SWAG LABS E-COMMERCE AUTOMATION PROJECT

DEPI Graduation Project

DKH2_SWD6_S2_Group2

Content

Our Team

1- Hisham Hossam Ahmed

2- Omar Talat El Said

3- Ziad Mohmed Helmy Mady

4-Marwa Mostafa Ibrahim

5- Yosef Emad Lotfy

What Is Software Testing?

Software Testing is the process of checking that a software application works correctly, has no errors, and performs the required tasks as expected by the user. The main goal is to ensure the quality of the software before it is released or used, in order to avoid problems and improve the user experience.

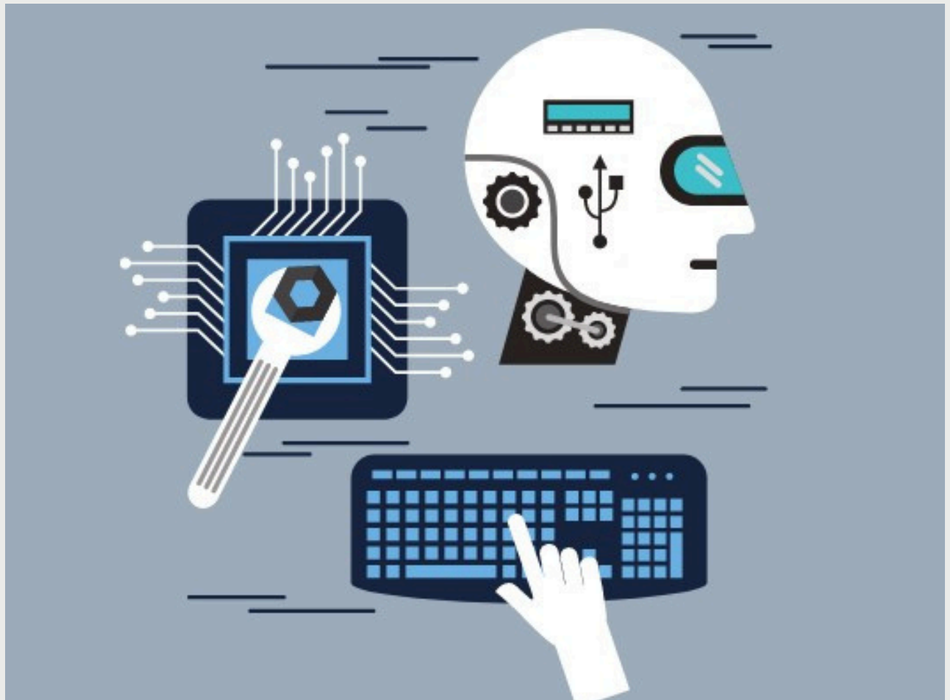
Manual VS Automation Testing

Manual Testing

It is done by humans without tools, ideal for simple or exploratory tests but slower and prone to errors

Automation Testing

It uses tools to run tests faster and more accurately, best for repetitive tasks but requires setup and technical skills.



When we use Manual Testing

- **When human observation and judgment are essential**: For aspects like usability, aesthetics, and exploratory analysis.
- **When understanding the user experience is the priority**: To ensure the software feels intuitive and meets user needs.
- **When dealing with uncertainty or novelty**: For exploring new features or applications where test cases aren't well-defined.
- **When the cost or effort of automation outweighs the benefits**: For small, short-lived, or frequently changing parts of the application.
- **When direct user feedback is required**: During user acceptance testing to gauge real-world usability.

When we use Automation Testing

- **Regression Testing:** To ensure new changes don't break existing features.
- **Repetitive Tasks:** For tests run frequently.
- **Performance Testing:** To evaluate system responsiveness under load.
- **Load Testing:** To assess behavior under expected and peak loads.
- **Stress Testing:** To find the system's breaking point.
- **Complex Test Cases:** With many steps or data inputs.
- **Data-Driven Testing:** Running the same test with multiple datasets.
- **Cross-Browser/Platform Testing:** To verify compatibility across different environments.

Used Automation Tools

- **Maven:** A build automation and dependency management tool for Java projects.
- **Selenium WebDriver:** A tool for automating web application testing using locators and commands.
- **TestNG Assertion Tool:** A testing framework for writing structured and repeatable test cases.

About The Project

The automation efforts for this project target the Swag Labs website, which is a demonstration e-commerce platform offered by Sauce Labs. This website serves as a learning ground for test automation engineers, particularly for practicing with tools like Selenium WebDriver, TestNG, and Maven. The structure of the automation framework also incorporates the Page Object Model (POM).

The Project Details

Login Page

```java
@Test
    public void testLoginWithValidData() {
        login.fillUserName("standard_user");
        login.fillPassword("secret_sauce");
        login.ClickButton();
        waitForInventoryPage();

        String currentUrl = driver.getCurrentUrl();
        Assert.assertEquals(currentUrl, "https://www.saucedemo.com/inventory.html");
        Assert.assertTrue(login.isLoginSuccessful(), "Login should be successful for standard_user.");
        System.out.println("testLoginWithValidData: PASSED");
    }
```

```java
@Test
    public void testLoginWithTechnicalProblems() {
        login.fillUserName("problem_user");
        login.fillPassword("secret_sauce");
        login.ClickButton();
        waitForInventoryPage();

        String currentUrl = driver.getCurrentUrl();
        Assert.assertEquals(currentUrl, "https://www.saucedemo.com/inventory.html");
        boolean isLoginSuccessful = login.isLoginSuccessful();
        Assert.assertTrue(isLoginSuccessful, "Login should be successful for problem_user.");
        System.out.println("testLoginWithTechnicalProblems: PASSED");
    }
```

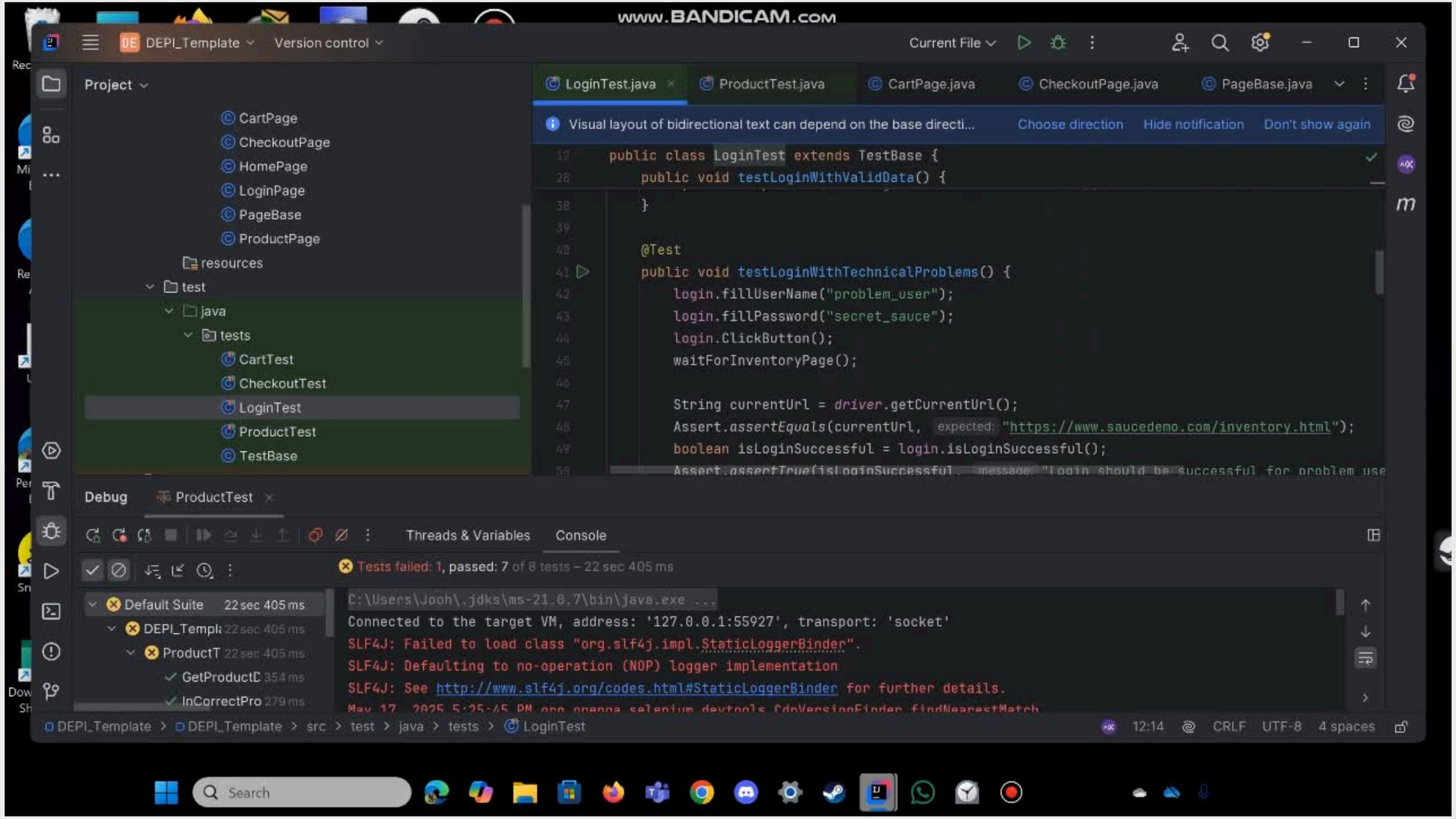
The Project Details

Login Page

```java
@Test
    public void testLoginWithErrorUser() {
        login.fillUserName("error_user");
        login.fillPassword("secret_sauce");
        login.ClickButton();
        waitForInventoryPage();

        Assert.assertTrue(login.isLoginSuccessful(), "Login should be successful for error_user.");
        System.out.println("testLoginWithErrorUser: PASSED");
    }

    @Test
    public void testLoginWithVisualUser() {
        login.fillUserName("visual_user");
        login.fillPassword("secret_sauce");
        login.ClickButton();
        waitForInventoryPage();

        Assert.assertTrue(login.isLoginSuccessful(), "Login should be successful for visual_user.");
        System.out.println("testLoginWithVisualUser: PASSED");
    }
```

The Project Details

Login Page



The Project Details

Product Page

```java
@Test(dataProvider = "sortOptions")
    public void verifySorting(String sortOption, boolean isNameSort) {
        LoginPage login = new LoginPage(driver);
        login.fillUserName("standard_user");
        login.fillPassword("secret_sauce");
        login.ClickButton();
        ProductPage productPage = new ProductPage(driver);
        productPage.selectSortOption(sortOption);
        if (isNameSort) {
            List<WebElement> elements = productPage.getProductNameElements();
            List<String> actualNames = new ArrayList<>();
            for (WebElement el : elements) {
                actualNames.add(el.getText());
            }
            List<String> sortedNames = new ArrayList<>(actualNames);
            if (sortOption.equals("Name (Z to A)")) {
                Collections.sort(sortedNames, Collections.reverseOrder());
            } else {
                Collections.sort(sortedNames);
            }
            Assert.assertEquals(actualNames, sortedNames);
        } else {
            List<WebElement> priceElements = productPage.getProductPriceElements();
            List<Double> actualPrices = new ArrayList<>();
            for (WebElement el : priceElements) {
                actualPrices.add(Double.parseDouble(el.getText().replace("$", "")));
            }
            List<Double> sortedPrices = new ArrayList<>(actualPrices);
            if (sortOption.equals("Price (high to low)")) {
                Collections.sort(sortedPrices, Collections.reverseOrder());
            } else {
                Collections.sort(sortedPrices);
            }
            Assert.assertEquals(actualPrices, sortedPrices);
        }
    }
```

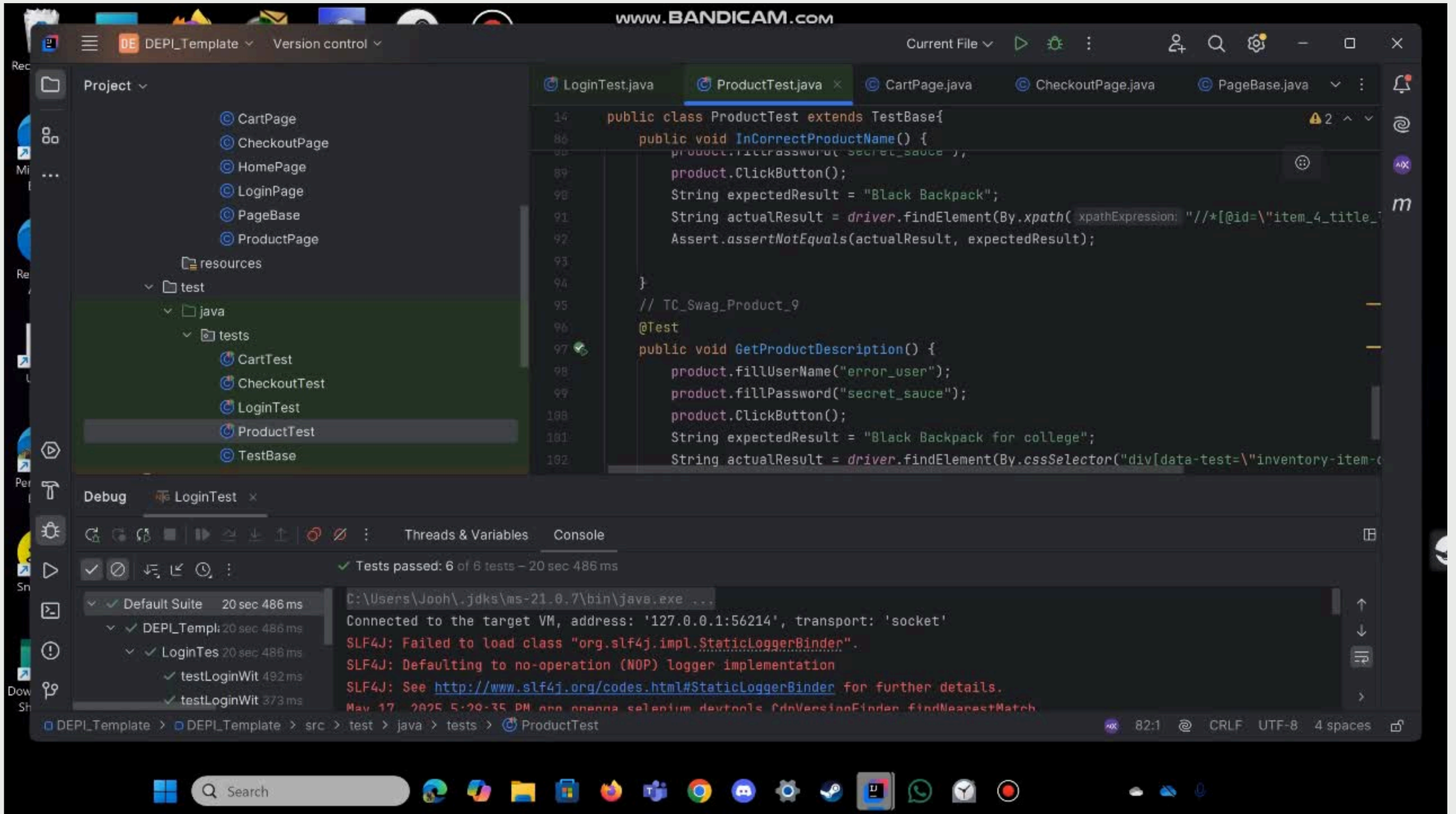
The Project Details

Product Page

```
14    public class ProductTest extends TestBase{  marwamostafa64 +1
60        public void verifySorting(String sortOption, boolean isNameSort) {
84                    actualPrices.add(Double.parseDouble(el.getText().replace( target: "$",  replacement: "")));
85                }
86                List<Double> sortedPrices = new ArrayList<>(actualPrices);
87                if (sortOption.equals("Price (high to low)")) {
88                    Collections.sort(sortedPrices, Collections.reverseOrder());
89                } else {
90                    Collections.sort(sortedPrices);
91                }
92                Assert.assertEquals(actualPrices, sortedPrices);
93            }
94        }
95        //TC_Swag_Product_005
96        @Test  marwamostafa64
97        public void verifyUniqueImages() {
98            LoginPage login = new LoginPage(driver);
99            login.fillUserName("problem_user");
100           login.fillPassword("secret_sauce");
101           login.ClickButton();
102
103           ProductPage productPage = new ProductPage(driver);
104           List<WebElement> images = productPage.getProductImageElements();
105
106           List<String> srcList = new ArrayList<>();
107           for (WebElement img : images) {
108               srcList.add(img.getAttribute( name: "src"));
109           }
110
111           Set<String> uniqueImages = new HashSet<>(srcList);
112           Assert.assertEquals(uniqueImages.size(), srcList.size(),  message: "Some product images are duplicated!");
113       }
```

The Project Details

Product Page



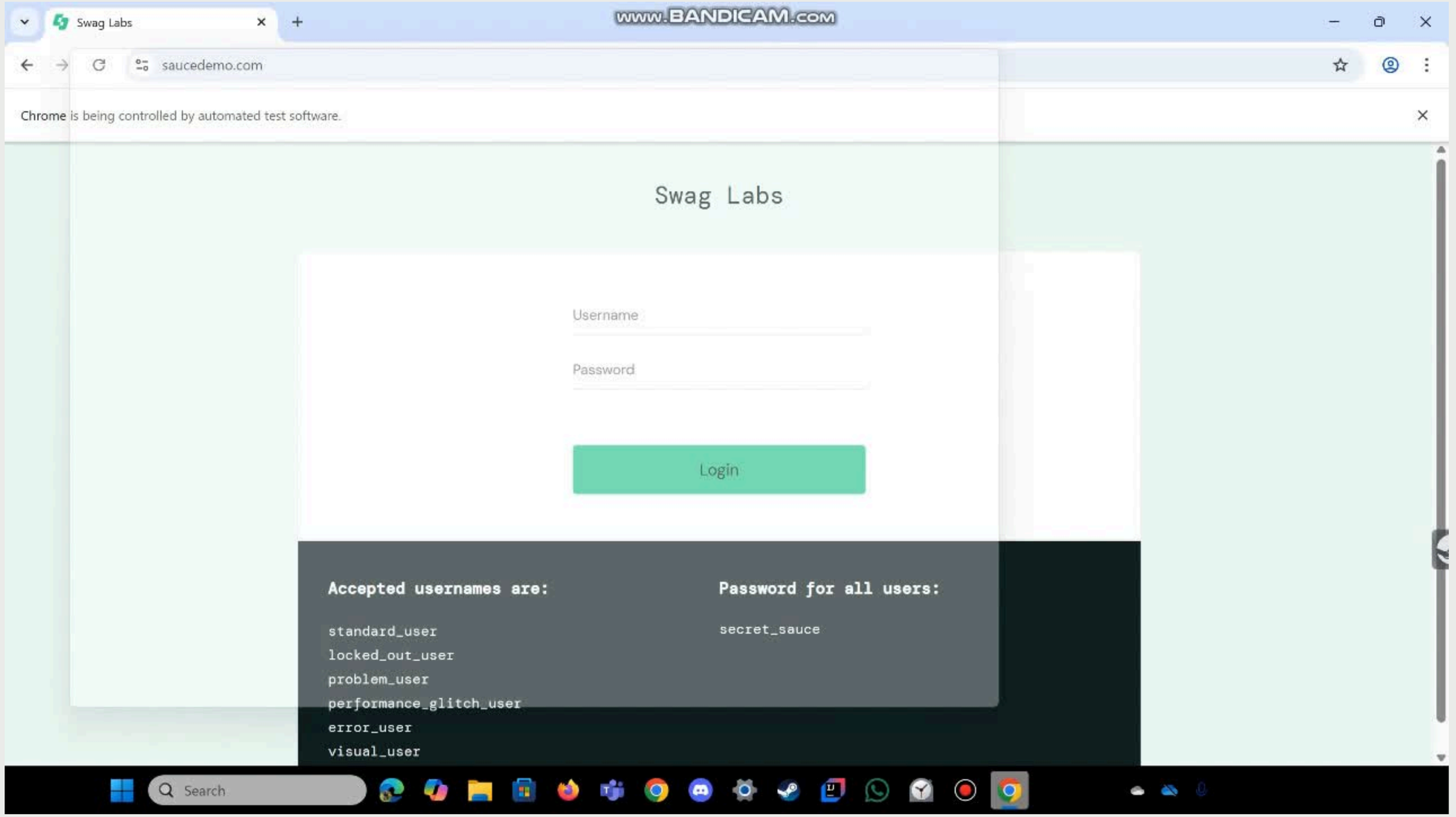
The Project Details

Cart Page

```
1   @Test
2       public void testLoginInCheckout() {
3           cart.openCart();
4           cart.clickCheckout();
5           Assert.assertTrue(cart.isOnCheckoutPage());
6       }
7       @Test
8       public void  testRemoveProductFunction(){
9           products.addFirstProductToCart();
10          products.goToCart();
11          cart.isRemoveButtonDisplayed();
12          cart.clickOnRemoveBtn();
13      }
```

The Project Details

Cart Page



The Project Details

CheckOut Page

```
public class CheckoutTest extends TestBase {
    @Test
    public void testStandardUserCheckoutFlow() throws InterruptedException {
        // Login
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

        driver.findElement(By.id("user-name")).sendKeys(STANDARD_USERNAME);
        driver.findElement(By.id("password")).sendKeys(PASSWORD);
        Thread.sleep( millis: 1000);
        driver.findElement(By.id("login-button")).click();
        Thread.sleep( millis: 1000);
        // Verify successful login
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.className("inventory_list")));
        Assert.assertTrue(driver.getCurrentUrl().contains("inventory.html"), message: "Login failed");

        // Add items to cart
        driver.findElement(By.id("add-to-cart-sauce-labs-backpack")).click();
        driver.findElement(By.id("add-to-cart-sauce-labs-bike-light")).click();
        Thread.sleep( millis: 1000);
        // Go to cart
        driver.findElement(By.className("shopping_cart_link")).click();

        wait.until(ExpectedConditions.visibilityOfElementLocated(By.className("cart_list")));

        // Verify items in cart
        Assert.assertEquals(driver.findElements(By.className("cart_item")).size(), expected: 2, message: "Cart should contain 2 items");
        Thread.sleep( millis: 1000);
        // Proceed to checkout
        driver.findElement(By.id("checkout")).click();
```

THANK YOU