МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ  
  
  
  
  
  
  
  
  
  
Дисципліна «Автоматизоване тестування»

Лабораторна робота № 4

Виконала: студентка гр. 8.1213  
 Абдуллаєва Севда   
 Самандар Кизи

Перевірив: Лебідь Максим   
 Володимирович

Запоріжжя  
2023

**Як результат виконання лабораторної роботи №4 було завантажено лише звіт, так як розмір архівного файлу перевищує 5 МБ. Всі файли завантажені в репозиторій:** [**https://github.com/SevdaAbdullaeva/Selenium-tests\_lab4.git**](https://github.com/SevdaAbdullaeva/Selenium-tests_lab4.git)

**Завдання**

1. Реалізувати скрипт для автоматизації наступних дій засобами Selenium:

* Відкрити сторінку http://suninjuly.github.io/math.html .
* Прочитати значення змінної x.
* Обчислити математичну функцію від x.
* Ввести відповідь в текстове поле.
* Вибрати checkbox "I'm the robot".
* Вибрати radiobutton "Robots rule!".
* Натиснути кнопку Submit.

**Код програми:**

const { Builder, By, Key, until } = require('selenium-webdriver');

// Функція для виконання автоматизації

async function automateTest() {

  const driver = await new Builder().forBrowser('chrome').build();

  try {

    await driver.get('http://suninjuly.github.io/math.html');

    const xElement = await driver.findElement(By.id('input\_value'));

    const x = await xElement.getText();

    const result = Math.log(Math.abs(12 \* Math.sin(parseFloat(x))));

    const answerInput = await driver.findElement(By.id('answer'));

    await answerInput.sendKeys(result);

    const robotCheckbox = await driver.findElement(By.id('robotCheckbox'));

    await robotCheckbox.click();

    const robotRadio = await driver.findElement(By.id('robotsRule'));

    await robotRadio.click();

    const submitButton = await driver.findElement(By.css('.btn-default'));

    await submitButton.click();

  } finally {

    await driver.quit();

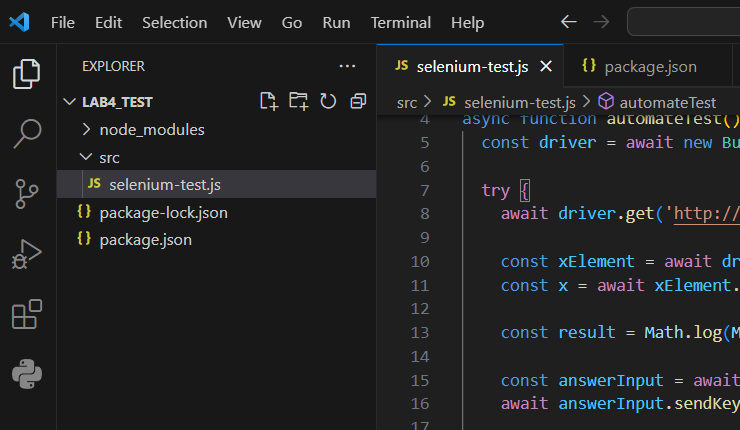
  }

}

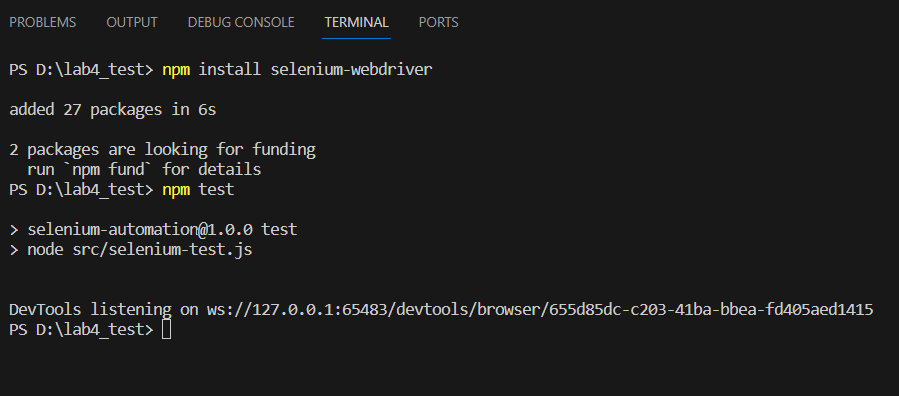
// Виклик функції для автоматизації

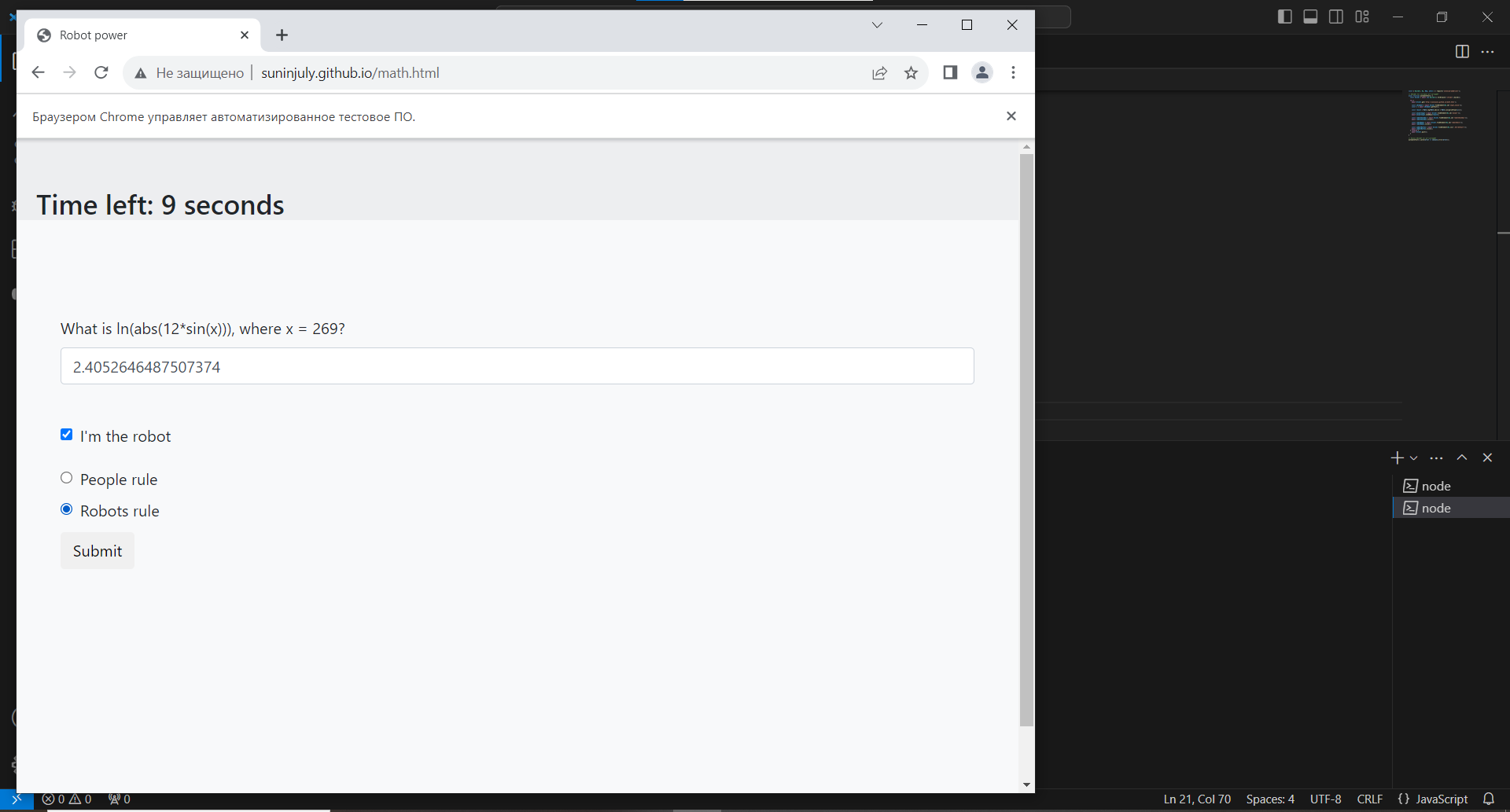
automateTest().catch(error => console.error(error));

**Структура файлів:**

****

**Результати роботи програми:**

****

****

1. Реалізувати скрипт тестування наступного функціонала ресурсу

<http://demo-store.seleniumacademy.com> :

* Перехід по всіх сторінках проєкту.
* Реєстрації нового користувача.
* Довільний функціонал за вибором студента.

**Код програми:**

// Перехід по всіх сторінках проєкту:

const { Builder, By } = require('selenium-webdriver');

async function navigateToAllPages() {

  const driver = await new Builder().forBrowser('chrome').build();

  try {

    await driver.get('http://demo-store.seleniumacademy.com');

    // Виконайте код для переходу на всі сторінки проекту, наприклад:

    await driver.findElement(By.linkText('Home')).click();

    await driver.findElement(By.linkText('About Us')).click();

    await driver.findElement(By.linkText('Contact Us')).click();

  } finally {

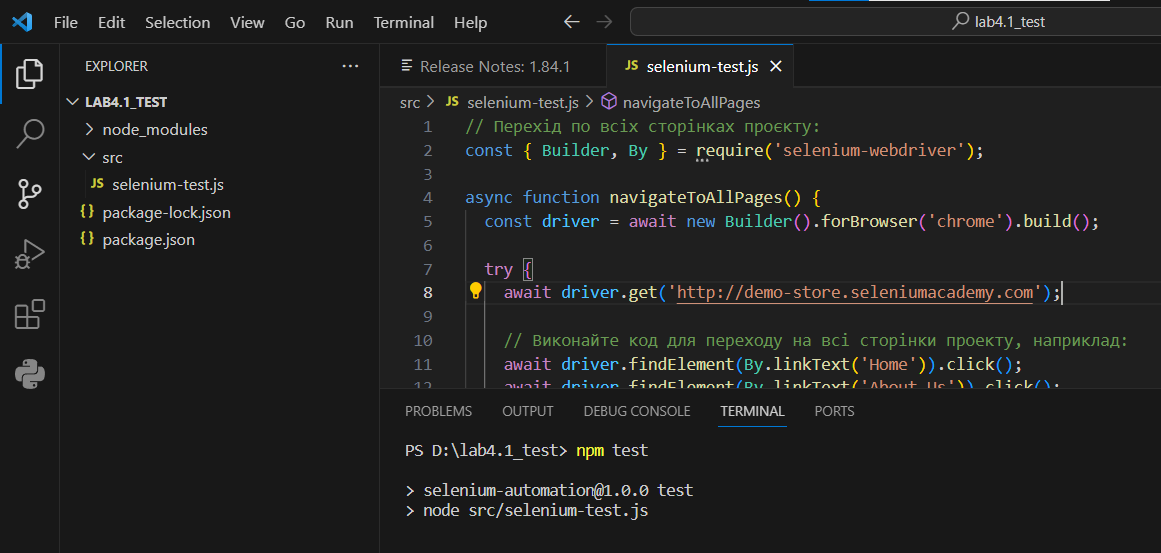
    await driver.quit();

  }

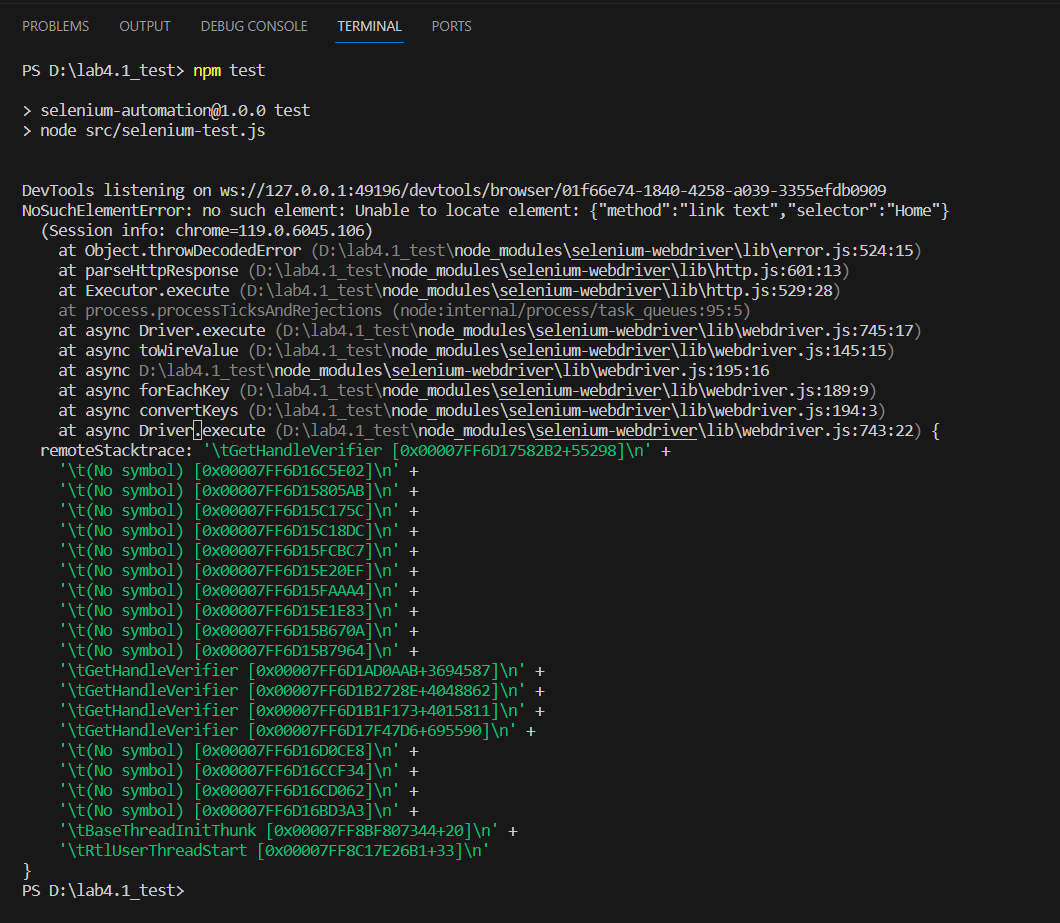
}

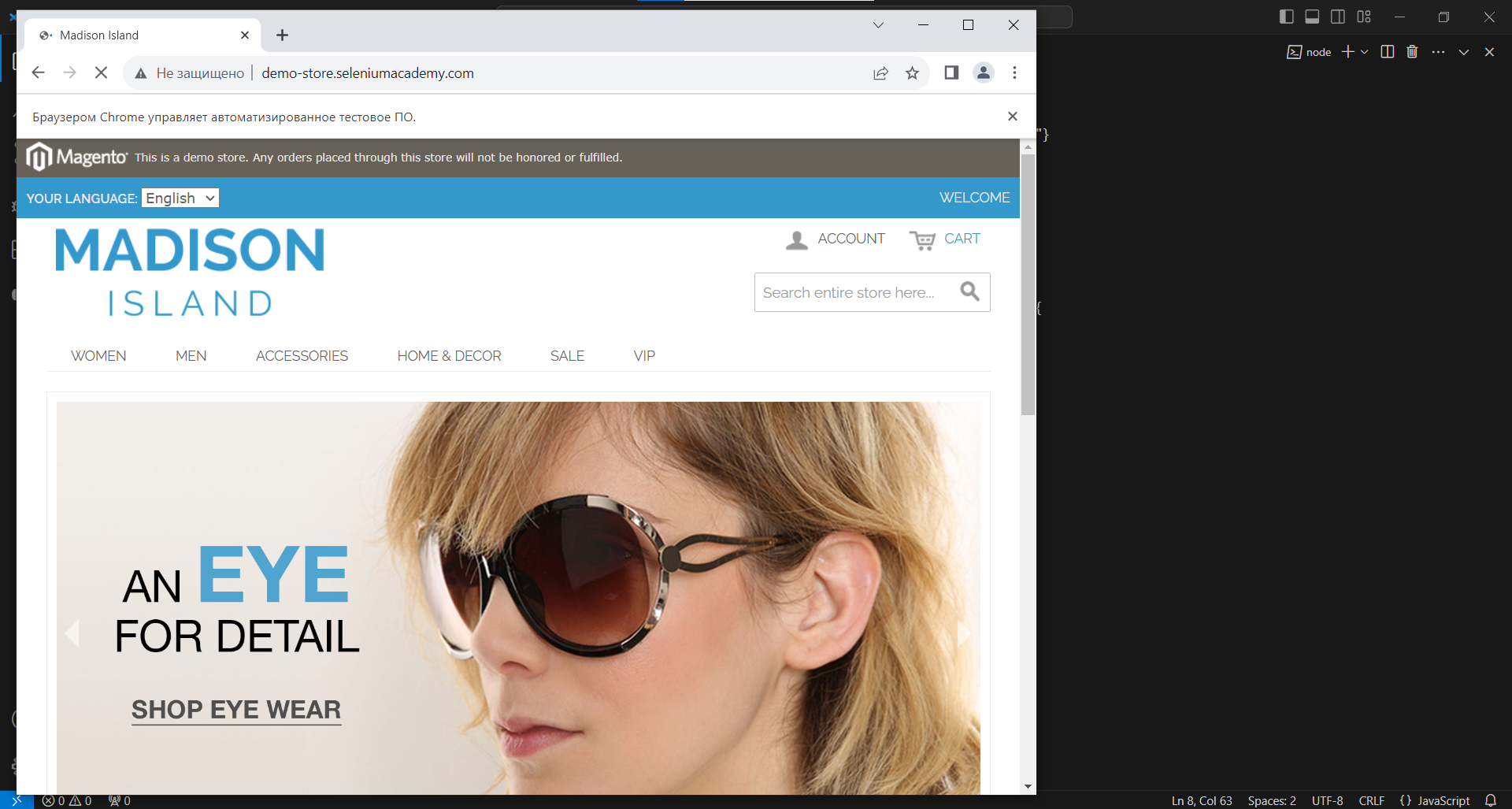
navigateToAllPages().catch(error => console.error(error));

**Структура файлів:**

****

**Результати роботи програми:**

****

****

**Код програми:**

// Реєстрація нового користувача:

const { Builder, By } = require('selenium-webdriver');

async function registerNewUser() {

  const driver = await new Builder().forBrowser('chrome').build();

  try {

    await driver.get('http://demo-store.seleniumacademy.com');

    // Знайдіть елементи та введіть дані для реєстрації, наприклад:

    await driver.findElement(By.linkText('Account')).click();

    await driver.findElement(By.linkText('Register')).click();

    await driver.findElement(By.id('firstname')).sendKeys('John');

    await driver.findElement(By.id('lastname')).sendKeys('Doe');

    await driver.findElement(By.id('email\_address')).sendKeys('john.doe@example.com');

    await driver.findElement(By.id('password')).sendKeys('password');

    await driver.findElement(By.id('confirmation')).sendKeys('password');

    await driver.findElement(By.css('.buttons-set .button')).click();

  } finally {

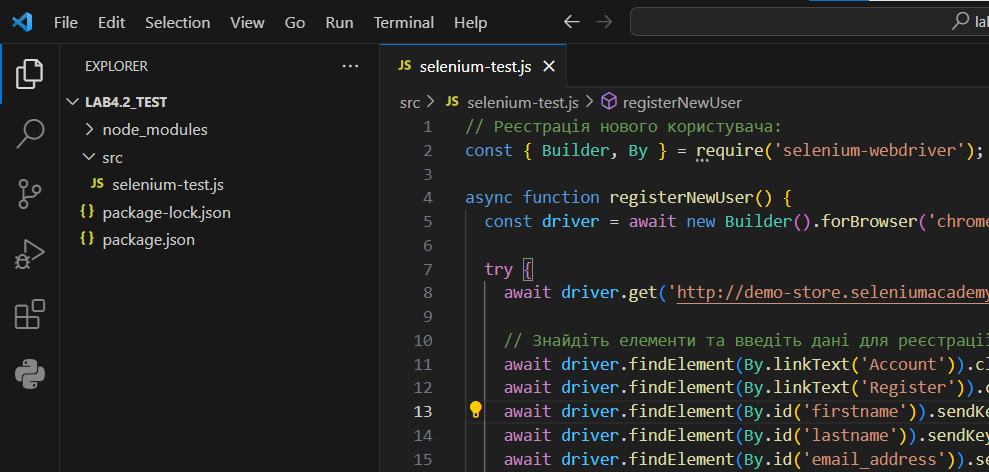
    await driver.quit();

  }

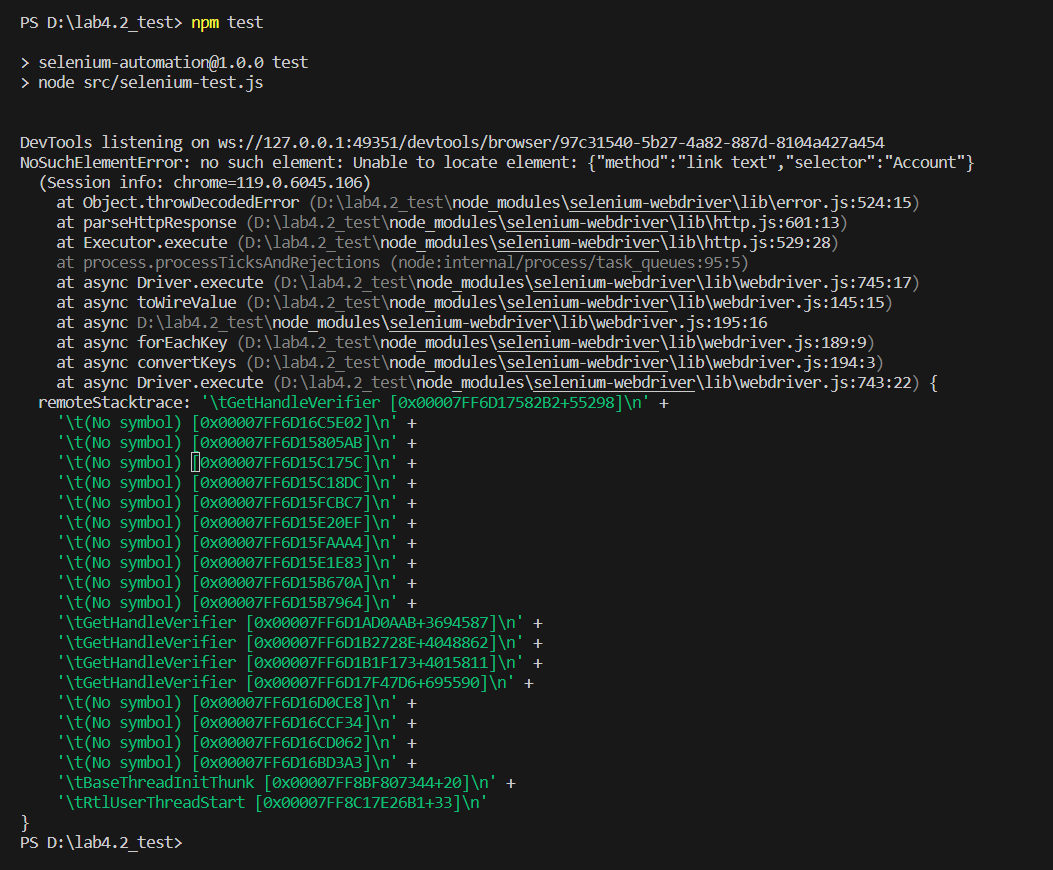
}

registerNewUser().catch(error => console.error(error));

**Структура файлів:**

****

**Результати роботи програми:**

****

**Код програми:**

// Довільний функціонал за вибором студента.

// Пошуку конкретного продукту на веб-сайті і додавання його до кошика:

const { Builder, By, Key } = require('selenium-webdriver');  // Додали Key

async function searchAndAddToCart() {

  const driver = await new Builder().forBrowser('chrome').build();

  try {

    await driver.get('http://demo-store.seleniumacademy.com');

    // Виконуємо пошук конкретного продукту (наприклад, "Sony Xperia") з використанням Key.RETURN

    const searchInput = await driver.findElement(By.id('search'));

    await searchInput.sendKeys('Sony Xperia', Key.RETURN);

    // Клікаємо на знайдений продукт

    await driver.findElement(By.linkText('Sony Xperia')).click();

    // Додаємо продукт до кошика

    await driver.findElement(By.css('.add-to-cart')).click();

    // Перевірка, що продукт доданий до кошика (може змінюватися відповідно до сайту)

    const successMessage = await driver.findElement(By.css('.success-msg'));

    if (await successMessage.isDisplayed()) {

      console.log('Продукт успішно доданий до кошика.');

    } else {

      console.log('Не вдалося додати продукт до кошика.');

    }

  } finally {

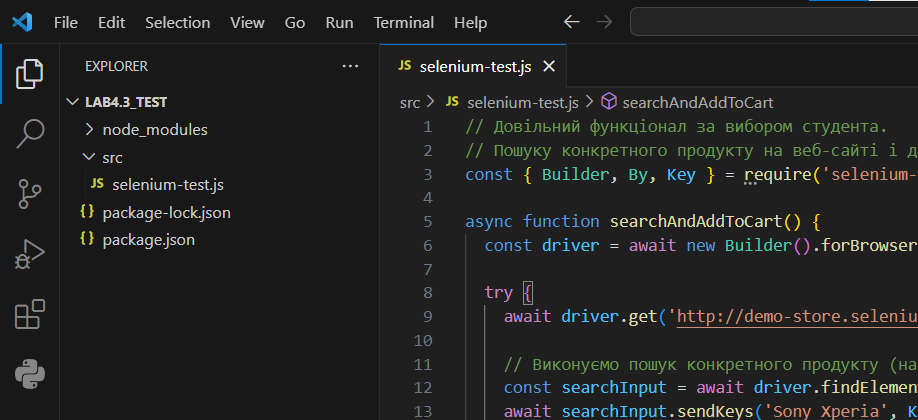
    await driver.quit();

  }

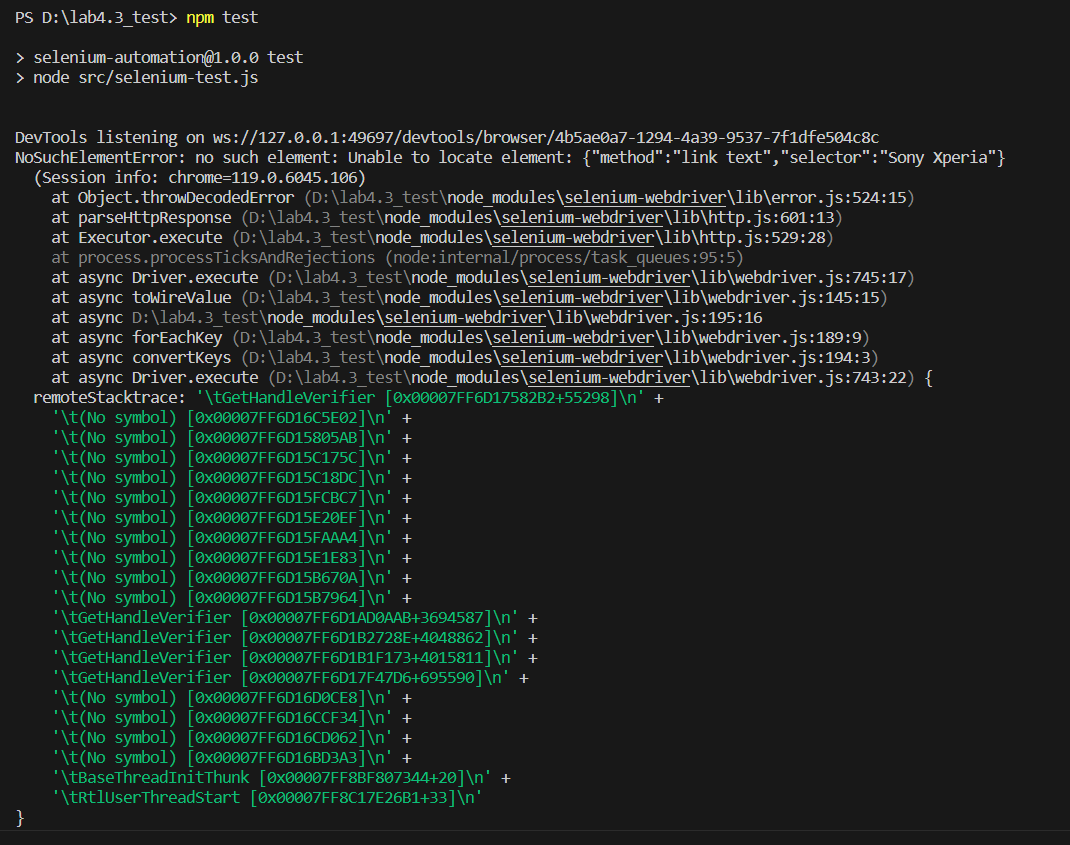
}

searchAndAddToCart().catch(error => console.error(error));

**Структура файлів:**

****

**Результати роботи програми:**

****

**Посилання на репозиторій:**

<https://github.com/SevdaAbdullaeva/Selenium-tests_lab4.git>

**Контрольні запитання**

1. **Наведіть види тестування.**

Тестування - це процес перевірки програмного продукту на відповідність вимогам і виявлення помилок. Існує багато різних видів тестування, які можуть бути застосовані в різних ситуаціях. Ось деякі з найпоширеніших видів тестування:

* Функціональне тестування (Functional Testing): Перевірка, чи виконує програма очікувані функції і вимоги. Включає тестування інтерфейсу, функціональність і взаємодію між компонентами програми.
* Негативне тестування (Negative Testing): Спроби використовувати програму в неправильний або некоректний спосіб для перевірки її стійкості та відповідності обробці помилок.
* Тестування користувацького інтерфейсу (UI Testing): Перевірка користувацького інтерфейсу, включаючи відображення, реакцію на взаємодію користувача та специфікації дизайну.
* Тестування продуктивності (Performance Testing): Оцінка продуктивності програми, включаючи тестування навантаження, швидкості відгуку, масштабованості та інших характеристик продуктивності.
* Тестування безпеки (Security Testing): Перевірка на вразливості та виявлення помилок в забезпеченні безпеки програми. Включає тестування на вразливості, тестування на проникнення та інші види тестування безпеки.
* Автоматизоване тестування (Automated Testing): Використання автоматизованих інструментів та скриптів для виконання тестів автоматично, замість ручного тестування.
* Тестування на мобільних пристроях (Mobile Testing): Тестування програм для мобільних пристроїв, включаючи тести для iOS та Android платформ.
* Тестування веб-додатків (Web Application Testing): Тестування веб-сайтів та веб-додатків, включаючи перевірку сумісності з різними браузерами та платформами.
* Тестування відновлення (Recovery Testing): Відновлення системи після збою або відновлення даних з резервних копій.
* Тестування вмісту (Content Testing): Перевірка відповідності вмісту програми вимогам та якості.
* Тестування на відмову (Failure Testing): Тестування стійкості програми до надмірного навантаження та відмови окремих компонентів.
* Тестування сумісності (Compatibility Testing): Перевірка сумісності програми з різними операційними системами, браузерами та апаратними засобами.
* Тестування масштабованості (Scalability Testing): Визначення здатності програми масштабуватися під велике навантаження або зростання даних.
* Тестування відновлення даних (Data Recovery Testing): Перевірка здатності відновити втрачені дані після відмови.
* Тестування резервних копій (Backup Testing): Перевірка роботи процедур створення та відновлення резервних копій даних.

1. **Особливості модульного та приймального тестування.**

Модульне тестування і приймальне тестування - це два різні види тестування програмного продукту, які використовуються для різних цілей і на різних етапах розробки та тестування. Воті їх основні особливості:

**Модульне тестування (Unit Testing):**

* Ціль: Модульне тестування спрямоване на перевірку окремих компонентів (модулів) програми, таких як функції, класи, методи, або інші великі частини коду, які можна тестувати окремо.
* Ізоляція: Під час модульного тестування інші компоненти програми, крім того, який тестується, можуть бути імітовані або підроблені, щоб забезпечити ізоляцію.
* Автоматизація: Модульні тести часто автоматизовані, і вони виконуються швидко після кожного змінного коду.
* Мета: Головною метою модульного тестування є виявлення помилок і дефектів на ранніх етапах розробки і впевненість в правильному функціонуванні окремих компонентів.

**Приймальне тестування (Acceptance Testing):**

* Ціль: Приймальне тестування спрямоване на перевірку того, чи відповідає готовий програмний продукт вимогам та очікуванням клієнта або користувачів. Воно визначає, чи програма готова до впровадження або випуску.
* Ізоляція: Приймальне тестування зазвичай виконується на весь програмний продукт, як цілісну систему, і не включає імітацію або підроблення частин програми.
* Вручні тести: Приймальне тестування може включати в себе ручні тести, автоматизовані тести або комбінацію обох.
* Мета: Головною метою приймального тестування є перевірка програмного продукту на відповідність бізнес-вимогам та очікуванням користувачів і підтвердження його готовності до впровадження.

**Масштаби тестування:**

* Модульне тестування зазвичай виконується на низькому рівні - на рівні окремих компонентів або класів.
* Приймальне тестування зазвичай виконується на вищому рівні - на рівні цілого програмного продукту або системи.

**Час виконання:**

* Модульне тестування виконується під час розробки і тестування окремих компонентів, і це процес, який часто автоматизується та виконується під час кожної ітерації.
* Приймальне тестування зазвичай виконується після завершення розробки всього програмного продукту або його значних частин.

1. **Призначення бібліотек Mocha та Chai.**

Mocha і Chai - це популярні бібліотеки для тестування на платформі Node.js, які використовуються для створення та виконання тестів в JavaScript. Кожна з цих бібліотек виконує свою функцію і має свої особливості:

**Mocha:**

* Призначення: Mocha - це фреймворк для написання, організації та виконання тестів в JavaScript. Він дозволяє створювати тести з різними видами викликів, такими як синхронні, асинхронні, або чергові тести.
* Особливості:
* Підтримка BDD (Behavior-Driven Development) та TDD (Test-Driven Development) структуризації тестів.
* Зручний засіб для створення хуків перед та після виконання тестів.
* Можливість використання різних звітних форматів для результатів тестування.

**Chai:**

* Призначення: Chai - це бібліотека для засобів, які використовуються для створення асерцій (тверджень) в тестах. Вона дозволяє вам визначати, які обгрунтування ви хочете перевірити в тестах та яким чином.
* Особливості:
* Підтримка різних стилів асерцій, таких як BDD (Behavior-Driven Development) та TDD (Test-Driven Development).
* Зручний синтаксис для порівняння значень та перевірки умов.
* Можливість встановлення різних плагінів та розширень для розширення можливостей асерцій.

В разі комбінації Mocha і Chai Ви можете створювати зрозумілі, легкі для читання та детально розроблені тести. Mocha використовується для організації та виконання тестів, включаючи управління хуками, а Chai надає засоби для створення асерцій для перевірки тверджень у ваших тестах.