Triangle\_Classifier Application: Test Plan

Test Plan draft ChatGPT LLM generated then manually updated

Python Application and Test code draft Copilot LLM generated then manually updated

**1. Introduction**

The **Triangle\_Classifier** application reads input strings and classifies them as one of the following types of triangles: "Scalene", "Isosceles", "Equilateral", or "NoTriangle". The application also handles error cases and exits on specific input commands ("Exit" or "Quit"). This test plan outlines the steps required to qualify the application for delivery, including all test suites and cases, pass/fail criteria, and test data.

**Application Specifics**

**Inputs**: New line separated strings. Words in a string are separated by white space.

WHEN there are three separate words in a string, AND when each word is a number, then the application outputs one word.

**Outputs**: one word from the set of words "Scalene", "Isosceles", "Equilateral", or "NoTriangle” on stdout. **Error** strings on stderr.

**Operation**: The choice of the output word is made by interpreting each number as a length of a side of a triangle.

In **ALL OTHER** cases the output is an error string indicating that the input could not be classified.

The error string also indicates the cause of the error.

Causes include what it is about the string that it cannot be classified OR execution errors.

The program exits with a stdout done message when the input string contains the word “**Exit**” or the word “**Quit**” (ignore case) at the beginning of the string.

Examples:

Input “1 1 1” Output “Equilateral”

Input “1” Output “Error: Two sides missing”

Input “Quit” Application exits.

**1.1 Objective**

The objective of this test plan is to ensure that the **Triangle\_Classifier** application:

* Correctly classifies valid triangle inputs.
* Accurately reports errors for invalid input strings.
* Properly exits on receiving "Exit" or "Quit".
* Functions as expected under various input scenarios.

**1.2 Scope**

The tests will cover:

- Functional validation for correct triangle classification.

- Error handling for invalid input strings.

- Proper application termination when given "Exit" or "Quit".

**2. Test Suites and Test Cases**

**2.1 Test Suite 1: Valid Triangle Classification**

**Test Case 1.1: Equilateral Triangle Integer**

**Description**: Verify that the application correctly identifies equilateral triangles where all three sides are equal.

* **Input**: `1 1 1`
* **Expected Output**: `Equilateral`
* **Pass Criteria**: Application outputs "**Equilateral**".

**Test Case 1.2: Isosceles Triangle Integer**

**Description** Verify that the application correctly identifies isosceles triangles where two sides are equal.

* **Input**: `2 2 3`
* **Expected Output** `Isosceles`
* **Pass Criteria**: Application outputs "Isosceles".

**Test Case 1.3: Scalene Triangle Integer**

**Description**: Verify that the application correctly identifies scalene triangles where all three sides are different.

* **Input**: `5 3 4`
* **Expected Output**: `Scalene`
* **Pass Criteria**: Application outputs "Scalene".

**Test Case 1.4: NoTriangle Integer**

**Description**: Verify that the application correctly identifies cases where the three sides do not form a triangle.

* **Input**: `2 3 1`
* **Expected Output**: `NoTriangle`
* **Pass Criteria**: Application outputs "NoTriangle".

**Test Case 1.5: Equilateral Triangle Float**

**Description**: Verify that the application correctly identifies equilateral triangles where all three sides are equal with tab white space

* **Input**: `12.3 12.3 12.3`
* **Expected Output**: `Equilateral`
* **Pass Criteria**: Application outputs "**Equilateral**".

**Test Case 1.6: Isosceles Triangle Float**

**Description** Verify that the application correctly identifies isosceles triangles where two sides are equal.

* **Input**: `45.6 23.4 23.4
* **Expected Output** `Isosceles`
* **Pass Criteria**: Application outputs "Isosceles".

**Test Case 1.7: Scalene Triangle Float**

**Description**: Verify that the application correctly identifies scalene triangles where all three sides are different with tab white space

* **Input**: `3.6 5.9 4.8`
* **Expected Output**: `Scalene`
* **Pass Criteria**: Application outputs "Scalene".

**Test Case 1.8: NoTriangle Float**

**Description**: Verify that the application correctly identifies cases where the three sides do not form a triangle

* **Input**: `7.8 1.1 2.3 `
* **Expected Output**: `NoTriangle`
* **Pass Criteria**: Application outputs "NoTriangle".

**Test Case 1.9: NoTriangle Zero**

**Description**: Verify that the application correctly identifies cases where at least one of three sides is zero, so does not form a triangle with tab white space

* **Input**: `1.1 2 0.0`
* **Expected Output**: `NoTriangle`
* **Pass Criteria**: Application outputs "NoTriangle".

**Test Case 1.10: NoTriangle Negative**

**Description**: Verify that the application correctly identifies cases where at-least one of three sides is negative, so does not form a triangle with tab white space

* **Input**: `7.7 -0.13 7.8`
* **Expected Output**: `NoTriangle`
* **Pass Criteria**: Application outputs "NoTriangle".

**2.2 Test Suite 2: Error Handling**

**Test Case 2.1: Missing Two Sides only**

**Description**: Verify that the application returns an error when fewer than three numbers are provided.

* **Input**: `1.1`
* **Expected Output**: `Error: Two sides missing`
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 2.2: Missing One Side only**

**Description**: Verify that the application returns an error when only two numbers are provided.

* **Input**: `5.6 10`
* **Expected Output**: `Error: One side missing`
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 2.3: Missing Three Sides only**

**Description**: Verify that the application returns an error when no numbers are provided.

* **Input**: ` `
* **Expected Output**: `Error: Three sides missing`
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 2.4: More than Three Sides numbers**

**Description**: Verify that the application returns an error when more than three numbers are provided.

* **Input**: `1 2 3.1 5.6`
* **Expected Output**: `Error: More than Three words provided`
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 2.5: Invalid three non-numbers**

**Description**: Verify that the application returns an error when only non-numeric input is provided.

* **Input**: `all three words`
* **Expected Output**: `Error: Non-numeric words`
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 2.6: Invalid two non-numbers**

**Description**: Verify that the application returns an error when only non-numeric input is provided.

* **Input**: `only 2two`
* **Expected Output**: `Error: Non-numeric words`
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 2.7: More than three non-numbers**

**Description**: Verify that the application returns an error when only non-numeric input is provided.

* **Input**: More 1than three3 inputs`
* **Expected Output**: `Error: More than Three words provided `
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 2.8: Missing One Side with non- number**

**Description**: Verify that the application returns an error when only two numbers are provided.

* **Input**: `5.6 number 10`
* **Expected Output**: `Error: One side missing with Non-numeric words`
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 2.9: Missing Two Sides with 1 non- number**

**Description**: Verify that the application returns an error when fewer than three numbers are provided.

* **Input**: `non 1.1`
* **Expected Output**: `Error: Two sides missing with Non-numeric words`
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 2.10: Missing Two Sides with 2 non- numbers**

**Description**: Verify that the application returns an error when fewer than three numbers are provided.

* **Input**: `non 1.1 number`
* **Expected Output**: `Error: Two sides missing with Non-numeric words`
* **Pass Criteria**: Application outputs the correct error message.

**2.3 Test Suite 3: Application Termination**

**Test Case 3.1: Quit Command**

**Description**: Verify that the application exits when the input string only has "Quit".

* **Input**: `Quit`
* **Expected Output**: triangle classifier done.
* **Pass Criteria**: Application terminates successfully with output message

**Test Case 3.2: Quit second word**

**Description**: Verify that the application has Error when the input string has "Quit" as second input.

* **Input**: `1 Quit 3`
* **Expected Output**: Error: One side missing with Non-numeric words
* **Pass Criteria**: Application outputs the correct error message

**Test Case 3.3: Quit first lowercase**

**Description**: Verify that the application exits when the input string on has "quit" as first word

* **Input**: `quit 2 3 `
* **Expected Output**: triangle classifier done.
* **Pass Criteria**: Application terminates successfully with output message

**Test Case 3.4: Quit starts first word**

**Description**: Verify that the application has Error when the input string has "Quit" as second input.

* **Input**: `Quit1 4 3.5`
* **Expected Output**: Error: One side missing with Non-numeric words
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 3.5: Exit Command**

**Description**: Verify that the application exits when the input string contains "Exit".

* **Input**: `Exit`
* **Expected Output**: triangle classifier done.
* **Pass Criteria**: Application terminates successfully with output message

**Test Case 3.6: Exit third word**

**Description**: Verify that the application has Error when the input string has "Exit" as third input.

* **Input**: `1 A Exit`
* **Expected Output**: Error: Two sides missing with Non-numeric words
* **Pass Criteria**: Application outputs the correct error message.

**Test Case 3.7: Exit first uppercase**

**Description**: Verify that the application exits when the input string contains "Exit".

* **Input**: `EXIT 3. 2`
* **Expected Output**: triangle classifier done.
* **Pass Criteria:** Application terminates successfully with output message

**Test Case 3.8: Exit ends first word**

**Description**: Verify that the application has Error when the input string has "Quit" as second input.

* **Input**: `1Exit 4 3.5`
* **Expected Output**: Error: One side missing with Non-numeric words
* **Pass Criteria:** Application outputs the correct error message.

**Test Case 3.9: Valid Invalid then Quit**

**Description**: Verify that the has Valid and Invalid Triangles then exits

* Input lines
  1. 23 34 45.6
  2. 23.4 78.4 19
  3. 1.0 3.0
  4. Quit
* Expected stdout lines
  1. Scalene
  2. NoTriangle
  3. triangle classifier done.
* Expected stderr lines
  1. Error: Missing One side
* Pass Criteria: Application terminates successfully with output and error messages

**Test Case 3.10: Valid Invalid then Exit**

**Description**: Verify that the has Valid and Invalid Triangles then exits

* Input lines

2. 3.4 5.6 3.40
3. 4.8
4. 5.7 5.70 5.700
5. Exit

* Expected stdout lines
  1. Isosceles
  2. Equilateral
  3. triangle classifier done.
* Expected stderr lines

1. Error: Missing Three sides
2. Error: Missing Two sides

* Pass Criteria: Application terminates successfully with output and error messages

**2.4 Test Suite 4: Boundary and Security**

**Test Case 4.1: Exit then data**

**Description**: Verify that the application exits and ignores input after exiting".

* Input lines

1. Exit
2. 3.1 5.2 8.01
3. 8.1

* Expected stdout lines

1. triangle classifier done

* Expected stderr line: None
* Pass Criteria: Application exits with done message and no other stdout/stderr data

**Test Cases float precision digits and decimals**

Noticed that this **Triangle\_Classifier** implementation has 15 digit-decimal precision

Test will validate that implementation round to 15 digit-decimal difference precision

Test Case table for 15 digit-decimal precision

* Input: via **Input** column
* Expected Output: via **Output** column.
* Pass Criteria: Application output matches **Expected Output**

|  |  |  |  |
| --- | --- | --- | --- |
| Test id | Test info; notes | Input | Output |
| 4\_02\_prec | 15 digits all same  Note: no rounding | 999999999999999 999999999999999 999999999999999 | Equilateral |
| 4\_03\_prec | 15 digits different values:  Note no rounding | 999999999999990 999999999999991 999999999999992 | Scalene |
| 4\_04\_prec | 16 digits same first 15 digits  Note: some numbers round | 9999999999999990 9999999999999991 9999999999999992 | Isosceles |
| 4\_05\_prec | 17 digits same first 16 digits  Note: some numbers round | 99999999999999990 99999999999999991 99999999999999992 | Isosceles |
| 4\_06\_prec | 18 digits same first 17 digits  Note: all numbers round | 999999999999999990 999999999999999991 999999999999999992 | Equilateral |
| 4\_07\_prec | 15 digits 14 decimals  Note: no rounding | 3.00000000000001 5.000 8.00 | Scalene |
| 4\_08\_prec | 16 digits 15 decimals  Note: rounding to 14 digits | 3.000000000000001 5.000 8.00 | NoTriangle |
| 4\_09\_prec | 15 digits 1 decimals vs 1 digits 0 decimals  Note: no rounding | 567891234567893.1 5 567891234567898 | Scalene |
| 4\_10\_prec | 15 digits 2 decimals vs 1 digits 0 decimals  Note: Round to 1 decimal | 567891234567893.01 5.00 567891234567898.00 | NoTriangle |
| 4\_11\_prec | 16 digits 0 decimal vs 1 digit 0 decimal  Note: no add round as have 15 digits difference | 4567891234567890 9 4567891234567898 | Scalene |
| 4\_12\_prec | 17 digits 0 decimal vs 1 digit 0 decimal  Note: add rounded ignoring the 1 digit num | 34567891234567890 9 34567891234567898 | NoTriangle |

**Test Cases other input formats**

Test will validate that application handles other input formats correction

With no **security issues** and no **execution internal** errors

These formats include: lists, xml, Json, special and Unicode values

Test Case table

* Input: via **Input** column
* Expected Output: via **Output** column when start with ‘Error:’ then error message otherwise output message
* Pass Criteria: Application terminates successfully with output and error messages

|  |  |  |  |
| --- | --- | --- | --- |
| Test id | Test info; notes | Input | Output |
| 4\_21\_other | Special Chars;  Note: special chars string is a word | 1 9 !@#$%^&\*(),.<>?/\\|`~{}[]'\" | Error: One side missing with Non-numeric words |
| 4\_22\_other | List of 3 numbers  Note: split into words the number end in , or ] | [1.5, 2.4, 3.6] | Error: Three sides missing with Non-numeric words |
| 4\_23\_other | List of 3 number and letter  Note: split into words has only 1 number | [1.5, 2.4 ,'abc'] | Error: Two sides missing with Non-numeric words |
| 4\_24\_other | XML tag with value  Note: splits into 4 words | <a>triangle 12 31 23</a> | Error: More than Three words provided |
| 4\_25\_other | JSON tag with value  split into words has 0 numbers | {a:1, b:true, c:'string'} | Error: Three sides missing with Non-numeric words |
|  |  |  |  |
| Note: | Space (\u0020) and Tab (\u0008)  Are in other test cases  So will not duplicate those tests |  |  |
| 4\_30\_other | Backspace and Unicode  Note: backspace is \b and \u0008  Linefeed is not a split white-space character | 7.4\u0008 1.56\b 3.89 | Error: Two sides missing with Non-numeric words |
| 4\_31\_other | Linefeed and Unicode  Note: Linefeed is \u000a  Linefeed is a split white-space character | 4.7\n 1.56\n 3.89\u000a | Scalene |
| 4\_32\_other | Formfeed and Unicode  Note: Formfeed is \f and \u000c  Formfeed is a split white-space character | 4.12 4.12\f 3.89\u000c | Isosceles |
| 4\_33\_other | Carriage Return and Unicode  Note: Carriage Return is \r and \u000d  Carriage Return is a split white-space character | 7.4\r 1.56\u000d 3.89 | NoTriangle |
| 4\_34\_other | Double Quote and Unicode  Note: Double Quote is \” and \u0022  Double Quote is not a split white-space character | 4.12\u0022 1.56\" 3.89 | Error: Two sides missing with Non-numeric words |
| 4\_35\_other | Single Quote and Unicode  Note: Single Quote is \’ and \u0027  Single Quote is not a split white-space character | 4.12\' 1.56\u0027 3.89 | Error: Two sides missing with Non-numeric words |
| 4\_36\_other | Backslash and Unicode  Note: Backslash is \\ and \u005c  Backslash is not a split white-space character | 4.12 1.56\\ 3.89\u005c | Error: Two sides missing with Non-numeric words |
| 4\_37\_other | Unicode Letter A  Note: Letter A is \u0041  Letter A makes its word non-numeric | 4.12\u0041 1.56 3.89 | Error: One side missing with Non-numeric words |
| 4\_38\_other | Unicode Number 6  Note: Number 6 is \u0036  Number 6 keep numeric words as numeric | 4.12 1.5\u0036 3.89 | Scalene |
| 4\_39\_other | Unicode UK Pound symbol  Note: UK Pound symbol is \u00A3  UK Pound makes its word non-numeric | 4.12 1.56 3.89\u00A3 | Error: One side missing with Non-numeric words |

**2.5 Test Suite 5: Load, Duration, Performance**

This test plan does specify Load, Duration or Performance test cases.

The following outlines how to create data and executed these tests

1. Create large on demand good input files that has lines with 3 numbers for different triangle types (NoTriangle, Equilateral, Isosceles, Scalene) and last line has Exit or Quit
2. Create large on demand error input file that has lines 3 words (numeric and non-numeric) and last line has Exit or Quit
3. Would like to that input file that has same output (stdout or stderr). This way able to check duration and load for same output.
4. Record the Machine’s environment (chip-type, speed, CPUs, RAM, OS, etc.)
5. Record Application build-id and current date-time
6. Ability to record application execution information (raw-time, cpu-time, memory usage, CPU-usage\_

**Performance tests**

Run application with each generated good and error files once checking that good file response are only stdout and error file response are stderr except for exit message on stdout

Input sends rate options:

* No delay after getting some response (check response later)
* No delay after check response
* Some delay after get and/or check response

Record number of input file lines and application execution information for charting good and error values

Could run parallel input-same-output file against application with expecting same output for each unique file.

Example:

* Input good Scalene file would have 3 numbers for Scalene triangles
* input good NoTriangle file would have 3 numbers for No triangles
* input Error only 2 numbers file would have lines only 2 numbers
* input Error 1 number and 2 non-numbers would have lines with 1 number and 2 non-numbers

**Duration Tests**

Similar to Performance with running input over long at some constant input delay rate (maybe none)

With monitoring application execution information periodically to check for memory leaks, CPU runaways, corrupted response data, etc.

This is normal done via running input files over and over again for the duration.

**Load Tests**

Similar to Performance with lower the input delay rate or adding more input-driver over time

To find when the response time for file and individual messages become unacceptable.

**3. Test Execution Sequence**

The test cases should be executed in the following sequence:

1. **Test Suite 1**: Validate classification of correct inputs.

2. **Test Suite 2**: Check for various error conditions and input validation.

3. **Test Suite 3**: Test application termination commands.

4. **Test Suite 4**: Test boundary and security cases.

5. **Test Suite 5**: Test load duration and performance case when build or environment changes.

**4. Criteria for Delivery**

The application will be considered qualified for delivery if:

- All test cases in **Test Suite 1** pass, meaning valid triangles are correctly classified.

- All test cases in **Test Suite 2** pass, meaning errors are accurately detected and reported.

- All test cases in **Test Suite 3** pass, meaning the application exits correctly on receiving "Exit" or "Quit".

- All test cases in **Test Suite 4** pass, meaning the application exits correctly boundary and security Cases

- All test cases in **Test Suite 5** pass with recording load, duration and performance results to compare with other builds.

**5. Pass/Fail Criteria**

Each test case is deemed to pass if the actual output matches the expected output. For error cases, the output message must correctly indicate the reason for failure, and for valid cases, the classification of the triangle must be correct.

- **Pass**: Application behavior matches the expected behavior.

- **Fail**: Any deviation from the expected behavior, including incorrect classification, missing output, or application crashes.

**6. Data Required to Determine Pass/Fail**

The pass/fail outcome will be determined based on:

- The correct classification output (one of: "Scalene", "Isosceles", "Equilateral", "NoTriangle").

- The error message content and format.

- The ability of the application to properly terminate on "Quit" or "Exit".

All outputs will be manually compared against expected results. Logs (if any) can also be used to trace and verify correct execution paths.

**DONE**