PROBLEM STATEMENT: DESIGNING A CALCULATOR

USER STORY:

<u>User Story for a Student:</u>

Title: Perform Quick Calculations for Homework

As a student,
I want to use a calculator program
So that I can efficiently solve math problems for my homework.

Acceptance Criteria:

1. I should be able to input mathematical expressions easily.
2.The calculator should support basic operations like addition, subtraction, multiplication, and division.
3.It should provide accurate results for both whole numbers and decimals.
4.The calculator should handle negative numbers appropriately.
5.I want the option to store and recall values from memory.
6.Clear functionality should reset the calculator for a new calculation.

<u>User Story for a Teacher:</u>

Title: Facilitate Classroom Demonstrations and Assessments

As a teacher,
I want a calculator program
So that I can efficiently demonstrate mathematical concepts to my students and assess their understanding.

Acceptance Criteria:

1.The calculator should support a variety of mathematical operations suitable for teaching purposes.
2.It should be user-friendly, allowing students to easily follow along during demonstrations.
3.The program should provide clear and accurate results, aiding in explaining concepts.
4.Memory functions should allow for storing and recalling values during lessons.
5.The calculator should handle both positive and negative numbers appropriately.
6.Clear functionality should reset the calculator for a new demonstration or assessment.
7.The program should be robust enough to handle a range of inputs without crashing.
8.If students attempt to divide by zero or provide invalid inputs, the calculator should handle these scenarios gracefully, providing an informative response.

TEST CASES:

Serial No	Test Method Name	Test Description	Input	Expected Output	Actual Output
1	test_addition	Used to calculate the addition of the two operands	Two values has been passed to the function	Should return the addition operation of the two operands	It has returned the addition operation of two operands
2	test_subtraction	Used to calculate the subtraction of the two operands	Two values has been passed to the function	Should return the subtraction operation of the two operands	It has returned the subtraction operation of two operands
3	test_multiplication	Used to calculate the multiplication of the two operands	Two values has been passed to the function	Should return the multiplication operation of the two operands	It has returned the multiplication operation of two operands
4	test_division	Used to calculate the division of the two operands	Two values has been passed to the function	Should return the division operation of the two operands and also should raise the ValueError of cannot divide by Zero if the second operand value is zero	It has returned the division operation of two operands And handles the divide by zero error

5	test_decimal_input	Used to calculate the desired operation of the float value rounded upto 3 decimal places	Two float values has been passed to the function	Should perform the desired operation with float values and return the output rounded with 3 decimal values	It has performed the desired operation with float values and has returned the output rounded with 3 decimal values
6	test_clear_function	Used to clear the input value	The function is called without passing any input values	Should clear the current_result value	It has cleared the current_result value
7	test_memory_functionality	Used to store and recall a value	A single value has been passed as input	should store and recall the last stored value	It stores and recalls the last stored value
9	test_user_input_validation_large_numbers	Used to handle large input values	This function gets the two large numbers as input	Should raise a value error	It has raised a value error indicating “Invalid Input”
9	test_user_input_negative_numbers	Used to handle the negative numbers of the input values	This function gets two negative numbers as input	Should perform the desired operation for the negative numbers	It has performed the desired operation for the negative value and returned the result

TEST RESULTS:

COMMANDS:

python -m unittest -v test_calculator.py

```
PS C:\Users\kirthika.h\Desktop\python> python -m unittest -v test_calculator.py
test_addition (test_calculator.TestCalculator.test_addition) ... ok
test_clear_function (test_calculator.TestCalculator.test_clear_function) ... ok
test_decimal_input (test_calculator.TestCalculator.test_decimal_input) ... ok
test_division (test_calculator.TestCalculator.test_division) ... ok
test_memory_functionality (test_calculator.TestCalculator.test_memory_functionality) ... ok
test_multiplication (test_calculator.TestCalculator.test_multiplication) ... ok
test_subtraction (test_calculator.TestCalculator.test_subtraction) ... ok
test_user_input_negative_numbers (test_calculator.TestCalculator.test_user_input_negative_numbers) ... ok
test_user_input_validation_large_numbers (test_calculator.TestCalculator.test_user_input_validation_large_numbers) ... ok

----------------------------------------------------------------------
Ran 9 tests in 0.005s

OK
```

python -m coverage run -m unittest test_calculator.py

```
PS C:\Users\kirthika.h\Desktop\python> python -m coverage run -m unittest test_calculator.py
.........
----------------------------------------------------------------------
Ran 9 tests in 0.001s

OK
```

python -m coverage report -m

```
PS C:\Users\kirthika.h\Desktop\python> python -m coverage report -m
Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
calculator.py           38      5    87%   3-4, 17, 25, 34
test_calculator.py      48      1    98%   67
--------------------------------------------------
TOTAL                   86      6    93%
```