Decision Analysis Recommendation - Unit Testing

Purpose

The purpose of this document is to recommend a framework that will perform unit tests on our application.

Technology Comparisons

- MSTest version 2.28
- NUnit version 3.13.2
- xUnit version 2.4.1

Metrics

Test Detection

- Analyzes the ability of each unit testing framework to detect test methods. For this metric we measured the time each framework took to detect 100 test methods. To account for variation, we performed 10 trials and took the average.

Parallel Testing Capability

- Parallel testing capability refers to whether or not the framework has the capability of testing multiple tests in parallel. We chose this metric as parallel testing significantly reduces test times.

Data Driven Testing Capability

- Data driven testing in a framework allows us to test different test functions with multiple sets of data. We chose this metric as this increases efficiency in testing.

Test Isolation

- Test isolation reduces the ability for outside factors to affect individual unit tests. This is important as this will look at bugs on a lower level as opposed to integration tests that will look at a higher level.

Sample Thrown Execution Speed

- We wanted to test the speed it took for each framework to detect a thrown exception. Each test determined whether the exception thrown was the correct exception. The execution speed was taken using the test view provided by Microsoft Visual Studio.

Benchmark Data

I. Test Detection

For each framework, 100 of the same tests with the following code were pasted into visual studio for each framework. We then measured the speed using a stopwatch.

```
Public void TestFunctionN(){
        Assert.Equals(1, 1);    // or Assert.Equal(1,1);
}
```

Table. 1 Detection Time for Each Framework

Trial	MSTest (s)	NUnit (s)	xUnit (s)
Trial 1	7.86	3.35	3.13
Trial 2	7.28	4.29	2.42
Trial 3	5.44	3.55	2.53
Trial 4	6.72	2.98	4.1
Trial 5	8.33	3.19	3.14
Trial 6	8.61	3.22	2.98
Trial 7	5.63	3.89	2.42
Trial 8	6.32	2.99	3.2
Trial 9	6.13	3.4	2.86
Trial 10	6.24	3.2	2.81
Average Time	6.86	3.41	2.96

II. Sample Thrown Execution Speed

Metric	MSTest		nUnit		xUnit	
	Pass	Fail	Pass	Fail	Pass	Fail
Thrown Execution Speed	7 ms	128 ms	17 ms	143 ms	6 ms	108 ms

Analysis Between MSTest, NUnit and xUnit

Metric	MSTest	NUnit	xUnit
Test Detection (0.55)	Detected 100 tests within 6.86 s (0.75)	Detected 100 tests within 3.41 s (0.95)	Detected 100 tests within 2.96 s (1)
Parallel Testing Capability (1)	Not by default (0.5)	Not by default (0.5)	Yes (1)
Data Driven Testing Capability (0.70)	Yes (1)	Yes (1)	Yes (1)
Test Isolation (0.75)	Creates a new instance for each test method (1)	Uses same instance to run all of the test methods (0.5)	Creates a new instance for each test method (1)
Sample Thrown Execution Speed (.25)	Executed test within 7 ms (0.95)	Executed test within 17 ms (0.5)	Executed test within 6 ms (1)
Total Weight	4.2	2.45	5

Recommendation

Based on the analysis between MSTest, NUnit and xUnit, it is clear that xUnit outperforms the others in the metrics we chose. Since we cared the most about parallel testing capabilities and test isolation, we emphasized the score more on those two. While MSTest performed close to xUnit, it did not do as well as xUnit in the two metrics mentioned. Since xUnit scored the most in all categories, we are recommending xUnit.