



**WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA**
z siedzibą w Rzeszowie

Projekt
Automatyczne testy funkcjonalne strony wsiz.edu.pl

Prowadzący:
dr inż. Leszek Puzio

Autor:
Piotr Kujda 64874
Kierunek:
Informatyka Niestacjonarne

Rzeszów, 2025 r.

Spis treści

1. Wstęp.....	3
1.1 Cel projektu	3
1.2 Zakres projektu	3
1.3 Środowisko i narzędzia.....	4
1.4 Identyfikacja elementów do testowania	5
2. Struktura projektu	6
3. Opis przypadków testowych.....	7
3.1 Tabela przypadków testowych	7
3.2. test_homepage_basic.py	8
3.3. test_language_switch.py.....	12
3.4. test_logo_redirect.py	15
3.5. test_search.py	17
3.6. test_top_bar_links.py.....	19
3.7. Uruchamianie testów	23
4. Propozycja usprawnień i naprawy napotkanych błędów	24
5. Podsumowanie i wnioski.....	25

1. Wstęp

1.1 Cel projektu

Celem projektu było przygotowanie i uruchomienie zestawu automatycznych testów interfejsu użytkownika dla strony internetowej Wyższej Szkoły Informatyki i Zarządzania w Rzeszowie dostępnej pod adresem <https://wsiz.edu.pl>. Testy zostały wykonane z wykorzystaniem biblioteki Selenium WebDriver w języku Python oraz frameworka pytest, co pozwoliło zautomatyzować czynności, które standardowo wykonywane byłyby manualnie w przeglądarce.

Projekt skupiał się na funkcjach widocznych i dostępnych dla użytkownika końcowego, bez ingerencji w zaplecze serwisu czy API. W szczególności sprawdzono działanie głównej nawigacji, linków kierujących do zewnętrznych usług uczelni, wyszukiwarki, przełącznika wersji językowych oraz odnośnika w postaci logo prowadzącego do strony głównej. Testy odwzorowują rzeczywiste zachowania użytkownika, takie jak kliknięcia kursorem, zmiana karty, przewijanie ekranu czy wpisywanie tekstu w pola formularzy.

W efekcie powstał kompletny zestaw skryptów, który umożliwia powtarzalną i szybką weryfikację kluczowych elementów strony. Zautomatyzowanie tych czynności pozwala łatwiej wykrywać błędy po zmianach na stronie oraz oszczędza czas w porównaniu z testowaniem manualnym.

1.2 Zakres projektu

Projekt obejmował przygotowanie i uruchomienie automatycznych testów interfejsu strony głównej WSiIZ oraz wybranych funkcjonalności dostępnych z poziomu użytkownika. Zakres został celowo ograniczony do elementów widocznych i możliwych do przetestowania bez bezpośredniego dostępu do zaplecza serwisu.

W projekcie uwzględniono:

- Weryfikację ładowania strony głównej
Sprawdzenie, czy serwis otwiera się poprawnie i czy kluczowe elementy, takie jak nagłówek i tytuł strony, są dostępne.
- Test działania menu głównego
Automatyczne przejścia do sekcji takich jak „Uczelnia”, „Dla studenta”, „Dla kandydata”, „Dla biznesu”, „Nauka i Badania” oraz „Dla otoczenia” wraz z potwierdzeniem, że wyświetlany jest odpowiedni content.
- Test przełącznika języków
Automatyczne przejście ze strony głównej do wersji EN, UA i RU, a następnie powrót do polskiej wersji.
- Test przekierowania po kliknięciu logo
Sprawdzenie, czy kliknięcie w logo uczelni przenosi użytkownika z dowolnej podstrony z powrotem na stronę główną.

- Test działania wyszukiwarki
Wpisanie frazy w pole wyszukiwania oraz potwierdzenie wyświetlenia wyników zgodnych z zapytaniem.
- Test linków górnego paska nawigacyjnego
Sprawdzenie odnośników kierujących do zewnętrznych usług uczelni, takich jak Wirtualna Uczelnia, e-Learning, Poczta, e-Usługi czy System Wydruków, z uwzględnieniem obsługi otwierania nowych kart.

1.3 Środowisko i narzędzia

Testy zostały wykonane w środowisku opartym na języku Python z wykorzystaniem biblioteki Selenium WebDriver, pozwalającej na automatyzację interakcji z przeglądarką. Projekt został uruchomiony lokalnie, bez wykorzystania zewnętrznych narzędzi CI/CD.

Do realizacji projektu wykorzystano następujące narzędzia i komponenty:

- Język programowania: Python 3.14
Główny język wykorzystywany do pisania skryptów testujących.
- Framework testowy: PyTest
Umożliwił organizację testów w postaci osobnych modułów oraz uruchamianie ich zbiorczo.
- Biblioteki:
 - selenium – obsługa przeglądarki, wyszukiwanie elementów, wykonywanie akcji użytkownika,
 - pytest – framework testowy, uruchamianie i raportowanie testów.
- Automatyzacja przeglądarki: Selenium WebDriver (Chrome)
Przeglądarka była uruchamiana lokalnie, w trybie graficznym, bez trybu headless, aby móc obserwować przebieg testów.
- IDE: JetBrains PyCharm
Środowisko wykorzystywane do pisania kodu, zarządzania strukturą projektu oraz wirtualnym środowiskiem.
- Wirtualne środowisko: venv
Oddzieliło zależności projektu od reszty systemu.
- System operacyjny: Windows 10
Środowisko wykonywania testów.

Dodatkowo konfiguracja testów zakłada:

- uruchamianie każdego testu w nowej instancji przeglądarki (czysty stan sesji),
- ręczne zatwierdzanie baneru dotyczącego cookies przed wykonaniem akcji testowych,
- brak modyfikacji stanu użytkownika.

1.4 Identyfikacja elementów do testowania

W projekcie skupiono się na testowaniu podstawowych funkcji dostępnych z poziomu strony głównej oraz kluczowych elementów interfejsu użytkownika. Wybrane scenariusze testowe obejmują elementy istotne z perspektywy pierwszego kontaktu użytkownika z witryną oraz nawigacji po najważniejszych sekcjach.

Testowane zostały następujące obszary:

a) Menu główne (nawigacja górna)

Każdy główny element menu został przetestowany pod kątem poprawnego przekierowania do odpowiedniej podstrony. Testy obejmowały kliknięcia w:

- Uczelnia
- Nauka i Badania
- Dla Studenta
- Dla Kandydata
- Dla Biznesu
- Dla Otoczenia

Celem było sprawdzenie, czy po kliknięciu link otwiera się poprawna podstrona, a adres URL ulega zmianie.

b) Górne linki serwisowe (panel narzędziowy)

Przetestowano odnośniki kierujące do zewnętrznych systemów uczelnianych, w tym:

- Wirtualna Uczelnia
- e-Learning
- Poczta
- e-Usługi
- Biblioteka
- System Wydruków
- e-Praktyki

Część z nich otwiera się w nowej karcie, co zostało uwzględnione w testach.

c) Wyszukiwarka strony

Przetestowano działanie ikony lupy, otwieranie pola wyszukiwania oraz poprawne wyświetlanie wyników po wpisaniu frazy *"informatyka"*.

d) Przełączanie wersji językowych strony

Sprawdzono poprawność przejścia z wersji polskiej do wersji:

- angielskiej (en.uitm.edu.eu),
 - ukraińskiej (ua.uitm.edu.eu),
 - rosyjskiej (ru.uitm.edu.eu),
- a następnie powrót do wersji polskiej poprzez kliknięcie ikony flagi.

e) Logo strony

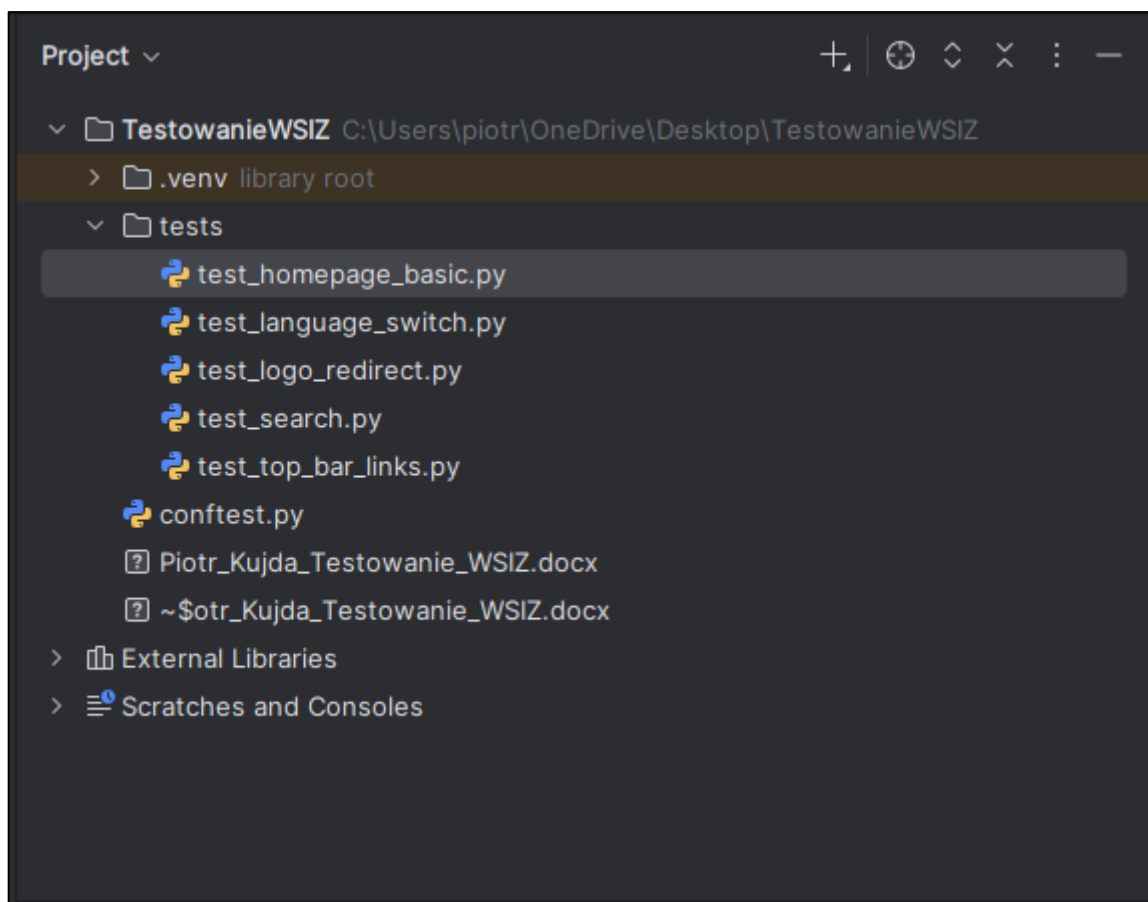
Przetestowano poprawność działania linku powrotnego do strony głównej poprzez kliknięcie w logo uczelni z poziomu podstrony.

2. Struktura projektu

Struktura katalogów (istotne elementy):

- conftest.py – plik z konfiguracją pytest, zawierający wspólny fixture wsiz_home tworzony dla każdego testu:
 - uruchamia nową instancję Chrome,
 - przechodzi na stronę <https://wsiz.edu.pl/>,
 - maksymalizuje okno,
 - po zakończeniu testu zamyka przeglądarkę.
- katalog tests/:
 - test_homepage_basic.py – testy nawigacji w głównym czerwonym menu,
 - test_language_switch.py – test przełączania wersji językowej PL ↔ EN/UA/RU,
 - test_logo_redirect.py – test przekierowania z podstrony na stronę główną po kliknięciu logo,
 - test_search.py – test pola wyszukiwania (ikona lupy),
 - test_top_bar_links.py – testy linków w górnym (czarnym) pasku: Wirtualna Uczelnia, e-Learning, Poczta, e-Usługi, Biblioteka, System wydruków, e-Praktyki.

- Każdy plik testowy korzysta z tego samego fixture'a `wsiz_home`, dzięki czemu każdy test działa w osobnej, czystej sesji przeglądarki.



Rysunek 1. Schemat struktury projektu.

3. Opis przypadków testowych

3.1 Tabela przypadków testowych

ID	Nazwa testu	Kroki	Oczekiwany rezultat
TC-01	Wyświetlanie strony głównej	Wejść na <code>wsiz.edu.pl</code>	Strona ładuje się poprawnie, nagłówki widoczne
TC-02	Nawigacja → Uczelnia	Kliknąć „Uczelnia” w menu	Ładuje się strona z treścią o uczelni
TC-03	Nawigacja → Nauka i Badania	Kliknąć „Nauka i Badania”	Strona z treścią o nauce/badaniach
TC-04	Nawigacja → Dla studenta	Kliknąć „Dla studenta”	Strona dla studentów
TC-05	Nawigacja → Dla kandydata	Kliknąć „Dla kandydata”	Strona dla kandydatów

TC-06	Nawigacja → Dla biznesu	Kliknąć „Dla biznesu”	Strona dla biznesu
TC-07	Nawigacja → Dla otoczenia	Kliknąć „Dla otoczenia”	Strona dla otoczenia
TC-08	Link → Wirtualna Uczelnia	Kliknąć link w górnym pasku	Przekierowanie do <i>my.wsiz.edu.pl</i>
TC-09	Link → e-Learning	Kliknąć link	Przekierowanie do <i>moodle.wsiz.edu.pl</i>
TC-10	Link → Poczta	Kliknąć link	Przekierowanie do portalu Microsoft (Outlook)
TC-11	Link → e-Usługi	Kliknąć link	Otwiera <i>euslugi.wsiz.pl</i> w nowej karcie
TC-12	Link → Biblioteka	Kliknąć link	Otwiera stronę biblioteki
TC-13	Link → System Wydruków	Kliknąć link	Otwiera <i>print.wsiz.rzeszow.pl</i> w nowej karcie
TC-14	Link → e-Praktyki	Kliknąć link	Przekierowanie do <i>portal.wsiz.edu.pl</i>
TC-15	Wyszukiwarka — wyszukiwanie	Kliknąć lupę → wpisać frazę → ENTER	Wyświetlone wyniki z frazą „informatyka”
TC-16	Zmiana języka PL → EN → PL	Wybrać EN, potem PL	Przejsie do wersji EN, potem powrót do <i>wsiz.edu.pl</i>
TC-17	Zmiana języka PL → UA → PL	Wybrać UA, potem PL	Przejsie do UA i powrót do PL
TC-18	Zmiana języka PL → RU → PL	Wybrać RU, potem PL	Przejsie do RU i powrót do PL
TC-19	Kliknięcie logo uczelni	Wejść w „Dla studenta”, kliknąć logo	Powrót na stronę główną

3.2. test_homepage_basic.py

Cel: Sprawdzenie, czy strona główna działa oraz czy linki w głównym menu przenoszą użytkownika do odpowiednich sekcji serwisu.

1. test_homepage_loads

- Wejście na stronę główną.

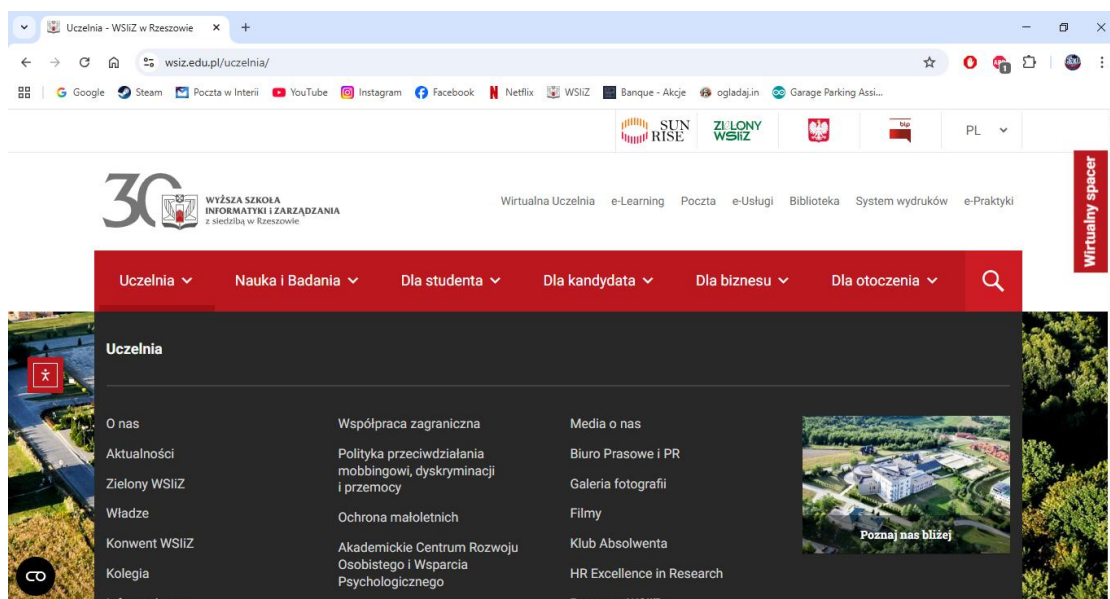


Rysunek 2. Strona główna Uczelni.

- Oczekiwanie na pojawienie się elementu header.
- Sprawdzenie, czy tytuł strony zawiera nazwę uczelni („WSiIZ” lub „Wyższa Szkoła Informatyki...”).

2. test_nawigacja_uczelnia

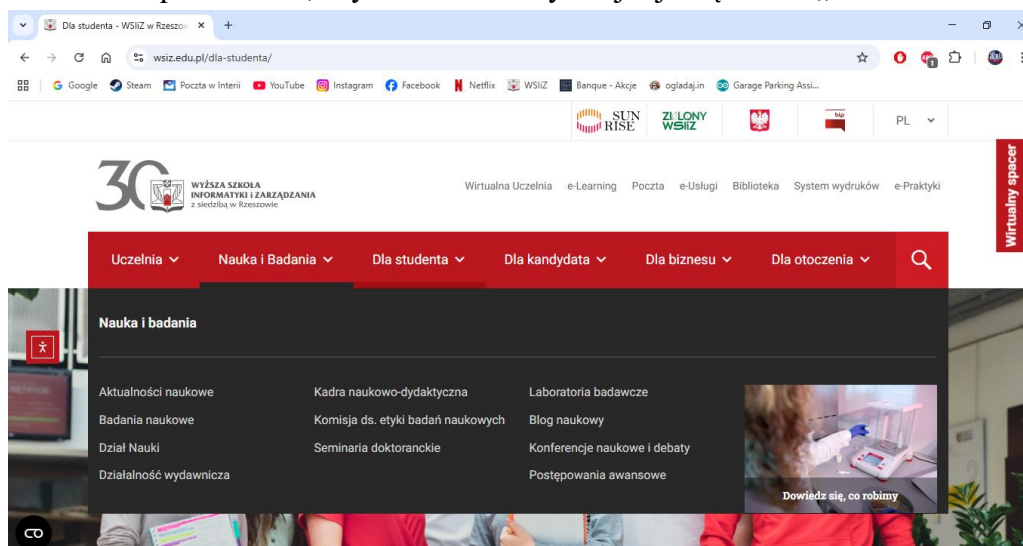
- Kliknięcie pozycji menu „Uczelnia” (po XPATH z mappingu MENU_XPATHS).
- Oczekiwanie na zmianę adresu URL.
- Sprawdzenie, czy w treści strony znajduje się słowo „uczelnia”.



Rysunek 3. Strona główna z zakładką „Uczelnia”.

3. test_nawigacja_nauka_i_badania

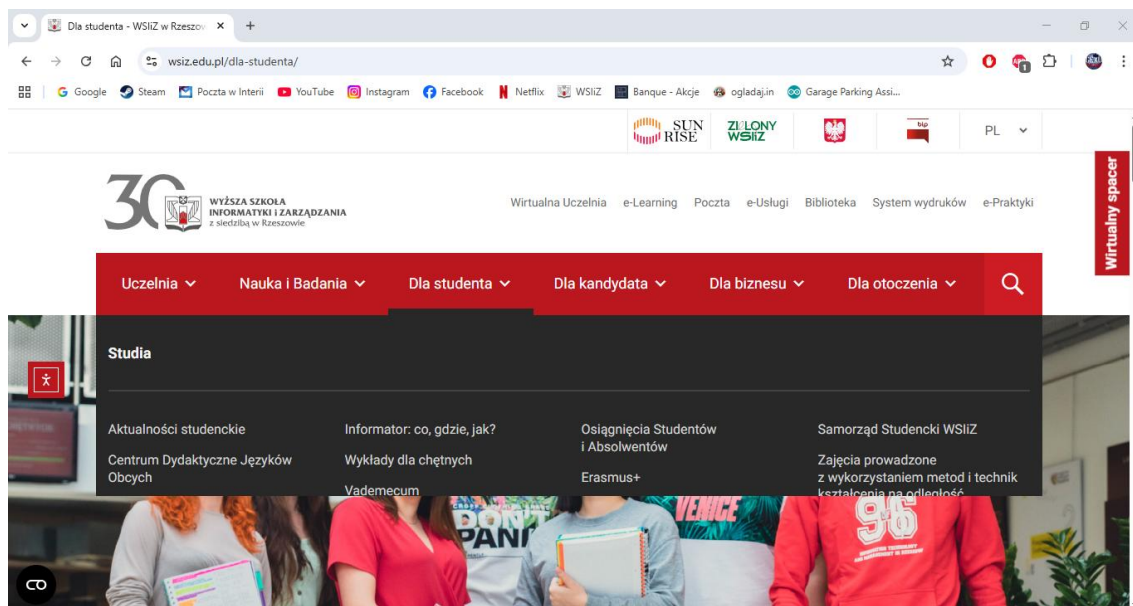
- Kliknięcie „Nauka i Badania”.
- Oczekiwanie na zmianę URL.
- Sprawdzenie, czy w treści strony znajduje się słowo „nauka”.



Rysunek 4. Strona główna z zakładką „Nauka i Badania”.

4. test_nawigacja_dla_studenta

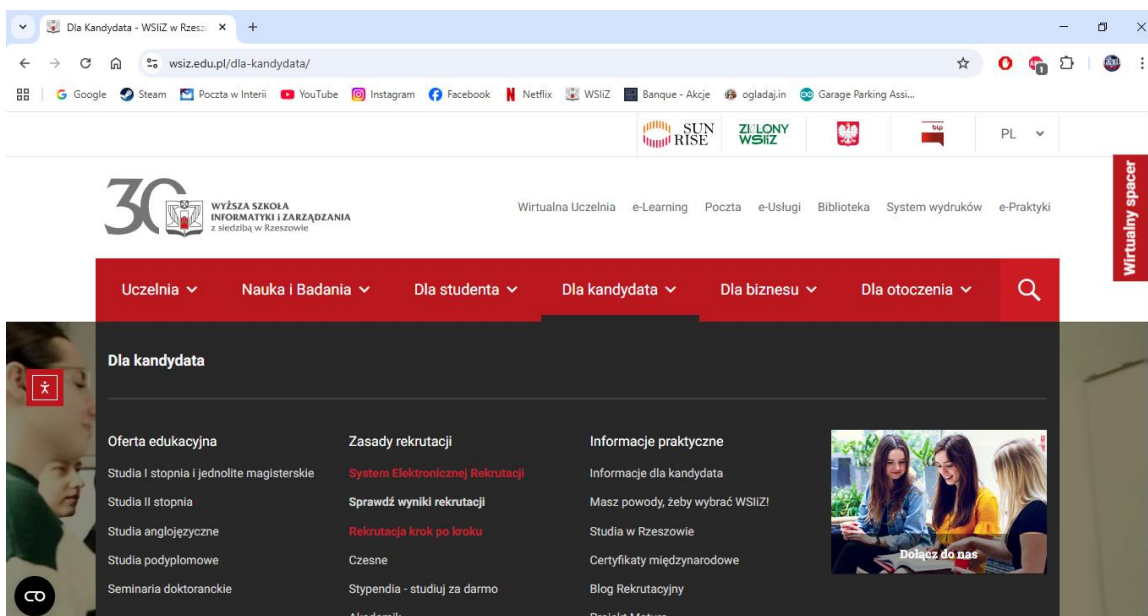
- Kliknięcie „Dla studenta”.
- Sprawdzenie, czy w treści pojawia się słowo „student”.



Rysunek 5. Strona główna z zakładką "Dla Studenta".

5. test_nawigacja_dla_kandydata

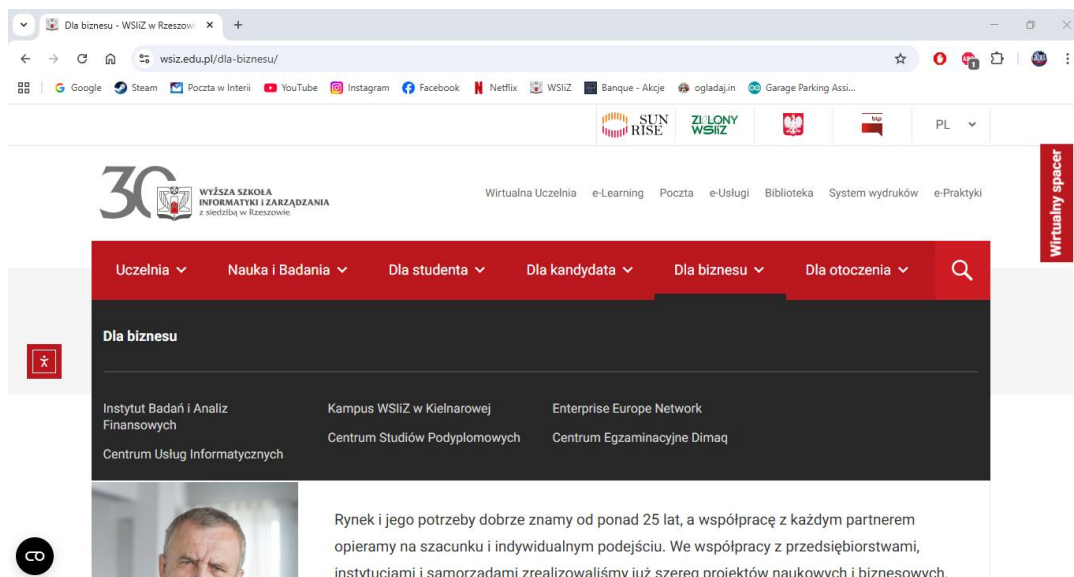
- Kliknięcie „Dla kandydata”.
- Sprawdzenie, czy w treści znajduje się słowo „kandydat”.



Rysunek 6. Strona główna z zakładką "Dla kandydata".

6. test_nawigacja_dla_biznesu

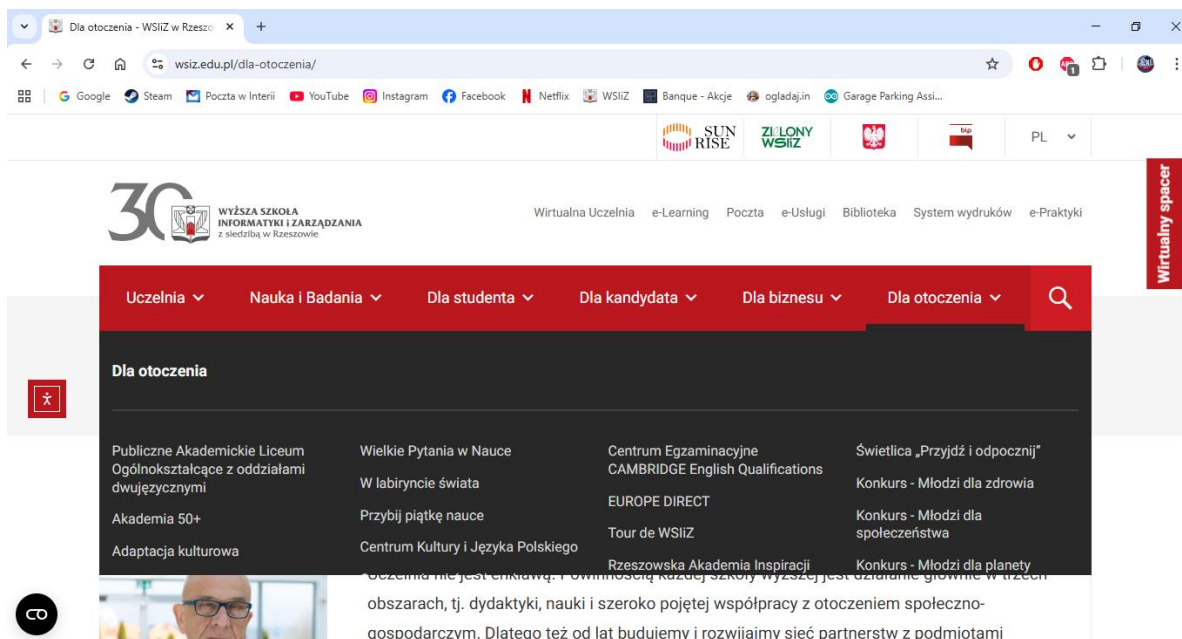
- Kliknięcie „Dla biznesu”.
- Sprawdzenie, czy w treści występuje słowo „biznes”.



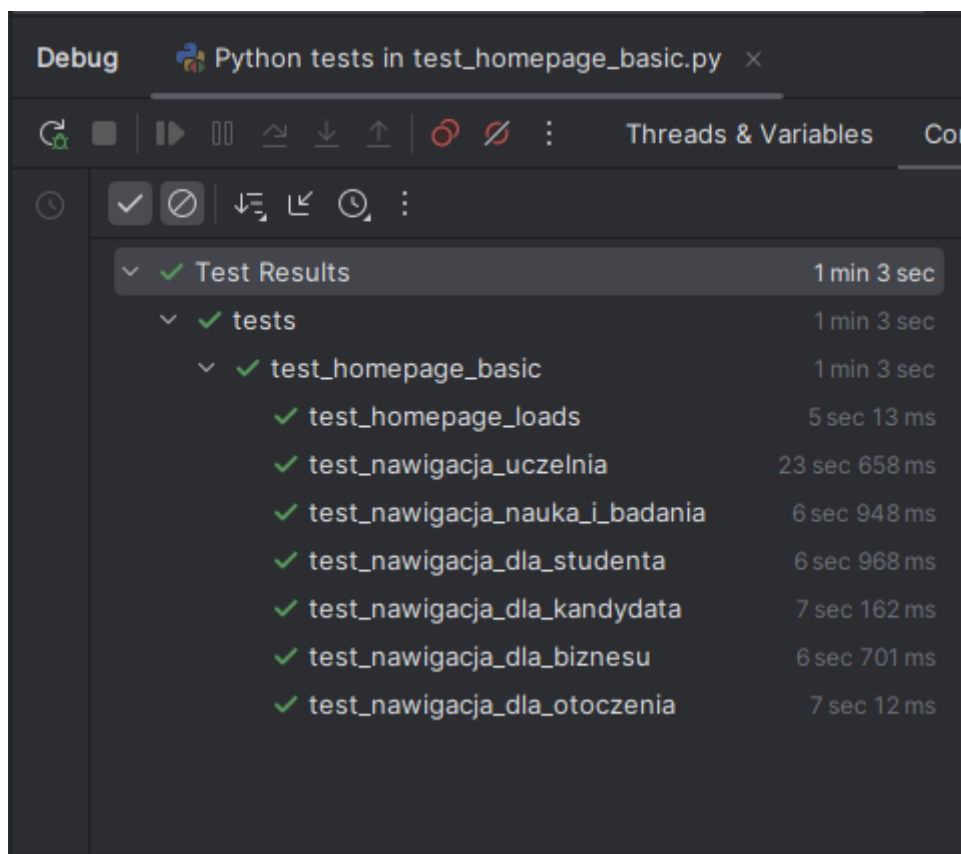
Rysunek 7. Strona główna z zakładką "Dla biznesu".

7. test_nawigacja_dla_otoczenia

- Kliknięcie „Dla otoczenia”.
- Sprawdzenie, czy w treści występuje słowo „otoczenia”.



Rysunek 8. Strona główna z zakładką "Dla otoczenia".



Rysunek 9. Końcowe wyniki przeprowadzonych testów.

3.3. test_language_switch.py

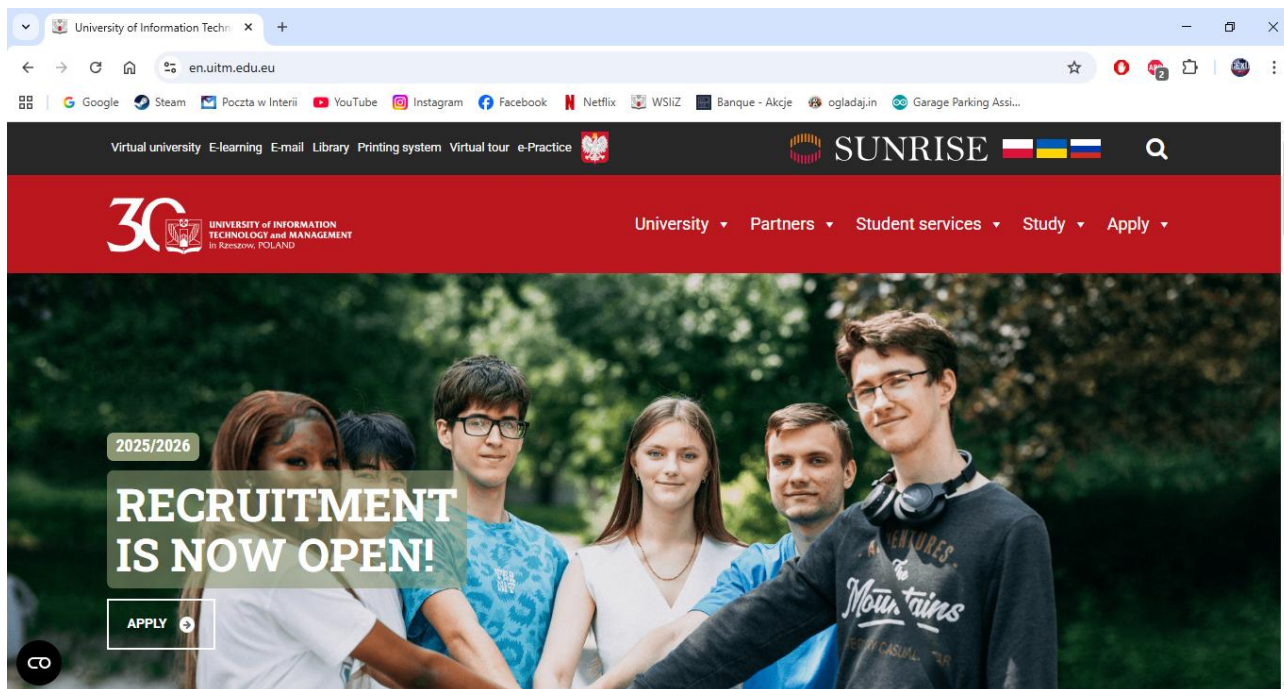
Cel: Weryfikacja działania przełącznika wersji językowych PL ↔ EN/UA/RU.

Dla każdego z trzech testów:

1. Początkowo użytkownik jest na stronie wsiz.edu.pl (wersja PL).
2. Skrypt znajduje element `<select id="dynamic_select">`.
3. Po atrybucie value wybierana jest odpowiednia wersja językowa:
 - en.uitm.edu.eu,
 - ua.uitm.edu.eu,
 - ru.uitm.edu.eu.
4. Oczekuje się przełączenia domeny (np. adres zawiera en.uitm.edu.eu).
5. Następnie znajdowana jest ikona polskiej flagi (po fragmencie ścieżki grafiki flaga_polska) i klikana, co powinno przywrócić adres w domenę wsiz.edu.pl.

Testy:

- test_language_switch_pl_en_pl



Rysunek 10. Główna strona Uczelni w języku angielskim.

- test_language_switch_pl_ua_pl



Rysunek 11. Główna strona Uczelni w języku ukraińskim.

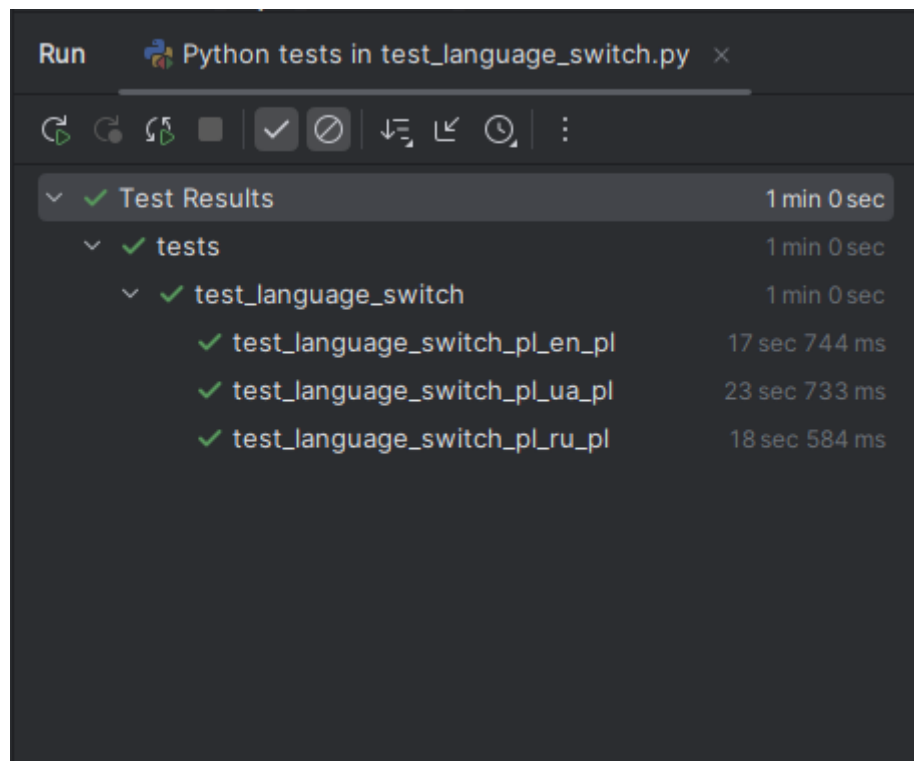
- test_language_switch_pl_ru_pl



Rysunek 12. Główna strona Uczelni w języku rosyjskim.

Każdy test sprawdza:

- czy po wyborze języka adres zawiera odpowiednią subdomenę,
- czy po kliknięciu polskiej flagi następuje powrót do wsiz.edu.pl.



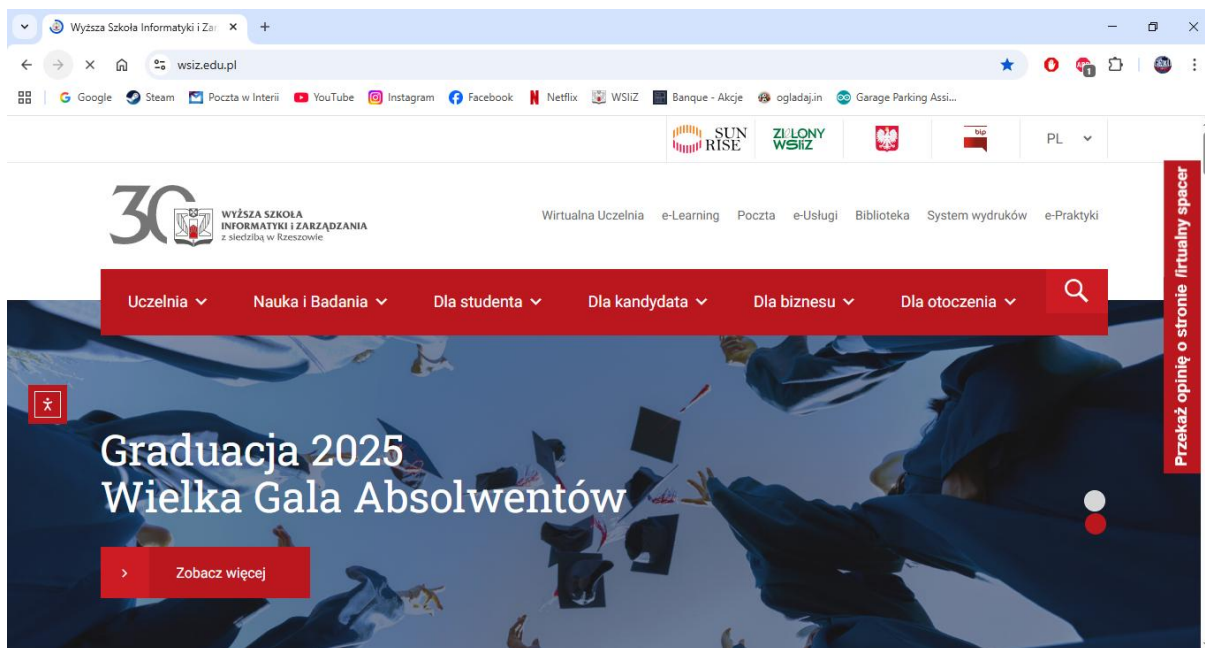
Rysunek 13. Końcowe wyniki przeprowadzonych testów.

3.4. test_logo_redirect.py

Cel: Sprawdzenie, czy kliknięcie logo uczelni przenosi z podstrony z powrotem na stronę główną.

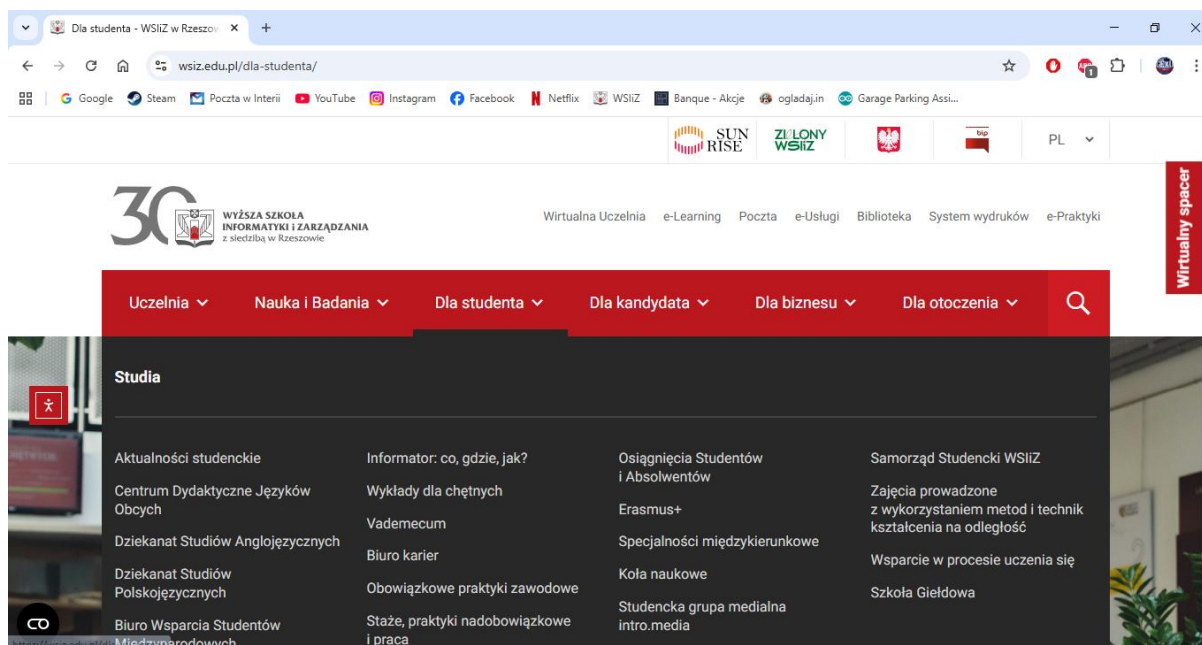
Scenariusz:

1. Wejście na stronę główną.



Rysunek 14. Główna strona Uczelni.

2. Kliknięcie pozycji menu „Dla studenta” (po linku tekstowym).



Rysunek 15. Główna strona z zakładką „Dla studenta”.

3. Czekanie aż adres URL będzie zawierał wsiz.edu.pl/dla-studenta.

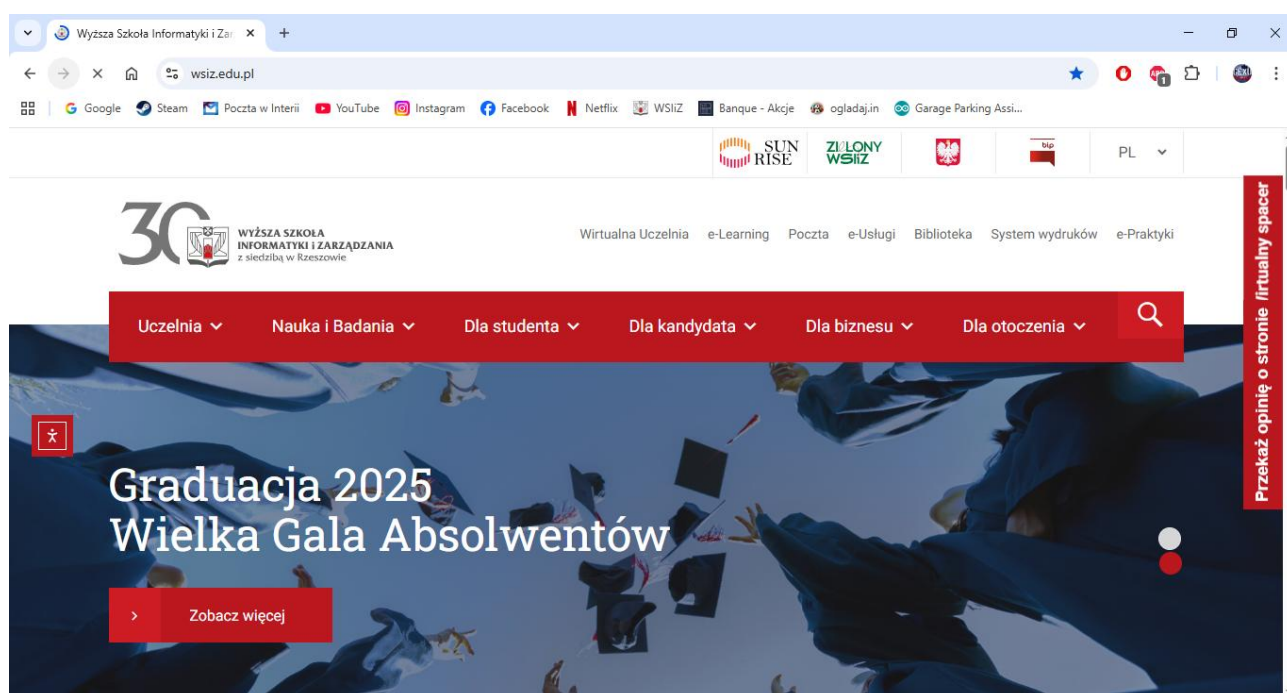
4. Krótkie odczekanie (5 sekund) na podstronie.

5. Wyszukanie elementu logo – link o XPATH `//*[@id='mega-menu-item-52539']/a`.
6. Przewinięcie do logo i kliknięcie go przy pomocy JavaScript (`driver.execute_script("arguments[0].click();", logo_link)`).

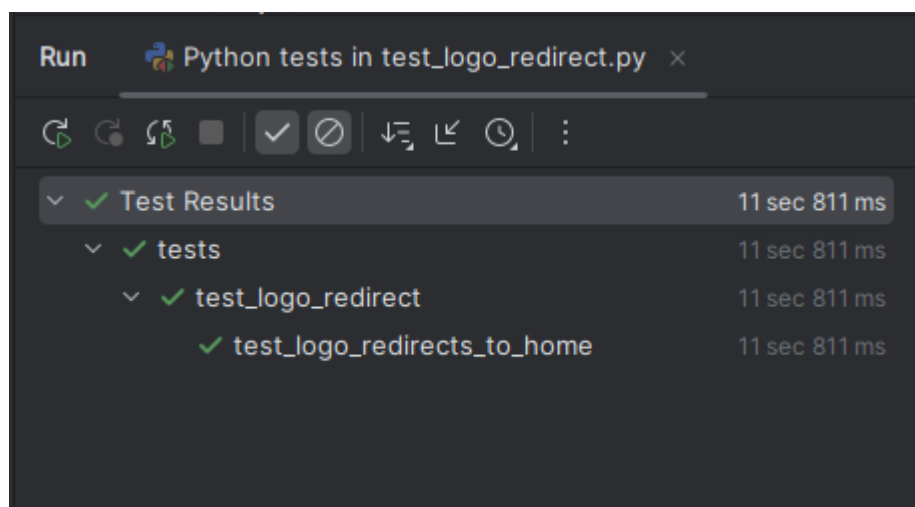


Rysunek 16. Logo Uczelni.

7. Oczekiwanie, aż adres będzie zaczynał się od `https://wsiz.edu.pl`.
8. Asercja, że rzeczywiście nastąpił powrót na stronę główną.



Rysunek 17. Główna strona Uczelni.



Rysunek 18. Końcowe wyniki przeprowadzonych testów.

3.5. test_search.py

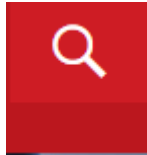
Cel: Sprawdzenie działania wyszukiwarki (ikona lupy w górnym menu).

Scenariusz:

1. Po załadowaniu strony krótkie oczekiwanie (`time.sleep(3)`).

2. Kliknięcie ikony lupy po XPATH:

- `SEARCH_TOGGLE_XPATH = "//*[@id='mega-menu-item-52922']/div/form/span".`

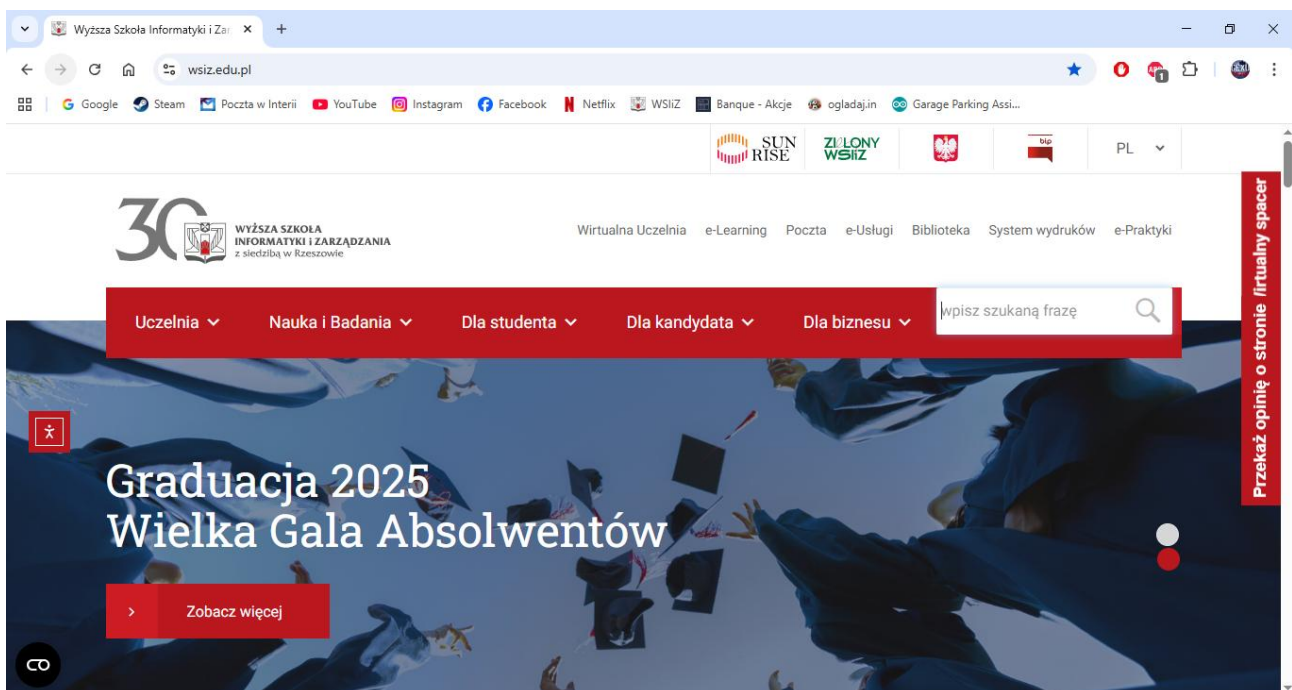


Rysunek 19. Ikona lupy.

3. Oczekiwanie (w miarę możliwości) na ustawienie atrybutu `aria-expanded="true"` – informacja, że pole wyszukiwania zostało rozwinięte.

4. Pobranie pola tekstowego po identyfikatorze:

- `SEARCH_INPUT_ID = "mega-search-52922".`



Rysunek 20. Strona główna Uczelni z możliwością wpisania frazy.

5. Ustawienie focusa na polu, wyczyszczenie zawartości.

6. Wpisanie frazy „informatyka” oraz naciśnięcie ENTER.

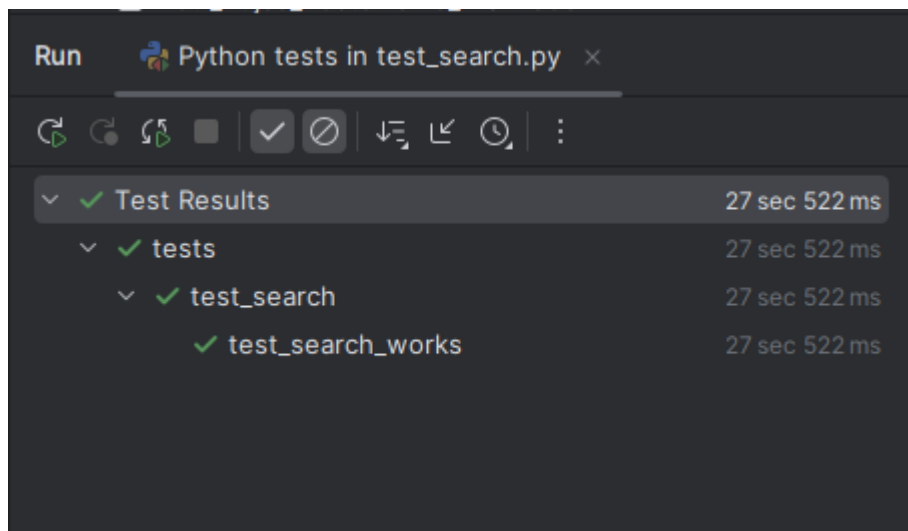


Rysunek 21. Główna strona Uczelni z wpisaną frazą „informatyka”.

7. Oczekiwanie, aż załadują się wyniki (sprawdzany fragment ?s= w URL lub obecność frazy w źródle strony).
8. Asercja, że na stronie wyników występuje tekst „informatyka”.



Rysunek 22. Wyniki wyszukiwania dla wpisanej frazy.



Rysunek 23. Końcowe wyniki przeprowadzonego testu.

3.6. test_top_bar_links.py

Cel: Weryfikacja poprawności odnośników w górnym (czarnym) pasku narzędziowym.

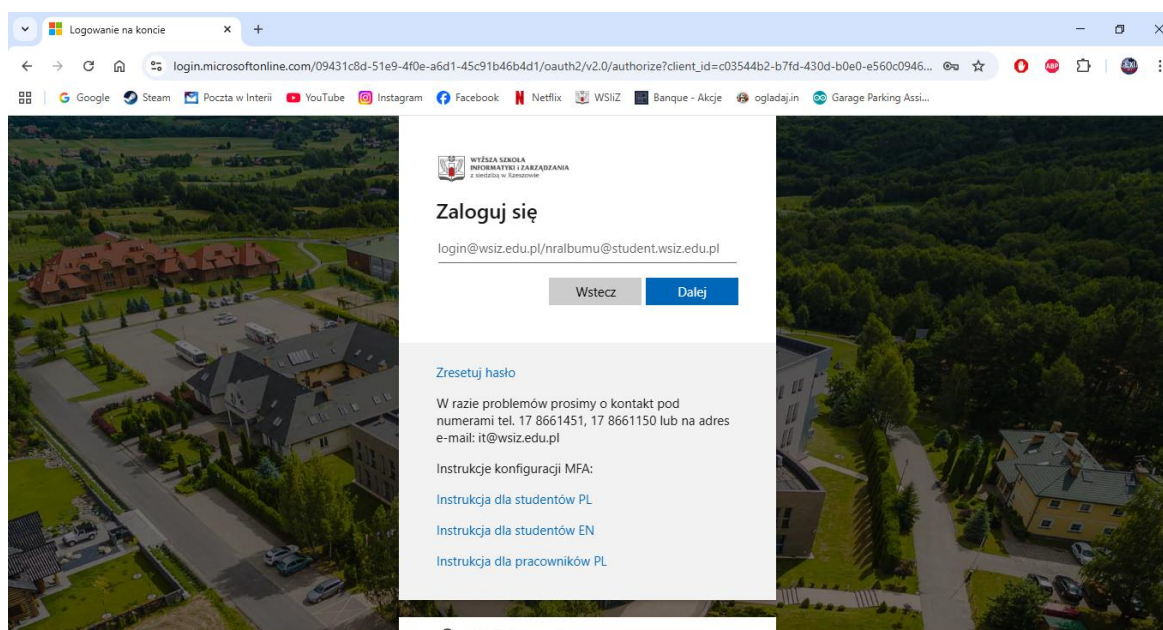
Dla wszystkich testów używane są osobne funkcje pomocnicze:

- `_click_link_same_tab_xpath(...)` – kliknięcie linku, który otwiera się w tej samej karcie (sprawdzana zmiana URL i obecność oczekiwanego fragmentu),
- `_click_link_new_tab_xpath(...)` – kliknięcie linku otwieranego w nowej karcie (sprawdzenie pojawienia się nowej karty, przełączenie na nią, weryfikacja URL).

Testowane linki:

1. test_top_link_wirtualna_uczelnia

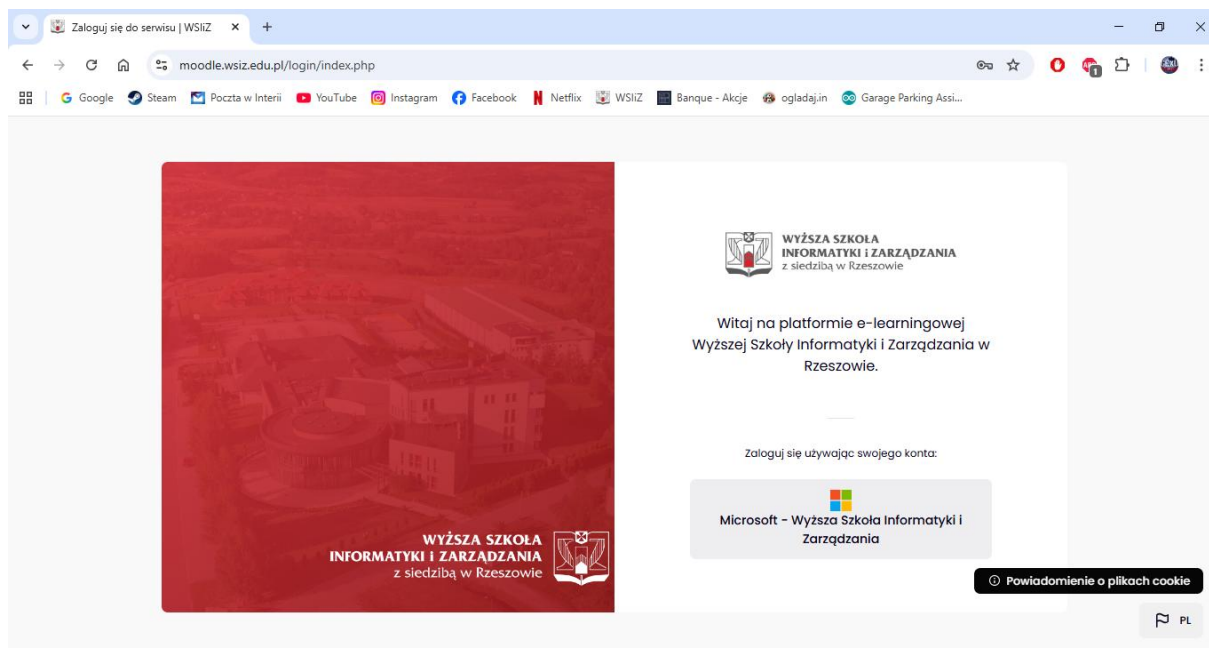
- XPATH: `//*[@id='mega-menu-item-52540']/a`
- Oczekiwane fragmenty URL: `my.wsiz.edu.pl`.



Rysunek 24. Główna strona "Wirtualna Uczelnia".

2. test_top_link_e_learning

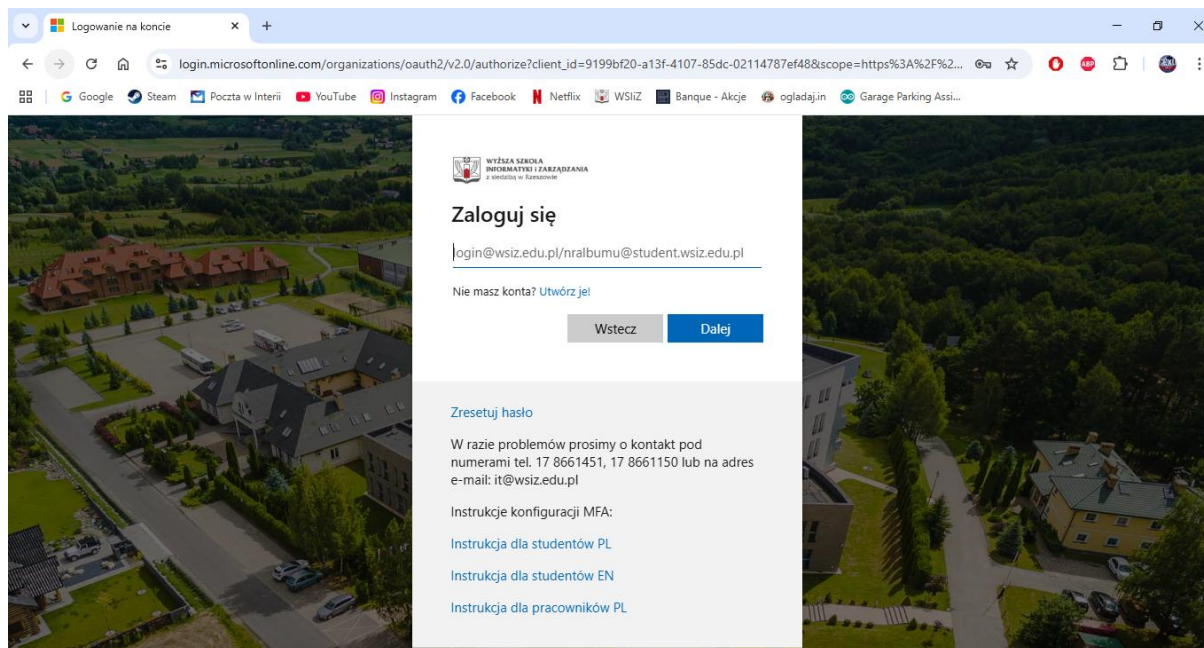
- XPATH: `//*[@id='mega-menu-item-52541']/a`
- Oczekiwane: `moodle.wsiz.edu.pl`.



Rysunek 25. Główna strona "e-Learning".

3. test_top_link_poczta

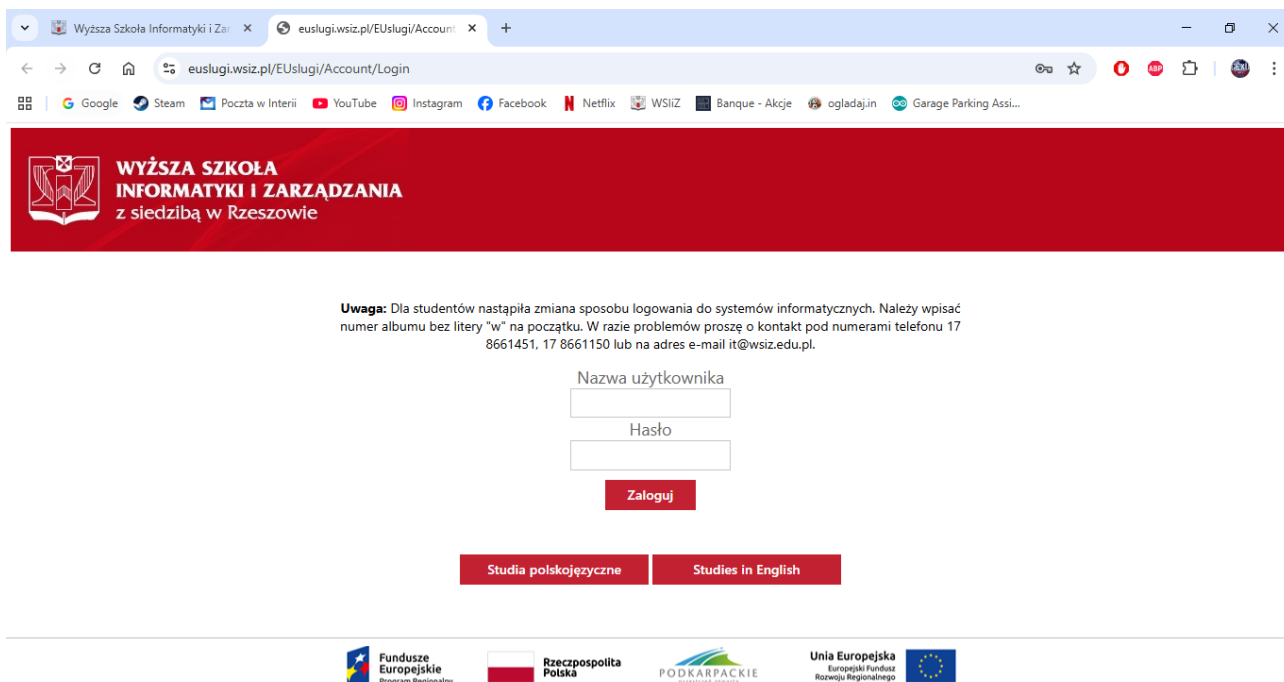
- XPATH: `//*[@id='mega-menu-item-52542']/a`
- Oczekiwane: `m.in. poczta.wsiz.edu.pl`, `outlook.office365.com`, `office.com`, `microsoftonline.com`.



Rysunek 26. Główna strona "Poczta".

4. test_top_link_e_uslugi (nowa karta)

- XPATH: //*[@id='mega-menu-item-52543']/a
- Oczekiwane: euslugi.wsiz.pl.



Rysunek 27. Główna strona “e-Usługi”.

5. test_top_link_biblioteka

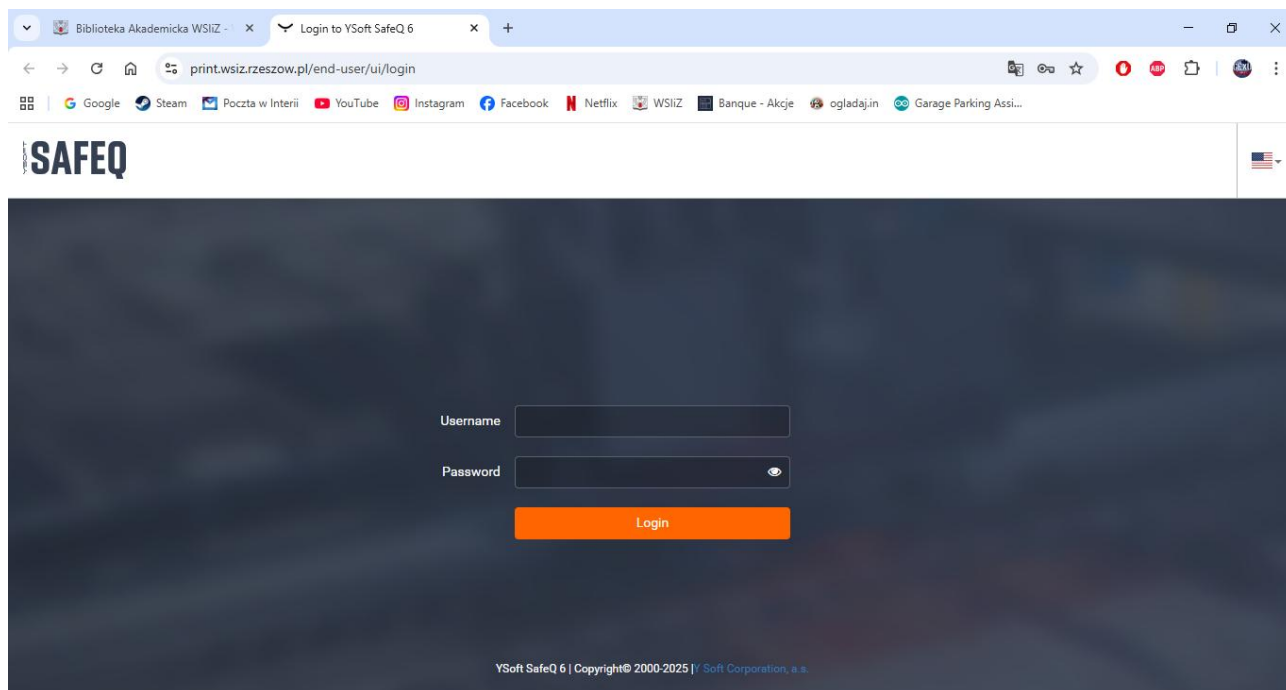
- XPATH: //*[@id='mega-menu-item-52544']/a
- Oczekiwane: wsiz.edu.pl/uczelnia/jednostki-wsiiz/biblioteka-wsiiz lub skrócona forma biblioteka-wsiiz.



Rysunek 28. Główna strona “Biblioteka”.

6. test_top_link_system_wydrukow (nowa karta)

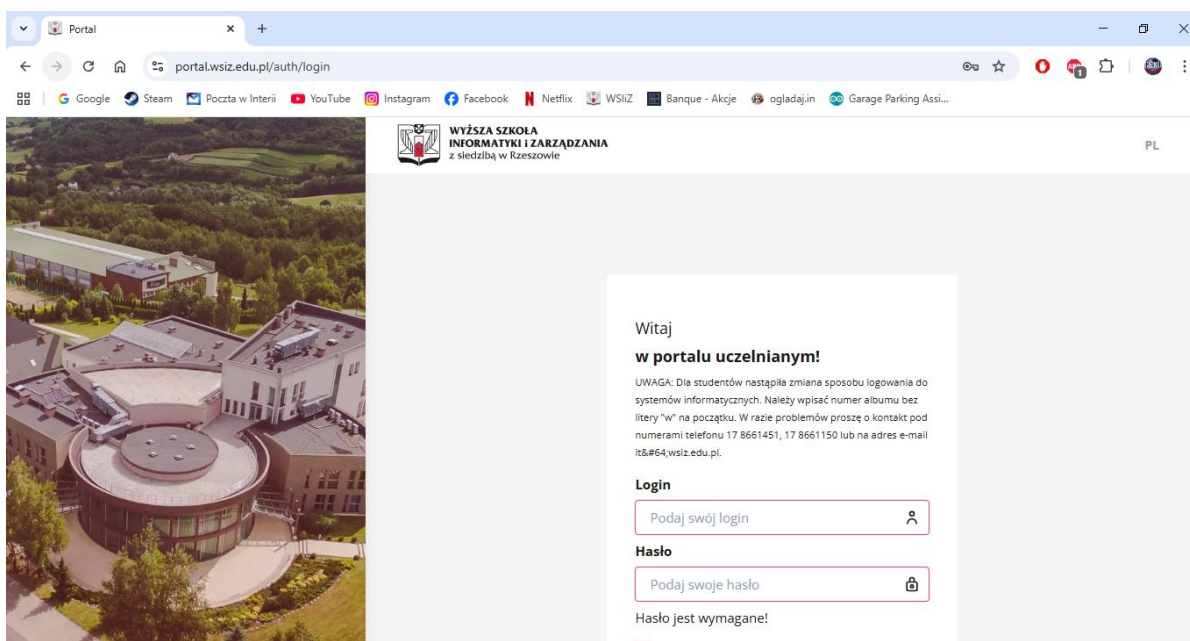
- XPATH: //*[@id='mega-menu-item-52545']/a
- Oczekiwane: print.wsiz.rzeszow.pl.



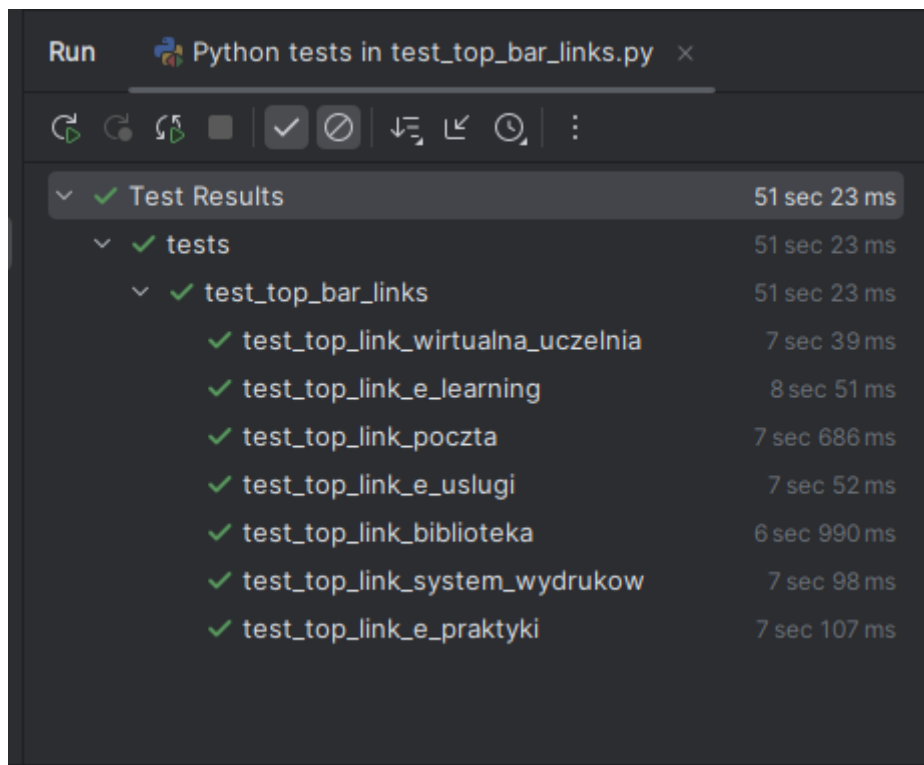
Rysunek 29. Główna strona "System wydruków".

7. test_top_link_e_praktyki

- XPATH: //*[@id='mega-menu-item-99388']/a
- Oczekiwane: portal.wsiz.edu.pl.



Rysunek 30. Główna strona "e-Praktyki".



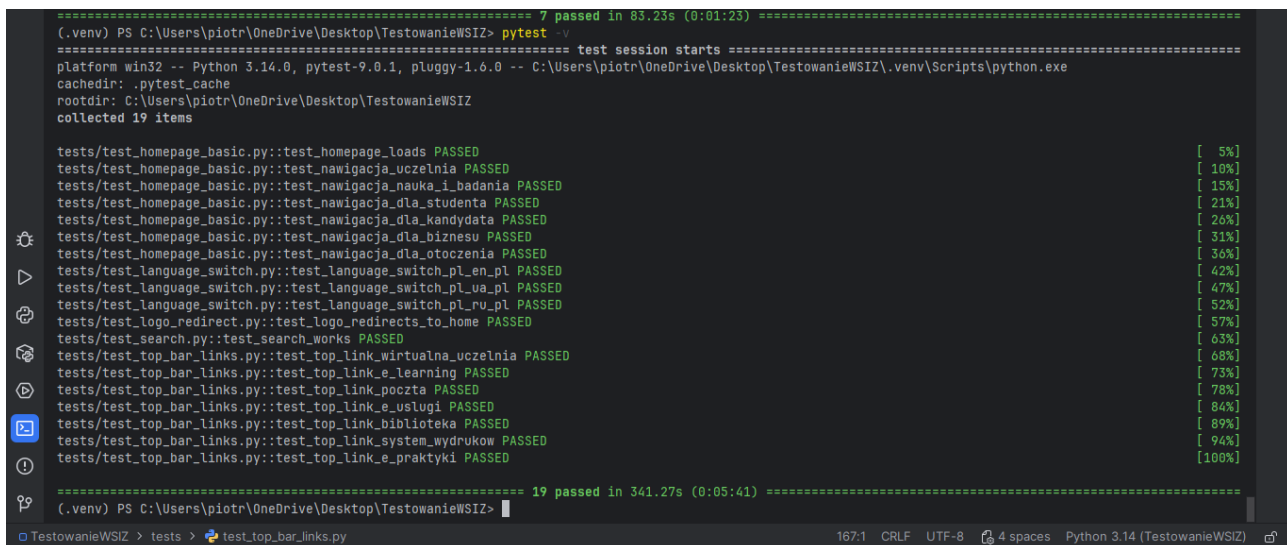
Rysunek 31. Końcowe wyniki przeprowadzonych testów.

3.7. Uruchamianie testów

Wszystkie testy uruchamiane są komendą:

```
pytest -v
```

Wynik końcowy z konsoli: 19 passed in 341.27s (0:05:41).



Rysunek 32. Końcowe wyniki wszystkich przeprowadzonych testów.

Oznacza, że wszystkie przygotowane przypadki testowe zakończyły się powodzeniem.

4. Propozycja usprawnień i naprawy napotkanych błędów

Podczas realizacji testów pojawiło się kilka problemów związanych głównie z dynamicznym zachowaniem strony oraz sposobem działania elementów JavaScript. Poniżej przedstawiono obserwacje oraz możliwe usprawnienia.

- Dynamiczne przeładowania strony po jej otwarciu

Strona główna czasami automatycznie przewija się i przeładowuje tuż po wejściu. Powodowało to losowe błędy, szczególnie w testach wyszukiwarki i nawigacji. Ostatecznie rozwiązano to poprzez dodanie krótkich opóźnień oraz przewijania do góry strony.

Możliwe usprawnienie:

- wdrożenie stabilnego oczekiwania opartego na `document.readyState == 'complete'`
- opóźnione inicjalizowanie elementów dynamicznych po stronie front-endu
- uproszczenie animacji startowych
- Elementy zasłanianie przez inne komponenty (baner cookies, sticky menu, animacje)

Kliknięcia czasem nie działały, ponieważ element był widoczny, ale przykryty niewidoczną warstwą (np. animacja menu lub niewidzialna nakładka). Rozwiązano to użyciem `execute_script("arguments[0].click()", element)` oraz przewijaniem widoku.

Możliwe usprawnienie:

- usunięcie nakładek pełnoekranowych albo ich opóźnienie
- zmiana `z-index` elementów, które nie powinny blokować UI
- zapewnienie możliwości kliknięcia przez standardową metodę Selenium
- Niestabilne zachowanie wyszukiwarki

Pole wyszukiwania sporadycznie nie przyjmowało wpisanego tekstu, szczególnie tuż po kliknięciu w ikonę lupy. Powodem była asynchroniczna inicjalizacja elementu oraz animacja otwierania.

Możliwe usprawnienie:

- wymuszenie focusu na elemencie po stronie frontu (`.focus()`)
- opóźnione wyświetlanie inputu dopiero gdy JS zakończy inicjalizację
- dodanie stałego identyfikatora lub flagi gotowości
- Testy oparte na lokalizatorach XPATH

Część testów oparta była o XPATH-y powiązane z pełną strukturą HTML. Działało to poprawnie, ale było bardziej podatne na zmiany layoutu niż lokalizowanie po tekście linku lub atrybutach.

Możliwe usprawnienie:

- stosowanie `By.LINK_TEXT` lub `By.CSS_SELECTOR` tam, gdzie możliwe
- dodawanie stabilnych atrybutów testowych typu `data-testid`
- Problemy z testami językowymi przy odświeżeniach domen

Przy zmianie wersji językowej pojawiały się błędy wynikające z przeładowania i przekierowań między domenami `wsiz.edu.pl` → `*.uitm.edu.eu`.

Możliwe usprawnienie:

- dodanie globalnego `wait_for_url_change()` zamiast twardych delayów
- wykorzystanie `WebDriverWait` z warunkiem na domenę zamiast string matching

5. Podsumowanie i wnioski

Projekt miał na celu opracowanie, uruchomienie i analizę automatycznych testów funkcjonalnych dla strony internetowej `wsiz.edu.pl` przy użyciu środowiska Python, biblioteki Selenium oraz frameworka testowego Pytest. W ramach prac stworzono pełny zestaw testów odwzorowujących rzeczywiste scenariusze użytkownika korzystającego z serwisu – od podstawowego ładowania strony, poprzez nawigację w menu, interakcję z wyszukiwarką, obsługę odnośników do usług zewnętrznych, aż po przełączanie wersji językowych oraz powrót do strony głównej z poziomu logo. Łącznie przygotowano 19 testów, wszystkie zakończone statusem „PASSED”.

Realizacja projektu pozwoliła przeanalizować zachowanie strony w warunkach automatyzacji. Witryna wykorzystuje rozbudowane menu, dynamiczne ładowanie elementów oraz warstwy nakładkowe (np. baner cookie, animacje przewijania, przechodzenie przez kotwice), co wprowadziło trudności w stabilnym klikaniu niektórych elementów. Problemem okazały się także losowe przeładowania strony tuż po wejściu oraz sytuacje, w których elementy były widoczne w DOM, lecz nie były fizycznie „klikane” z powodu nałożonych warstw. W celu zapewnienia stabilności wykorzystano m.in.:

- klikanie przez JavaScript zamiast standardowej metody `.click()`,
- przewijanie widoku do klikanych elementów,
- oczekiwanie na faktyczną zmianę adresu URL zamiast tylko pojawienia się elementu w DOM,
- kilkusekundowe pauzy na manualne zatwierdzenie banera cookie.

Wnioskiem z obserwacji jest fakt, że automatyzacja testów aplikacji webowych wymaga nie tylko znajomości Selenium, ale także dostosowania logiki testów do specyfiki działania strony. Serwis `wsiz.edu.pl` nie posiada atrybutów testowych (np. `data-testid`), co utrudnia jednoznaczne i odporne na zmiany selekcjonowanie elementów. Część linków otwiera się w nowej karcie, co wymaga

dodatkowej obsługi przełączania kontekstu WebDriver. Sam baner cookie nie zawsze wczytuje się w tym samym momencie, co również wpływa na deterministyczność testów.

Rezultaty testów wskazują, że kluczowe funkcje strony działają poprawnie, jednak pewne obszary mogłyby zostać zoptymalizowane pod kątem UX i automatyzacji. Szczególnie przydatne mogłyby być:

- stabilne identyfikatory elementów niezależne od warstwy wizualnej strony,
- wyeliminowanie niepotrzebnych przeładowań po otwarciu witryny,
- uwzględnienie oficjalnego API lub endpointów dla wyszukiwarki,
- bardziej przewidywalne zachowanie przy zmianie języka.

Podsumowując, postawione cele projektu zostały osiągnięte, a wykonane testy stanowią solidną podstawę do dalszego rozwoju automatyzacji. Projekt dostarczył także praktycznego doświadczenia z obsługą Selenium i realnymi problemami testowania współczesnych, dynamicznych interfejsów webowych. W przyszłości zestaw testów może stać się częścią ciągłego nadzoru nad dostępnością i poprawnością działania witryny, co przełożyłoby się na szybsze wykrywanie błędów i poprawę jakości usług oferowanych użytkownikom strony.