ZÁRÓDOLGOZAT TESZTELÉS

Készítették:

Kovács Dániel

Braczkó Tamás

Novák Dominik Viktor

Konzulens:

Farkas Zoltán

Miskolc

Tesztelési Dokumentáció

A webshop tesztelése kiemelkedő fontosságú, mivel az online vásárlások általában összetettek, és a felhasználói élmény kritikus szerepet tölt be. Ez a dokumentum összegzi az alkalmazás működésének tesztelésére alkalmazott módszereket.

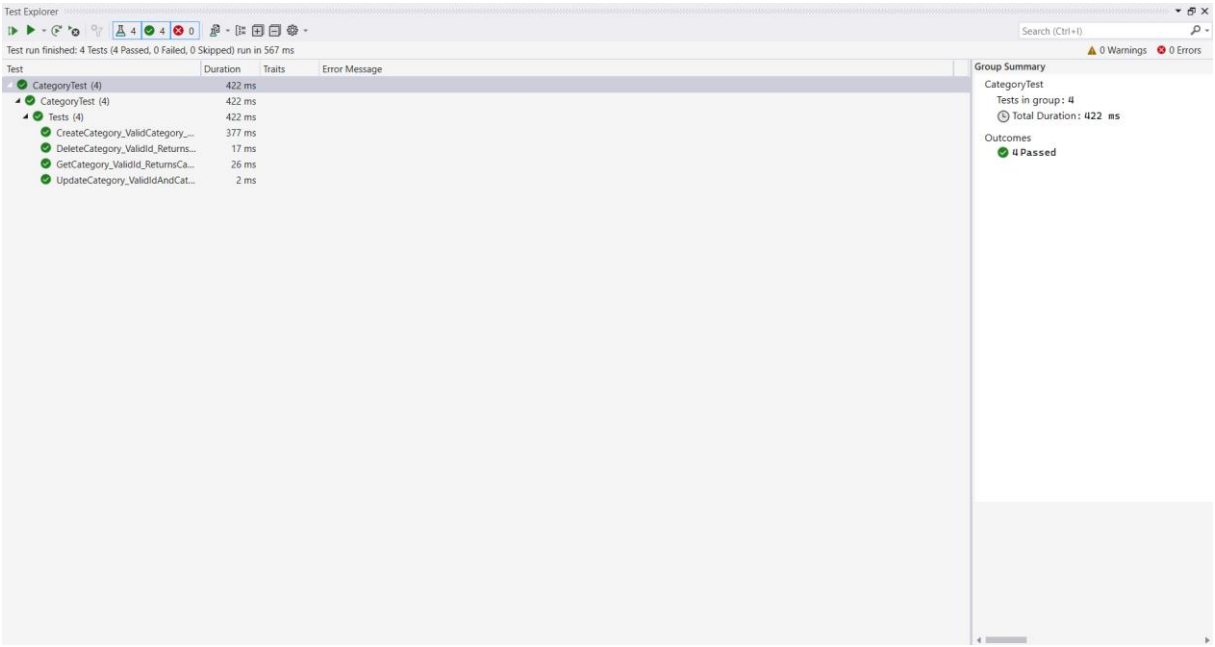
A tesztelési folyamatok célja a webshop stabilitásának, funkcionalitásának és felhasználói élményének biztosítása, valamint az esetleges hibák és problémák azonosítása.

- Unit tesztek a Backend funkcionalitásokra:
    - A backend rész egységtesztelése során ellenőrizzük a webshop logikáját és adatfeldolgozási folyamatait. Ezek a tesztek biztosítják, hogy a webshop minden része megfelelően működjön, és az elvárt eredményeket produkálja.
- Teljesítményelemzés a Frontend részről:
    - A frontend teljesítményének elemzése segít azonosítani az oldalak betöltési sebességével és a felhasználói interakciókkal kapcsolatos problémákat. A Lighthouse Chrome bővítmény használatával mérjük és értékeljük a frontend teljesítményét, hogy a webshop gyorsan és zökkenőmentesen működjön.
- Selenium automatizált tesztek a regisztráció és belépés funkciókra:
    - A Selenium segítségével automatizált teszteket futtatunk a webshop regisztrációs és belépési funkcióira. Ezek a tesztek ellenőrzik, hogy a regisztráció és belépés folyamatai megfelelően működnek-e, és az elvárt eredményeket produkálják-e.

- Főoldal Teljesítményének Értékelése Lighthouse Segítségével.
- Felhasználói Interakciók Tesztelése CommentForm Komponenssel Jest Használatával.

Unit tesztek a Backenden

4 teszt volt értékélve, mind a 4 sikeresen lefutott



GetCategory_ValidId_ReturnsCategory

```
[Test]
0 references
public async Task GetCategory_ValidId_ReturnsCategory()
{
    // Arrange
    var validId = 900;
    var expectedCategory = new Category { Id = validId, Name = "Test Category" };

    _context.Categories.Add(expectedCategory);
    await _context.SaveChangesAsync();

    // Act
    var result = await _controller.GetCategory(validId);

    // Assert
    Assert.IsNotNull(result.Value);
    Assert.AreEqual(expectedCategory, result.Value);
}
```

A teszt célja annak ellenőrzése, hogy a GetCategory metódus helyesen visszaadja-e a megfelelő kategóriát a megadott érvényes azonosító alapján. A teszt először létrehoz egy érvényes azonosítót és egy várt kategóriát. Ezután hozzáadjuk ezt a kategóriát az adatbázishoz és mentjük a változtatásokat. Ezután meghívjuk a GetCategory metódust a létrehozott érvényes azonosítóval. Végül ellenőrizzük, hogy a visszakapott eredmény nem null, és hogy megegyezik-e az elvárt kategóriával.

CreateCategory_ValidCategory_ReturnsCategory

```
[Test]
0 references
public async Task CreateCategory_ValidCategory_ReturnsCategory()
{
    // Arrange
    var newCategory = new Category { Name = "New Category" };

    // Act
    var actionResult = await _controller.CreateCategory(newCategory);

    // Assert
    Assert.IsNotNull(actionResult.Result);
    var createdAtActionResult = actionResult.Result as CreatedAtActionResult;
    Assert.IsNotNull(createdAtActionResult);
    var createdCategory = createdAtActionResult.Value as Category;
    Assert.IsNotNull(createdCategory);
    Assert.AreEqual(newCategory.Name, createdCategory.Name);
}
```

A teszt célja annak ellenőrzése, hogy a CreateCategory metódus helyesen létrehoz-e egy új kategóriát és visszaadja-e azt a megfelelő módon. A teszt először létrehoz egy új kategóriát a "New Category" névvel. Ezután meghívjuk a CreateCategory metódust a létrehozott új kategóriával. Végül ellenőrizzük, hogy a visszakapott akció eredménye nem null, és hogy a létrehozott eredmény megfelelő típusú és tartalmú-e. Ezen belül ellenőrizzük, hogy a visszakapott kategória neve megegyezik-e az új kategória nevével.

UpdateCategory_ValidIdAndCategory_ReturnsNoContent

```
[Test]
0 references
public async Task UpdateCategory_ValidIdAndCategory_ReturnsNoContent()
{
    // Arrange
    var validId = 900;
    var existingCategory = new Category { Id = validId, Name = "Existing Category
    _context.Categories.Add(existingCategory);
    await _context.SaveChangesAsync();

    var updatedCategory = await _context.Categories.FindAsync(validId);
    updatedCategory.Name = "Updated Category";

    // Act
    var result = await _controller.UpdateCategory(validId, updatedCategory);

    // Assert
    Assert.IsInstanceOf<NoContentResult>(result);
}
```

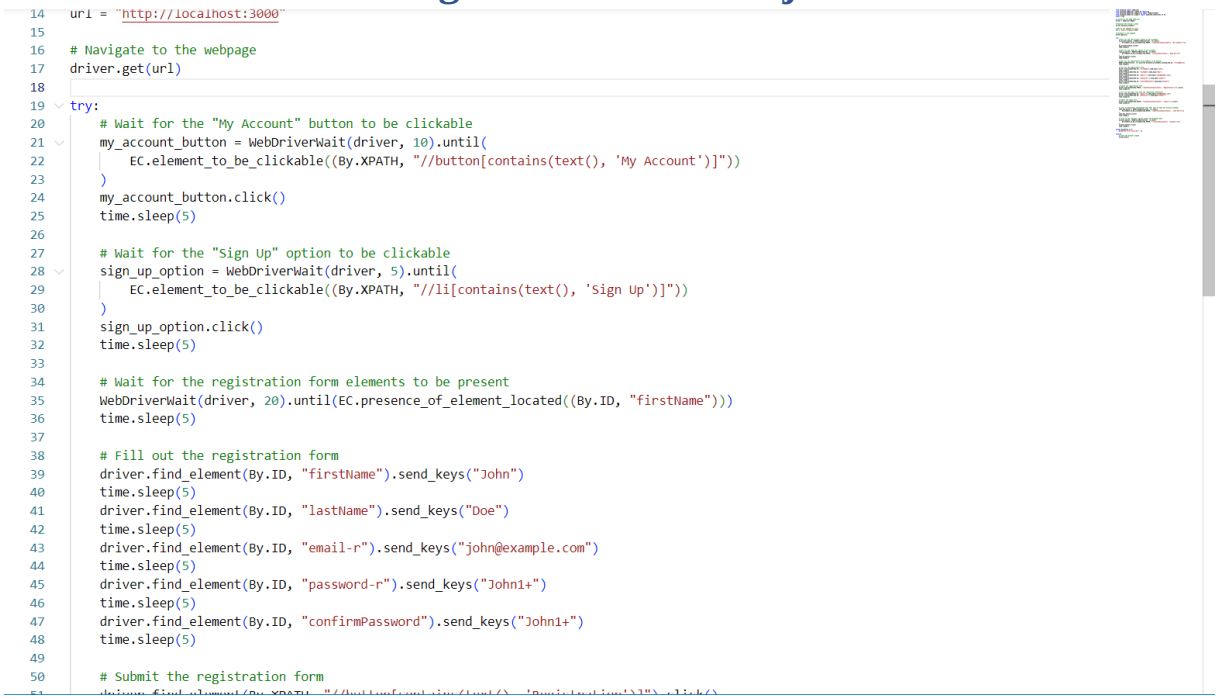
A teszt célja annak ellenőrzése, hogy az UpdateCategory metódus helyesen frissíti-e a meglévő kategóriát az adott érvényes azonosító alapján, és hogy a megfelelő visszajelzést adja-e vissza a frissítés sikerességére. A teszt először létrehoz egy meglévő kategóriát az adott érvényes azonosítóval és az "Existing Category" nevével. Ezután hozzáadjuk ezt a kategóriát az adatbázishoz és mentjük a változtatásokat. Ezután létrehozunk egy frissített kategóriát az eredeti azonosítóval és egy új névvel. Ezután meghívjuk az UpdateCategory metódust a frissített kategóriával és az azonosítóval. Végül ellenőrizzük, hogy a visszakapott eredmény típusa egy NoContentResult objektum-e, amely azt jelzi, hogy a frissítés sikeresen megtörtént, és nincs szükség további tartalomra a válaszban.

DeleteCategory_ValidId_ReturnsNoContent

```
[Test]
0 references
public async Task DeleteCategory_ValidId_ReturnsNoContent()
{
    // Arrange
    var validId = 900;
    var existingCategory = new Category { Id = validId, Name = "Existing Category" };

    _context.Categories.Add(existingCategory);
    await _context.SaveChangesAsync();

    // Act
    var result = await _controller.DeleteCategory(validId);

    // Assert
    Assert.IsInstanceOf<NoContentResult>(result);
}
```

A teszt célja annak ellenőrzése, hogy a DeleteCategory metódus helyesen törli-e a meglévő kategóriát az adott érvényes azonosító alapján, és hogy a megfelelő visszajelzést adja-e vissza a törlés sikerességére. A teszt először létrehoz egy meglévő kategóriát az adott érvényes azonosítóval és az "Existing Category" nevével. Ezután hozzáadjuk ezt a kategóriát az adatbázishoz és mentjük a változtatásokat. Ezután meghívjuk a DeleteCategory metódust az adott érvényes azonosítóval. Végül ellenőrizzük, hogy a visszakapott eredmény típusa egy NoContentResult objektum-e, amely azt jelzi, hogy a törlés sikeresen megtörtént, és nincs szükség további tartalomra a válaszban.

Selenium a Regisztrációhoz és Bejelentkezéshes

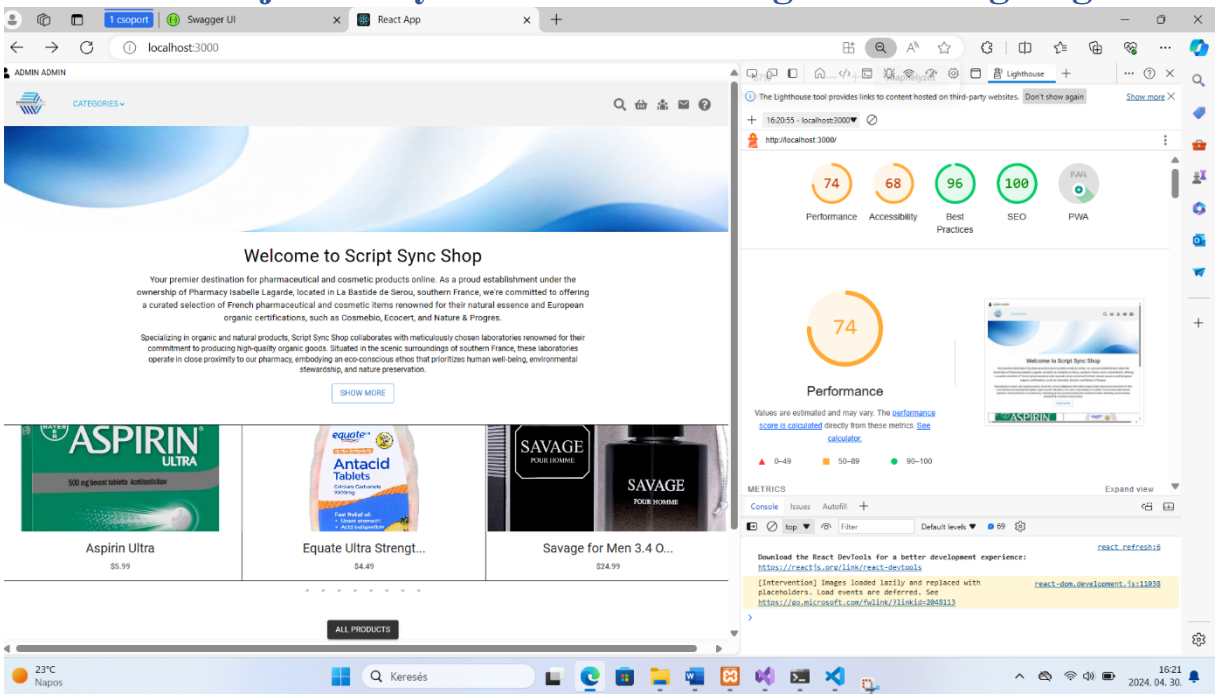


```
url = "http://localhost:3000"

# Navigate to the webpage
driver.get(url)

try:
    # Wait for the "My Account" button to be clickable
    my_account_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.XPATH, "//button[contains(text(), 'My Account')]"))
    )
    my_account_button.click()
    time.sleep(5)

    # Wait for the "Sign Up" option to be clickable
    sign_up_option = WebDriverWait(driver, 5).until(
        EC.element_to_be_clickable((By.XPATH, "//li[contains(text(), 'Sign Up')]"))
    )
    sign_up_option.click()
    time.sleep(5)

    # Wait for the registration form elements to be present
    WebDriverWait(driver, 20).until(EC.presence_of_element_located((By.ID, "firstName")))
    time.sleep(5)

    # Fill out the registration form
    driver.find_element(By.ID, "firstName").send_keys("John")
    time.sleep(5)
    driver.find_element(By.ID, "lastName").send_keys("Doe")
    time.sleep(5)
    driver.find_element(By.ID, "email-r").send_keys("john@example.com")
    time.sleep(5)
    driver.find_element(By.ID, "password-r").send_keys("John1+")
    time.sleep(5)
    driver.find_element(By.ID, "confirmPassword").send_keys("John1+")
    time.sleep(5)

    # Submit the registration form
```

Részlet a regisztrációból

Ez a Selenium Python szkript automatizálja a felhasználói felületen végzett műveleteket egy webalkalmazásban. A szkript feladata a felhasználó regisztrációs és bejelentkezési folyamatának szimulálása. Először inicializálja és maximalizálja az Edge böngészőt, majd elnavigál a megadott URL-re. A regisztrációs részben megvárja, hogy a "My Account" gomb kattintható legyen, majd a "Sign Up" opcióra kattintva kitölti és elküldi a regisztrációs űrlapot. Ezután a bejelentkezési folyamatban a szkript a regisztrált hitelesítési adatokkal kitölti és elküldi a bejelentkezési űrlapot. Végül a szkript a felhasználó fiókjához navigál, megnyitva az "Account" opciót. Ennek eredményeként a szkript segíti a fejlesztőket a regisztrációs és bejelentkezési folyamatok tesztelésében és ellenőrzésében, miközben ezeket a feladatokat automatizálja, lehetővé téve a gyors és hatékony tesztelést.

Főoldal Teljesítményének Értékelése Lighthouse Segítségével



A Lighthouse egy Google által fejlesztett automatizált eszköz, amely segít a webes alkalmazások teljesítményének, hozzáférhetőségének, legjobb gyakorlatainak és SEO-jának értékelésében. Segítségével gyorsan és hatékonyan lehet tesztelni és javítani a webhelyek teljesítményét és felhasználói élményét.

Felhasználói Interakciók Tesztelése CommentForm Komponenssel Jest Használatával

```
1  import { render, screen } from '@testing-library/react';
2  import userEvent from '@testing-library/user-event';
3  import CommentForm from './Comment';
4
5  describe('CommentForm', () => {
6    test('renders CommentForm component without crashing', () => {
7      render(<CommentForm productId="testProductId" />);
8    });
9
10   test('updates state on user input', async () => {
11     render(<CommentForm productId="testProductId" />);
12
13     const textarea = screen.getByRole('textbox');
14     await userEvent.type(textarea, 'Test comment');
15
16     expect(textarea.value).toBe('Test comment');
17
18     const rating = screen.getByTestId('rating');
19     const thirdStar = rating.children[2];
20     userEvent.click(thirdStar);
21
```

A CommentForm React komponens tesztjei a @testing-library/react és @testing-library/user-event könyvtárak segítségével vannak megvalósítva. Az első teszt célja, hogy ellenőrizze, hogy a CommentForm komponens rendeltetésszerűen megjelenik-e, és nem okoz-e hibát a renderelés során. A második teszt az állapot frissítését teszteli felhasználói interakciók segítségével, ellenőrizve, hogy a beviteli mezők és a csillagok kiválasztása helyesen működik-e. Ezek a tesztek segítenek biztosítani, hogy a CommentForm komponens megfelelően működjön, javítva az alkalmazás stabilitását és megbízhatóságát.