Test Approach for Automated testing of Swag Labs

This repository contains an automated testing project for Swag Labs using Java, Cucumber, BDD, Selenium, Gherkin, Chrome Driver, Maven, and IntelliJ IDE. The tests are designed to cover various scenarios on the Sauce Demo website (https://www.saucedemo.com/v1/index.html).

Table of Contents

Introduction

Test automation is the process of using software tools and frameworks to automate test execution, compare actual outcomes to expected outcomes, and generate test reports. This helps to increase efficiency, accuracy, and repeatability of testing processes, as well as reducing the time and costs associated with manual testing.

In here we are going to test SWAG Labs (Sauce Demo - https://www.saucedemo.com/v1/index.html). We are going to use tools and frameworks like Cucumber BDD (Behaviour-driven development) with Selenium for the automation. This project focuses on automating tests for the Swag Labs website using Cucumber BDD and Selenium.

Scope

The project covers the following areas of the Swag Labs website:

- Login page for all accepted users and error scenarios.
- Product page, including sorting features and product detailed view.
- Your cart page for all accepted users.

Features

- Utilizes Cucumber for behaviour-driven development (BDD) to write tests in a human-readable format.
- Uses Selenium for interacting with web elements and performing automated actions.
- Organizes test scenarios using Gherkin syntax.
- Includes a wide range of test cases covering login, product features, cart functionality, and more.
- Provides easy-to-understand bug reports and test outcomes.

Tools and Technologies Used

This test automation project for Swag Labs utilizes a variety of tools and technologies to streamline the testing process and ensure efficient and accurate results.

- **Java**: The primary programming language used for writing the test automation code.
- **Cucumber**: A Behaviour-Driven Development (BDD) tool that allows writing test scenarios in a human-readable format.
- **Selenium**: A popular open-source framework for automating web browser interactions and testing.
- **Gherkin**: A plain-text language that uses keywords to define test scenarios in a structured way, making them easy to read and understand.
- **Chrome Driver**: The WebDriver implementation for Google Chrome, enabling Selenium to automate Chrome browser actions.
- **Maven**: A build and dependency management tool that simplifies project setup and maintenance.
- **Intellij IDE**: An integrated development environment for Java that provides a workspace for coding, testing, and debugging.

These technologies are combined to create an effective testing framework that automates test cases, performs comparisons between expected and actual outcomes, and generates detailed test reports.

Test Cases

The test cases are written in the Gherkin syntax, which offers a clear representation of the test scenarios, and Each test case covers various aspects of the Swag Labs website, ensuring comprehensive coverage.

Please refer the Test Cases mentioned below for complete details:

- Check the login functionality with accepted username and passwords.

- Login to Swag Labs application and add a product to the cart and complete the order.
- Login to Swag labs application and add multiple products to the cart and complete the order.
- Log in to the Swab Labs application, add an item to the cart, proceed to check out, and then click on "remove" to remove the item from the cart.
- Login to the Swag Labs application, add an item to the cart, fill out the form, and verify the checkout information before submitting.

Installation

Ensure that you have the following tools and technologies installed on your machine:

- Java
- Maven
- Intellij IDE
- Selenium Need to add Selenium jar files to class path!
- Cucumber
- Chrome driver
- Junit

Usage

1. Open the project in Intellij or your preferred IDE (Recommended IntelliJ idea)
2. Navigate to the `src/test/java` directory.
3. Locate the test files with the. feature extension.
4. Run the test files using Test Runner to execute the automated test scenarios.

BDD Implementation Steps:

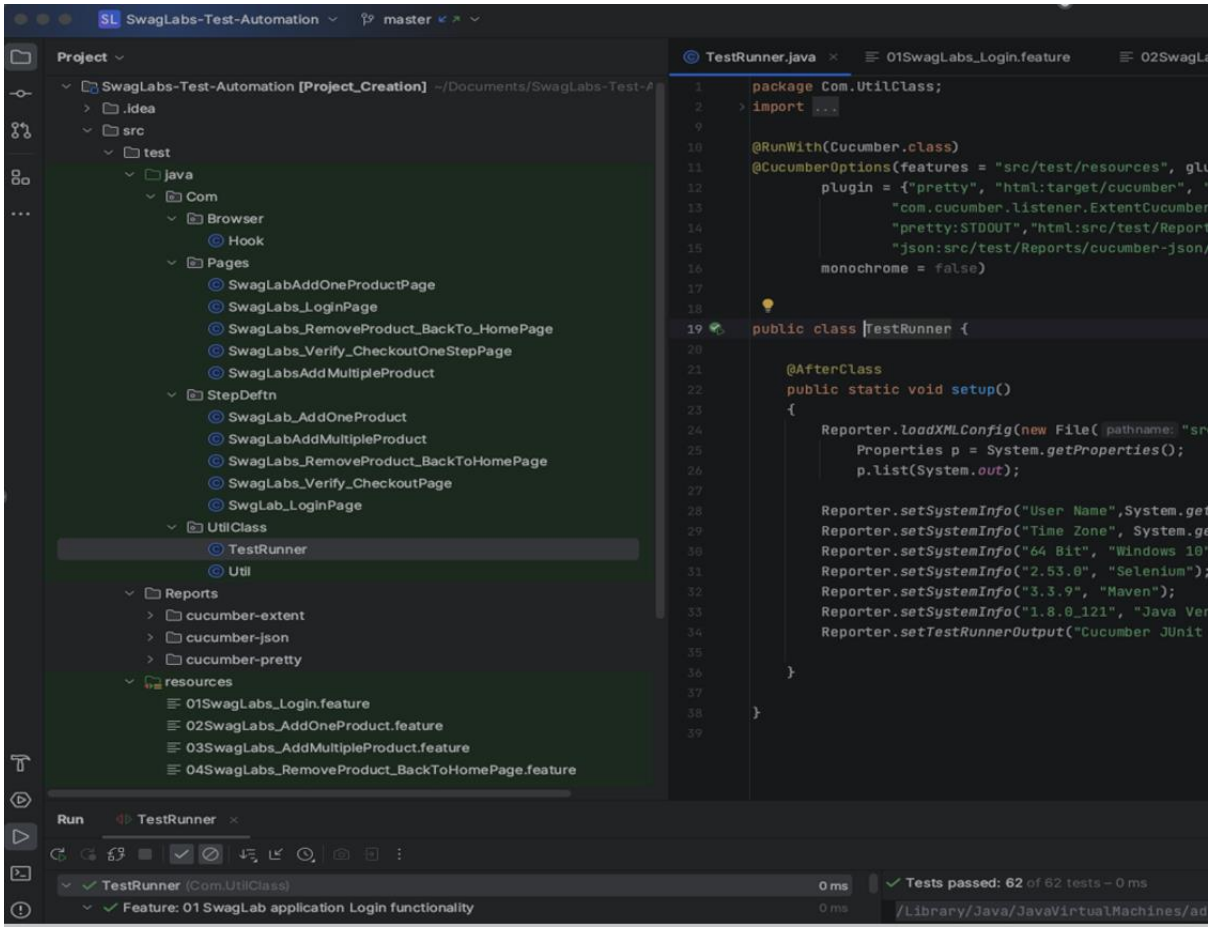
GitHub Repo details:

GitHub - bharatidevaramani/TISS-SwagLabs-Test-Automation

URL: https://github.com/bharatidevaramani/TISS-SwagLabs-Test-Automation

Project structure:

The below mentioned screenshot has been captured from IDE and that explains that framework has been structured and implemented for Swag Labs application.



1. ***src/test /java/Com/Browser****:* The folder is used to store the page objects that are required by my project, such as locators, configuration files, methods for every page, application properties files, or test data files.

2. **src/test /java/Com/page***:* The folder is used to store the page objects that are required by my project, such as Locators, configuration files, method for every page on the application properties files, or test data files.

3. **src/test/Java /Com /UtilClass/TestRunner.java*:*** The TestRunner class is responsible for executing the test cases and generating the test reports. It serves as an entry point for running BDD tests.

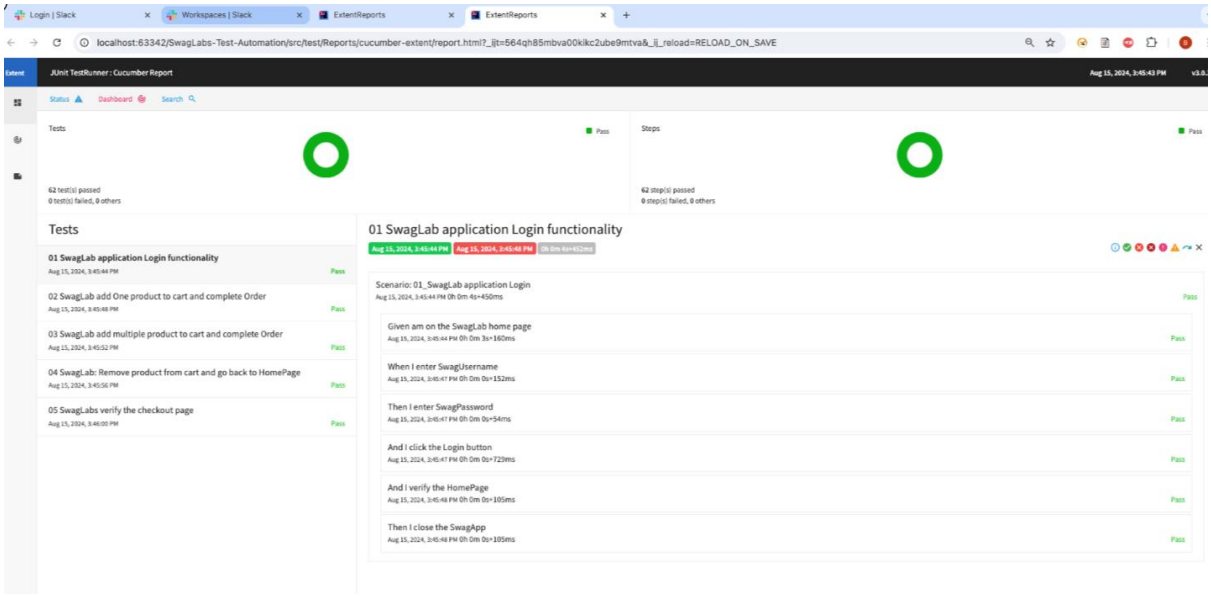
4. **src/test/Java /Com /UtilClass/Util.Java:** This folder contains all the common methods to enter the data in a text field, click on a web button, select an item from a dropdown, wait for the page to load, select the radio button, and check the checkbox, etc.

5. src/test/java/Com/stepDeftn: The Step Definitions folder contains the step definition files for BDD scenarios. Each file corresponds to a specific feature or functionality and defines the step definitions that map to the Gherkin syntax in your feature files.

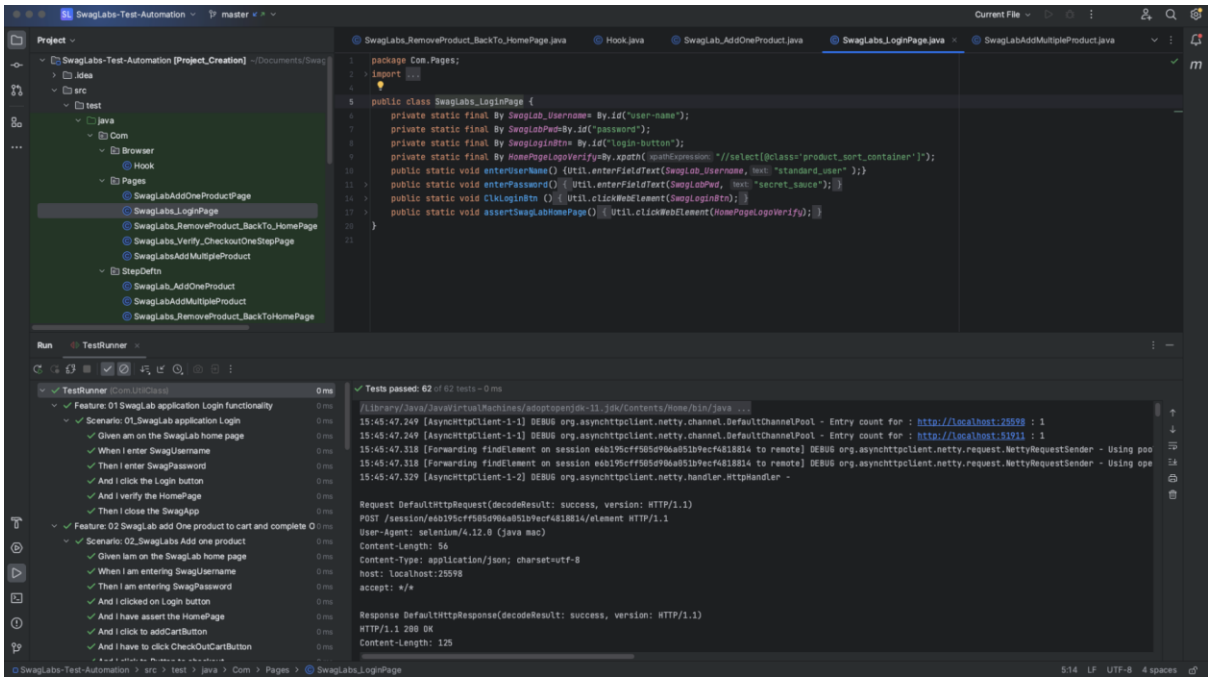
6. ***src/test/ java/Com/resources:*** This folder contains the feature files written in Gherkin syntax. Each feature file represents a specific feature or functionality that we want to test.

7. ***src/test/ java/Com/Reports:*** The folder contains the reports of the test execution, and these reports are familiar to people, even those who are non-technical, such as business and other stakeholders.

Screenshot of the sample test execution report:



Screenshot for Test Runner logs in the console:



Feature files:

```
Feature:01 SwagLab application Login functionality

  Scenario:01_SwagLab application Login
    Given am on the SwagLab home page
    #When I enter "<SwagUsername>"
    #|SwagUsername|
    #|standard_user|
    #|locked_out_user|
    #|problem_user|
    #|performance_glitch_user|
    When I enter SwagUsername
    Then I enter SwagPassword
    And I click the Login button
    And I verify the HomePage
    Then I close the SwagApp

    #Examples:
    #|SwagUsername|            |SwagPassword|
    #|standard_user|           |secret_sauce|           |
    #|locked_out_user|         |secret_sauce|
    #|problem_user|            |secret_sauce|
    #|performance_glitch_user| |secret_s
```

```
Feature:02 SwagLab add One product to cart and complete Order

  Scenario:02_SwagLabs Add one product
    Given Iam on the SwagLab home page
    When I am entering SwagUsername
    Then I am entering SwagPassword
    And I clicked on Login button
    And I have assert the HomePage
    And I click to addCartButton
    And I have to click CheckOutCartButton
    And I click to Button to checkout
    And I enter FirstName
    And I enter LastName
    And I enter PostCode
    And I click continue button
    And I click Finish button
    And I verify order successful
    Then I would close the SwagApp
```

```
Feature:03 SwagLab add multiple product to cart and complete Order

  Scenario:03_SwagLabs Add multiple product
    Given I access SwagLab home page
    When I entered SwagUsername
    Then I entered SwagPassword
    And I press Login button
    And I asserted the HomePage
    And I clicked to addToCartButton
    And I clicked to addToCart SecondItem
    And I clicked to CheckOutCartButton
    And I clicked to CheckOutButton
    And I entered FirstName
    And I entered LastName
    And I entered PostCode
    And I clicked continue button
    And I clicked Finish button
    And I verified order successful
    Then I had close the SwagApp
```

```
Feature:04 SwagLab: Remove product from cart and go back to HomePage

  Scenario:04_SwagLabs remove item from cart and back to Home page
    Given Iam on the SwagLab homepage
    When I enter SwagLabUsername
    Then I enter SwagLabPassword
    And I click on SwagLogin button
    And I verify SwagLab HomePage
    And I click addCartButton swagLab
    And I click swag CheckOutCartButton
    And I click remove Button
    And I click ContinueShopping Button
    And I check shopping page for SwagLab
    And I have click close the SwagApp
```

```
Feature:05 SwagLabs verify the checkout page

  Scenario:05_SwagLabs verify the checkout page
    Given I land on SwagLab home page
    When I am submitting Username
    Then I am submitting Password
    And I submit click to Login button
    And I check the swag lab HomePage
    And I navigate to add Cart Button
    And I navigate to CheckOutCartButton
    And I navigating to CheckOutButton
    And I add FirstName to the field
    And I add LastName to the field
    And I add PostCode to the field
    And I navigating to continue button
    And I have verified the checkout information
    Then I logoff and close the SwagApp
```

Steps to run this test Framework:

1. Clone the Repo from GitHub
2. Launch the IntelliJ Idea (Recommended)
3. Open the Source code from where it has been cloned.
4. Let all the maven dependencies and indexes get updated (it may take 5 to 8 minutes to get update)
5. Go to File from IntelliJ idea > Project structure > window Opens and make sure that you have selected right SDK and JDK version 11 and above.
6. Go to Build tab on the menu in IntelliJ idea and Click on Build Project, wait for all the indexes to be updated.
7. Go to Build tab on the menu in IntelliJ idea and Click on Re-Build Project, wait for all the indexes to be updated.
8. Then go to the project structure - src/test/Java /Com /UtilClass/TestRunner.java.
9. Click on Run button from Test Runner class and Test should execute all the 5 test cases automated.

Note: Test reports might not be clear from this document as I have captured a screenshot, so I have shared separate images for the overall project structure and reports for this test automation.