**Practical :** Basic Math Operation Functionality and Usability Testing

**Test Cases Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Test Steps** | **Inputs** | **Expected Output** |
| TC01 | Open the math application | N/A | Application loads successfully |
| TC02 | Input numbers and click Add | x: 5, y: 3 | Result: 8 |
| TC03 | Input numbers and click Subtract | x: 10, y: 4 | Result: 6 |
| TC04 | Input numbers and click Multiply | x: 4, y: 5 | Result: 20 |
| TC05 | Input numbers and click Divide | x: 12, y: 4 | Result: 3 |
| TC06 | Input numbers and click Divide | x: 5, y: 0 | Error: Division by zero is undefined |
| TC07 | Input numbers and click Add | x: 0, y: 0 | Result: 0 |
| TC08 | Input numbers and click Subtract | x: -5, y: -3 | Result: -2 |
| TC09 | Input numbers and click Multiply | x: -2, y: 3 | Result: -6 |
| TC10 | Input numbers and click Divide | x: -10, y: -2 | Result: 5 |

White Box Testing

Lab 01 DeskTop Math Cal Testing using Unittest

* Lab01\_test\_unittest\_operations.py
* import unittest
* # Functions to perform operations
* def add(x, y):
* return x + y
* def subtract(x, y):
* return x - y
* def multiply(x, y):
* return x \* y
* def divide(x, y):
* if y == 0:
* raise ValueError("Division by zero is undefined.")
* return x / y
* # Test cases class
* class TestMathOperations(unittest.TestCase):
* def test\_add(self):
* self.assertEqual(add(5, 3), 8)
* self.assertEqual(add(-2, -3), -5)
* self.assertEqual(add(0, 7), 7)
* def test\_subtract(self):
* self.assertEqual(subtract(10, 4), 6)
* self.assertEqual(subtract(5, 8), -3)
* self.assertEqual(subtract(-5, -3), -2)
* def test\_multiply(self):
* self.assertEqual(multiply(4, 5), 20)
* self.assertEqual(multiply(-3, 6), -18)
* self.assertEqual(multiply(0, 9), 0)
* def test\_divide(self):
* self.assertEqual(divide(12, 4), 3)
* self.assertEqual(divide(-15, 3), -5)
* self.assertEqual(divide(0, 5), 0)
* with self.assertRaises(ValueError) as context:
* divide(5, 0)
* self.assertEqual(str(context.exception), "Division by zero is undefined.")
* self.assertEqual(divide(-10, -2), 5)
* # Run the tests
* if \_\_name\_\_ == '\_\_main\_\_':
* unittest.main()
* # to run test: python -m unittest test\_unittest\_operations.py
* # run with detai report: python -m unittest -v test\_operations\_unittest.py

Lab 02 DeskTop Math Cal Testing using Pytest

* Lab02\_test\_pytest\_operations.py
* import pytest
* # Functions to perform operations
* def add(x, y):
* return x + y
* def subtract(x, y):
* return x - y
* def multiply(x, y):
* return x \* y
* def divide(x, y):
* if y == 0:
* raise ValueError("Division by zero is undefined.")
* return x / y
* # Test cases
* @pytest.mark.parametrize("input1, input2, expected, operation", [
* (5, 3, 8, "add"),
* (-2, -3, -5, "add"),
* (0, 7, 7, "add"),
* (10, 4, 6, "subtract"),
* (5, 8, -3, "subtract"),
* (-5, -3, -2, "subtract"),
* (4, 5, 20, "multiply"),
* (-3, 6, -18, "multiply"),
* (0, 9, 0, "multiply"),
* (12, 4, 3, "divide"),
* (-15, 3, -5, "divide"),
* (0, 5, 0, "divide"),
* (5, 0, None, "divide"),  # Placeholder for exception
* (-10, -2, 5, "divide"),
* ])
* def test\_operations(input1, input2, expected, operation):
* if operation == "add":
* assert add(input1, input2) == expected
* elif operation == "subtract":
* assert subtract(input1, input2) == expected
* elif operation == "multiply":
* assert multiply(input1, input2) == expected
* elif operation == "divide":
* if input2 == 0:
* with pytest.raises(ValueError, match="Division by zero is undefined."):
* divide(input1, input2)
* else:
* assert divide(input1, input2) == expected
* # To run the tests, use the command: pytest <filename>.py
* # To install: pip install pytest
* # To run test:pytest test\_pytest\_operations.py
* # to run with detail report: pytest -v test\_pytest\_operations.py

Lab 02 DeskTop Math Cal Testing Using Robot

* math\_operations.py
* # math\_operations.py
* def add(x, y):
* return float(x) + float(y)
* def subtract(x, y):
* return float(x) - float(y)
* def multiply(x, y):
* return float(x) \* float(y)
* def divide(x, y):
* if y == 0:
* raise ValueError("Division by zero is undefined.")
* return float(x) / float(y)
* Lab04\_test\_robot\_operations\_math.robot

\*\*\* Settings \*\*\*

Library math\_operations.py

\*\*\* Variables \*\*\*

@{ADD\_POSITIVE} 5 3

@{ADD\_NEGATIVE} -2 -3

@{ADD\_ZERO} 0 7

@{SUBTRACT\_POSITIVE} 10 4

@{SUBTRACT\_NEGATIVE} 5 8

@{MULTIPLY\_POSITIVE} 4 5

@{MULTIPLY\_NEGATIVE} -3 6

@{DIVIDE\_POSITIVE} 12 4

\*\*\* Test Cases \*\*\*

Test Simple List

Log ${ADD\_POSITIVE}[0]

Log ${ADD\_POSITIVE}[1]

Test Addition

${result}= Add ${ADD\_POSITIVE}[0] ${ADD\_POSITIVE}[1]

Should Be Equal As Numbers ${result} 8

${result}= Add ${ADD\_NEGATIVE}[0] ${ADD\_NEGATIVE}[1]

Should Be Equal As Numbers ${result} -5

${result}= Add ${ADD\_ZERO}[0] ${ADD\_ZERO}[1]

Should Be Equal As Numbers ${result} 7

Test Subtraction

${result}= Subtract ${SUBTRACT\_POSITIVE}[0] ${SUBTRACT\_POSITIVE}[1]

Should Be Equal As Numbers ${result} 6

${result}= Subtract ${SUBTRACT\_NEGATIVE}[0] ${SUBTRACT\_NEGATIVE}[1]

Should Be Equal As Numbers ${result} -3

Test Multiplication

${result}= Multiply ${MULTIPLY\_POSITIVE}[0] ${MULTIPLY\_POSITIVE}[1]

Should Be Equal As Numbers ${result} 20

${result}= Multiply ${MULTIPLY\_NEGATIVE}[0] ${MULTIPLY\_NEGATIVE}[1]

Should Be Equal As Numbers ${result} -18

Test Division

# Test normal division

${result}= Divide ${DIVIDE\_POSITIVE}[0] ${DIVIDE\_POSITIVE}[1]

Should Be Equal As Numbers ${result} 3

Grey Box Testing

Lab 05 Local Web Math Cal App using Selenium Framework

* Server.py
* from flask import Flask, request, jsonify, send\_from\_directory
* import os
* app = Flask(\_\_name\_\_)
* @app.route('/')
* def index():
* """Serve the index.html file."""
* return send\_from\_directory(os.getcwd(), 'index.html')
* @app.route('/add', methods=['POST'])
* def add\_route():
* data = request.json
* result = float(data['x']) + float(data['y'])
* return jsonify({'result': result})
* @app.route('/subtract', methods=['POST'])
* def subtract\_route():
* data = request.json
* result = float(data['x']) - float(data['y'])
* return jsonify({'result': result})
* @app.route('/multiply', methods=['POST'])
* def multiply\_route():
* data = request.json
* result = float(data['x']) \* float(data['y'])
* return jsonify({'result': result})
* @app.route('/divide', methods=['POST'])
* def divide\_route():
* data = request.json
* if data['y'] == 0:
* return jsonify({'error': "Division by zero is undefined."}), 400
* result = float(data['x']) / float(data['y'])
* return jsonify({'result': result})
* if \_\_name\_\_ == '\_\_main\_\_':
* app.run(debug=True)
* index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Math Operations</title>

    <script>

        async function performOperation(operation) {

            const x = parseFloat(document.getElementById('num1').value);

            const y = parseFloat(document.getElementById('num2').value);

            let url = `/${operation}`;

            let response;

            try {

                response = await fetch(url, {

                    method: 'POST',

                    headers: {

                        'Content-Type': 'application/json'

                    },

                    body: JSON.stringify({ x, y })

                });

                const result = await response.json();

                if (response.ok) {

                    document.getElementById('result').innerText = `Result: ${result.result}`;

                } else {

                    document.getElementById('result').innerText = `Error: ${result.error}`;

                }

            } catch (error) {

                document.getElementById('result').innerText = `Error: ${error.message}`;

            }

        }

    </script>

</head>

<body>

    <h1>Math Operations</h1>

    <div>

        <input type="number" id="num1" placeholder="Enter first number" required>

        <input type="number" id="num2" placeholder="Enter second number" required>

    </div>

    <div>

        <button onclick="performOperation('add')">Add</button>

        <button onclick="performOperation('subtract')">Subtract</button>

        <button onclick="performOperation('multiply')">Multiply</button>

        <button onclick="performOperation('divide')">Divide</button>

    </div>

    <h2 id="result"></h2>

</body>

</html>

* Lab06\_test\_selenium\_operations.py

Black Box Testing

Lab 04 Local Web Math Cal App using Robot Framework

* Server.py
* index.html
* Lab05\_test\_robot\_web\_operations.robot

\*\*\* Settings \*\*\*

Library SeleniumLibrary

Suite Setup Open Browser http://127.0.0.1:5000 chrome

Suite Teardown Close Browser

\*\*\* Test Cases \*\*\*

Test Addition

[Documentation] Test the addition operation through the web interface

Input Number 5 3

Click Element //button[text()='Add']

Verify Result Result: 8

Test Subtraction

[Documentation] Test the subtraction operation through the web interface

Input Number 10 4

Click Element //button[text()='Subtract']

Verify Result Result: 6

Test Multiplication

[Documentation] Test the multiplication operation through the web interface

Input Number 4 5

Click Element //button[text()='Multiply']

Verify Result Result: 20

Test Division

[Documentation] Test the division operation through the web interface

Input Number 12 4

Click Element //button[text()='Divide']

Verify Result Result: 3

Test Division By Zero

[Documentation] Test division by zero handling through the web interface

Input Number 5 0

Click Element //button[text()='Divide']

Verify Result Error: Division by zero is undefined.

\*\*\* Keywords \*\*\*

Input Number

[Arguments] ${num1} ${num2}

Input Text id=num1 ${num1}

Input Text id=num2 ${num2}

Verify Result

[Arguments] ${expected\_result}

${result}= Get Text id=result

Should Contain ${result} ${expected\_result}