**Test Case Description:**

**Application/Feature Tested:**

* **Application**: Google Search
* **Feature**:
  + Search functionality for different types of inputs (valid, invalid, and edge cases).
  + Dynamic search suggestions based on the query entered.

**Goals:**

* Validate that the search bar works correctly for valid inputs, handles invalid data gracefully, and accepts edge-case inputs without crashing.
* Verify that search suggestions are dynamically populated for relevant inputs and absent for irrelevant or gibberish inputs.

**Test Strategy:**

**Steps to Test the Features:**

1. **Search Functionality**:
   * **Positive Test**:
     + Enter a valid query (e.g., "Selenium WebDriver").
     + Validate that relevant results are displayed.
   * **Boundary Test**:
     + Enter an edge-case input like a single character or an extremely long string (e.g., 500 characters).
     + Validate that the input is accepted and processed correctly without errors.
2. **Search Suggestions**:
   * **Positive Test**:
     + Enter a valid query (e.g., "Selenium").
     + Validate that relevant suggestions are dynamically displayed.
   * **Boundary Test**:
     + Enter a single character (e.g., "S").
     + Validate that suggestions for popular searches starting with that character are displayed.

**Testing Tools and Frameworks:**

* **Programming Language**: Java
* **Framework**: TestNG
* **Dependencies**: Selenium WebDriver
* **Build Tool**: Maven
* **Browser**: Google Chrome

**Test Environment:**

* **ChromeDriver**: Matches the installed browser version.
* **IDE**: Eclipse
* **Execution**: Automated through TestNG suite.

**Rationale**

**Why These Tests?**

1. **Positive Tests**:
   * Validate that the primary functionality (search and suggestions) works as expected.
2. **Boundary Tests**:
   * Check system behaviour with edge cases, such as excessively long or minimal input.

**How They Align with Business Logic:**

* The primary purpose of Google Search is to provide relevant results quickly and reliably. Testing these scenarios ensures the application's usability, reliability, and robustness for different types of user behaviour.

**Prioritization:**

* Search functionality is critical; hence, positive and boundary tests are high-priority.
* Suggestion accuracy is important but less critical than the search, so suggestion-related tests are medium-priority.

**Test Design:**

**Test Case 1: Search Functionality**

| **Field** | **Details** |
| --- | --- |
| **Test Case ID** | TC\_001 |
| **Test Title** | Validate Google Search Functionality |
| **Description** | Test the search bar for valid, invalid, and edge inputs. |
| **Preconditions** | ChromeDriver is set up, and the Google homepage is accessible. |
| **Steps** | 1.) Navigate to the Google homepage  2.) Enter a valid search query  3.) Validate results  4.) Repeat with invalid and edge-case inputs.</li></ol> |
| **Expected Outcome** | Valid inputs return relevant results.  Invalid inputs do not crash the app or return meaningful results.  Edge cases are handled gracefully |
| **Actual Outcome** | It matches the expected results. |

**Test Case 2: Search Suggestions**

| **Field** | **Details** |
| --- | --- |
| **Test Case ID** | TC\_002 |
| **Test Title** | Validate Google Search Suggestions |
| **Description** | Test the dynamic population of search suggestions based on input. |
| **Preconditions** | ChromeDriver is set up, and Google homepage is accessible. |
| **Steps** | 1.) Navigate to the Google homepage  2.) Start typing a query  3.) Validate suggestions, Repeat with invalid and edge-case inputs. |
| **Expected Outcome** | Valid inputs dynamically populate relevant suggestions. Invalid inputs do not display suggestions; edge cases (e.g., single characters) dynamically populate suggestions. |
| **Actual Outcome** | It matches the expected results. |
| **Priority** | Medium |