Automatyzacja przypadków testowych przy pomocy Selenium Webdriver

Przypadek testowy 1:

Tytuł:
Logowanie oraz wylogowanie się ze strony poprawnym użytkownikiem

Środowisko:
Google Chrome (Version 100.0.4896.127), Ubuntu 20.04.3

Warunek wstępny: Użytkownik posiada konto na stronie (e-mail: aaa@jigdrsrs.com, hasło: AAApass567)

Kroki:
1. Otwórz stronę: automationpractice.com
2. Kliknij **Sign in**
3. Wprowadź poprawny e-mail
4. Wprowadź poprawne hasło
5. Kliknij **Sign in**
6. Kliknij **Sign out**

Oczekiwany rezultat:
Użytkownik poprawnie zalogował się oraz wylogował się na stronie automationpractice.com

Automatyzacja przypadku testowego przy pomocy Selenium Webdriver:

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from time import sleep

my_email = "aaa@jigdrsrs.com"
my_password = "AAApass567"

class ProjectLogOut(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.get("http://automationpractice.com/")
        self.driver.maximize_window()
        self.driver.implicitly_wait(4)

    def test_log_out_from_website(self):
        driver = self.driver
        sign_in = driver.find_element(By.PARTIAL_LINK_TEXT, "Sign in")
        sign_in.click()

        registered_email = driver.find_element(By.ID, "email")
        registered_email.send_keys(my_email)

        registered_password = driver.find_element(By.ID, "passwd")
        registered_password.send_keys(my_password)

        sign_in_button = driver.find_element(By.ID, "SubmitLogin")
        sign_in_button.click()
```

```
        sleep(5)

        sign_out = driver.find_element(By.PARTIAL_LINK_TEXT, "Sign out")
        sign_out.click()

        sleep(5)

    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Wnioski końcowe:
Automat wykonał poprawnie test. Użytkownik poprawnie zalogował oraz wylogował się ze swojego konta.

Przypadek testowy 2:

Tytuł:
Logowanie się na stronie niepoprawnym adresem mailowym

Środowisko:
Google Chrome (Version 100.0.4896.127), Ubuntu 20.04.3

Warunek wstępny:
Użytkownik posiada konto na stronie (e-mail: aaa@jigdrsrs.com, hasło: AAApass567)

Kroki:
1. Otwórz stronę: automationpractice.com
2. Kliknij **Sign in**
3. Wprowadź błędny e-mail - aa@jigdrsrs.com
4. Wprowadź hasło
5. Kliknij **Sign in**

Oczekiwany rezultat:
Użytkownik nie zalogował się na stronie

Automatyzacja przypadku testowego przy pomocy Selenium Webdriver:

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from time import sleep

invalid_email = "aa@jigdrsrs.com"
my_password = "AAApass567"

class ProjectLogin(unittest.TestCase):
    def setUp(self):
```

```
        self.driver = webdriver.Chrome()
        self.driver.get("http://automationpractice.com/")
        self.driver.maximize_window()
        self.driver.implicitly_wait(4)

    def test_sign_in_to_website(self):
        driver = self.driver
        sign_in = driver.find_element(By.PARTIAL_LINK_TEXT, "Sign in")
        sign_in.click()

        registered_email = driver.find_element(By.ID, "email")
        registered_email.send_keys(invalid_email)

        registered_password = driver.find_element(By.ID, "passwd")
        registered_password.send_keys(my_password)

        sign_in_button = driver.find_element(By.ID, "SubmitLogin")
        sign_in_button.click()

        sleep(3)

    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Wnioski końcowe:
Automat wykonał poprawnie test. Użytkownikowi nie udało się zalogować na stronie. Pojawia się komunikat: „Authentication failed”.

Przypadek testowy 3:

Tytuł:
Sprawdzenie czy użytkownik może zalogować się na swoje konto zmieniając wielkość liter hasła

Środowisko:
Google Chrome (Version 100.0.4896.127), Ubuntu 20.04.3

Warunek wstępny:
Użytkownik posiada już zarejestrowane konto na stronie automationpractice.com (e-mail: aaa@jigdrsrs.com, hasło: AAApass567)

Kroki:
1. Otwórz stronę: automationpractice.com
2. Kliknij **Sign in**
3. Wprowadź e-mail
4. Wprowadź hasło zmieniając wielkość liter
5. Kliknij **Sign in**

Oczekiwany rezultat:

Użytkownik nie zalogował się na stronie

Automatyzacja przypadku testowego przy pomocy Selenium Webdriver:

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from time import sleep

my_email = "aaa@jigdrsrs.com"
my_password = "AAApass567"

class ProjectLogin(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.get("http://automationpractice.com/")
        self.driver.maximize_window()
        self.driver.implicitly_wait(4)

    def test_sign_in_to_website(self):
        driver = self.driver
        sign_in = driver.find_element(By.PARTIAL_LINK_TEXT, "Sign in")
        sign_in.click()

        registered_email = driver.find_element(By.ID, "email")
        registered_email.send_keys(my_email)

        registered_password = driver.find_element(By.ID, "passwd")
        my_new_password = my_password.upper()
        registered_password.send_keys(my_new_password)

        sign_in_button = driver.find_element(By.ID, "SubmitLogin")
        sign_in_button.click()

        sleep(6)

    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Wnioski końcowe:
Automat wykonał poprawnie test. Użytkownik nie może zalogować się na swoje konto. Pojawia się komunikat: "Authentication failed". Pole Password w zakładce Sign in jest wrażliwe na zmianę wielkości liter.

Przypadek testowy 4:

Tytuł:
Tworzenie nowego konta za pomocą zarejestrowanego już adresu mailowego.

Środowisko:
Google Chrome (Version 100.0.4896.127), Ubuntu 20.04.3

Warunek wstępny:
Użytkownik posiada już zarejestrowane konto na podany adres mailowy (e-mail: aaa@jigdrsrs.com)

Kroki:
1. Otwórz stronę: automationpractice.com
2. Kliknij **Sign in**
3. Wprowadź e-mail w polu Email address
4. Kliknij **Create an account**

Oczekiwany rezultat: Nowe konto nie zostaje utworzone. Pojawia się informacja, że konto na podany adres jest już zarejestrowane.

Automatyzacja przypadku testowego przy pomocy Selenium Webdriver:

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from time import sleep

my_email = 'aaa@jigdrsrs.com'

class ProjectLogin(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.get("http://automationpractice.com/")
        self.driver.maximize_window()
        self.driver.implicitly_wait(4)

    def test_sign_in_to_website(self):
        driver = self.driver
        sign_in = driver.find_element(By.PARTIAL_LINK_TEXT, 'Sign in')
        sign_in.click()

        email_address = driver.find_element(By.ID, 'email_create')
        email_address.send_keys(my_email)

        create_an_account = driver.find_element(By.ID, 'SubmitCreate')
        create_an_account.click()

        sleep(6)

    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Wnioski końcowe:
Automat wykonał poprawnie test. Na stronie nie można utworzyć więcej niż jednego konta na ten sam adres mailowy. Pojawia się komunikat: „An account using this email address has already been registered. Please enter a valid password or request a new one.”

Przypadek testowy 5:

Tytuł:
Dodawanie produktu do koszyka

Środowisko:
Google Chrome (Version 100.0.4896.127), Ubuntu 20.04.3

Warunek wstępny:
Użytkownik nie jest zalogowany na stronie.

Kroki:
1. Otwórz stronę: automationpractice.com
2. Wybierz kategotię **Dresses**
3. Wybierz kategorię **Casual Dresses**
4. Wybierz produkt **Printed Dress**
5. Dodaj produkt do koszyka
6. Sprawdź czy w koszyku znajduję się wybrany produkt

Oczekiwany rezultat:
Użytkownik dodał produkt do koszyka

Automatyzacja przypadku testowego przy pomocy Selenium Webdriver:

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from time import sleep


class ProjectSearchAndAddingProduct(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.get("http://automationpractice.com/")
        self.driver.maximize_window()
        self.driver.implicitly_wait(4)

    def test_choose_and_add_product(self):
        driver = self.driver
        choose_dresses = driver.find_element(By.XPATH,
"/html/body/div/div[1]/header/div[3]/div/div/div[6]/ul/li[2]/a")
        choose_dresses.click()

        choose_casual_dresses = driver.find_element(By.XPATH,
"/html/body/div/div[2]/div/div[3]/div[2]/div[2]/ul/li[1]/div[1]/a/img")
        choose_casual_dresses.click()
```

```
        choose_printed_dress = driver.find_element(By.LINK_TEXT, "Printed Dress")
        choose_printed_dress.click()

        select_printed_dress = driver.find_element(By.ID, "add_to_cart")
        select_printed_dress.click()

        go_to_cart = driver.find_element(By.XPATH,
"/html/body/div/div[1]/header/div[3]/div/div/div[4]/div[1]/div[2]/div[4]/a")
        go_to_cart.click()

        sleep(5)

    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Wnioski końcowe:
Automat wykonał poprawnie test. Wybrany produkt został dodany do koszyka.