 29-Mar-2024. Available: https://www.selenium.dev/documentation/webdriver/. [Accessed: 09-Jun-2024].

[109] “WebDriver.” *W3C*, 2018. Available: https://www.w3.org/TR/webdriver1/. [Accessed: 09-Jun-2024].

[110] “Getting started,” *Selenium*. Available: https://www.selenium.dev/documentation/webdriver/getting_started/. [Accessed: 09-Jun-2024].

[111] “Write your first Selenium script,” *Selenium*. Available: https://www.selenium.dev/documentation/webdriver/getting_started/first_script/. [Accessed: 09-Jun-2024].

[112] “Supported Browsers,” *Selenium*, 20-Sep-2022. Available: https://www.selenium.dev/documentation/webdriver/browsers/. [Accessed: 09-Jun-2024].

[113] “Downloads,” *Selenium*. Available: https://www.selenium.dev/downloads/. [Accessed: 09-Jun-2024].

[114] “Selenium WebDriver Tutorial - javatpoint,” *www.javatpoint.com*. Available: https://www.javatpoint.com/selenium-webdriver. [Accessed: 09-Jun-2024].

[115] “WebDriver For Mobile Browsers,” *Selenium*, 12-Jan-2022. Available: https://www.selenium.dev/documentation/legacy/selenium_2/mobile/. [Accessed: 09-Jun-2024].

[116] C. Tozzi , “Can Selenium Be Used for Mobile Testing?” *Sauce Labs Inc.*, 04-Feb-2023. Available: https://saucelabs.com/resources/blog/can-selenium-be-used-for-mobile-testing.
[Accessed: 09-Jun-2024].

[117] “Welcome - Appium Documentation.” *Appium Documentation*. Available: https://appium.io/docs/en/latest/. [Accessed: 10-Jun-2024].

[118] “Appium in a Nutshell - Appium Documentation.” *Appium Documentation*. Available: https://appium.io/docs/en/latest/intro/. [Accessed: 10-Jun-2024].

[119] “Selendroid: Selenium for Android.” *selendroid.io*. Available: http://selendroid.io/. [Accessed: 10-Jun-2024].

[120] “Mobile Testing - Selendroid Framework.” *www.tutorialspoint.com*. Available: https://www.tutorialspoint.com/mobile_testing/mobile_testing_selendroid_framework.htm#:~:text=Selendroid%20is%20a%20test%20automation,the%20Selenium%20Webdriver%20client%20API. [Accessed: 10-Jun-2024].

[121] “Appium vs Selendroid,” *GeeksforGeeks*, 10-Dec-2023. Available: https://www.geeksforgeeks.org/appium-vs-selendroid/. [Accessed: 10-Jun-2024].

References (continued)

[122] adegeo, “Install .NET on Windows - .NET,” *Microsoft Learn*, 20-Jun-2024. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/install/windows. [Accessed: 29-Aug-2024].

[123] tdykstra, “.NET CLI,” 20-Jun-2024. *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/tools/. [Accessed: 29-Aug-2024].

[124] tdykstra, “dotnet command - .NET CLI,” *Microsoft Learn*, 11-Oct-2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/tools/dotnet. [Accessed: 29-Aug-2024].

[125] tdykstra, “dotnet test command - .NET CLI,” *Microsoft Learn*, 27-Mar-2024. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/tools/dotnet-test. [Accessed: 29-Aug-2024].

[126] “Using xUnit to Test your C# Code,” *Auth0 - Blog*. [Online]. Available: https://auth0.com/blog/xunit-to-test-csharp-code/. [Accessed: 29-Aug-2024].

[127] B. Code, “Unit Testing in C# with xUnit: Complete guide,” *Medium*, 11-Apr-2024. [Online]. Available: https://medium.com/@codebob75/unit-testing-in-c-with-xunit-complete-guide-18ee2b919b05. [Accessed: 29-Aug-2024].

[128] “Getting started: .NET Core with command line,” *xUnit.net*. [Online]. Available: https://xunit.net/docs/getting-started/v2/netcore/cmdline. [Accessed: 29-Aug-2024].

[129] “Getting started: .NET Framework with command line,” *xUnit.net*. [Online]. Available: https://xunit.net/docs/getting-started/v2/netfx/cmdline. [Accessed: 29-Aug-2024].

[130] “Capturing Output,” *xUnit.net*. [Online]. Available: https://xunit.net/docs/capturing-output. [Online]. [Accessed: 29-Aug-2024].

[131] erikdietrich, “The history of C#,” *Microsoft Learn*, 07-Mar-2024. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history. [Accessed: 29-Aug-2024].

[132] “Download - Stable | Mono.” [Online]. Available: https://www.mono-project.com/download/stable/. [Accessed: 29-Aug-2024].

[133] “Microsoft Offloads The Mono Project To Wine.” [Online]. Available: https://www.phoronix.com/news/Microsoft-Gives-Mono-To-Wine. [Accessed: 29-Aug-2024].

[134] “How YOU can get started with .NET Core and C# in VS Code.” [Online]. Available: https://softchris.github.io/pages/dotnet-core.html#resources. [Accessed: 30-Aug-2024].

[135] “Home,” *xUnit.net*. [Online]. Available: https://xunit.net/. [Accessed: 03-Sept-2024].

[136] “Apache License, Version 2.0,” *Open Source Initiative*. [Online]. Available: https://opensource.org/license/apache-2-0. [Accessed: 03-Sept-2024].

[137] “Why Did We Build xUnit 1.0?,” *xUnit.net*. [Online]. Available: https://xunit.net/docs/why-did-we-build-xunit-1.0. [Accessed: 03-Sept-2024].

[138] “Using xUnit to Test your C# Code,” *Auth0 - Blog*. [Online]. Available: https://auth0.com/blog/xunit-to-test-csharp-code/. [Accessed: 03-Sept-2024].

[139] “Getting Started: .NET Framework with Visual Studio,” *xUnit.net*. [Online]. Available: https://xunit.net/docs/getting-started/v2/netfx/visual-studio. [Accessed: 03-Sept-2024].

[140] S. Dhakne, “Why Should You Use xUnit? A Unit Testing Framework For .NET.” [Online]. Available: https://www.clariontech.com/blog/why-should-you-use-xunit-a-unit-testing-framework-for-.net. [Accessed: 03-Sept-2024].

[141] “NUnit vs xUnit vs MSTest: .NET Unit Testing Framework Comparison.” [Online]. Available: https://daily.dev/blog/nunit-vs-xunit-vs-mstest-net-unit-testing-framework-comparison. [Accessed: 03-Sept-2024].

References (continued)

[142] A. Kumar, “Unit Testing with xUnit.” [Online]. Available: https://www.c-sharpcorner.com/article/unit-testing-with-xunit-in-net/. [Accessed: 03-Sept-2024].

[143] “Testing with the xUnit Framework - Overview (2 of 12),” *Microsoft Learn*, 29-Nov-2022. [Online]. Available: https://learn.microsoft.com/en-us/shows/visual-studio-toolbox/testing-with-the-xunit-framework-overview-2-of-12-automated-software-testing. [Accessed: 03-Sept-2024].

[144] “Testing with the xUnit Framework - Overview (2 of 12),” *Microsoft Learn*, 29-Nov-2022. [Online]. Available: https://learn.microsoft.com/en-us/shows/visual-studio-toolbox/testing-with-the-xunit-framework-overview-2-of-12-automated-software-testing. [Accessed: 03-Sept-2024].

[145] V. Magalhães, “Test Coverage Analysis with Coverlet in .NET,” *Medium*, 23-Sept-2023. [Online]. Available: https://victormagalhaes-dev.medium.com/test-coverage-analysis-with-coverlet-in-net-2e38df3c6ed7. [Accessed: 03-Sept-2024].

[146] G. Barré, “Computing code coverage for a .NET project - Gérald Barré,” *Meziantou’s blog*, 15-Apr-2024. [Online]. Available: https://www.meziantou.net/computing-code-coverage-for-a-dotnet-project.htm. [Accessed: 03-Sept-2024].

[147] gewarren, “dotnet-coverage code coverage tool - .NET CLI - .NET”, 24-Feb-2023. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/additional-tools/dotnet-coverage. [Accessed: 04-Sept-2024].

[148] “ReportGenerator - Code coverage reports.” [Online]. Available: https://reportgenerator.io/. [Accessed: 04-Sept-2024].

[149] E. Dahl, "Code Coverage Reports for .NET Projects.” [Online]. Available: https://knowyourtoolset.com/2024/01/coverage-reports/. [Accessed: 04-Sept-2024].

[150] “Code Coverage in .NET,” *my tech ramblings*, 10-Jul-2024. [Online]. Available: https://www.mytechramblings.com/posts/code-coverage-in-dotnet/. [Accessed: 04-Sept-2024].

[151] C. Allen, “Generating Code Coverage Metrics for .NET Framework Applications,” *Coding with Calvin - Calvin Allen*, 18-Aug-2022. [Online]. Available: https://www.codingwithcalvin.net/generating-code-coverage-metrics-for-net-framework-applications/. [Accessed: 04-Sept-2024].

[152] J. Carracedo, “Code coverage in .Net Core,” *Medium*, 09-May-2021. [Online]. Available: https://jke94.medium.com/code-coverage-in-net-core-840bcd96b1e9. [Accessed: 04-Sept-2024].

[153] “NET Core Code Coverage,” *DEV Community*, 07-Mar-2021. [Online]. Available: https://dev.to/eduardstefanescu/net-core-code-coverage-1dfn. [Accessed: 04-Sept-2024].

[154] L. Bennett, “9 Best Code Coverage Tools for Java, Python, C, C++, C#, .NET,” 13-Aug-2024. [Online]. Available: https://www.guru99.com/code-coverage-tools.html. [Accessed: 04-Sept-2024].

[155] “Cobertura.” [Online]. Available: https://cobertura.github.io/cobertura/. [Accessed: 04-Sept-2024].

[156] R. Singh, “What is Cobertura and use cases of Cobertura ?,” *DevOpsSchool.com*, 15-Feb-2024. [Online]. Available: https://www.devopsschool.com/blog/what-is-cobertura-and-use-cases-of-cobertura/. [Accessed: 04-Sept-2024].

[157] “ReportGenerator - Code coverage reports.” [Online]. Available: https://reportgenerator.io/usage. [Accessed: 04-Sept-2024].

References (continued)

This page intentionally left blank.

End of document