**Case Study: Testing a Calculator Application**

**Introduction**

This case study outlines the approach taken to test a basic calculator application. The calculator supports fundamental operations like addition, subtraction, multiplication, and division. Our goal is to ensure the calculator operates correctly under various scenarios, including typical usage, edge cases, and error handling.

**Objectives**

* Validate the accuracy of arithmetic operations.
* Ensure the application handles edge cases and error conditions gracefully.
* Confirm the user interface behaves as expected.
* Verify performance and responsiveness under load.

**Scope**

* **Operations**: Addition, Subtraction, Multiplication, Division
* **Input Types**: Positive numbers, Negative numbers, Zero, Large numbers
* **Edge Cases**: Division by zero, Overflow conditions, Invalid inputs
* **User Interface**: Button clicks, Display updates, Error messages

**Test Environment**

* **Device**: Desktop and mobile devices
* **Operating System**: Windows, macOS, Linux, iOS, Android
* **Browser**: Chrome, Firefox, Safari, Edge

**Test Plan**

**1. Functional Testing**

**Objective**: Validate that the calculator performs all basic operations correctly.

**Test Cases**:

* **Addition**
  + 2 + 3 = 5
  + -2 + -3 = -5
* **Subtraction**
  + 5 - 3 = 2
  + -5 - (-3) = -2
* **Multiplication**
  + 4 \* 5 = 20
  + -4 \* -5 = 20
* **Division**
  + 10 / 2 = 5
  + -10 / -2 = 5

**2. Edge Case Testing**

**Objective**: Test the calculator's behavior with unusual or extreme inputs.

**Test Cases**:

* **Large Numbers**
  + 999999999 + 1 = 1000000000
  + 999999999 \* 999999999 = 999999998000000001
* **Zero as Input**
  + 0 + 0 = 0
  + 0 - 0 = 0
  + 0 \* 0 = 0
  + 0 / 1 = 0
* **Negative Numbers**
  + -5 + 5 = 0
  + -10 - 5 = -15
  + -6 \* 3 = -18
  + -10 / 2 = -5
* **Invalid Inputs**
  + Entering multiple operators consecutively (e.g., 2 ++ 3)
  + Non-numeric input (e.g., letters or special characters)

**3. User Interface Testing**

**Objective**: Ensure the user interface operates correctly and is user-friendly.

**Test Cases**:

* **Button Functionality**
  + Each button (0-9, +, -, \*, /, =) should respond correctly to clicks.
  + The "C" (clear) button should reset the display to 0.
* **Display**
  + Numbers and results should display correctly.
  + Error messages should display clearly when appropriate.
* **Responsive Design**
  + The calculator should display and function correctly on various devices and screen sizes.

**4. Performance Testing**

**Objective**: Ensure the calculator operates efficiently under load.

**Test Cases**:

* **Speed**
  + Verify the response time for each operation.
  + Test the application under heavy load (e.g., multiple rapid inputs).
* **Resource Usage**
  + Monitor CPU and memory usage during operations.

**Tools Used**

* **Automated Testing Tools**: Selenium for UI testing, JUnit for functional testing
* **Manual Testing**: For exploratory and edge case testing

**Results**

* **Functional Tests**: All basic operations were validated successfully.
* **Edge Cases**: The calculator handled large numbers, zero, and negative inputs correctly. An error message was displayed for division by zero and invalid inputs.
* **UI Tests**: Buttons responded correctly, and the display updated as expected. The design was responsive across devices.
* **Performance Tests**: The calculator responded quickly and used minimal resources under load.