Freddie Can't Into Downvote

projekt automatyzacji przypadku testowego z wykorzystaniem

Selenium Webdriver z językiem Python

Toruń 2022

Opracowała Małgorzata Kwiatkowska Szargiej

Spis treści

I. Cele projektu

Projekt skupia się na stworzeniu automatyzacji testów strony piosenki „Crazy Little Thing Called Love” (lub jakiejkolwiek wybranej przez testującego) w serwisie youtube. Zadaniem testera jest sprawdzenie możliwości zamieszczenia komentarza, kliknięcia upvote lub downvote bez logowania w serwisie.
W projekcie wykorzystano narzędzia:

- Selenium Webdriver 
- PyCharm 
- radość zabawy w Selenium

Kod dostępny pod adresem:
https://github.com/Najgorzata/SELENIUMFreddieCantIntoDownVote

ENJOY!

II. Scenariusz testowy

ID: 001

Tytuł: Zbadanie możliwości zagłosowania na piosenkę (upvote) lub przeciw piosence (downvote) w serwisie YouTube bez zalogowania w przeglądarce Firefox/Chrome.

Warunki wstępne: Uruchomiona przeglądarka Firefox lub Chrome w zależności od decyzji testera. Użytkownik nie jest zalogowany.

Pierwszy przypadek testowy

ID Scenariusza testowego: 001

ID przypadku testowego: 01

Opis przypadku: Niezalogowany użytkownik nie powinien mieć możliwości zagłosowania na video (upvote)

Warunki wstępne: użytkownik jest niezalogowany

Kroki:

1. Wejdź na stronę "https://www.youtube.com/"
2. Zaakceptuj politykę prywatności
3. W polu szukaj wpisz: „Crazy little thing called love Queen”
4. Znajdź pierwsze dostępne na liście video i kliknij w nie
5. Znajdź przycisk „Podoba mi się”
6. Kliknij przycisk „Podoba mi się”

Oczekiwany rezultat:

Pojawia się popup logowania

Drugi przypadek testowy

ID Scenariusza testowego: 001

ID przypadku testowego: 02

Opis przypadku: Niezalogowany użytkownik nie powinien mieć możliwości zagłosowania przeciwko video (downvote)

Warunki wstępne: użytkownik jest niezalogowany

Kroki:

1. Wejdź na stronę "https://www.youtube.com/"
2. Zaakceptuj politykę prywatności
3. W polu szukaj wpisz: „Crazy little thing called love Queen”
4. Znajdź pierwsze dostępne na liście video i kliknij w nie
5. Znajdź przycisk „Nie podoba mi się”
6. Kliknij przycisk „Nie podoba mi się”

Oczekiwany rezultat:

Pojawia się popup logowania

Trzeci przypadek testowy

ID Scenariusza testowego: 001

ID przypadku testowego: 03

Opis przypadku: Niezalogowany użytkownik nie powinien mieć możliwości skomentowania video

Warunki wstępne: użytkownik jest niezalogowany

Kroki:

1. Wejdź na stronę "https://www.youtube.com/"
2. Zaakceptuj politykę prywatności
3. W polu szukaj wpisz: „Crazy little thing called love Queen”
4. Znajdź pierwsze dostępne na liście video i kliknij w nie
5. Znajdź pole dodawania komentarza
6. Kliknij w nie

Oczekiwany rezultat:

Użytkownik jest przeniesiony na stronę logowania

III. Ścieżki kodu przypadków testowych

Przygotowanie testu, import potrzebnych bibliotek oraz ustanowienie potrzebnych zmiennych.
Z uwagi na długi czas ładowania strony oraz jej specyfikę wykorzystano metodę time.sleep, która mimo wydłużenia czasu testów, była najbardziej efektywna, pozwalała zaoszczędzić linijki kodu oraz zmniejszyć wrażliwość testu na zmiany na stronie. Alternatywą dla tej metody była (w mojej ocenie) metoda WebDriverWait, która (w części przypadków) wymagałaby ustalenia miejsca na stronie, na które WebDriver miałby czekać aby upewnić się, że strona jest załadowana w całości, a które nie zawsze dało się powiązać z szukanym elementem.

```
#Import bibliotek
import unittest
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait

piosenka = 'Crazy little thing called love Queen'
przegladarka = input("Wybierz przeglądarkę wpisz chrome lub firefox: ")
```

Ustanowienie klasy:

```
class TestingFreddie(unittest.TestCase):
```

Zdefiniowanie metody setup, wspólnej dla każdego z testów. Test zakłada interakcję z testującym, ale można wybór przeglądarki zdefiniować stałą zmienną z poziomu kodu.

W niektórych przypadkach wykorzystano metodę CSS_SELECTOR z uwagi na bardzo rozbudowaną ścieżkę XPATH, wrażliwą na zmiany w kodzie.

```
    def setUp(self):
        #Wybór przeglądarki
        if przegladarka == "firefox":
            self.browser = webdriver.Firefox()
        elif przegladarka == "chrome":
            self.browser = webdriver.Chrome()
        else:
            print("Przeglądarka nieobsługiwana")
        #Uruchomienie YT
        self.browser.get("https://www.youtube.com/")
        self.browser.maximize_window()
        self.browser.implicitly_wait(10)
        # Wyszukanie i zaakceptowanie cookies
        agree_button = self.browser.find_element(By.CSS_SELECTOR,'ytd-button-
renderer.ytd-consent-bump-v2-lightbox:nth-child(2) > a:nth-child(1) > tp-yt-
paper-button:nth-child(1) > yt-formatted-string:nth-child(1)')
        agree_button.click()
        # Wyszukanie utworu
        time.sleep(2)
        szukaj = self.browser.find_element(By.XPATH,
'//input[@name="search_query"]')
        szukaj.send_keys(piosenka)
        time.sleep(2)
        self.browser.find_element(By.XPATH, '//button[@id="search-icon-
legacy"]').click()
        self.browser.find_element(By.XPATH, '//a[@id="video-title"][1]').click()
```

ID Scenariusza testowego: 001

ID przypadku testowego: 01

Oczekiwany rezultat badany jest poprzez sprawdzenie, czy pojawia się okienko popup monitujące o potrzebie zalogowania się.

```
def testUpvote(self):
    #Poczekaj na załadowanie
    time.sleep(2)
    #Znajdź i kliknij łapkę w górę
    upVote = self.browser.find_element(By.CSS_SELECTOR,
                                       'ytd-toggle-button-renderer.style-
scope:nth-child(1) > a:nth-child(1) > yt-icon-button:nth-child(1) > button')
    upVote.click()
    # Jeśli wyświetla się monit o zalogowanie, pass
    if self.browser.find_element(By.XPATH,
'//iframe[@name="passive_signin"]').is_displayed():
        pass
```

ID Scenariusza testowego: 001

ID przypadku testowego: 02

Oczekiwany rezultat badany jest poprzez sprawdzenie, czy pojawia się okienko popup monitujące o potrzebie zalogowania się.

```
def testDownvote(self):
    # Poczekaj na załadowanie
    time.sleep(2)
    #Znajdź i kliknij łapkę w dół
    downVote = self.browser.find_element(By.CSS_SELECTOR, 'ytd-toggle-button-
renderer.style-scope:nth-child(2) > a:nth-child(1) > yt-icon-button:nth-
child(1)')
    downVote.click()
    #Jeśli wyświetla się monit o zalogowanie, pass
    if self.browser.find_element(By.XPATH,
'//iframe[@name="passive_signin"]').is_displayed():
        pass
```

ID Scenariusza testowego: 001

ID przypadku testowego: 03

Z uwagi na specyfikę strony YT oraz różne rozdzielczości ekranu, należało wykorzystać scrolling do ekranu umożliwiającego wyświetlenie pola komentarza, w przeciwnym wypadku pole komentarza nie było załadowane i nie można było go wyszukać. Metoda została dopasowana do rozdzielczości ekranu laptopa z uwagi na jego mniejszy rozmiar.

Oczekiwany rezultat badany jest poprzez sprawdzenie, czy kliknięcie spowoduje przeładowanie się strony video do strony logowania. Wybrano metodę is_displayed jako bardziej intuicyjną i mniej wrażliwą na zmiany w kodzie, ale można było podobny rezultat osiągnąć sprawdzając asercję elementu wyświetlanego i oczekiwanego.

```
def testComment(self):
    # Poczekaj na załadowanie
    time.sleep(2)
    self.browser.execute_script("window.scrollTo(0, 1000)")
    # Znajdź pole komentarza
    WebDriverWait(self.browser,
15).until(EC.presence_of_element_located((By.XPATH, '//*[@id="placeholder-
area"]'))).click()
    #Jeśli wyświetla się pole logowania, pass
    if self.browser.find_element(By.XPATH,
'//input[@id="identifierId"]').is_displayed():
        pass
```

TearDown i zamknięcie skryptu.

```
    def tearDown(self):
        self.browser.quit()
if __name__ == '__main__':
    unittest.main()
```

IV Uwagi końcowe

Automatyzacja przypadku testowego (test funkcjonalny) powiodła się.

Testowana strona wymaga, w niektórych przypadkach, stosowania rozbudowanych lokalizatorów XPATH i CSS Selector, przez co test jest wrażliwy na zmiany w strukturze strony. Konstrukcja testu pozwala jednak na wprowadzanie szybkich zmian w lokalizatorach.