International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 06 | June - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Power of Automation Testing

Aman Simon Santhosh
Electronics and communication
R.V College of Engineering
Bangalore, India
amansimons.ec19@rvce.edu.in

Varun M
Electronics and communication
R.V College of Engineering
Bangalore, India
varunm.ec19@rvce.edu.in

Karan Rahul Shah
Electronics and communication
R.V College of Engineering
Bangalore, India
karanrahulshah.ec19@rvce.edu.in

Dr Kiran V
Associate Professor
R.V College of Engineering
Bangalore, India
kiranv@rvce.edu.in

***Abstract*—Testing is a very fundamental concept of analysing all the functionalities to ensure the smooth running of the software. The most Basic form of testing is manual testing where a user will test all the functionalities but this is very time - consuming. This is very automation testing plays a big role, it will test multiple functionalities at the same time including stress test and volume testing with the help of a framework. Libraries such as the cucumber BDD framework are used to make the coding of the framework less tedious and more understandable for people from non-coding backgrounds. Cucumber is written gherkin language which is easily understandable. The framework has the capability of testing multiple parameters of the interface and gives case-by-case output to see which parameters have failed, hence helping fix these parameters. The implementation of application programming interfaces and web services in the current framework can help increase its volume and productivity of testing.**

***Index Terms*—Behaviour - driven development (BDD), User Interface (UI), Integrated Development Environment (IDE).**

I. Introduction

The creation of high-quality software is the ultimate goal of software development. Software of superior quality has attributes including affordability, dependability, and user satisfaction. The act of testing involves running a program with the goal of identifying errors. This is a vital and necessary step in the software development process to find all the faults. Successful and thorough testing lowers the system cost. To undertake testing activities, software development companies hire testing and quality assurance specialists. Software testing is the process of assessing and confirming that a software application or product performs as intended. Bugs can be avoided, development expenses are decreased, and performance is increased. It is an essential step in the life cycle of software development that guarantees the capability of the software product being created.

Overview of Testing

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs, and improving performance. It is a critical process in the software development life cycle that ensures the quality of the software product being developed.

- **White Box Testing:** White box testing is a kind of software testing that focuses on examining a system's internal architecture and implementation data. Gaining access to the software's supply code, structure, and design is required for what is also known as structural testing or glass box testing.
- **Black Box Testing:** On the other hand, black box testing is a software testing approach that only considers the device's exterior behaviour, paying no attention to its internal design or implementation specifics. Black-box testing is a technique for software testing that looks at an application's functioning without peeping into its internal workings or structures. Black container testing is performed by testers who do not have access to the source code and who view the system as black where they examine inputs and outputs to assess the machine's capabilities, usability, and compliance with specifications. • **Grey Box Testing:** White-box testing and black-box testing are combined in grey-box testing. The purpose of this testing is to look for any bugs caused by inappropriate application usage or structure. Without having knowledge of each detailed aspect of the internal implementation, this method enables the discovery of hidden defects, testing of specific code pathways, and verification of the device's average operation.

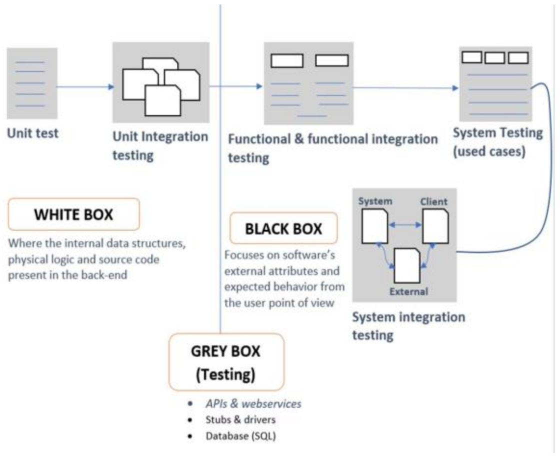

Unit test
Unit Integration testing
Functional & functional integration testing
System Testing (used cases)
WHITE BOX
Where the internal data structures, physical logic and source code present in the back-end
BLACK BOX
Focuses on software's external attributes and expected behavior from the user point of view
System
Client
External
System integration testing
GREY BOX (Testing)
APIs & webservices
Stubs & drivers
Database (SQL)

Fig. 1. Overview of Testing

The term "test" refers to how a program reacts to any input. A program must test both valid and invalid input. There are two ways to conduct testing activities: Manual testing & automation testing. Any kind of software testing can be carried out either manually or automatically utilizing the right tools.

A. Manual Testing

Manual testing is the process of verifying and validating software applications or systems through manual interactions, without the use of automated tools or scripts. In order to find bugs, flaws, or inconsistencies, human testers run test cases, replicate user behaviors, and analyze the software's performance. When performing manual testing, a tester frequently follows a written test plan that guides them through a number of significant test cases. In software testing, a test case is a list of conditions created for a specific application that the tester runs to confirm that the software applications are functioning properly. Defects, errors & bugs in the domain of usability testing and GUI testing can be found via manual testing. Before its testing can be automated, a new application must first be manually tested. Although manual testing is more labor-intensive, it is essential to determine the viability of automation. The use of any testing tool is not necessary for manual testing.

Problems with manual testing

Time-Consuming: Test case execution is laborious and time-consuming since it is done by human resources.

Huge investment in human resources: Since manual testing requires the manual execution of test cases, more testers are needed.

Less reliable: Because human error might occur during manual testing, tests may not be carried out accurately every time.

Non-programmable: Complex tests that retrieve concealed information cannot be written using programming.

Manual testing is prone to inaccuracy and can become boring.

B. Automation Testing

Automation testing is the process of performing test cases without the need for manual testing. To develop and run test cases and compare the actual result with the projected result, specialized software is used. Once tests are automated, they may be easily and often executed. In order to run test cases by computers with the least amount of human involvement and oversight, test scripts must be created using scripting languages like Python and JavaScript. It is possible to automate test development and design to save money and minimize human work.

Benefits of automation testing

Fast: It moves more quickly than manual testing.

Cost-effective: Because automation tools are used to run test cases, fewer testers are needed for automation testing.

Repeatable: Using testing tools, the same test scenario (record and playback) may be repeated [9].

Reusable: Test suites may be applied to several software versions.

Programmable: Testers can create complex tests that reveal hidden data.

Reliability: Automation tests are more dependable since they always carry out the exact same task.

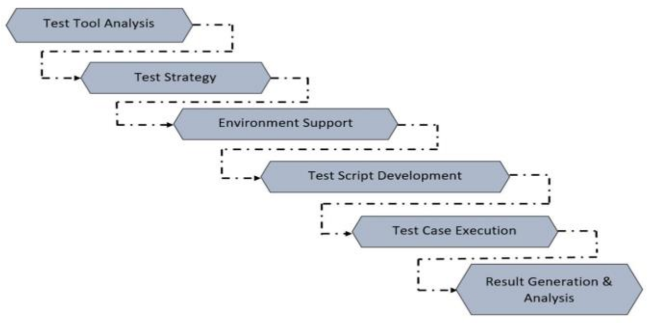
II. Background of Testing tools for WEB/UI Automation

A. Testing Framework

A set of instructions or standards used for developing and designing test cases is known as a testing framework. A framework is made up of a variety of procedures and equipment intended to aid QA specialists in conducting tests more quickly. A testing framework that involves planning, organizing, and carrying out automated tests for online applications is provided by a set of tools and libraries for automation testing.

A Web/UI Automation Framework is hard and fast of pointers, equipment, and practices that offer a dependent method to automate the testing and interplay with internet consumer interfaces.

B. Process of Automation testing



1) **Test Tool Analysis:** Any process begins with a definition, thus test engineers for test automation should specify the desired result before entering.
2) **Test Strategy:** Choose the test automation framework that will be applied. A framework provides a methodical way to arrange test scripts, handle test data management, manage test execution, and produce test results.
3) **Environment Support:** Determine the hardware, software, and network configurations required to recreate the production environment by understanding the requirements of the application or system under test.
4) **Test Script Development :** Using the chosen automation tool or framework, write automation test scripts. Create scripts that simulate user behavior, system interactions, and validation of expected results.
5) **Test Execution:** Run the automated test scripts against the application or system under test.
6) **Result Generation & Analysis:** Analyze the test results generated by the selenium IDE automation tool.

C. JUnit

JUnit is a popular open-source testing framework developed primarily for unit testing Java applications. It offers a quick and effective way to create and run repeatable, automated tests for single pieces of code, such as methods, classes, or components.

By enabling developers to design test cases, run them, and evaluate the results, JUnit aids in confirming the behavior and validity of the code that is being tested. It offers a collection of tools, assertions, and annotations that make testing easier and help developers create test cases more successfully. Annotations are used by JUnit to identify methods as test cases, setup or teardown procedures, and to define the order in which tests should be run or expected errors. The annotations @Test, @Before, @After, @BeforeClass, and @AfterClass are a few examples of those that are often used.

D. Cucumber

A cucumber is a popular tool that facilitates BehaviorDriven Development(BDD) by allowing developers to write executable specifications in a natural language format. It promotes collaboration between stakeholders, such as developers, testers, and business analysts, by providing a common language to describe and validate application behavior. Cucumber is based on the Gherkin language, which uses a natural language syntax to describe the behavior of a system in terms of its features and scenarios.

The basic syntax of Gherkin includes the following keywords:

Given, When, Then: These keywords are used to define the steps of a scenario.

And, But: These keywords can be used to add additional steps to a scenario or to combine multiple Given, When, or Then steps together.

E. Selenium

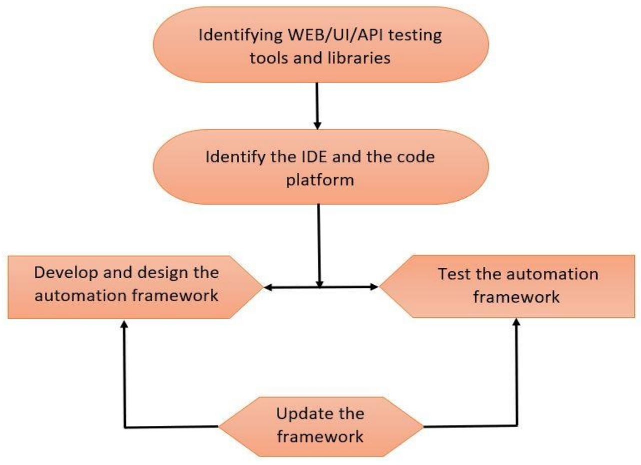
A portable software testing tool called Selenium is used for automation testing. It is a framework made up of several web application testing tools. Selenium gives off a Selenium IDE that acts as a record/playback tool for creating tests without having to learn a scripting language. In order to construct test cases in a variety of well-liked programming languages, such as C#, Java, Groovy, Perl, PHP, Python, and Ruby, it has a test domain-specific language (Selenese). The majority of contemporary web browsers may then be used to perform all these prepared test cases. Any operating system platform, including Windows, Linux, and iOS, may run Selenium. It is open-source software. Selenium is a suite of tools for web automation testing.

Selenium IDE

An integrated development environment for Selenium scripts is called Selenium IDE. Users are able to record, edit, and debug tests with this Chrome plugin. The Selenium IDE is a full-featured IDE and not just a recording tool. Comparable commercial solutions include QTP, Silk Test, Test Partner, and Selenium IDE (Integrated Development Environment).

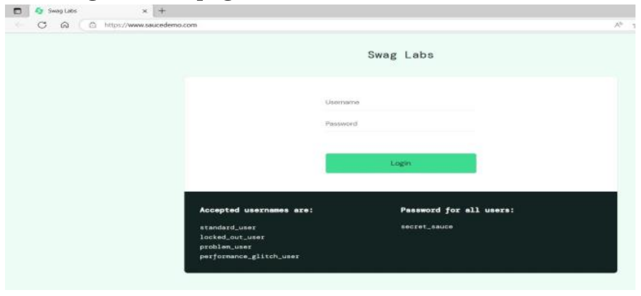
III. METHODOLOGY

The goal is to design a reusable Framework that can be used precisely over any other application without spending additional time and is independent of the application. The Framework's capabilities, component parts and their coupling relationship must all be specified in order for the Framework to function as a whole.


Identifying WEB/UI/API testing tools and libraries
Identify the IDE and the code platform
Develop and design the automation framework
Test the automation framework
Update the framework

IV. STEPS FOR IMPLEMENTATION OF TEST CASESUSING INTELLIJ IDE

Let us consider a website for entering a username and password to login. For this, we have to design a framework for automating the web page.

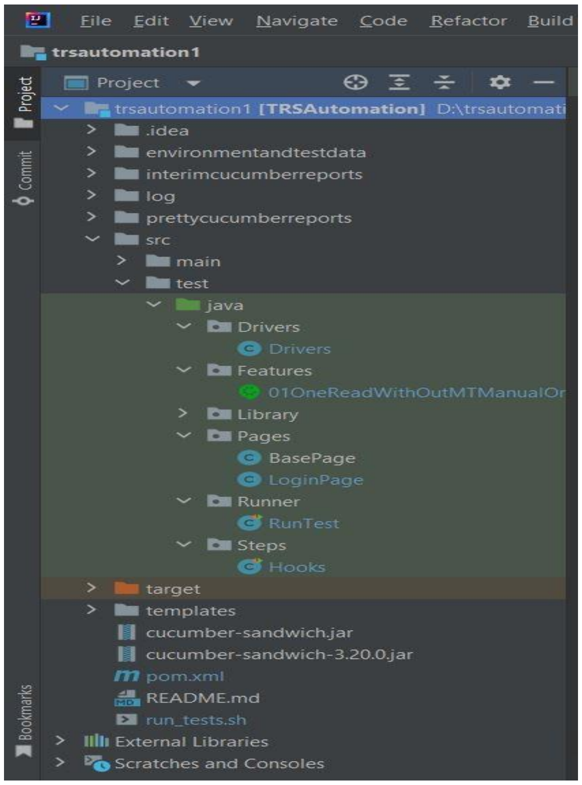


Accepted usernames are:
standard_user
locked_out_user
problem_user
performance_glitch_user

Password for all users:
secret_sauce

A. Development of framework on IntelliJ IDE

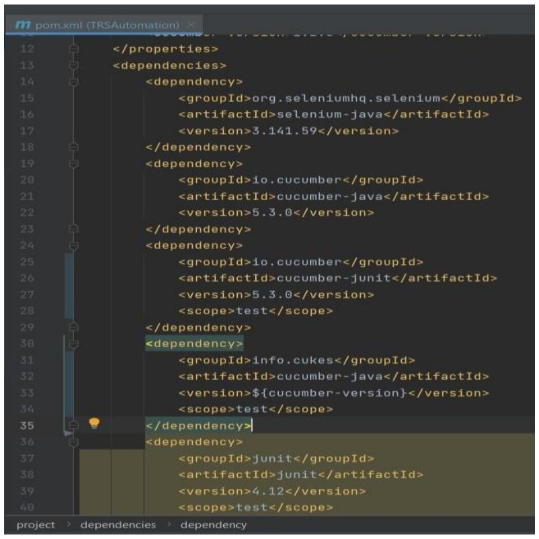
The structure of the framework designed for automation testing is shown in the below figure. The complete framework is written under src/test/java for automation testing. The flow of the framework is shown in the below snapshot.



```
trsautomation1
Project
trsautomation1 [TRSAutomation] D:\trsautomati
  > .idea
  > environmentandtestdata
  > interimcucumberreports
  > log
  > prettycucumberreports
  v src
    > main
    v test
      v java
        v Drivers
            Drivers
        v Features
            01OneReadWithOutMTManualOr
        > Library
        v Pages
            BasePage
            LoginPage
        v Runner
            RunTest
        v Steps
            Hooks
  > target
  > templates
    cucumber-sandwich.jar
    cucumber-sandwich-3.20.0.jar
    pom.xml
    README.md
    run_tests.sh
> External Libraries
> Scratches and Consoles
```

1)pom.xml file to add dependencies

Before starting to write the test cases for the webpage, adding the correct dependencies is mandatory to develop an automation framework.



pom.xml (TRSAutomation)

```
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>3.141.59</version>
        </dependency>
        <dependency>
            <groupId>io.cucumber</groupId>
            <artifactId>cucumber-java</artifactId>
            <version>5.3.0</version>
        </dependency>
        <dependency>
            <groupId>io.cucumber</groupId>
            <artifactId>cucumber-junit</artifactId>
            <version>5.3.0</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-java</artifactId>
            <version>${cucumber-version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
```

Dependencies to be added:

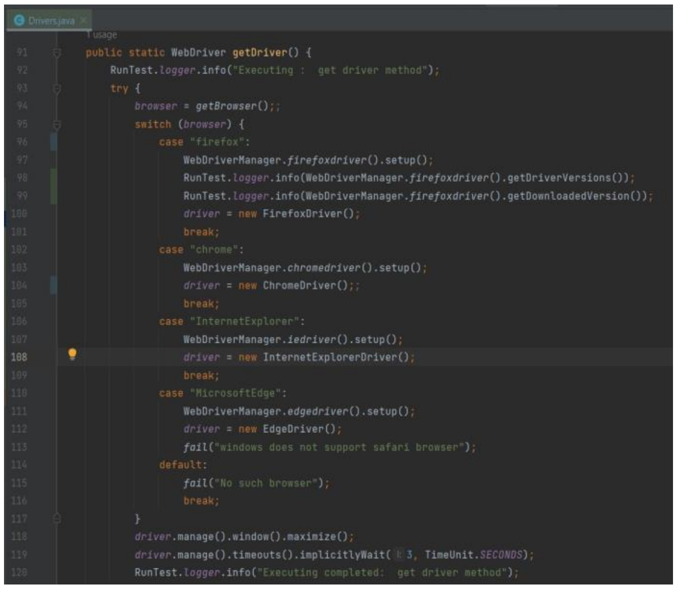
- selenium-java
- cucumber-java
- java-client
- junit & cucumber-junit
- cucumber-runner
- log4j & log4j-core

Build in functions to be added:

- maven-compiler-plugin
- maven-surefire-plugin
- surefire-junit47

2) Add the drivers and Feature file

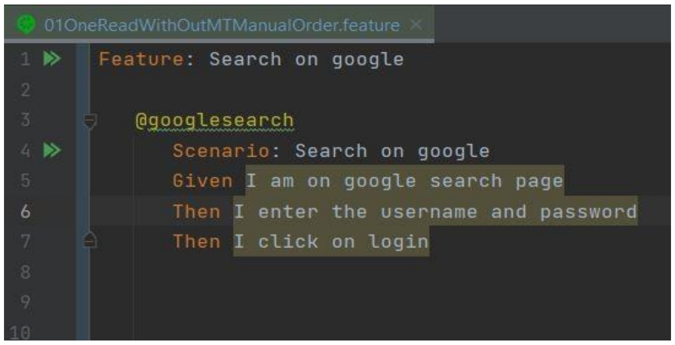
- Write a code to add the drivers to get the execution on the browser. Using the switch case write a code to select the browser that has to be executed in i.e chrome, fire fox, Internet Explorer, Microsoft Edge, etc.



Drivers.java

```
public static WebDriver getDriver() {
    RunTest.logger.info("Executing :  get driver method");
    try {
        browser = getBrowser();;
        switch (browser) {
            case "firefox":
                WebDriverManager.firefoxdriver().setup();
                RunTest.logger.info(WebDriverManager.firefoxdriver().getDriverVersions());
                RunTest.logger.info(WebDriverManager.firefoxdriver().getDownloadedVersion());
                driver = new FirefoxDriver();
                break;
            case "chrome":
                WebDriverManager.chromedriver().setup();
                driver = new ChromeDriver();;
                break;
            case "InternetExplorer":
                WebDriverManager.iedriver().setup();
                driver = new InternetExplorerDriver();
                break;
            case "MicrosoftEdge":
                WebDriverManager.edgedriver().setup();
                driver = new EdgeDriver();
                fail("windows does not support safari browser");
            default:
                fail("No such browser");
                break;
        }
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        RunTest.logger.info("Executing completed:  get driver method");
```

- Under src-test-java Develop a feature file step by step that has to be done for automation using cucumber language.

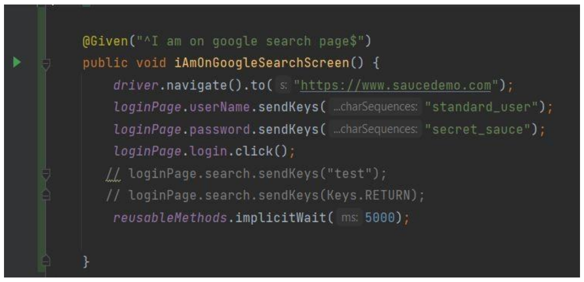


01OneReadWithOutMTManualOrder.feature

```
Feature: Search on google

  @googlesearch
    Scenario: Search on google
    Given I am on google search page
    Then I enter the username and password
    Then I click on login
```

3) Steps File

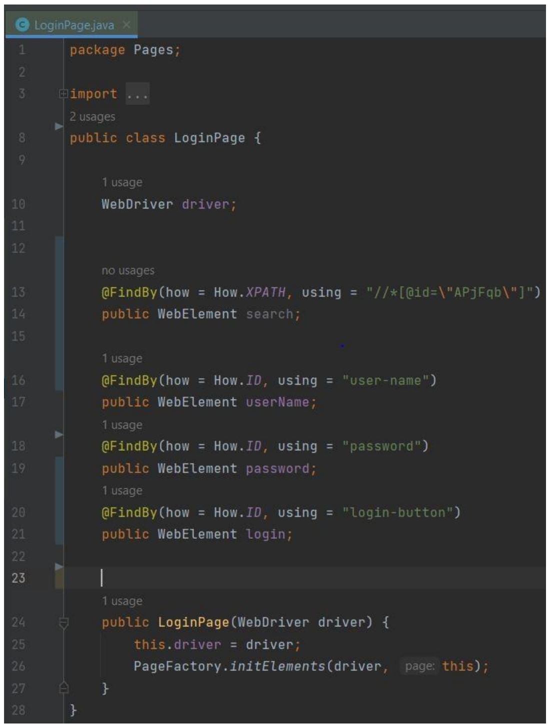
Create a step file for the feature file that has been written in the cucumber language. Paste the URL of the website and the Xpath of the username and password that is been automated.



```java
@Given("^I am on google search page$")
public void iAmOnGoogleSearchScreen() {
    driver.navigate().to("https://www.saucedemo.com");
    loginPage.userName.sendKeys("standard_user");
    loginPage.password.sendKeys("secret_sauce");
    loginPage.login.click();
   // loginPage.search.sendKeys("test");
   // loginPage.search.sendKeys(Keys.RETURN);
    reusableMethods.implicitWait(5000);

}
```

4) Login Page

Write the test cases that have to be automated in Java. Copy the Xpath of the webpage and paste it to its respective test case.



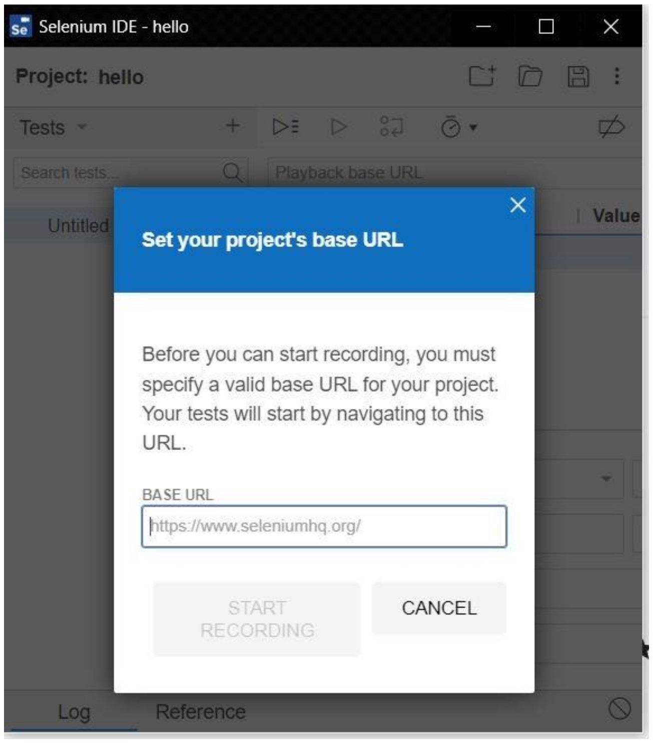
```java
package Pages;

import ...
2 usages
public class LoginPage {

    1 usage
    WebDriver driver;


    no usages
    @FindBy(how = How.XPATH, using = "//*[@id=\"APjFqb\"]")
    public WebElement search;

    1 usage
    @FindBy(how = How.ID, using = "user-name")
    public WebElement userName;
    1 usage
    @FindBy(how = How.ID, using = "password")
    public WebElement password;
    1 usage
    @FindBy(how = How.ID, using = "login-button")
    public WebElement login;

    
    1 usage
    public LoginPage(WebDriver driver) {
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }
}
```

V. Web Automation using Selenium IDE

A. Process to activate selenium IDE

Chrome has an add-on functionality called Selenium IDE. The procedures should be performed in order to activate selenium. Choose Tools from the menu bar in Firefox. Choosing selenium Ide in the tools popup. The screen in the figure will now show up. A recording button is located on the right side of the screen.

The URL of the website that has to be automated has to be pasted in Selenium IDE.



Set your project's base URL

Before you can start recording, you must specify a valid base URL for your project. Your tests will start by navigating to this URL.

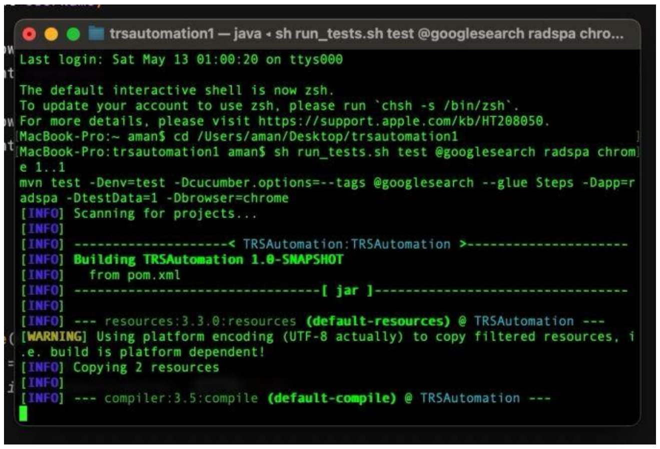
BASE URL: https://www.seleniumhq.org/

START RECORDING   CANCEL

VI. Results

A. Excecution of IntelliJ IDE

- Running the command ”sh run tests.sh test @googlesearch radspa chrome 1..1” on the terminal to execute the test cases.

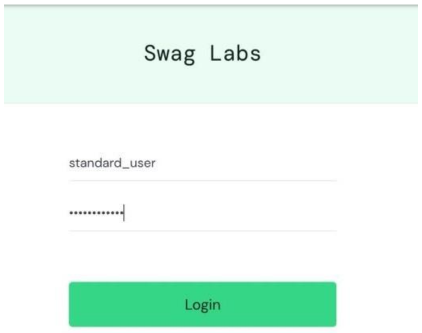


```
trsautomation1 — java ◂ sh run_tests.sh test @googlesearch radspa chro...
Last login: Sat May 13 01:00:20 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro:~ aman$ cd /Users/aman/Desktop/trsautomation1
MacBook-Pro:trsautomation1 aman$ sh run_tests.sh test @googlesearch radspa chrom
e 1..1
mvn test -Denv=test -Dcucumber.options=--tags @googlesearch --glue Steps -Dapp=r
adspa -DtestData=1 -Dbrowser=chrome
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------< TRSAutomation:TRSAutomation >--------------------
[INFO] Building TRSAutomation 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- resources:3.3.0:resources (default-resources) @ TRSAutomation ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i
.e. build is platform dependent!
[INFO] Copying 2 resources
[INFO]
[INFO] --- compiler:3.5:compile (default-compile) @ TRSAutomation ---
```

Install java-8 and Maven to the terminal to run the test cases. Maven is an effective construct automation and project control device mostly used for Java-primarly based projects. Maven plugins amplify the capability of Maven by way of imparting additional desires and obligations. Maven binaries are often added to your system’s PATH environment variable when it is installed on your computer.

As a result, you may immediately execute Maven commands from the console.

- The below figure shows the web page that is automated.

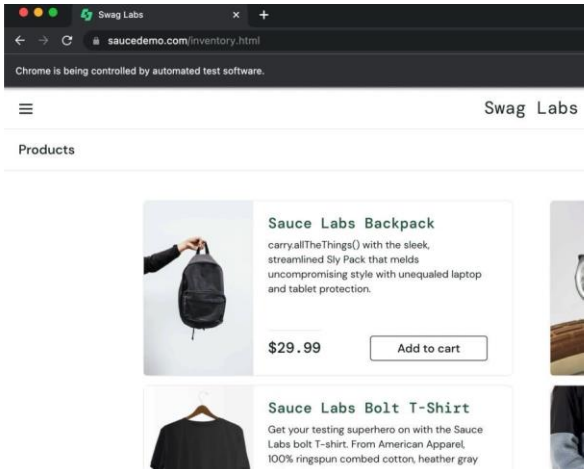

Swag Labs

standard_user

············

Login

- On clicking on login the page gets logged in to select the products.


Swag Labs

saucedemo.com/inventory.html

Chrome is being controlled by automated test software.

Swag Labs

Products

Sauce Labs Backpack

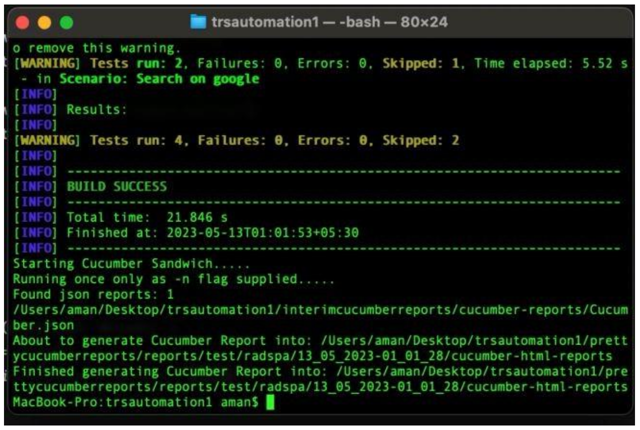
carry.allTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection.

$29.99 Add to cart

Sauce Labs Bolt T-Shirt

Get your testing superhero on with the Sauce Labs bolt T-shirt. From American Apparel, 100% ringspun combed cotton, heather gray

- Finally the terminal window shows the test cases that are passed while automating the web page. All the test cases are passed with zero failures and zero errors


trsautomation1 — -bash — 80×24

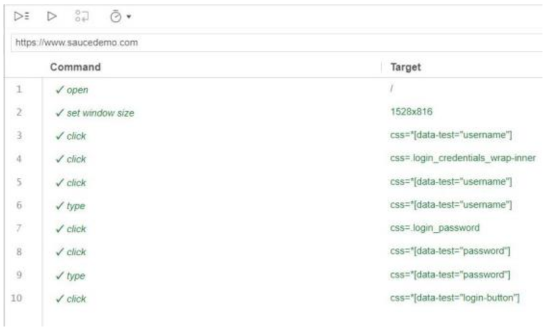
```
o remove this warning.
[WARNING] Tests run: 2, Failures: 0, Errors: 0, Skipped: 1, Time elapsed: 5.52 s
 - in Scenario: Search on google
[INFO]
[INFO] Results:
[INFO]
[WARNING] Tests run: 4, Failures: 0, Errors: 0, Skipped: 2
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  21.846 s
[INFO] Finished at: 2023-05-13T01:01:53+05:30
[INFO] ------------------------------------------------------------------------
Starting Cucumber Sandwich.....
Running once only as -n flag supplied.....
Found json reports: 1
/Users/aman/Desktop/trsautomation1/interimcucumberreports/cucumber-reports/Cucum
ber.json
About to generate Cucumber Report into: /Users/aman/Desktop/trsautomation1/prett
ycucumberreports/reports/test/radspa/13_05_2023-01_01_28/cucumber-html-reports
Finished generating Cucumber Report into: /Users/aman/Desktop/trsautomation1/pre
ttycucumberreports/reports/test/radspa/13_05_2023-01_01_28/cucumber-html-reports
MacBook-Pro:trsautomation1 aman$ 
```

All the test cases are passed with zero failures and zero errors.

B. Execution on Selenium IDE

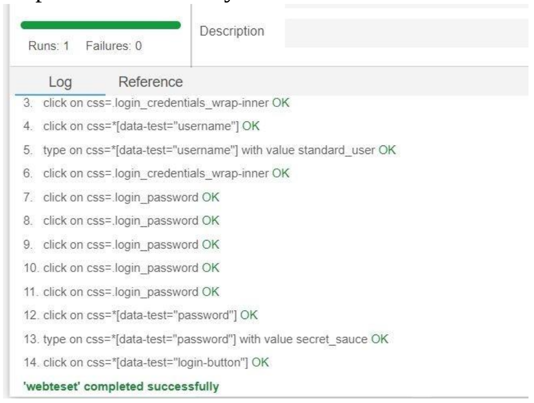
There is a red color recording button on the right side of the selenium environment. Once activating recording mode, then what so ever data is entered in the web page will automatically be recorded in the Selenium IDE.

- When the current web page is tested manually by typing the username and password to log in to the page. Thus it shows the step-by-step process involved while testing.


https://www.saucedemo.com

	Command	Target
1	✓ open	/
2	✓ set window size	1528x816
3	✓ click	css="[data-test="username"]
4	✓ click	css=.login_credentials_wrap-inner
5	✓ click	css="[data-test="username"]
6	✓ type	css="[data-test="username"]
7	✓ click	css=.login_password
8	✓ click	css="[data-test="password"]
9	✓ type	css="[data-test="password"]
10	✓ click	css="[data-test="login-button"]

- Stop the recording by clicking the red button after testing the page manually. When the current page is running on Selenium IDE, the log window shows the page test is completed successfully.

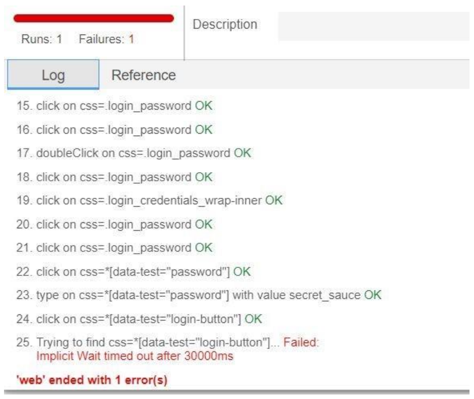

Runs: 1 Failures: 0

Description

Log Reference

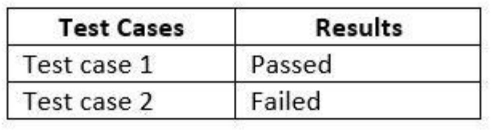
```
3. click on css=.login_credentials_wrap-inner OK
4. click on css="[data-test="username"] OK
5. type on css="[data-test="username"] with value standard_user OK
6. click on css=.login_credentials_wrap-inner OK
7. click on css=.login_password OK
8. click on css=.login_password OK
9. click on css=.login_password OK
10. click on css=.login_password OK
11. click on css=.login_password OK
12. click on css="[data-test="password"] OK
13. type on css="[data-test="password"] with value secret_sauce OK
14. click on css="[data-test="login-button"] OK
'webteset' completed successfully
```

- The test cases that failed while typing the wrong username or password.



```
Runs: 1    Failures: 1

15. click on css=.login_password OK
16. click on css=.login_password OK
17. doubleClick on css=.login_password OK
18. click on css=.login_password OK
19. click on css=.login_credentials_wrap-inner OK
20. click on css=.login_password OK
21. click on css=.login_password OK
22. click on css=*[data-test="password"] OK
23. type on css=*[data-test="password"] with value secret_sauce OK
24. click on css=*[data-test="login-button"] OK
25. Trying to find css=*[data-test="login-button"]... Failed:
    Implicit Wait timed out after 30000ms
'web' ended with 1 error(s)
```

- Below the table shows results carried out while testing the webpage on Selenium IDE.



Test Cases	Results
Test case 1	Passed
Test case 2	Failed

Test case 1 passed when the correct username and password is typed.

Test case 2 failed when anyone of username or password is typed incorrectly.

VII. Conclusion

Frameworks for web automation have completely changed and streamlined how firms operate. To automate numerous operations connected to web development, testing, and deployment, these frameworks provide a complete collection of tools and functionalities. The current status of web automation frameworks has been thoroughly evaluated, and it is clear that these frameworks offer a wide range of advantages, such as greater quality assurance, increased productivity, and increased efficiency.

The huge increase in productivity that web automation frameworks provide is one of their main benefits. Developers may focus on more important areas of their projects by automating repetitive operations like form filling, data extraction, and content management. Faster development cycles and a shorter time to market provide businesses an advantage over rivals.

Web automation frameworks also help to increase productivity throughout the test- ing and development phases. They offer a standardized method for developing websites, enabling programmers to adhere to best practices and keep their code consistent. These frameworks also provide built-in reporting, logging, and error-handling functions that help find problems quickly and get them fixed.

Another area where web automation frameworks shine is quality control. Frameworks offer thorough testing coverage by automating test scenarios, ensuring that applications work as intended across a range of browsers, devices, and platforms. This lowers the possibility of defects and compatibility problems, resulting in more dependable and robust online apps. The general stability and performance of online applications are improved through continuous integration and regression testing, which are both made possible by automation frameworks

In conclusion, web automation frameworks have established themselves as crucial solutions for companies looking to streamline their online development procedures. These frameworks give development teams the tools they need to produce higher-quality web apps faster by enhancing productivity, efficiency, and quality assurance. Web automation frameworks will surely be essential to advancing innovation and influencing the direction of web development as technology develops further. Organizations may stay ahead in the competitive and dynamic digital landscape by embracing and utilizing these frameworks.

References

[1] A. Bezbaruah, B. Pratap and S. B. Hake, ”Automation of Tests and Comparative Analysis between Manual and Automated testing,” 2020 IEEE Students Conference on Engineering & Systems (SCES), Prayagraj, India, 2020, pp. 1-5, doi: 10.1109/SCES50439.2020.9236748.

[2] R. M. Sharma, “Quantitative Analysis of Automation and Manual Testing,” vol. 4, no. 1, pp. 252–257, 2014.

[3] R. Broer Bahaweres et al., ”Behavior-driven development (BDD) Cucumber Katalon for Automation GUI testing case CURA and Swag Labs,” 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), Jakarta, Indonesia, 2020, pp. 87-92, doi: 10.1109/ICIMCIS51567.2020.9354325.

[4] Wandan, Zeng, Jiang Ningkang, and Zhou Xubo. ”Design and Implementation of a Web Application Automation Testing Framework.” In 2009 Ninth International Conference on Hybrid Intelligent Systems, vol. 2, pp. 316-318. IEEE, 2009.

[5] Uppal, Nidhika, and Vinay Chopra. ”Design and implementation in selenium ide with web driver.” International Journal of Computer Application 46.12 (2012): 8-11.

[6] D. Winkler, R. Hametner, T. Ostreicher and S. Biffl, ”A framework for¨ automated testing of automation systems,” 2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010), Bilbao, Spain, 2010, pp. 1-4, doi: 10.1109/ETFA.2010.5641264.

[7] Z. Kurbatova, Y. Golubev, V. Kovalenko and T. Bryksin, ”The IntelliJ Platform: A Framework for Building Plugins and Mining Software Data,” 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW), Melbourne, Australia, 2021, pp. 14-17, doi: 10.1109/ASEW52652.2021.00016.

[8] E. Pelivani and B. Cico, ”A comparative study of automation testing tools for web applications,” 2021 10th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2021, pp. 1-6, doi: 10.1109/MECO52532.2021.9460242.

[9] F. Wang and W. Du, ”A Test Automation Framework Based on WEB,” 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, Shanghai, China, 2012, pp. 683-687, doi: 10.1109/ICIS.2012.21.10.1109/MECO52532.2021.9460242.

[10] A. Holmes and M. Kellogg, ”Automating functional tests using Selenium,” AGILE 2006 (AGILE’06), Minneapolis, MN, USA, 2006, pp. 6 pp.-275, doi: 10.1109/AGILE.2006.19.

[11] Ezhuthachan, S.D. and Tailor, J.K., 2022. Test Case to Test Automation with Selenium IDE: a Case Study. Compendium of Management Case Studies, 263, p.2022.

[12] Kurbatova, Zarina, et al. ”The intellij platform: a framework for building plugins and mining software data.” 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW). IEEE, 2021.

[13] Kozak, Ilona, and Andrii Berko. ”Three-module framework for automated software testing.” 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT). IEEE, 2022.

[14] Z. Sun, Y. Zhang and Y. Yan, ”A Web Testing Platform Based on Hybrid Automated Testing Framework,” 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chengdu, China, 2019, pp. 689-692, doi: 10.1109/IAEAC47372.2019.8997684.

[15] Z. Wandan, J. Ningkang and Z. Xubo, ”Design and Implementation of a Web Application Automation Testing Framework,” 2009 Ninth International Conference on Hybrid Intelligent Systems, Shenyang, China, 2009, pp. 316-318, doi: 10.1109/HIS.2009.175.