Accessibility Testing Script Documentation

Overview

This Python script automates the process of running accessibility tests on web pages using the Axe accessibility engine through Selenium WebDriver. The script captures accessibility violations for WCAG (Web Content Accessibility Guidelines) levels 2.0 (A, AA, AAA), generates JSON reports for each tested page, and consolidates the results into a single HTML report that includes tables and charts for easier analysis.

The report includes a visual summary of violations by WCAG level and impact (Critical, Serious, Minor), and a detailed table of specific violations. Additionally, the report includes company logos to enhance its presentation.

Requirements

1. **Python Packages**:
    - `selenium`: To automate browser interactions.
    - `axe-selenium-python`: To run accessibility audits using Axe.
    - `matplotlib`: To generate charts for visualization.
    - `pandas`: To handle and manipulate violation data.
    - `json`: To handle JSON reports generated by Axe.
    - `datetime`: To generate the report date.
2. **System Dependencies**:
    - Chrome browser and ChromeDriver (must be installed and configured in the system path).
3. **Credentials**:
    - A valid login URL, username, and password to authenticate on the website being tested.

Code Breakdown

Imports

The script imports various libraries necessary for file handling, browser automation, accessibility checks, chart generation, and data processing:

python
```
import re
import matplotlib.pyplot as plt
from selenium import webdriver
from selenium.webdriver.common.by import By
from axe_selenium_python import Axe
import time
import json
import pandas as pd
from datetime import datetime
```

Functions

1. `sanitize_filename(url)`

This function sanitizes URLs to create valid filenames by replacing special characters (`<>:"/\|?*`) with underscores.

python
```
def sanitize_filename(url):
    return re.sub(r'[<>:"/\\|?*]', '_', url)
```

2. `run_accessibility_test(urls, wcag_levels, login_url, username, password, snapapp_logo_url, bluevector_logo_url)`

This is the main function that logs into the target site, performs the accessibility tests, saves the results as JSON files, and calls the report generation function.

- **Parameters**:
  - `urls`: A list of URLs to be tested.
  - `wcag_levels`: List of WCAG levels to check for (`wcag2a`, `wcag2aa`, `wcag2aaa`).
  - `login_url`: URL to the login page of the application.
  - `username`, `password`: Login credentials.
  - `snapapp_logo_url`, `bluevector_logo_url`: URLs of the SnapApp and BlueVector logos to be included in the report.

python

```
def run_accessibility_test(urls, wcag_levels, login_url, username,
password, snapapp_logo_url, bluevector_logo_url):
    # Login and accessibility audit logic
```

- **Process**:
    1. **WCAG Level Validation**: Ensures only valid WCAG levels are used.
    2. **Login Automation**: Navigates to the login page and authenticates using the provided credentials.
    3. **Accessibility Tests**: For each page in the `urls` list, it runs Axe accessibility tests for each WCAG level, collects results, and stores them in a JSON file.
    4. **JSON File Storage**: Results are stored as JSON files, and a count of violations is printed for each page.
    5. **HTML Report Generation**: Once all tests are complete, it calls `generate_consolidated_html_report()` to create a summary HTML report.

3. `generate_consolidated_html_report(json_files, html_file, snapapp_logo_url, bluevector_logo_url)`

Generates an HTML report summarizing the accessibility test results.

- **Parameters**:
    - `json_files`: List of tuples containing JSON file paths and the corresponding URLs.
    - `html_file`: The output HTML file path.
    - `snapapp_logo_url`, `bluevector_logo_url`: URLs of the logos for display in the report.

python

```
def generate_consolidated_html_report(json_files, html_file,
snapapp_logo_url, bluevector_logo_url):
    # HTML report generation logic
```

- **Process**:
    1. **Violation Summary**: Initializes a summary dictionary for each WCAG level and violation type (Critical, Serious, Minor).
    2. **Detailed Violations**: Processes JSON files to extract individual violations and group them by WCAG level and impact.

3. **HTML Structure**: Builds an HTML file with sections for summary charts, a table of violations, and company logos.
4. **Chart Integration**: Adds bar charts and pie charts summarizing the violation data.

4. `generate_charts_and_summary(violation_summary)`

This function generates a summary table, bar chart, and pie chart from the collected violation data.

- **Parameters**:
  - `violation_summary`: A dictionary summarizing violations by WCAG level and impact.

python
```
def generate_charts_and_summary(violation_summary):
    # Chart generation logic
```

- **Outputs**:
  - A summary table in HTML format.
  - Paths to the saved bar chart and pie chart images.

Example Usage

The script can be used as follows:

python
```
run_accessibility_test(
    urls=[
        'https://example.com/page1',
        'https://example.com/page2'
    ],
    wcag_levels=['wcag2a', 'wcag2aa', 'wcag2aaa'],
    login_url='https://example.com/login',
    username='your_username',
    password='your_password',
    snapapp_logo_url= 'https://example.com/snapapp_logo.png',
    bluevector_logo_url= 'https://example.com/bluevector_logo.png'
)
```

Output

1. **JSON Reports**: Accessibility violations for each page and WCAG level are stored in JSON files named `accessibility_report_<wcag_level>_<sanitized_url>.json`.
2. **HTML Report**: A consolidated HTML report is generated with a summary table, bar chart, pie chart, and detailed violation data.