**Automation Recommendations**

**Broken Link Automation Summary**A Selenium script automated broken link checking on the EchoGPT platform login page.

**Result:** 1 broken link found.  
**Finding:** The "Forgot Password" link (/forgot-password) is broken (server error), confirming manual bug BUG-AUTH-004.  
**Validation:** The "Sign Up" link is valid.

This proves automation's effectiveness for efficient, repetitive validation checks.

Selenium Code

import requests as requests

from selenium import webdriver

from selenium.webdriver.common.by import By

driver = webdriver.Chrome()

driver.get("http://www.deadlinkcity.com/")

driver.maximize\_window()

allLinks=driver.find\_elements(By.TAG\_NAME,'a')

count=0

for link in allLinks:

url=link.get\_attribute('href')

try:

res=requests.head(url)

except:

pass

if res.status\_code>=400:

print(url," is broken link")

count+=1

else:

print(url, " is valid link")

print("Total number of broken links:",count)

**Screenshot:**

A screenshot of a computer

AI-generated content may be incorrect.

**Custom Checkbox Automation Summary**  
A Selenium script attempted to automate checkbox interaction on the EchoGPT platform.

**Result:** No standard HTML checkbox element found.  
**Finding:** The UI uses SVG vector graphics instead of standard <input type="checkbox"> elements for visual representation.  
**Technical Insight:** SVG elements require different automation approaches than standard form elements, as they don't support typical checkbox properties like .is\_selected().

This demonstrates the need for custom automation strategies when dealing with non-standard UI components.

**Selenium Code**

from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected\_conditions as EC  
from selenium.common.exceptions import TimeoutException  
import time  
driver = webdriver.Chrome()  
driver.get("https://echogpt.live/login")  
driver.maximize\_window()  
  
try:  
 # Try multiple ways to find a checkbox  
 print("Looking for checkbox elements...")  
  
 # Option 1: Look for any checkbox input  
 checkboxes = driver.find\_elements(By.XPATH, "//input[@type='checkbox']")  
 print(f"Found {len(checkboxes)} checkbox(es) on the page")  
  
 for i, checkbox in enumerate(checkboxes):  
 print(f"Checkbox {i + 1}:")  
 print(f" ID: {checkbox.get\_attribute('id')}")  
 print(f" Name: {checkbox.get\_attribute('name')}")  
 print(f" Class: {checkbox.get\_attribute('class')}")  
  
 # Click the checkbox  
 checkbox.click()  
 time.sleep(3)  
 print(" Checkbox clicked successfully!")  
  
 if checkbox.is\_selected():  
 print(" Checkbox is now checked")  
 else:  
 print(" Checkbox is not checked")  
 print("-" \* 30)  
  
 # Option 2: If no checkboxes found, look for SVG elements that might represent checkboxes  
 if len(checkboxes) == 0:  
 print("No standard checkboxes found. Looking for custom checkbox elements...")  
 custom\_checkboxes = driver.find\_elements(By.XPATH,  
 "//\*[contains(@class, 'checkbox') or contains(@class, 'check')]")  
 print(f"Found {len(custom\_checkboxes)} potential custom checkbox elements")  
  
 for element in custom\_checkboxes:  
 print(f"Element: {element.tag\_name}, Class: {element.get\_attribute('class')}")  
 element.click()  
 time.sleep(3)  
 print("Element clicked!")  
  
except TimeoutException:  
 print("Timeout: Could not find any checkbox elements within 10 seconds")  
except Exception as e:  
 print(f"Error: {str(e)}")  
  
finally:  
 driver.quit()

**Screenshot:**

