Automatyzacja testów

teoria

Testowanie dzielimy na manualne i automatyczne

Dlaczego warto stosować automatyzację testów?

Testowanie dzielimy na manualne i automatyczne

Dlaczego warto stosować automatyzację testów?

- Testowanie ręczne jest czasochłonne, pracochłonne, a w przypadku skomplikowanego oprogramowania może również stać się kosztowne, gdy używamy go wyłącznie.

Testowanie dzielimy na manualne i automatyczne

Dlaczego warto stosować automatyzację testów?

- Testowanie ręczne jest czasochłonne, pracochłonne, a w przypadku skomplikowanego oprogramowania może również stać się kosztowne, gdy używamy go wyłącznie.

- Zautomatyzowane testowanie usprawnia procesy, skraca czas potrzebny na testowanie i eliminuje nieefektywności, takie jak spędzanie przez programistów żmudnych godzin na testowaniu funkcjonalności oprogramowania.

Agenda

1. Czym jest automatyzacja testów?
2. Piramida automatyzacji testów.
3. Kryteria udanego procesu testów automatycznych.
4. Rodzaje testów automatycznych.
5. Jakie procesy i rodzaje testów należy automatyzować?
6. Osoby zaangażowane w proces automatyzacji?
7. Proces automatyzacji testów i jego wdrożenie.
8. Błędne przekonania o automatycznych testach.
9. Uwagi odnośnie procesu automatyzacji.
10. Narzędzia automatyzacji testów.

Czym jest automatyzacja testów?

Testy automatyczne to proces używania narzędzi programistycznych, które uruchamiają nowo opracowane oprogramowanie lub aktualizacje przez serię testów w celu zidentyfikowania potencjalnych błędów w kodowaniu, wąskich gardeł i innych przeszkód w wydajności.

Czym jest automatyzacja testów?

Testy automatyczne to proces używania narzędzi programistycznych, które uruchamiają nowo opracowane oprogramowanie lub aktualizacje przez serię testów w celu zidentyfikowania potencjalnych błędów w kodowaniu, wąskich gardeł i innych przeszkód w wydajności.

Podczas testowania nowego oprogramowania lub jego aktualizacji, **testy manualne mogą być drogie i żmudne**. Natomiast testy automatyczne są mniej kosztowne i zajmują mniej czasu(są szybsze).

Czym jest automatyzacja testów?

Zautomatyzowane testy mogą **pomóc w szybszym wykrywaniu awarii** przy mniejszej szansie na błąd człowieka. Plus, są one **łatwiejsze do uruchomienia wiele razy** dla każdej zmiany lub aż do uzyskania pożądanych wyników.

Czym jest automatyzacja testów?

Zautomatyzowane testy mogą **pomóc w szybszym wykrywaniu awarii** przy mniejszej szansie na błąd człowieka. Plus, są one **łatwiejsze do uruchomienia wiele razy** dla każdej zmiany lub aż do uzyskania pożądanych wyników.

Automatyzacja **przyspiesza również proces wprowadzania oprogramowania na rynek**. Automatyzacja umożliwia dokładne testowanie w określonych obszarach, dzięki czemu umożliwia zajęcie się typowymi problemami przed przejściem do kolejnej fazy testów.

Piramida automatyzacji testów

Piramida automatyzacji testów pomaga zrozumieć, jak często powinniśmy wykonywać każdy typ testu. Najbardziej sprawdza się w metodyce agile.

Piramida automatyzacji testów

Piramida automatyzacji testów pomaga zrozumieć, jak często powinniśmy wykonywać każdy typ testu. Najbardziej sprawdza się w metodyce agile.

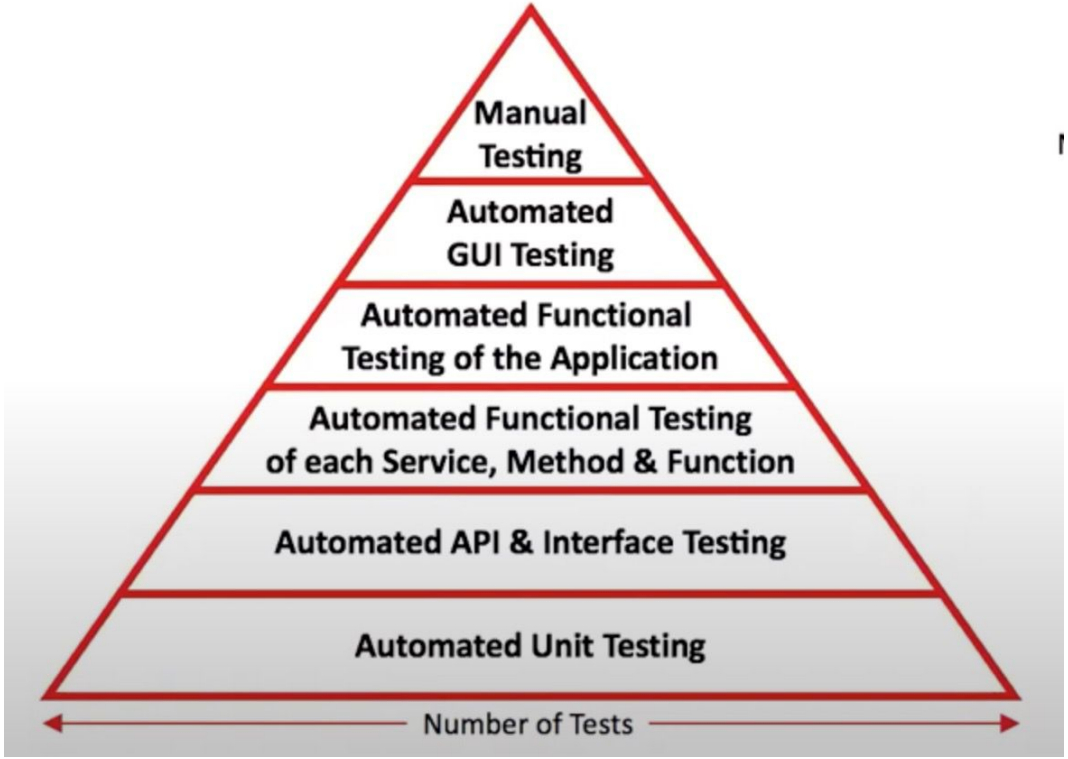
Piramida automatyzacji testów dzieli testowanie na cztery poziomy. Dolna warstwa reprezentuje testy, które powinno się wykonywać najczęściej. Poziomy stają się tym mniejsze, im bardziej zbliżają się do szczytu piramidy, reprezentując testy, które powinno się wykonywać rzadziej.

Piramida automatyzacji testów

Oto rodzaje testów, które piramida automatyzacji testów uwzględnia. Na które testy powinniśmy poświęcić najwięcej zasobów, od najbardziej do najmniej:

- Testy jednostkowe
- Testy integracyjne
- Testy API
- Testy UI

Piramida automatyzacji testów



Testy jednostkowe

Testy jednostkowe polegają na rozbiciu oprogramowania deweloperskiego na strawne jednostki w celu zidentyfikowania wszelkich błędów lub problemów z wydajnością.

Testy jednostkowe

Testy jednostkowe polegają na rozbiciu oprogramowania deweloperskiego na strawne jednostki w celu zidentyfikowania wszelkich błędów lub problemów z wydajnością.

Testy jednostkowe pomagają w identyfikacji błędów zanim proces tworzenia oprogramowania posunie się za daleko. Ten rodzaj testowania ma miejsce podczas najwcześniejszych etapów rozwoju oprogramowania, izolując i rozwiązując problemy przed przejściem do testów (Zwykle wykonywane przez programistów).

Testy jednostkowe

Testy jednostkowe polegają na rozbiciu oprogramowania deweloperskiego na strawne jednostki w celu zidentyfikowania wszelkich błędów lub problemów z wydajnością.

Testy jednostkowe pomagają w identyfikacji błędów zanim proces tworzenia oprogramowania posunie się za daleko. Ten rodzaj testowania ma miejsce podczas najwcześniejszych etapów rozwoju oprogramowania, izolując i rozwiązując problemy przed przejściem do testów (Zwykle wykonywane przez programistów).

Testy jednostkowe to rodzaj testów, które powinny być **wykonywane najczęściej**, ponieważ zapewniają one, że wszystkie najmniejsze komponenty oprogramowania działają poprawnie przed zintegrowaniem ich z całością.

Testy integracyjne

Po sprawdzeniu, czy każdy pojedynczy element oprogramowania działa poprawnie, nadszedł czas na ich połączenie, aby określić, czy wszystkie działają razem.

Testy integracyjne

Po sprawdzeniu, czy każdy pojedynczy element oprogramowania działa poprawnie, nadszedł czas na ich połączenie, aby określić, czy wszystkie działają razem.

Testy integracyjne walidują interakcje pomiędzy komponentami, także w obrębie tego samego programu. Istotne jest, aby wszystkie zintegrowane komponenty poprawnie współdziałały z oprogramowaniem lub z usługami zewnętrznymi, takimi jak usługi internetowe. Ponieważ podczas testów jednostkowych znajdujemy większość błędów w kodzie, testów integracyjnych powinno być mniej niż jednostkowych.

Testy API

Testowanie interfejsu programu aplikacyjnego (API) sprawdza, czy **dwa odrębne komponenty** oprogramowania **mogą się ze sobą komunikować** w różnych okolicznościach

Testy API

Testowanie interfejsu programu aplikacyjnego (API) sprawdza, czy **dwa odrębne komponenty** oprogramowania **mogą się ze sobą komunikować** w różnych okolicznościach

Rodzaje testów obejmują:

- Badania walidacyjne
- Testy funkcjonalne
- Testy bezpieczeństwa
- Testy obciążeniowe

Testy UI

Testowanie interfejsu użytkownika (UI) (znane również jako testowanie GUI) zapewnia, że **oprogramowanie działa z różnymi interfejsami użytkownika**, takimi jak systemy operacyjne, przeglądarki i inne miejsca, w których użytkownicy końcowi wchodzą z nim w interakcje.

Testy UI oceniają takie cechy jak funkcjonalność, projekt wizualny, wydajność i użyteczność.Testy automatyzacji UI **eliminują konieczność zakupu wielu urządzeń do testowania**. Automatyzacja testów UI bierze pod uwagę doświadczenie użytkownika końcowego i pomaga kształtować oprogramowanie tak, aby pasowało do tej interakcji.

Testy UI

Ramy automatyzacji testów UI powinny zawierać scenariusze testowe związane z wąskimi gardłami systemu i procesu. Ponieważ wszystkie poprzednie kroki testowania powinny zidentyfikować i naprawić większość problemów, jakie może mieć oprogramowanie, **testowanie UI powinno być najmniej czasochłonnym testem.** Narzędzia automatyzacji UI pozwalają zaoszczędzić jeszcze więcej czasu.

Kryteria udanego procesu testów automatycznych.

1. Posiadanie dedykowanego zespołu
2. Posiadanie odpowiednich narzędzi
3. Posiadanie dedykowanego budżetu
4. Wdrażanie silnych ram testowania

Kryteria udanego procesu testów automatycznych.

1. Posiadanie dedykowanego zespołu

Posiadanie dedykowanego zespołu do testowania oprogramowania jest niezbędne. Deweloperzy, testerzy i zespół zapewnienia jakości mogą być zaangażowani w różne części procesu testowania, aby zapewnić, że nic nie zostanie pominięte na każdym poziomie testowania.

Kryteria udanego procesu testów automatycznych.

2. Posiadanie odpowiednich narzędzi

Wybór odpowiednich narzędzi do automatyzacji testów jest kluczowy. Narzędzia do testów automatycznych działają najlepiej, gdy:

- Są łatwe w użyciu
- Działają na różnych systemach operacyjnych/przeglądarkach
- Wspierają język skryptowy
- Można wielokrotnie ich używać dla wielu testów
- Wspierają wykorzystanie dużych zbiorów danych

Kryteria udanego procesu testów automatycznych.

3. Posiadanie dedykowanego budżetu

Jeśli już inwestujesz w rozwój oprogramowania, posiadanie dedykowanego budżetu na oprogramowanie do automatyzacji testów, rozwój i szkolenia może zaoszczędzić pieniądze w dłuższej perspektywie. Firma spędzi mniej czasu na ręcznym testowaniu i szybciej uruchomi oprogramowanie.

Kryteria udanego procesu testów automatycznych.

4. Wdrażanie silnych ram testowania

Co to jest **framework testowy**?
Ramy testowania, które zawierają wytyczne, najlepsze praktyki, narzędzia i zasady testowania, mogą zaoszczędzić czas i wysiłek. Dobry framework do automatyzacji stron internetowych powinien integrować różne funkcje jak:

- Biblioteki
- Dane z badań
- Moduły wielokrotnego użytku
- Integracja z narzędziami innych firm

Rodzaje testów automatycznych

1. Testy funkcjonalne

Rodzaje testów automatycznych

1. Testy funkcjonalne
2. Testy niefunkcjonalne

Rodzaje testów automatycznych

1. Testy funkcjonalne
2. Testy niefunkcjonalne
3. Analiza kodu

Rodzaje testów automatycznych

1. Testy funkcjonalne
2. Testy niefunkcjonalne
3. Analiza kodu
4. Testy jednostkowe

Rodzaje testów automatycznych

1. Testy funkcjonalne
2. Testy niefunkcjonalne
3. Analiza kodu
4. Testy jednostkowe
5. Testy integracyjne

Rodzaje testów automatycznych

1. Testy funkcjonalne
2. Testy niefunkcjonalne
3. Analiza kodu
4. Testy jednostkowe
5. Testy integracyjne
6. Testy dymu(smoke)

Rodzaje testów automatycznych

1. Testy funkcjonalne
2. Testy niefunkcjonalne
3. Analiza kodu
4. Testy jednostkowe
5. Testy integracyjne
6. Testy dymu(smoke)
7. Testy wydajności

Rodzaje testów automatycznych

1. Testy funkcjonalne
2. Testy niefunkcjonalne
3. Analiza kodu
4. Testy jednostkowe
5. Testy integracyjne
6. Testy dymu(smoke)
7. Testy wydajności
8. Testy regresji

Rodzaje testów automatycznych

1. Testy funkcjonalne
2. Testy niefunkcjonalne
3. Analiza kodu
4. Testy jednostkowe
5. Testy integracyjne
6. Testy dymu(smoke)
7. Testy wydajności
8. Testy regresji
9. Testy API

Jakie procesy i rodzaje testów należy automatyzować?

Celem każdego scenariusza automatyzacji jest **przyspieszenie czasu testowania** i **zmniejszenie kosztów**, więc automatyzacja oparta na danych jest niezbędna. Przykłady procesów, w których automatyzacja może pomóc:

1. **Badanie powtarzalne** - każdy test obejmujący sekwencyjne i regularne powtarzanie korzysta z testów automatycznych po prostu dlatego, że może działać szybciej niż testy ręczne.

Jakie procesy i rodzaje testów należy automatyzować?

Celem każdego scenariusza automatyzacji jest **przyspieszenie czasu testowania** i **zmniejszenie kosztów**, więc automatyzacja oparta na danych jest niezbędna. Przykłady procesów, w których automatyzacja może pomóc:

1. **Badanie powtarzalne** - każdy test obejmujący sekwencyjne i regularne powtarzanie korzysta z testów automatycznych po prostu dlatego, że może działać szybciej niż testy ręczne.
2. **Badania wysokiego ryzyka** - Automatyzacja pozwala wyizolować potencjalne punkty awarii i zająć się nimi zanim zaczniesz zmieniać jakikolwiek kod.

Jakie procesy i rodzaje testów należy automatyzować?

Celem każdego scenariusza automatyzacji jest **przyspieszenie czasu testowania** i **zmniejszenie kosztów**, więc automatyzacja oparta na danych jest niezbędna. Przykłady procesów, w których automatyzacja może pomóc:

3. **Czasochłonne testy** - Testowanie ręczne trwa dłużej i jest podatne na błędy. Automatyzacja testów zmniejsza siłę roboczą potrzebną do ich przeprowadzenia oraz szanse na niewykrycie istotnych błędów.

Jakie procesy i rodzaje testów należy automatyzować?

Celem każdego scenariusza automatyzacji jest **przyspieszenie czasu testowania** i **zmniejszenie kosztów**, więc automatyzacja oparta na danych jest niezbędna. Przykłady procesów, w których automatyzacja może pomóc:

3. **Czasochłonne testy** - Testowanie ręczne trwa dłużej i jest podatne na błędy. Automatyzacja testów zmniejsza siłę roboczą potrzebną do ich przeprowadzenia oraz szanse na niewykrycie istotnych błędów.

4. **Aplikacje o wielu twarzach** - Kiedy oprogramowanie ma wiele interakcji z innymi aplikacjami lub oprogramowaniem, istnieje więcej możliwości konfliktów. Automatyzacja zapewnia ich wychwycenie

Osoby zaangażowane w proces automatyzacji.

Testy automatyzacyjne **rzadko są zadaniem dla jednego pracownika**. Oto kilka przykładów osób, które powinny być zaangażowane w każdy proces testowania automatyzacji:

1. **Programiści(testy jednostkowe)**

Osoby zaangażowane w proces automatyzacji.

Testy automatyzacyjne **rzadko są zadaniem dla jednego pracownika**. Oto kilka przykładów osób, które powinny być zaangażowane w każdy proces testowania automatyzacji:

1. **Programiści(testy jednostkowe)**
2. **Testerzy automatyczni**

Osoby zaangażowane w proces automatyzacji.

Testy automatyzacyjne **rzadko są zadaniem dla jednego pracownika**. Oto kilka przykładów osób, które powinny być zaangażowane w każdy proces testowania automatyzacji:

1. **Programiści(testy jednostkowe)**
2. **Testerzy automatyczni**
3. **Zespół zapewnienia jakości - QA (używanie frameworka)**

Osoby zaangażowane w proces automatyzacji.

Testy automatyzacyjne **rzadko są zadaniem dla jednego pracownika**. Oto kilka przykładów osób, które powinny być zaangażowane w każdy proces testowania automatyzacji:

1. **Programiści(testy jednostkowe)**
2. **Testerzy automatyczni**
3. **Zespół zapewnienia jakości - QA (używanie frameworka)**
4. **Interesariusze (beta testy)**

Proces automatyzacji i jego wdrożenie.

1. Zdefiniowanie celów testu
2. Ustalenie priorytetów testów
3. Możliwość zastosowania w różnych platformach
4. Łatwość testowania
5. Usprawniona komunikacja
6. Zapewnienie jakości

Błędne przekonania o automatycznych testach.

1. Automatyzacja zastępuje testowanie ręczne

Błędne przekonania o automatycznych testach.

1. Automatyzacja zastępuje testowanie ręczne

Najlepsza analogia dotycząca automatyzacji zastępującej zadania ręczne pochodzi z fałszywego pomysłu, że zmywarki mogą wyeliminować całe ręczne zmywanie naczyń. Jednak zawsze znajdą się naczynia, które wymagają ręcznego mycia.

Ta sama koncepcja dotyczy testów automatyzacji w oprogramowaniu. **Automatyzacja przyspiesza typowe scenariusze testowe i zmniejsza obciążenie testowe.** Nie eliminuje to jednak potrzeby istnienia testerów manualnych, szczególnie na etapie rozwiązywania problemów, gdzie deweloper jest w stanie lepiej zidentyfikować źródła błędów.

Błędne przekonania o automatycznych testach.

2. Automatyzacja eliminuje błędy

Błędne przekonania o automatycznych testach.

2. Automatyzacja eliminuje błędy

Nawet najlepsze testy nie wyeliminują błędów czy awarii systemu. Niektóre wady kodu są nieodłącznym elementem procesu. Inne błędy w kodowaniu aktywują się tylko w bardzo specyficznych scenariuszach.

Używanie testów automatycznych jest jak sygnalizacja świetlna, która czyni skrzyżowania znacznie bezpieczniejszymi, ale nie eliminuje wypadków, wąskich gardeł i korków.

Błędne przekonania o automatycznych testach.

3. Automatyzacja wymaga doświadczenie w tworzeniu

Błędne przekonania o automatycznych testach.

3. Automatyzacja wymaga doświadczenie w tworzeniu

Podczas gdy niektóre testy automatyczne są bardziej skomplikowane i wymagają doświadczonego programisty, wiele pakietów testowych pozwala początkującym pisać proste testy automatyczne.

Uwagi odnośnie procesu automatyzacji

1. Testowanie nie jest procesem dla wszystkich
2. Pośpiech sprzyja błędom
3. Nawet testy mają błędy

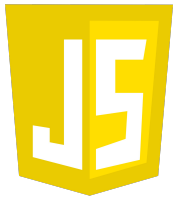
Uwagi odnośnie procesu automatyzacji

1. Testowanie nie jest procesem dla wszystkich
2. Pośpiech sprzyja błędom
3. Nawet testy mają błędy
4. Automatyczne testy są podatne na regułę pestycydów
5. Automatyczne testy regresji przyśpieszają proces wytwarzania oprogramowania

Narzędzia automatyzacji testów

Języki programowania

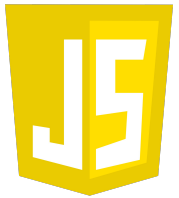
1. Python
2. Java
3. Javascript



Narzędzia automatyzacji testów

Języki programowania

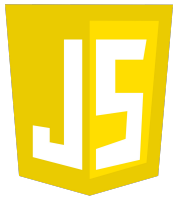
1. Python - Pytest, Selenium, Robot Framework
2. Java
3. Javascript



Narzędzia automatyzacji testów

Języki programowania

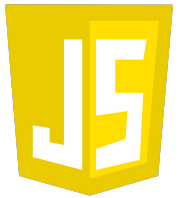
1. Python - Pytest, Selenium, Robot Framework
2. Java - Selenium, Cucumber, Junit
3. Javascript



Narzędzia automatyzacji testów

Języki programowania

1. Python - Pytest, Selenium, Robot Framework
2. Java - Selenium, Cucumber, Junit
3. Javascript - Cypress, Playwright, Postman



test_todo_api.py

02_pytest_REST_API > test_todo_api.py > ...

```python
import requests
import uuid

ENDPOINT = "https://todo.pixegami.io/"


def test_can_call_endpoint():
    response = requests.get(ENDPOINT)
    assert response.status_code == 200


def test_can_create_task():
    payload = new_task_payload()
    create_task_response = create_task(payload)
    assert create_task_response.status_code == 200
    data = create_task_response.json()
    task_id = data["task"]["task_id"]

    get_task_response = get_task(task_id)
    assert get_task_response.status_code == 200
    get_task_data = get_task_response.json()
    assert get_task_data["content"] == payload["content"]
    assert get_task_data["user_id"] == payload["user_id"]
```

Testy automatyczne w pytest

pytest

Testy automatyczne w Selenium

02_locating_elements.py

TWT_tutorial > 02_locating_elements.py > ...

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

# search order: id -> class -> name
# Task: search "test" in search bar and print summary of all results from
# website https://techwithtim.net.

driver = webdriver.Chrome()
driver.get("https://techwithtim.net")
print(driver.title)

search = driver.find_element(By.NAME, "s")
search.send_keys("test")
search.send_keys(Keys.RETURN)


try:
    main = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.ID, "main"))
    )
    articles = main.find_elements(By.TAG_NAME, "article")
    for article in articles:
        header = article.find_element(By.CLASS_NAME, "entry-summary")
        print(header.text)
finally:
    driver.quit()

```

Testowanie automatyczne w praktyce

Testowanie poczty elektronicznej

Przypadki testowe związane z projektem

1. Zweryfikuj, czy pole emailowe jest obecne na stronie.
2. Sprawdź, czy tekst etykiety jest wyświetlany wraz z polem mailowym.
3. Zweryfikuj, czy tekst etykiety dla adresu email jest wyrównany z polem mailowym.
4. Sprawdź, czy w polu mailowym jest dodany tekst zastępczy (placeholder).

Testowanie automatyczne w praktyce

Testowanie poczty elektronicznej

Przypadki testowe funkcjonalne

1. Zweryfikuj dostęp do pola adresu email po kliknięciu w to pole.
2. Sprawdź, czy użytkownicy mogą wpisywać adresy email w polu emailowym.
3. Zweryfikuj, czy użytkownik może wkleić adres email za pomocą Ctrl + V.
4. Sprawdź, czy użytkownik może wkleić adres email za pomocą prawego przycisku myszy, klikając w pole emailowe i wybierając opcję "Wklej".
5. Zweryfikuj, czy zaimplementowano walidację pola emailowego.
6. Sprawdź, czy w przypadku podania nieprawidłowego adresu email, wyświetlany jest komunikat o błędzie.

Testowanie automatyczne w praktyce

Testowanie poczty elektronicznej

Przypadki testowe pozytywne

1. Zweryfikuj poprawność pola emailowego, wpisując poprawny adres email. (codebrainers@gmail.com)
2. Sprawdź, czy adres email musi zawierać znak @.
3. Zweryfikuj, czy pole emailowe akceptuje adres email zawierający znak plus +.
4. Sprawdź, czy pole emailowe waliduje obecność domeny w adresie email. (codebrainers@gmail.com) Upewnij się, czy adres email zawiera kropkę.