МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ  
  
  
  
  
  
  
  
  
  
Дисципліна «Автоматизоване тестування»

Лабораторна робота № 5

Виконала: студентка гр. 8.1213  
 Абдуллаєва Севда   
 Самандар Кизи

Перевірив: Лебідь Максим   
 Володимирович

Запоріжжя  
2023

**Як результат виконання лабораторної роботи №5 було завантажено лише звіт, так як розмір архівного файлу перевищує 5 МБ. Всі файли завантажені в репозиторій:** [**https://github.com/SevdaAbdullaeva/Cypress-test\_lab5.git**](https://github.com/SevdaAbdullaeva/Cypress-test_lab5.git)

**Завдання**

1. Реалізувати скрипт для автоматизації наступних дій Cypress:

* 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:**

describe('Math Test', () => {

    it('Performs math test', () => {

      cy.visit('http://suninjuly.github.io/math.html');

      // Прочитати значення x

      cy.get('#input\_value').invoke('text').then((x) => {

        const result = calculateMathFunction(x);

        // Ввести відповідь в текстове поле

        cy.get('#answer').type(result);

        // Вибрати checkbox "I'm the robot"

        cy.get('[type="checkbox"]').check();

        // Вибрати radiobutton "Robots rule!"

        cy.get('[type="radio"]').check('robots');

        // Натиснути кнопку Submit

        cy.get('button[type="submit"]').click();

      });

    });

  });

  // Функція для обчислення математичної функції

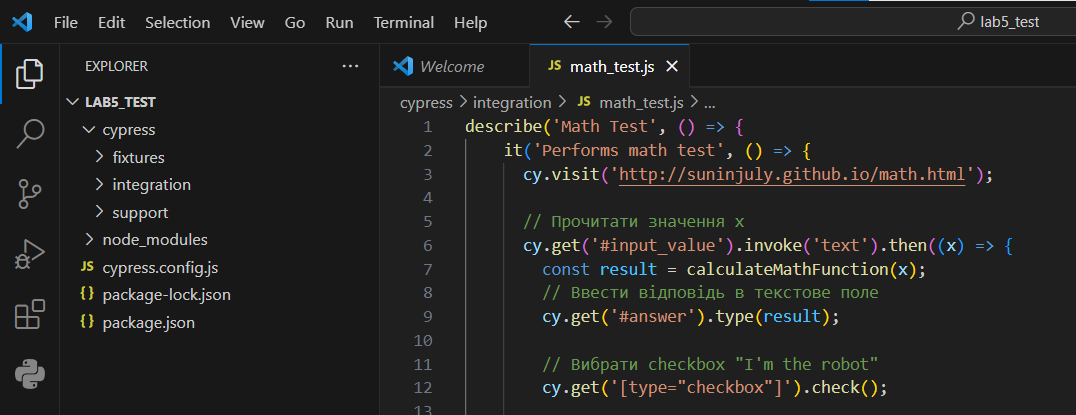
  function calculateMathFunction(x) {

    // Ваш код для обчислення математичної функції тут

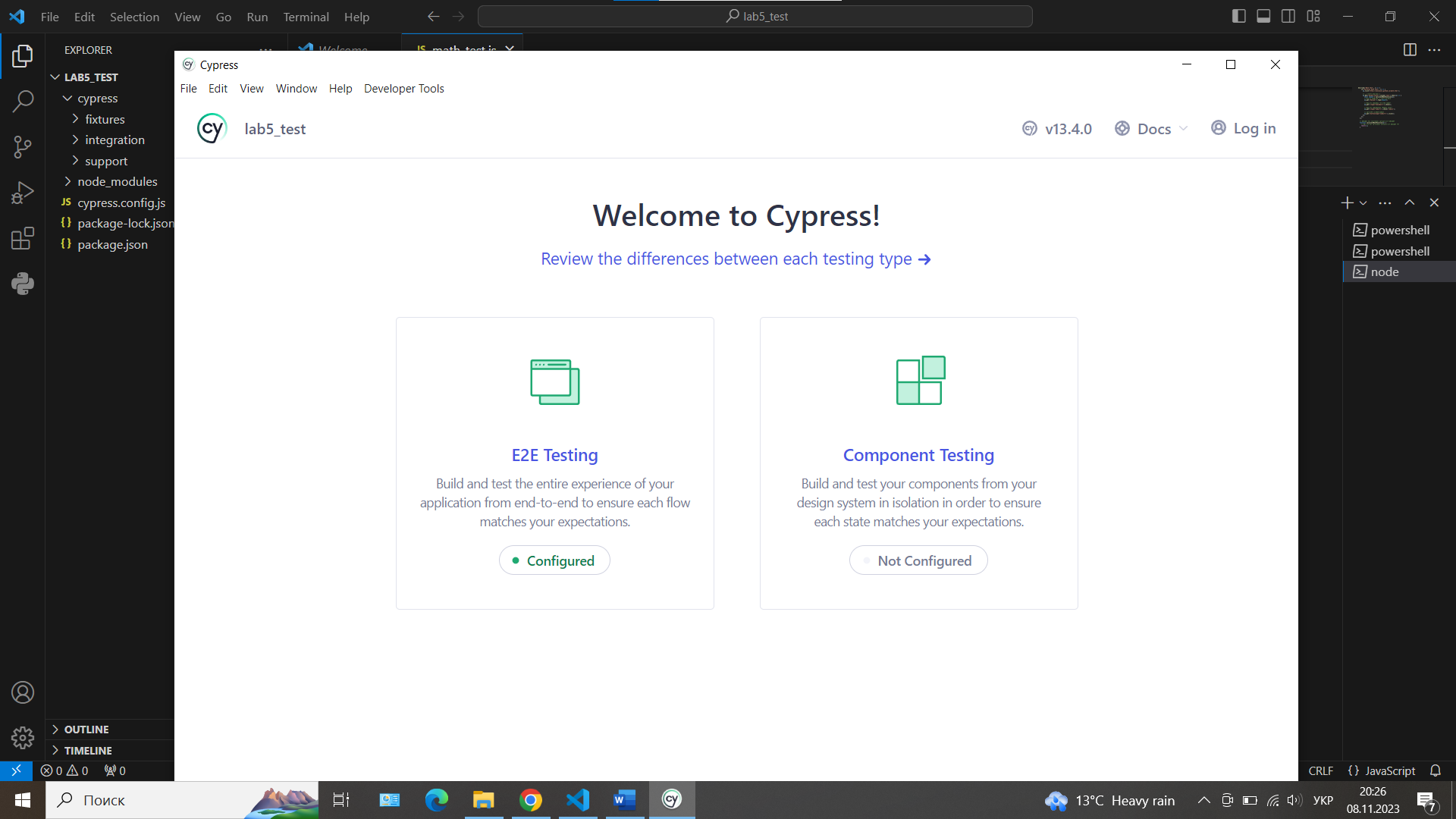
    return x;

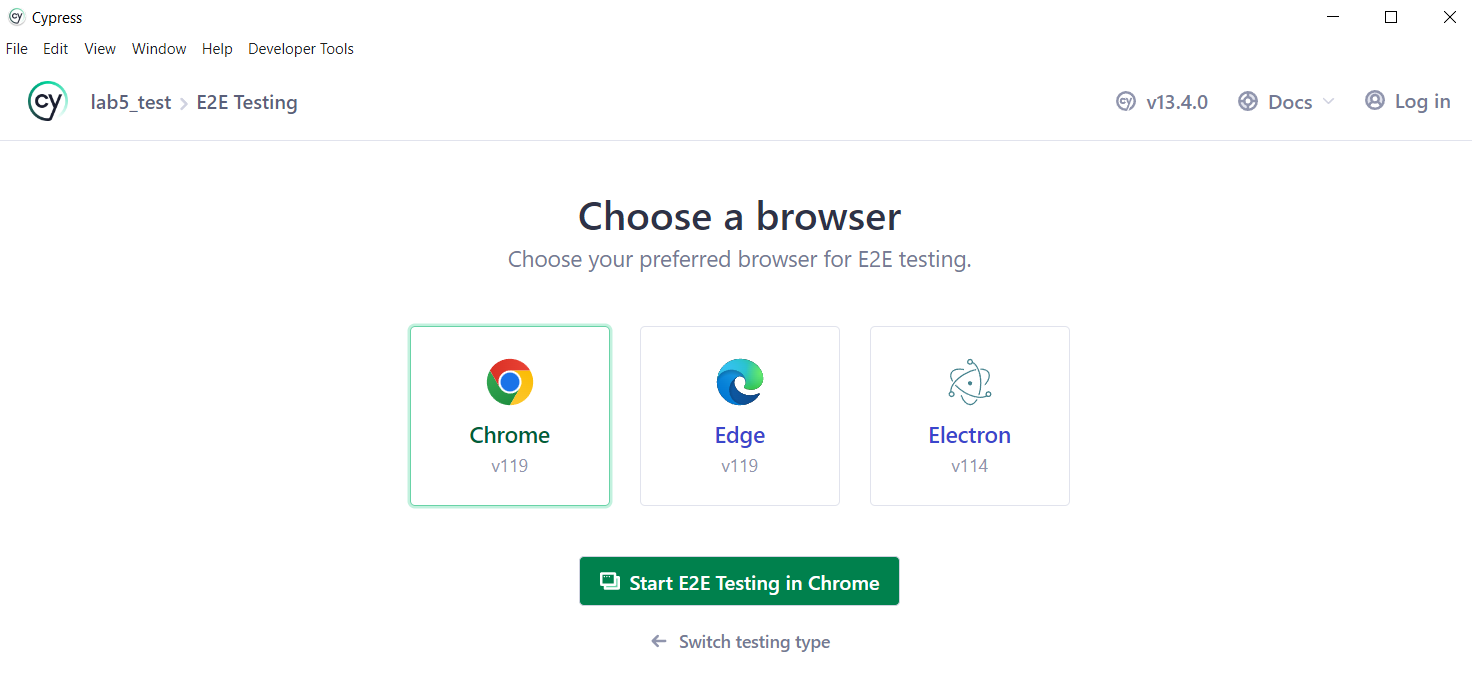
  }

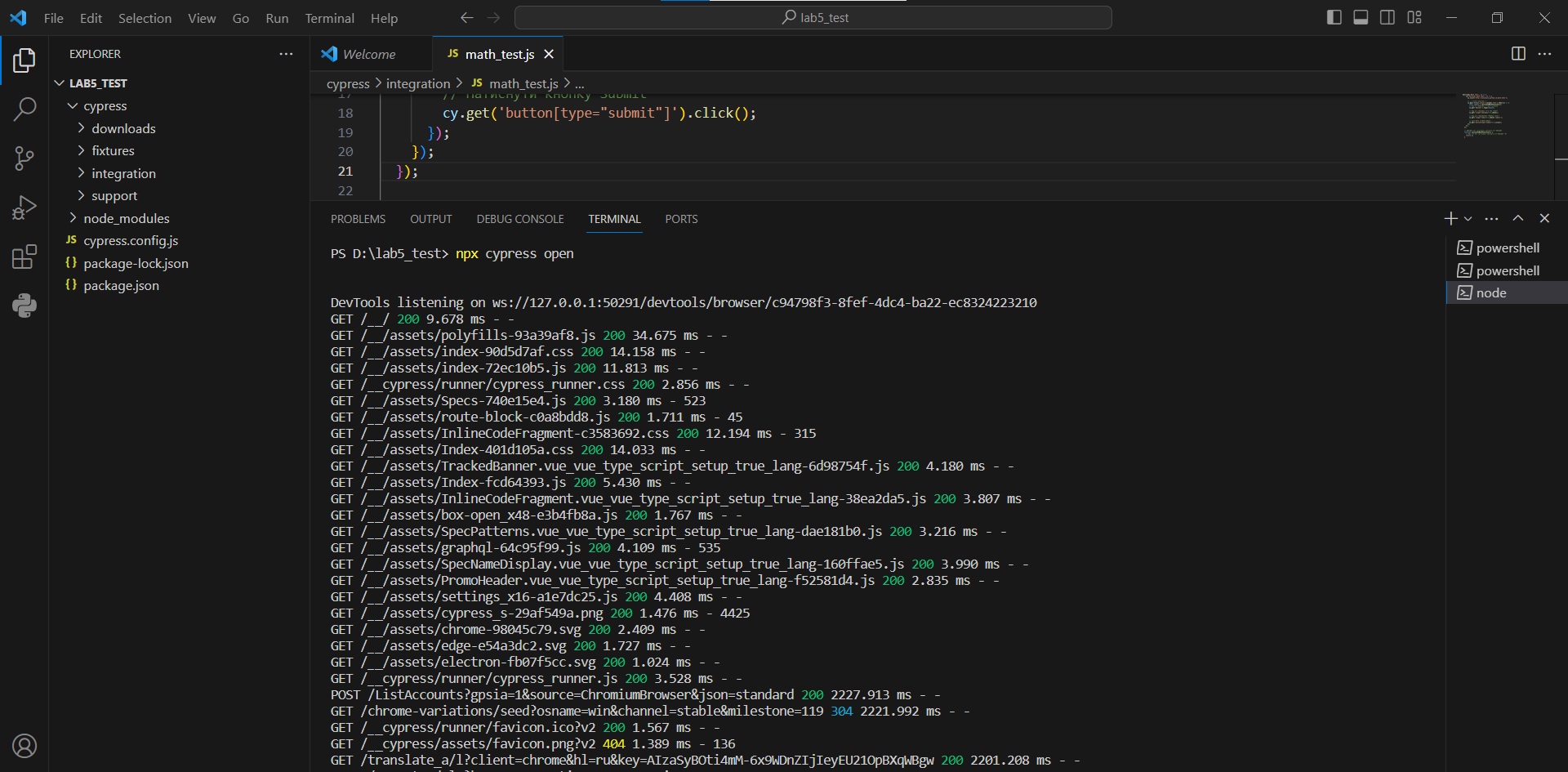
**Структура файлів:**

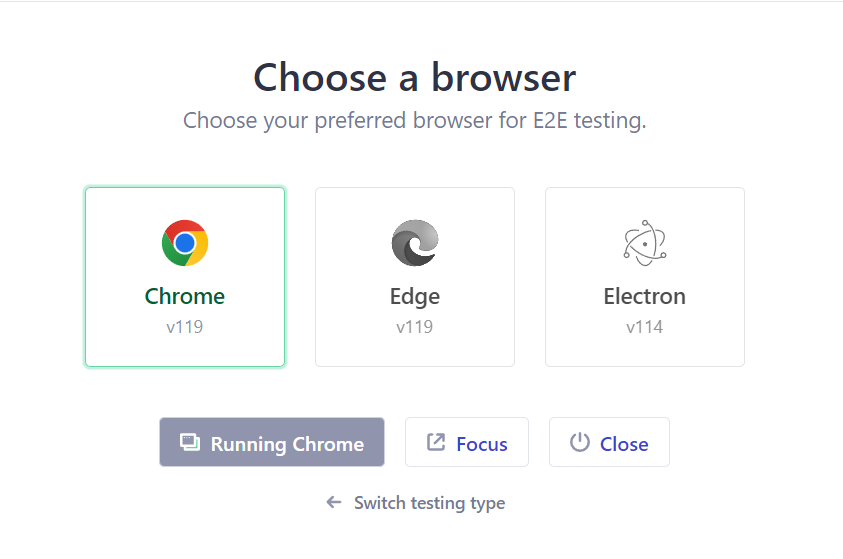
****

**Результати роботи програми:**

****

****

****

****

1. Реалізувати скрипт тестування наступного функціонала ресурсу

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

* Пошук довільного товару.
* Тестування роботи корзини.
* Довільний функціонал за вибором студента.

**Код програми:**

// Пошук довільного товару:

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

const chrome = require('selenium-webdriver/chrome');

// Функція для створення і налаштування драйвера

async function setupDriver() {

  const chromeOptions = new chrome.Options();

  chromeOptions.addArguments('--start-maximized'); // Максимізує вікно браузера

  const driver = new Builder()

    .forBrowser('chrome')

    .setChromeOptions(chromeOptions)

    .build();

  return driver;

}

// Основна функція для виконання тесту

async function searchProductTest() {

  const driver = await setupDriver();

  try {

    // Відкрити веб-сторінку

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

    // Знайти поле для введення тексту пошуку

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

    // Ввести пошуковий запит (наприклад, "shirt")

    searchBox.sendKeys('shirt', Key.RETURN);

    // Очікування завантаження результатів пошуку

    await driver.wait(until.titleContains('Search results'), 5000);

    // Вивести заголовок сторінки з результатами пошуку

    const pageTitle = await driver.getTitle();

    console.log('Page Title:', pageTitle);

  } finally {

    // Завершити роботу драйвера

    driver.quit();

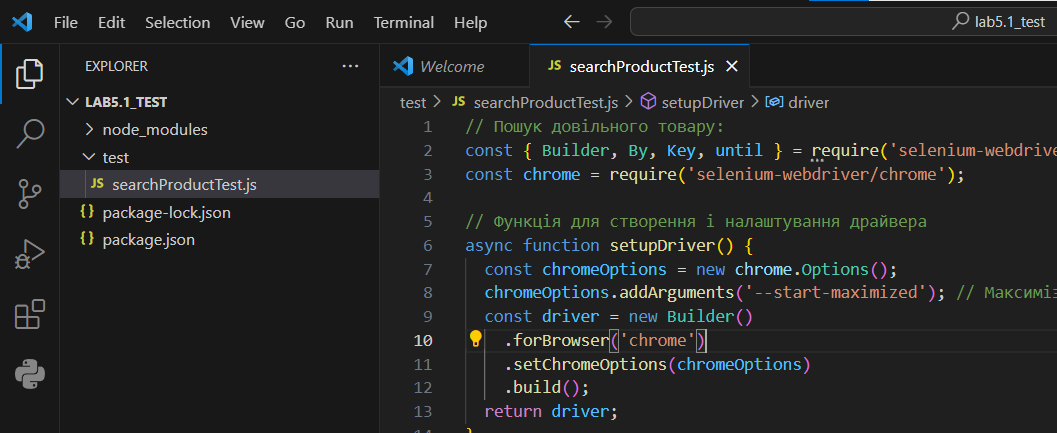
  }

}

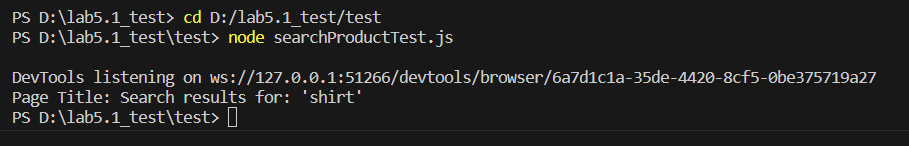
// Виклик функції для тестування

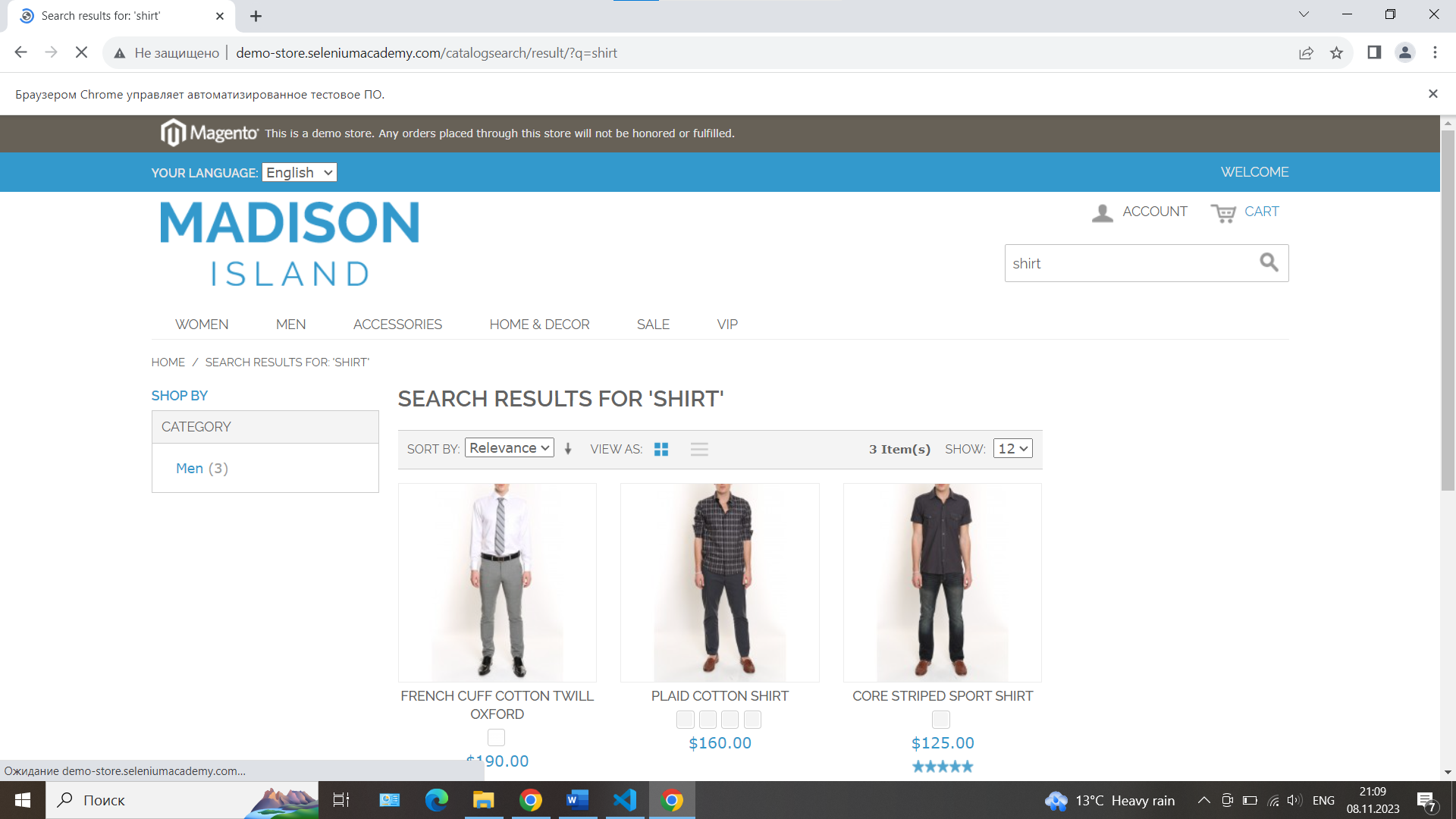
searchProductTest();

**Структура файлів:**

****

**Результати роботи програми:**

****

****

**Код програми:**

// Тестування роботи корзини:

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

const chrome = require('selenium-webdriver/chrome');

// Функція для створення і налаштування драйвера

async function setupDriver() {

  const chromeOptions = new chrome.Options();

  chromeOptions.addArguments('--start-maximized'); // Максимізує вікно браузера

  const driver = new Builder()

    .forBrowser('chrome')

    .setChromeOptions(chromeOptions)

    .build();

  return driver;

}

// Основна функція для виконання тесту

async function cartTest() {

  const driver = await setupDriver();

  try {

    // Відкрити веб-сторінку

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

    // Знайти товар, який ви хочете додати в корзину (наприклад, "Magic Mouse")

    const productLink = await driver.findElement(By.linkText('Magic Mouse'));

    // Клікнути на посилання товару

    productLink.click();

    // Дочекатися завантаження сторінки товару

    await driver.wait(until.titleContains('Magic Mouse'), 5000);

    // Знайти кнопку "Add to Cart"

    const addToCartButton = await driver.findElement(By.id('product-addtocart-button'));

    // Клікнути на кнопку "Add to Cart"

    addToCartButton.click();

    // Дочекатися відобразження повідомлення про успішне додавання товару в корзину

    await driver.wait(until.elementLocated(By.className('success-msg')), 10000);

    // Знайти посилання "View and Edit Cart"

    const viewCartLink = await driver.findElement(By.linkText('View and Edit Cart'));

    // Клікнути на посилання "View and Edit Cart"

    viewCartLink.click();

    // Дочекатися завантаження сторінки корзини

    await driver.wait(until.titleContains('Shopping Cart'), 5000);

    // Вивести заголовок сторінки корзини

    const pageTitle = await driver.getTitle();

    console.log('Page Title:', pageTitle);

  } finally {

    // Завершити роботу драйвера

    driver.quit();

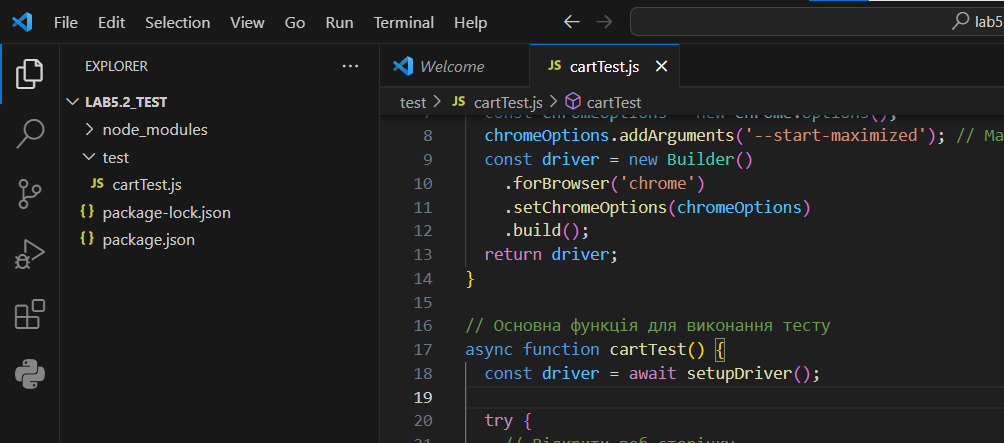
  }

}

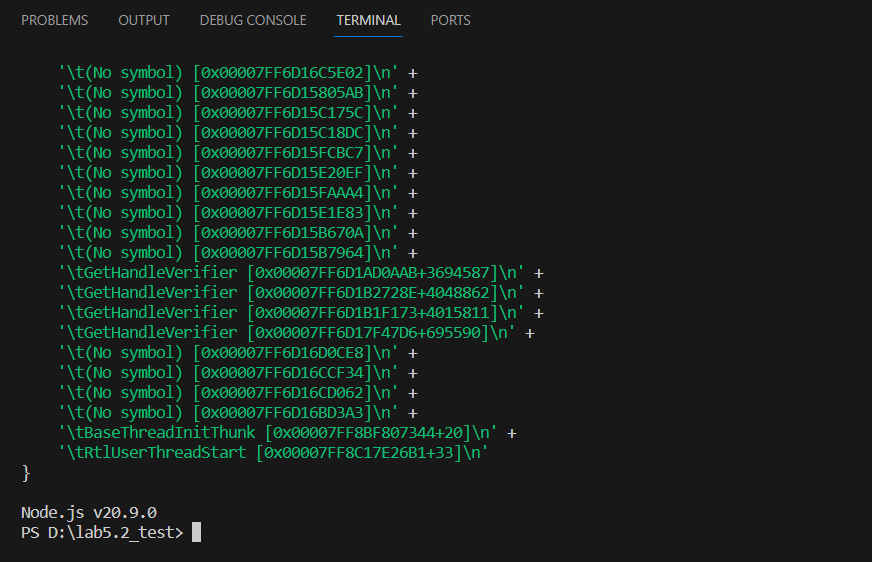
// Виклик функції для тестування

cartTest();

**Структура файлів:**

****

**Результати роботи програми:**

****

**Код програми:**

// Перевірку, чи відображається повідомлення про успішне додавання товару до корзини:

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

const chrome = require('selenium-webdriver/chrome');

// Функція для створення і налаштування драйвера

async function setupDriver() {

  const chromeOptions = new chrome.Options();

  chromeOptions.addArguments('--start-maximized'); // Максимізує вікно браузера

  const driver = new Builder()

    .forBrowser('chrome')

    .setChromeOptions(chromeOptions)

    .build();

  return driver;

}

// Основна функція для виконання тесту

async function checkCartMessageTest() {

  const driver = await setupDriver();

  try {

    // Відкрити веб-сторінку

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

    // Знайти товар, який ви хочете додати в корзину (наприклад, "Magic Mouse")

    const productLink = await driver.findElement(By.linkText('Magic Mouse'));

    // Клікнути на посилання товару

    productLink.click();

    // Дочекатися завантаження сторінки товару

    await driver.wait(until.titleContains('Magic Mouse'), 5000);

    // Знайти кнопку "Add to Cart"

    const addToCartButton = await driver.findElement(By.id('product-addtocart-button'));

    // Клікнути на кнопку "Add to Cart"

    addToCartButton.click();

    // Дочекатися відобразження повідомлення про успішне додавання товару в корзину

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

    // Перевірка, чи повідомлення існує на сторінці

    if (await successMessage.isDisplayed()) {

      console.log('Success message displayed: Item added to cart');

    } else {

      console.log('Success message not displayed');

    }

  } finally {

    // Завершити роботу драйвера

    driver.quit();

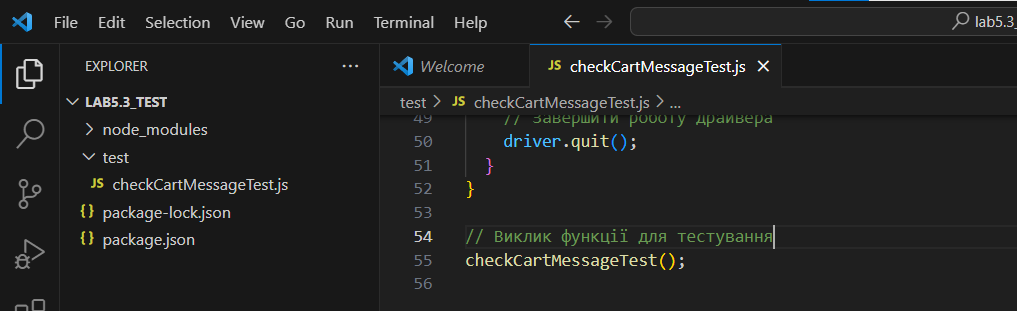
  }

}

// Виклик функції для тестування

checkCartMessageTest();

**Структура файлів:**

****

**Результати роботи програми:**

****

**Посилання на репозиторій:**

<https://github.com/SevdaAbdullaeva/Cypress-test_lab5.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.