**Ryan Strouse - Test Plan**

Depending on whether or not this app is behind a login page, I would provide user credential information along with where to navigate to start testing.

Environment: QA, Stage, Prod

Website: theappcanbefoundhere.com

Username // Pass: math.user // mathpass01!

**Testing**

Input the correct answer and click the “Answer” button – should receive “That's right! Try another one.” prompt

New values should appear on the page after a successful answer and prompt

Input the wrong answer and click the “Answer” button – should receive “Sorry, that is not correct. Please try again.”

Values should stay the same if you enter in the wrong answer

Verify either message prompt disappears after a few seconds (correct or incorrect answer)

Verify message prompt can be closed as the user by clicking it (correct or incorrect answer)

Attempt to input characters other than numbers or decimal – should not be able to enter

Attempt to input special characters (!@#$%^) etc. and click the “Answer” button – should not be able to enter

Attempt to input SQL injection code and click the “Answer” button – those characters should not be allowed to be entered and nothing exposed via developers console

Verify the user interface makes sense and can be navigated.

Verify the “Answer” button should be grayed out until valid numbers are input into field

Verify the “Answer” button is clickable after valid numbers are placed into input field (even if incorrect answer)

Verify the input text “Answer” slides up and shrinks the size while entering in valid numbers. The field should stay like that as long as numbers are entered.

Verify if there is a minimum or maximum value that can be entered (even if incorrect)

Verify page loads with addition as the default operation

**Notes**

I would attach screenshots of before and after for each iteration of the test to match with the expected outcomes.

The example stated that the two numbers were being auto generated, assuming the user is not inputting numbers into those boxes as well. Otherwise, you could test whether adding negative numbers together will check for a correct answer.

If the page had accessibility classes and formatting would make sure to use a third-party application like WAVE Evaluation Tool to verify the expected behavior. Such as reading to the user or highlighting input boxes or buttons.

Seemed odd you could input two dash “-“ characters. One would make sense if you were adding negative values, but why two? The “Answer” button was grayed out either way.

**Unit Tests**

<https://github.com/RyanAStrouse/math-tutor-app/blob/strouse-test/src/app/services/calculator-service/calculator.service.ts>

You could create some checks in the calculator to verify a number being generated is a valid number (between 1 and 10)

public void TestNumberValidator()

{

string goodNumber = “8”

string badNumber = “11”

NumberValidator validator = new NumberValidator();

Assert.IsTrue(validator.IsValid(goodNumber));

Asswer.IsFalse(validator.IsValid(badPhone));

}

<https://github.com/RyanAStrouse/math-tutor-app/blob/strouse-test/src/app/services/message-service/message.service.ts>

You could have a check for if the success or error message displays based on the answer

public string MessageCheck(string message)

{

bool success = MessageCheck(message)

if (success == false)

return " Sorry, that is not correct. Please try again. "

else

return " That\'s right! Try another one."

}