**Selenium Automation Test: Table Search Validation**

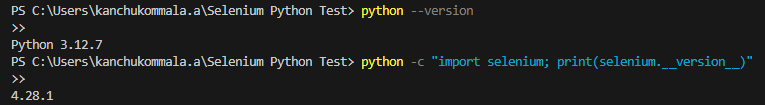
**Overview**

This Selenium automation script is designed to test the search functionality of the **Table Sorting and Searching** page on [Selenium Playground](https://www.lambdatest.com/selenium-playground/table-sort-search-demo).  
The script performs the following:

* Opens the browser and maximizes the window.
* Navigates to the test URL.
* Validates the page title to ensure the correct page is loaded.
* Searches for entries related to **"New York"** in the table.
* Confirms that exactly **5** results are displayed out of **24 total entries**.
* Closes the browser after execution.

**Tools Used**

* **Python** (v3.12.7)
* **Selenium WebDriver** (latest version)
* **WebDriver Manager** (for dynamic Chrome Driver installation)
* **Google Chrome** (Version **133.0.6943.98**)
* **Pytest** – A robust test framework for structuring and running test cases efficiently.
* **VS Code** – The integrated development environment (IDE) for writing and running the script.
* **Anaconda Interpreter** – Used for executing the Python script within a virtual environment.



1.Tools used are latest and stable

**Approach**

**Step 1: WebDriver Setup & Optimization**

* Used WebDriver manager to avoid manual ChromeDriver installation.
* Browser maximization ensures all elements are visible before interaction.
* Implemented Pytest fixtures to reuse the WebDriver session, preventing redundant reinitialization.

A screen shot of a computer program

AI-generated content may be incorrect.

2.Setup webdriver & Browser

**Step 2: Navigation & Page Validation**

* The script opens the required URL and ensures the page is loaded correctly.
* Explicit waits (WebDriver Wait) are used instead of static sleeps to handle dynamic page elements.
* A validation step ensures the correct page is loaded before executing further tests.

A screen shot of a computer program

AI-generated content may be incorrect.

3.Navigation & Page Validation

**Step 3: Perform Search Operation**

* The script **locates the search box** and enters "New York".
* Waits until search results are updated dynamically in the table.
* **Extracts visible rows** and verifies the **expected 5 entries** appear correctly.

**A screen shot of a computer code

AI-generated content may be incorrect.**

4.Perform Search Operation

**Step 4: Validate Results**

* The script **counts the number of displayed rows** in the table.
* It verifies that **5 entries** are displayed, matching the expected behavior.

A computer screen shot of a program

AI-generated content may be incorrect.

5.Validate Results

**Step 5: Handle Errors & Close Browser**

* If an error occurs, it is **printed with a clear message**.
* The browser remains open until **manual confirmation** (pressing Enter).

A screen shot of a computer program

AI-generated content may be incorrect.

6.Handle Errors & Close Browser

**Best Practices Followed**

**Dynamic WebDriver Handling**

* Instead of relying on **manually downloading** the driver, we use **WebDriver manager** to automatically fetch the compatible **Chrome Driver**.

**Explicit Waits for Stability**

* The script uses **explicit waits** (WebDriver Wait) to ensure elements are **fully loaded before interacting**, avoiding flaky test failures.

**Assertions for Validation**

* The script uses **assertions** to validate:  
  ✔ Correct **page title**  
  ✔ Correct **search result count**

**Graceful Exception Handling**

* A try-except-finally block ensures that any **unexpected failures are logged** without abrupt crashes.

**Minimized Duplicate Code**

* Variables are used for **XPaths and frequently accessed elements** to improve **code readability and maintainability**.

**How to Run the Script**

**Prerequisites**

Ensure you have the following installed:

1. Python 3.12+
2. Google Chrome (Latest version)
3. Required dependencies (install via pip): pip install Selenium WebDriver-manager

**Execution Steps**

**Run the Script in VS Code**

* **Open VS Code** and load your Selenium script (selenium\_test.py).
* **Select Python Interpreter** (if not already selected):
* Press **Ctrl+Shift+P** → **Select Interpreter** → Choose **Anaconda (conda) Python** (which I have used to execute the script)
* **Run the Script**: Open the **Terminal in VS Code** and run: python selenium\_test.py
* Or use the **Run button** in VS Code.
* The browser will open, execute the test, and wait for you to **press Enter** to close it.

**Conclusion**

* This structured approach ensures **high test accuracy, maintainability, and efficiency** in Selenium automation.
* If further enhancements are required, such as **logging, reporting, or CI/CD integration**, the script can be extended accordingly.