Test Plan

RE: Technology One Technical Test

Submitted by: Eugene Ordeniza

Contents

1. Document Overview

1.1 Purpose

This document defines the testing strategy, scope, and procedures for the Conversion Algorithm in NumericalTranslator.razor.cs file within the TechOne Technical Test project. This document intends to ensure the correctness of the numeral-to-text logic, performance, and error handling.

1.2 Scope

In Scope:

1. Convert Whole Numbers:
    a. Translate integers from 0 up to the maximum value supported by a “long” data type (Sextillion range).
    b. Handle numbered structure grouping (e.g. Thousands, Millions, Billions, Trillions, etc.).
2. Handling Decimal Values
    a. When provided with a decimal value (e.g., 123.45), convert the part after the decimal point to represent cents (e.g., "FORTY-FIVE CENTS").
3. Handling Zero Gracefully
    a. Return “Zero” if the value is 0
4. Handling Negative Values
    a. Prefix the output with “Negative” to denote a number below 0
5. Return Readable English Output
    a. Include proper use of “and” for grammatical correctness (e.g., One Hundred and One).
    b. Use hyphens for compound numbers below 100 (e.g., Twenty-One).
6. Support Large-Scale Grouping
    a. Recognise numeric groups up to at least **Sextillion** (10^21).

Out of Scope:

1. Non-numeric Inputs
    a. The function assumes the input is a valid number. Validation is offloaded but will still be handled gracefully.
2. Floating-Point Rounding
3. Localisation or Multiple Currencies
4. Custom Formatting or Styling
    a. Output is expected to be of plain-text, viewable by the UI
5. Invalid Range Handling
    a. Extreme values like the minimum of long are not a realistic number for the use case of dollars and cents.

2. Test Strategy

2.1 Testing Levels

These are the levels at which testing was performed:

LEVEL	DESCRIPTION	EXAMPLE
UNIT	Tests smallest code units	ConversionAlgorithm(123)
SYSTEM	Tests entire web-system	User Submission Form

2.2 Testing Types

These are the types of tests within the unit-level of testing:

- Single-Digit Tests
- Teen and Tens Tests
- Hundreds Tests
- Thousand Tests
- Millions and Billions Tests
- Trillions and Beyond
- Negative Number Tests
- Miscellaneous Tests (Hyphenation, Spacing and Zero Handling)
- Edge Cases (Long maximums and minimums)

3. Test Environment

- Hardware/OS: Windows 10+, .NET 8 SDK
- Tools: Visual Studio 2022, MSTest Framework
- Dependencies: Newtonsoft.Json (13.0.4), Microsoft.AspNetCore.Components (8.0.0), Microsoft.NET.Test.Sdk (17.12.0), MSTest.TestAdapter (3.4.0)
- Database/API: None

3.1 Environment Setup

```
dotnet restore
dotnet build
dotnet test --logger:"console;verbosity=detailed"
```

4. Test Cases

For unit test cases, please refer to Table 1 in the Appendix.

5. Entry and Exit Criteria

Entry Criteria

#	Entry Criteria	Description
1	Application Build Complete	The Number-to-Words converter web application compiles successfully without build errors.
2	Functional Requirements Defined	All core functionalities (numeric input, conversion algorithm, text output) are clearly documented.
3	Unit Test Framework Setup	MSTest framework is properly configured and test discovery works within Visual Studio and CLI (dotnet test).
4	Conversion Algorithm Implemented	The ConversionAlgorithm() function for converting integers to words is implemented.
5	Test Data Prepared	Representative test data covering positive, negative, zero, and edge cases (e.g., billions, sextillions) is available.
6	Development Environment Ready	All necessary development tools (.NET SDK, IDE, and dependencies) are installed and functional.

Exit Criteria

#	Exit Criteria	Description
1	All Planned Tests Executed	100% of defined test cases (unit tests) have been run.
2	All Critical Tests Passed	All high-priority tests (basic digits, hundreds, thousands, and negatives) pass successfully.
3	Major Bugs Resolved	No critical or high-severity defects remain unresolved.
4	Coverage Threshold Met	Unit test coverage reaches at least 80% of all logical branches in the conversion algorithm.
5	Output Format Verified	All outputs match expected wording, including correct hyphenation and spacing.
6	Code Review Completed	Conversion algorithm and test code have passed peer review for readability and maintainability.
7	Documentation Delivered	Test Plan, Approach Document, and README are complete and approved.

6. Risk Mitigation Plan

#	Risk Description	Impact	Likelihood	Mitigation Strategy	Contingency Plan
1	**Incorrect word conversion logic** — e.g., missing "and" or incorrect hyphen placement.	High	Medium	Implement detailed unit tests for all number ranges (1–999, 1000–999,999, etc.)	Perform manual validation of edge outputs (e.g., 101, 110, 1005) before final submission.
2	**Unrecognized or negative input not handled properly** (e.g., -1 or 0).	Medium	Medium	Include validation logic and test cases for negative and zero inputs.	Default to user-friendly messages such as "Negative One" or "Zero."
3	**Performance degradation for very large numbers (e.g., quintillions).**	Medium	Low	Optimize recursive algorithm to avoid redundant calls.	Fallback to iterative logic if recursion exceeds stack limits.
4	**Compatibility issue with MSTest framework or .NET SDK** causing test discovery failure.	High	Low	Pin versions in .csproj and verify tests run via CLI (dotnet test).	Switch temporarily to xUnit or NUnit if MSTest remains unstable.
5	**Build or dependency conflict** (e.g., missing Newtonsoft.Json).	Medium	Medium	Use explicit version bindings and run dotnet restore regularly.	Clean and rebuild project; if unresolved, manually reinstall missing packages.
6	**UI not displaying conversion output correctly** due to binding or form issues.	High	Medium	Test frontend manually using various inputs. Use developer console to verify data binding.	Log errors and fallback to basic HTML output display.
7	**Deadline pressure reduces test coverage or documentation quality.**	High	Medium	Prioritize high-value test cases and outline documentation early.	Submit minimal viable solution, document known limitations, and plan post-submission fixes.
8	**Algorithm does not handle decimals (e.g., 123.45 for currency conversion).**	Medium	High	Extend algorithm with fractional parsing (split by ".").	Return whole number first, append "cents" logic later if time allows.

Appendix

Table 1: Test Case Table – Number to Words Conversion

Test ID	Test Description	Input	Expected Output	Category	Result
TC001	Convert single digit 1	1	One	Single Digit	Pass
TC002	Convert single digit 9	9	Nine	Single Digit	Pass
TC003	Convert 19	19	Nineteen	Single Digit	Pass
TC004	Convert 12 (teen)	12	Twelve	Teen	Pass
TC005	Convert 17 (teen)	17	Seventeen	Teen	Pass
TC006	Convert 10	10	Ten	Tens	Pass
TC007	Convert 20	20	Twenty	Tens	Pass
TC008	Convert 29	29	Twenty-Nine	Tens	Pass
TC009	Convert 80	80	Eighty	Tens	Pass
TC010	Convert 33	33	Thirty-Three	Tens	Pass
TC011	Convert 99	99	Ninety-Nine	Tens	Pass
TC012	Convert 100	100	One Hundred	Hundreds	Pass
TC013	Convert 200	200	Two Hundred	Hundreds	Pass
TC014	Convert 123	123	One Hundred and Twenty-Three	Hundreds	Pass
TC015	Convert 105	105	One Hundred and Five	Hundreds	Pass
TC016	Convert 110	110	One Hundred and Ten	Hundreds	Pass
TC017	Convert 215	215	Two Hundred and Fifteen	Hundreds	Pass
TC018	Convert 999	999	Nine Hundred and Ninety-Nine	Hundreds	Pass
TC019	Convert 1000	1000	One Thousand	Thousands	Pass

TC020	Convert 2000	2000	Two Thousand	Thousands	Pass
TC021	Convert 20000	20000	Twenty Thousand	Thousands	Pass
TC022	Convert 200000	200000	Two Hundred Thousand	Thousands	Pass
TC023	Convert 200005	200005	Two Hundred Thousand Five	Thousands	Pass
TC024	Convert 1005	1005	One Thousand Five	Thousands	Pass
TC025	Convert 1001	1001	One Thousand One	Thousands	Pass
TC026	Convert 1234	1234	One Thousand Two Hundred and Thirty-Four	Thousands	Pass
TC027	Convert 10000	10000	Ten Thousand	Thousands	Pass
TC028	Convert 100000	100000	One Hundred Thousand	Thousands	Pass
TC029	Convert 120305	120305	One Hundred and Twenty Thousand Three Hundred and Five	Recursive	Pass
TC030	Convert 1001001	1001001	One Million One Thousand One	Recursive	Pass
TC031	Convert 1000000	1000000	One Million	Millions	Pass
TC032	Convert 1000001	1000001	One Million One	Millions	Pass
TC033	Convert 10000000	10000000	Ten Million	Millions	Pass

TC034	Convert 100000000	100000000	One Hundred Million	Millions	Pass
TC035	Convert 100000001	100000001	One Hundred Million One	Millions	Pass
TC036	Convert 9876543210	9876543210	Nine Billion Eight Hundred and Seventy-Six Million Five Hundred and Forty-Three Thousand Two Hundred and Ten	Billions	Pass