**TEST STRATEGY**

Testing the Calculator web application will encompass functional, security, performance, usability, compatibility, localization, interface and exploratory testing.

**TEST CATEGORIES:**

1. **FUNCTIONAL**

The following approaches will be considered for testing the functionality of the Calculator application:

* 1. Equivalence Partitioning:

Divide inputs into categories (ie. valid, invalid, short, long) and test each category.

* 1. Boundary Value Analysis:

Test the limits of acceptable input, such as the minimum and maximum allowed values or lengths.

* 1. Negative Testing:

Provide invalid inputs to see how the application handles them, looking for errors, vulnerabilities or application crashes.

* 1. Stress Testing:

Use random or unpredictable inputs, in very large numbers or at high speeds, usually through automation, to expose unexpected behavior.

1. **SECURITY**
   1. XSS Testing:

Inject JavaScript code into text fields to see if it can be executed in the browser, allowing malicious scripts to be run.

* 1. SQL Injection Testing:

Use special characters or SQL commands in the input to see if the application is vulnerable to SQL injection attacks.

1. **PERFORMANCE**

Evaluate the performance of the application using different automation tools or browser’s Inspect menu option.

1. **USABILITY**

Evaluate user’s experience with the application, ensuring an easy and intuitive interface.

1. **COMPATIBILITY**

Ensure the application functions correctly on different browsers, different versions of browsers, different operating systems and different types of devices.

1. **LOCALIZATION**

Ensure the application and its interface function correctly in different languages and different time zones.

1. **INTERFACE**

Ensure that different components of the application communicate with each other as expected, including APIs and web services.

1. **EXPLORATORY**

Exercise ad-hoc inputs and events on the application, that other test categories do not cover, to identify potential issues or areas that might be prone to defects.

**TEST ENVIRONMENT:**

* OS: Windows 10 Home (64-bit)
* Browser: Chrome, version 137.0.7151.56 (64-bit)
* Language: Python, version 3.13.2
* Version control: Git
* Repository: <https://github.com/llkrm/calculator.git>

**TEST CASE DEVELOPMENT**

**Manual testing:**

Test usability through manual testing, ensuring that the GUI meets the application requirements and the user has a smooth and intuitive experience.

Exploratory tests should also be run manually.

**Automated testing:**

Test the functionality, security, compatibility, localization, interface and performance of the application using test automation.

For the purpose of this POC, we use Python and Selenium Webdriver, on the test environment listed above.

The automated tests developed in this repository cover mainly functional and some aspects of security testing. Compatibility and localization are much bigger in scope, as they depend on the number of browsers and versions that would need to be supported as well as the operating systems, devices, languages and regional settings. They are not covered by this POC.

How to run automated tests:

* Install Python 3
* Install “pip”, “selenium”, “2captcha-python”, “webdriver-manager” and “pytest” packages
* Navigate to the root of the Calculator project
* Open the terminal and run:

pytest -v -s .\tests\.

Note: You need to solve CAPTCHA manually, for every test file, as my code in “conftest.py” does not work with this site’s captcha.

* Each test file or test case can be run independently as well.
* Each test case has its own documentation that can be accessed via Python console as such:

>>> help(test\_invalid\_inputs.TestInvalidInputs.test\_random\_keys)

**Design for automated tests:**

1. Buttons
   1. Left-click and right-click on each button.
2. Keys
   1. Input the corresponding keys for each button of the calculator app, in the output field.
   2. Input “enter” and “backspace” keys to test their functionality on the output field
   3. Input all alphabet letters to test for unexpected values in the output field, other than the shortcuts that could not be disabled.
3. Operations
   1. Operations (add, subtract, multiply, divide) with integer numbers, positive and negative, using buttons.
   2. Operations (add, subtract, multiply, divide) with decimal numbers, positive and negative, using buttons.
   3. Operations (add, subtract, multiply, divide) with very large and very small numbers, positive and negative, using keys.
   4. Operations (add, subtract, multiply, divide) with zero.
4. Invalid inputs
   1. Invalid keys that should not produce any result in the output field.
   2. Random keys sent 100 times to stress the application.