

# INDEX

S.NO	DATE	Name of the Experiment	Page No.	Staff Sign
1		Develop the test plan for testing an e-commerce web/mobile application (www.amazon.in).		
2		Design the test cases for testing the e-commerce application		
3		Test the e-commerce application and report the defects in it.		
4		Develop the test plan and design the test cases for an inventory control system		
5		Execute the test cases against a client server or desktop application and identify the defects.		
6		Test the performance of the e-commerce application.		
7		Automate the testing of e-commerce applications using Selenium.		
8		Integrate TestNG with the above test automation.		
9		Build a data-driven framework using Selenium and TestNG		

Ex.no : 01

Date :

Develop the test plan for testing an e-commerce web/mobile application  
([www.amazon.in](http://www.amazon.in)).

**Aim :**

To Develop the test plan for testing an e-commerce web/mobile application  
([www.amazon.in](http://www.amazon.in)).

**Test plan :**

**Scope**

The scope of this test plan includes testing the core functionalities of the e-commerce application, such as browsing products, adding items to the cart, placing orders, and managing user accounts. It also covers testing across different platforms, including web and mobile.

**Test Objectives**

- 1) Validate the functionality of the e-commerce application.
- 2) Verify that the application is user-friendly and provides a smooth shopping experience.
- 3) Ensure that the application is secure and protects user data.
- 4) Test the application's compatibility across different browsers and devices.
- 5) Evaluate the performance and scalability of the application under various load conditions.
- 6) Identify and report any defects or issues found during testing.

**Test Environment**

- 1) Web browsers: Chrome, Firefox, Safari, and Edge.
- 2) Mobile devices: iOS and Android.
- 3) Test management tool: Jira or any other preferred tool.
- 4) Test automation tool: Selenium WebDriver or any other preferred tool.
- 5) Load testing tool: JMeter or any other preferred tool.

**Test Approach**

**1) Requirements Analysis:**

- Review the functional and non-functional requirements of the e-commerce application.
- Identify testable features and prioritize them based on criticality.

**2) Test Design:**

- Create test scenarios and test cases for each identified feature.
- Include positive and negative test cases to cover different scenarios.
- Define test data and test environment setup requirements.

**3) Test Execution:**

- Execute test cases manually or using test automation tools.
- Log defects in the test management tool and track their status.
- Perform regression testing after each bug fix or application update.

**4) Performance Testing:**

- Design and execute performance tests to evaluate the application's response time, throughput, and scalability.

- Simulate different user loads and monitor system resources during testing.
- Identify performance bottlenecks and suggest improvements if needed.

#### **5) Security Testing:**

- Conduct security testing to identify vulnerabilities and ensure the application protects user data.
- Test for common security issues like SQL injection, cross-site scripting (XSS), and authentication vulnerabilities.
- Implement security best practices and follow industry standards.

#### **6) Compatibility Testing:**

- Test the application on different browsers, versions, and mobile devices.
- Verify that the application renders correctly and functions as expected across various platforms.

#### **Test Deliverables**

- 1) Test plan document.
- 2) Test scenarios and test cases.
- 3) Test execution reports.
- 4) Defect reports with severity and priority.
- 5) Performance test reports.
- 6) Security test reports.

#### **Result :**

Thus the Test Plan for Testing an E-commerce Web/Mobile Application, the goal is to ensure that the application functions as intended, meets user requirements, and provides a seamless shopping experience.

Ex.no : 02

Date :

Design the test cases for testing the e-commerce application

**Aim :**

To Design the test cases for testing the e-commerce application

**Types of Ecommerce Testing :**

Ecommerce websites function like any other web/mobile website. Hence it undergoes various types of testing, such as:

- **Functional Testing:** If the tester wants to check if the checkout feature is working on the website and scoped only to that feature, it is called functional testing.
- **Usability Testing:** When a customer wants to buy an item, searching, adding to the cart, and making payment should be simple and easy. This is where usability testing kicks in.
- **Performance Testing:** When a single user is trying to access an ecommerce website, then it should be extremely responsive.
- **Mobile Application Testing:** As there is a lot of penetration into the mobile space, many users access the e-commerce websites on mobiles, So the website must have a good user experience across different mobile.

**General Ecommerce Test Cases**

1. The user should be able to navigate to all the pages in the website
2. There should be a fallback page for any page load errors
3. Verify that all the links and banners work properly
4. Search results should be displayed with the most relevant item being shown first
5. All data related to the product – title, price, images, and description are all visible clearly
6. Maintain a session for each user and test verify the session times out after a while.

**Login/Registration test cases for Online Shopping Website**

The image shows the Amazon India login page. At the top is the 'amazon.in' logo. Below it is the heading 'Login'. There are two input fields: 'Email or mobile phone number' and 'Password'. To the right of the password field is a link that says 'Forgot Password'. Below the password field is an orange 'Login' button. Underneath the button is a checkbox labeled 'Keep me signed in.' followed by a link 'Details' with a dropdown arrow. At the bottom, there is a link 'New to Amazon?' and a button that says 'Create your Amazon account'.

- Test for valid username and password
- Test “Forgot Password” and Reset Password functionality

- Validate If user is registered or not, and if not, provide an option to create an account
- Show Login screen by default for registered user
- Test that all fields are mandatory

### Seller Product Creation Test Cases

For sellers who want to create their catalog, they should be able to add their products, and this flow should be tested.

- Test that authenticated sellers can access authorized product creation panels under authorized categories.
- Test the maximum product creation limit to avoid adding more products to the catalog.
- Test that seller's products are visible after some time
- Test the product creation process is working fine for the seller
- Test that there are no duplicate products
- Search Bar Test Cases for Online Shopping Website.

### Search Bar Test Cases for Online Shopping Website



- Test what parameters the search is based on – for example, product name, brand name, category, etc.
- Test if the search results are relevant
- Number of results to be displayed per page
- Test whether the search API is being called at every keystroke. (This isn't recommended, as it would unnecessarily cause multiple API calls to the database)

### Filter Results

#### Avg. Customer Review



#### Amazon Fashion

- ☐ Our Brands
- ☐ Premium Brands
- ☐ Popular Brands

#### Brand

- ☐ Under Armour
- ☐ Simple Joys by Carter's
- ☐ Disney
- ☐ Amazon Essentials
- ☐ Fruit of the Loom
- ☐ Hanes
- ☐ Resinta

- Test if the user can filter based on all the parameters on the page

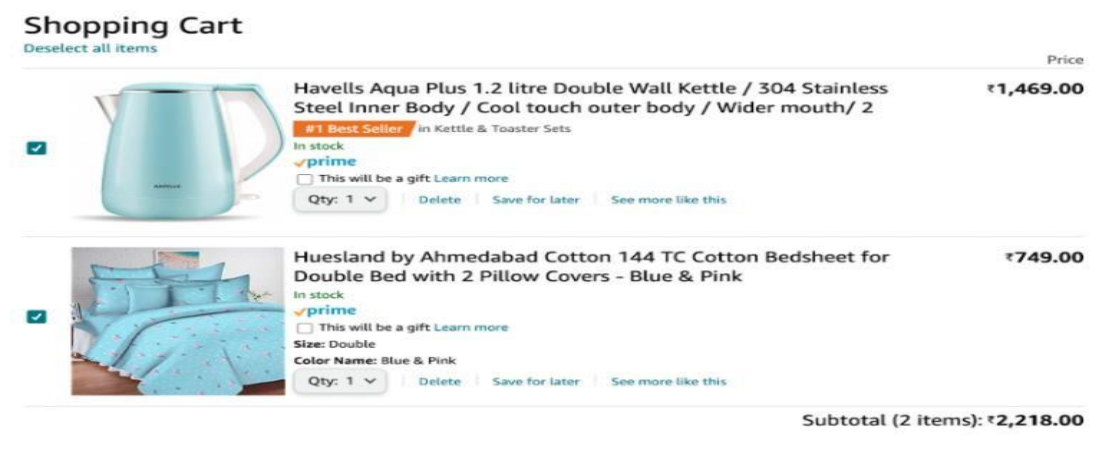
- The user should be able to see results with default search criteria when at least one of the filter parameters isn't mandatory
- Validation messages for invalid filter criteria
- When at least one filter criteria are required, and the user selects none, proper error messages are shown

## Product Detail Test Cases



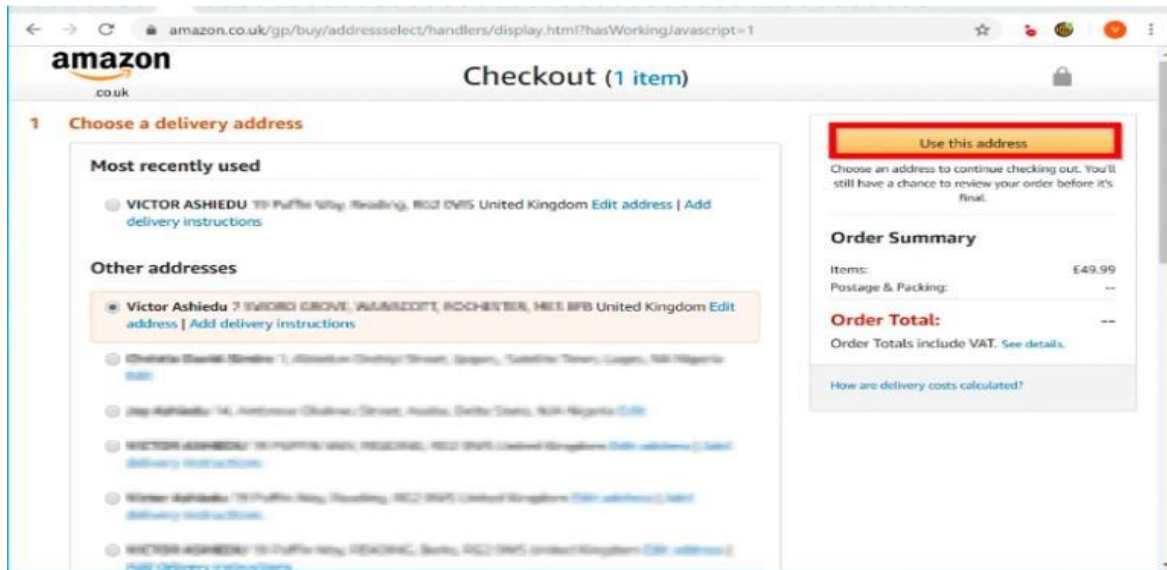
- Test that all the product details are displayed correctly and that no empty/invalid details are displayed.
- Product Images should be optimized for size and dimensions, which further helps in performance testing.
- All the links(size, Pincode check, etc) about the product should be functional.

## Shopping Cart



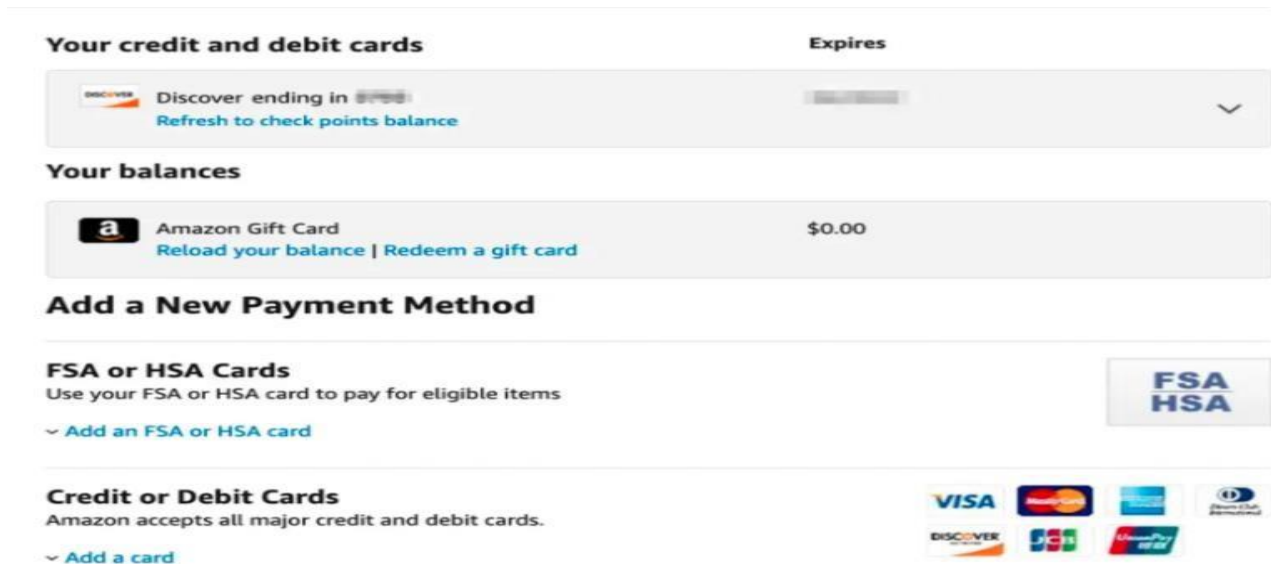
- Test that all items are added into the cart
- Test that all added items have at least a quantity, price, and delete option associated with it
- Test that the user can increase/decrease the quantity from the cart
- If a user adds the same item to the cart the amount of that item should increase in the cart
- On closing the tab/leaving the site, the items should still be in the cart upon returning to the website.

## Checkout Page



- User should be able to add coupons
- User should be shown the total amount with the necessary breakup as applicable.
- User should be able to select the desired payment method
- On adding more items or increasing/decreasing quantity, the total should change accordingly
- Calculate shipping costs based on shipping methods
- Right address should be selected and the user should be able to edit/add the address

## Payments



- Perform security testing if in case the user's credit card details need to be saved
- For returning customers, they should be redirected to log in for checkout
- User should be logged out after the session times out
- Emails/test confirmation when the order is confirmed

**Result :**

Thus the eCommerce website has many more tests or features, and testing each is very time-consuming and requires effort.



Ex.no : 03

Date :

Test the e-commerce application and report the defects in it.

### Aim :

To test the e-commerce application and report the defects in it.

### Testing E-Commerce Websites

E-commerce applications/sites include web applications and mobile applications too. So, they undergo all the typical test types.

- Functional Testing
- Usability Testing
- Security Testing
- Performance Testing
- Database Testing
- Mobile Application Testing
- A/B testing.

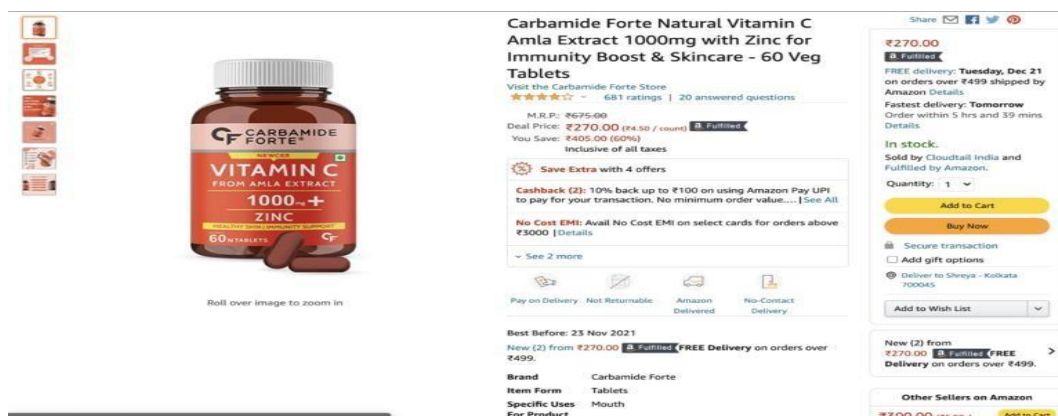
### E-Commerce Testing Checklist

We have listed important segments and test cases for eCommerce website testing below.

- **Home Page:** Test that the correct text and images show up, whether static or dynamic. Links to important pages (catalog, account login, cart) should also be visible and functional.
- **Search and Navigation:** Users should be able to search for relevant terms and be directed to the exact page they are looking for. They should also be able to navigate to important sections (product categories, cart, account info, etc.) with a couple of clicks. Test to check for any bugs that may prevent a frictionless experience in this regard. Let's take the example of the Amazon India website.



- **Catalog of products and services:** All products and services should be listed clearly, with adequate descriptions and explanatory images. Images should be easy to magnify, and the Add To Cart option should be upfront and seamlessly responsive.



- **Order processing mechanism:** Once orders are placed, products and their details must match what the user selected. They should be able to choose a preferred shipping method, and their addresses should be correctly mapped to the order. Additionally, return and exchange policies should be accessible for reading before placing an order.

**Payment Function/User Data:** At this point, a number of variables need to be tested:

- a) The privacy, security, and accuracy of customer data. This is where security testing comes in.
- b) Required validations: First and last name, card number, CSV, OTP, etc.
- c) Currency conversion (if the purchase is made in a foreign currency, does it convert correctly to local currency?)
- d) Can payment be canceled within a specific time frame?
- e) Can customers pay in installments?
- f) Generation of order confirmation and receipt
- g) Is the site flawlessly and securely integrated with an external payment system? Does the system work every time? What happens to the money if it is debited but the payment fails?

The image shows a payment form interface. At the top right, there is a hamburger menu icon. Below it, the 'Total' is displayed as '\$ 10.25 USD'. The form contains several input fields: 'First name' and 'Last name' (both with dropdown arrows), 'Card number' (with a dropdown arrow), 'Card type' (showing 'CW' with a dropdown arrow), and 'Expiry date' (showing 'MM/YY' with a dropdown arrow). Below the card number field, there is a row of payment logos including Mastercard, VISA, American Express, Discover, and others. At the bottom of the form is a large green button labeled 'SUBMIT PAYMENT'.

## Defects of eCommerce Testing

1. Often, eCommerce websites use content management systems like Shopify, Woocommerce to build the site in the shortest possible time.
2. However, these platforms offer integration with third-party services for various purposes – gift cards, social media features, online payment management, etc.
3. However, too many third-party integrations increase testing effort, as each of them needs to seamlessly and securely communicate and interact with the site.
4. With new devices and browser versions coming into existence as fast as you can blink, eCommerce website testing must keep up.

**Result**

Thus Test the e-commerce application and report the defects in it was tested done and verified successfully.

Ex.no : 04  
Date :

Develop the test plan and design the test cases for an inventory control system.

**Aim :**

To Develop the test plan and design the test cases for an inventory control system.

**Key steps to create an inventory management system**

- 1) Engineer requirements and design an inventory system.
- 2) Plan the project in detail.
- 3) Develop inventory software and run QA.
- 4) Integrate the software with other systems.
- 5) Migrate inventory data.
- 6) Deploy the inventory system.
- 7) Conduct user training.
- 8) Ensure after-launch support.

**Test Cases for Inventory Management System**

**Test cases for Inventory management system**

Sr no	Test cases	Action	Steps	Input data	Expected result	Actual result	Status
1	TC-1	Invoice no	Enter invoice no	Input *1021*	It should accepted invoice no.	Invoice no .is accepted	Pass
2	TC-2	Bill date	Enter bill date	Input*27/11/2021*	It should accepted bill date	Bill date is accepted	Pass
3	TC-3	Item name	Select item name	-	Item name should be automatically reflected	Item name is reflecting automatically	Pass
4	TC-4	Available item stock	Click on textbox	-	It should reflect automatically item stock	Item stock is reflecting automatically	Pass
5	TC-5	Quantity	Enter item quantity	Input*5000*	Item quantity should be accepted	Item quantity is accepting	Pass
6	TC-6	Price	Click on textbox	-	Price should be reflected automatically	Price is reflecting automatically	Pass
7	TC-7	Total	Click on textbox	-	Total should be reflected automatically	Total is reflecting automatically	Pass
8	TC-8	Receive bill date	Enter receive bill date	Input*29/1/2021*	Receive bill date should be accepting	Receive bill date is accepting	Pass
9	TC-9	Add item	Click on add item	-	It should be add item reflecting in database	Add item In reflecting database s	Pass

**Result**

Thus the test plan and design the test cases for an inventory control system was done and verified successfully.

Ex.no : 05

Date :

Execute the test cases against a client server or desktop application and identify the defects.

**Aim :**

To Execute the test cases against a client server or desktop application and identify the defects.

**Client Server Testing**

Client-server testing is a software testing methodology focused on evaluating the performance, functionality, and reliability of applications and systems built on a client-server architecture.

**Types Of Testing to Perform in Client-Server Test**

Performing a comprehensive range of testing types is essential in client-server testing to ensure the reliability, performance, and security of systems. Here are the primary types of testing to consider:

**Functional Testing**

- Unit Testing: Evaluate individual client and server components to verify that they perform their specific functions correctly.
- Integration Testing: Assess how well client and server components integrate and work together, ensuring that data is exchanged accurately.
- System Testing: Conduct end-to-end testing to validate that the entire client-server system functions as expected in real-world scenarios.

**Performance Testing**

- Load Testing: Measure the system's response and behavior under varying levels of user load to identify performance bottlenecks.
- Stress Testing: Push the system beyond its intended capacity to determine breaking points and assess its ability to recover.
- Scalability Testing: Evaluate how well the system scales to accommodate growing numbers of clients or data transactions while maintaining performance.

**Security Testing**

- Authentication Testing: Verify that the authentication mechanisms (e.g., username/password, tokens) work correctly and securely.
- Authorization Testing: Ensure that users can only access resources and functionalities they are permitted to, and unauthorized access is prevented.
- Encryption Testing: Confirm that data transmission between the client and server is properly encrypted to protect sensitive information.

**Client-Server Testing Techniques**

**Manual Testing and Its Types**

Manual testing is a fundamental testing approach where human testers execute test cases without

the use of automation tools or scripts.

- Functional Testing
- Usability Testing
- Exploratory Testing

### Automated Testing

Automated testing involves the use of specialized tools and scripts to perform testing tasks automatically. It is especially valuable for repetitive or complex test scenarios.

### Black-Box Testing

Black-box testing focuses on evaluating an application's functionality without knowledge of its internal code or structure. Testers interact with the application's interface and assess how it responds to different inputs and conditions.

### White-Box Testing

White-box testing is an approach where testers have access to the internal code and structure of the application. They design tests to evaluate the correctness of the code, its logic, and the execution paths within the application.

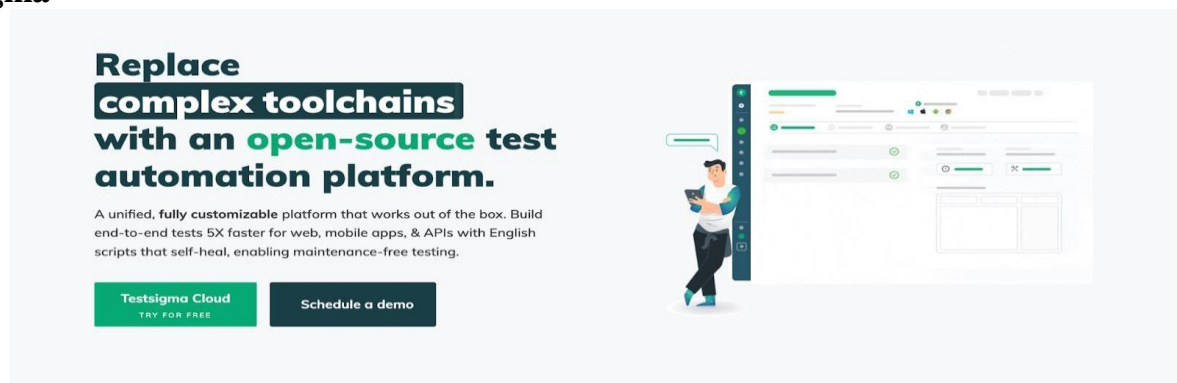
### Mocking and Simulation

Mocking and simulation involve creating simulated or mock components to mimic the behavior of real components or services that an application relies on.

### Tools for Client-Server Testing :

Here are a few essential tools commonly used for client-server testing, along with a brief explanation of how each tool aids in the testing process.

### Testsigma

The image is a promotional banner for Testsigma Cloud. On the left, there is a text block with a headline: "Replace complex toolchains with an open-source test automation platform." Below this, a smaller line of text reads: "A unified, fully customizable platform that works out of the box. Build end-to-end tests 5X faster for web, mobile apps, & APIs with English scripts that self-heal, enabling maintenance-free testing." At the bottom left, there are two buttons: a green one labeled "Testsigma Cloud TRY FOR FREE" and a dark blue one labeled "Schedule a demo". On the right side of the banner, there is an illustration of a person in a white shirt and dark pants standing next to a large, stylized screen. The screen displays a complex dashboard with various charts, graphs, and data points, representing a test automation interface.

Testsigma, as an intelligent test automation platform for web and mobile applications, offers significant advantages for automating client-server testing. With its no-code, scriptless automation approach, Testsigma simplifies the test automation process, making it accessible to both technical and non-technical team members.

## Selenium

The Selenium website features a blue header with navigation links: About, Downloads, Documentation, Projects, Support, Blog, and English. A search bar is located on the right. Below the header, a green banner reads "SeleniumConf Chicago 2023 is a wrap! Watch the Videos". The main content area has a green background with the text "Selenium automates browsers. That's it! What you do with that power is entirely up to you." followed by a sub-header "Getting Started". Three columns are provided: "Selenium WebDriver" (red icon), "Selenium IDE" (blue icon), and "Selenium Grid" (purple icon), each with a brief description of their use cases.

Selenium is a popular open-source automation testing tool primarily used for web applications. It enables testers to automate interactions with web-based client applications, making it invaluable for client-side testing.

## JMeter

The Apache JMeter website features a blue header with the Apache Software Foundation logo and the JMeter logo. A navigation menu on the left includes links for About, Download, Documentation, Tutorials, and Community. The main content area has a blue background with the text "Apache JMeter™" and a sub-header "What can I do with it?". Below this, a list of features is provided, including the ability to load and performance test many different applications/server/protocol types, full featured Test IDE, and CLI mode.

Apache JMeter is a versatile open-source tool designed for load testing, stress testing, and performance testing of web applications and servers. JMeter helps assess how the server component of a client-server application performs under various loads and concurrent user connections.

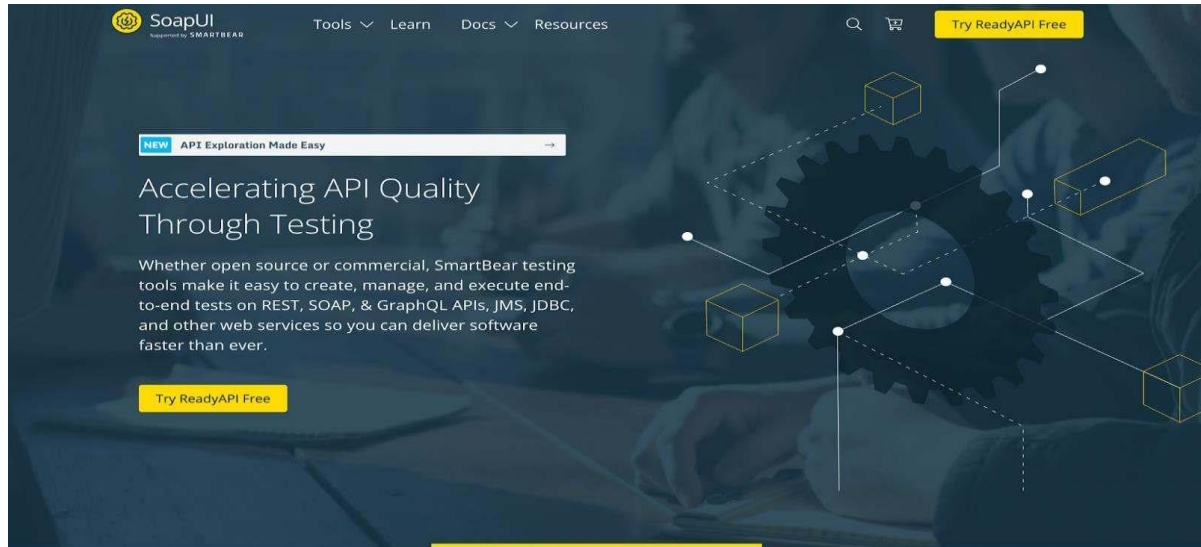
## Wireshark

The Wireshark website features a blue header with the Wireshark logo and navigation links: News, Learn, SharkFest, Get Acquainted, Get Help, Develop, Shop, and Members. A "Donate" button is located on the right. The main content area has a blue background with the text "The world's most popular network protocol analyzer" and a sub-header "Wireshark 4.0 Overview". A "Get started" button is located at the bottom left.

Wireshark is a widely used open-source network protocol analyzer that aids in network monitoring and troubleshooting. For client-server testing, Wireshark is instrumental in capturing and analyzing network traffic between the client and server components.



# SoapUI



SoapUI is an open-source tool designed specifically for testing web services, making it suitable for client-server applications that rely on RESTful APIs or SOAP-based services. With SoapUI, testers can create and execute functional and load tests for the server-side components of client-server applications.

## Executing test cases and identifying defects:

- Use a checklist: A checklist can help you to make sure that you execute all of the test cases and that you don't miss any important steps.
- Take your time: Don't rush through the test cases. Take your time to carefully follow the instructions and to identify any defects.
- Be thorough: Don't just test the happy path. Test all of the different scenarios, including error conditions and edge cases.
- Use a bug tracking system: A bug tracking system can help you to keep track of the defects you find and to communicate with the developers.

**Result**

Thus the client-server testing is a critical process in ensuring the functionality, performance, and security of applications operating on this architecture.

**Aim :**

To Test the performance of the e-commerce application.

**Procedures / Performance testing :****Goals of eCommerce Performance Testing**

- **Reduce Risks:** Many times, making major and considerable changes to a site can cause notable strategic changes or even trigger significant losses.
- **Increase Conversion Rates:** By testing every aspect of an application and ensuring a smooth visitor experience through site optimization, the application conversion rate is bound to increase.
- **Improve User Engagement:** Testing tells which page element or process affects a user's onsite journey and helps in rectifying the issues faster. The better the user experience, the greater the onsite engagement.

**Benefits of eCommerce Application Performance Testing**

- Coupon codes or gift vouchers are provided to increase product sales and performance tests ensure that coupon codes work well when applied to users in bulk.
- Multiple systems like email servers, social network sites, and enterprise content management systems are involved in the backend and the performance test ensures flawless functioning of various features like product images/videos, and social media in such an integrated system.
- eCommerce applications usually have more users than any other application and validation of users is a must to filter out genuine customers. Performance test helps in validating multiple sign-up and invoice email notifications in parallel.

**Common Performance Testing Issues**

- **Slow DB server:** Database size increases with the increase in product items of all ages, so ensure to optimize DB queries.
- **Faster Integrated Systems Communication:** Multiple systems like email servers, social media accounts, and payment channels are involved in the functioning of an eCommerce application to ensure that integrated systems work well under a heavy load.
- **System Scaling:** Brands cannot predict user load during peak hours, hence, the need of the hour is to be ready with tested scalable systems.

**Main Features to be Tested in eCommerce Applications**

- **Searching For a Product:** eCommerce sites provide various search filters to make the search experience faster, which include price range, product type, country, and age. Hence, ensure that searching for a product among billions of records does not halt the system and return the expected records.

- Virtual Reality: Modern eCommerce sites provide advanced-level features to visualize products. Online shoppers can now see how they would look wearing an item virtually. Hence, we need to ensure the smooth movement of virtual images for a better experience.
- Testing Payment Gateway Integration: Failed transactions are one of the main reasons why most customers exit an application without completing the purchase. So it's important to ensure the proper functioning of payment gateways.
- Product Details Page: A fast product listing is important because every consumer will be attracted to the product by how it's presented and how it's visible to them.

**Result**

Thus the Test the performance of the e-commerce application was done and verified successfully.

Ex.no : 07

Date :

Automate the testing of e-commerce applications using Selenium.

**Aim :**

To automate the testing of e-commerce applications using Selenium.

**Procedures :**

**Positive scenario :**

**Automate the 'User Registration and Login' of Amazon like an e-commerce website**

1. Test Case - Automate the User Registration process of an e-commerce website.

**Steps to Automate:**

- i. Open this URL <http://automationpractice.com/index.php>
- ii. Click on the sign-in link.
- iii. Enter your email address in the 'Create and Account' section.
- iv. Click on Create an Account button.
- v. Enter your Personal Information, Address, and Contact info.
- vi. Click on the Register button.
- vii. Validate that the user is created.

**Selenium code for User Registration:**

This code is contributed by Uday. We have provided the code for only positive scenarios, you should try automating negative scenarios yourself.

**Code :**

```
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;
import io.github.bonigarcia.wdm.WebDriverManager;

public class EcomSignUp {
    public static void main(String[] args) {
        WebDriverManager.chromedriver().setup();
        WebDriver driver=new ChromeDriver();
        String URL="http://automationpractice.com/index.php";
        driver.get(URL);
        driver.manage().timeouts().implicitlyWait(2000, TimeUnit.MILLISECONDS);
        driver.manage().window().maximize();
        //Click on Sign in
        driver.findElement(By.linkText("Sign in")).click();
        //Enter email address
```

```

driver.findElement(By.cssSelector("[name='email_create']")).sendKeys("test1249@test.com"
);
//Click on "Create an account"
driver.findElement(By.xpath("//button[@name=\"SubmitCreate\"]")).click();
//Select Title
driver.findElement(By.xpath("//input[@id=\"id_gender1\"]")).click();
driver.findElement(By.name("customer_firstname")).sendKeys("Test User");
driver.findElement(By.name("customer_lastname")).sendKeys("Vsoft");
driver.findElement(By.id("passwd")).sendKeys("PKR@PKR");
// Enter your address
driver.findElement(By.id("firstname")).sendKeys("Test User");
driver.findElement(By.id("lastname")).sendKeys("Vsoft");
driver.findElement(By.id("company")).sendKeys("Vsoft");
driver.findElement(By.id("address1")).sendKeys("Test 81/1,2nd cross");
driver.findElement(By.id("city")).sendKeys("XYZ");
// Select State
WebElement statedropdown=driver.findElement(By.name("id_state"));
Select oSelect=new Select(statedropdown);
oSelect.selectByValue("4");
driver.findElement(By.name("postcode")).sendKeys("51838");
// Select Country
WebElement countrydropDown=driver.findElement(By.name("id_country"));
Select oSelectC=new Select(countrydropDown);
oSelectC.selectByVisibleText("United States");
//Enter Mobile Number
driver.findElement(By.id("phone_mobile")).sendKeys("234567890");
driver.findElement(By.xpath("//input[@name=\"alias\"]")).clear();
driver.findElement(By.xpath("//input[@name=\"alias\"]")).sendKeys("Office");
driver.findElement(By.name("submitAccount")).click();
StringuserText=driver.findElement(By.xpath("//*[ @id=\"header\"]/div[2]/div/div/nav/div[1]/a")).getText();
// Validate that user has created
if(userText.contains("Vsoft")) {
    System.out.println("User Verified,Test case Passed");
}
else {
    System.out.println("User Verification Failed,Test case Failed");
}
}
}

```

## Negative Scenarios

### 2. Test Case - Verify invalid email address error.

#### Steps to Automate:

- i. Open this URL <http://automationpractice.com/index.php>
- ii. Click on the sign-in link.
- iii. Enter an invalid email address in the email box and click enter.

- iv. Validate that an error message is displaying saying "Invalid email address."

### 3. Test Case - Verify error messages for mandatory fields.

#### Steps to Automate:

- i. Open this URL <http://automationpractice.com/index.php>
- ii. Click on the sign-in link.
- iii. Enter your email address and click the Register button.
- iv. Leave the mandatory fields (marked with \*) blank and click the Register button.
- v. Verify that an error has been displayed for the mandatory fields.

### 4. Test Case - Verify error messages for entering incorrect values in fields.

#### Steps to Automate:

- i. Open this URL <http://automationpractice.com/index.php>
- ii. Click on the sign-in link.
- iii. Enter your email address and click the Register button.
- iv. Enter incorrect values in fields like., enter numbers in first and last name, city field, etc., and enter alphabets in Mobile no, Zip postal code, etc., and click on the 'Register' button.
- v. Verify that error messages for respective fields are displaying.
- vi. Try automating the above scenarios using Selenium commands, if you face any difficulty please refer to the Selenium Tutorial series.

### 5. Automate the 'Search Product' feature of Amazon like e-commerce website with Selenium

#### (1).Test Case - Automate the 'Search Product' feature of the e-commerce website with Selenium.

#### Steps to Automate:

- i. Open link <http://automationpractice.com/index.php>
- ii. Move your cursor over the Women's link.
- iii. Click on the sub-menu 'T-shirts'
- iv. Get the Name/Text of the first product displayed on the page.
- v. Now enter the same product name in the search bar present at the top of the page and click the search button.
- vi. Validate that the same product is displayed on the searched page with the same details which were displayed on T-Shirt's page.

#### Automation Code for Product Search:

The following code is contributed by Uday

```

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;
import io.github.bonigarcia.wdm.WebDriverManager;
public class EcomPractice2 {
    public static void main(String[] args) throws InterruptedException{
        WebDriverManager.chromedriver().setup();
        WebDriver driver=new ChromeDriver();
        String URL="http://automationpractice.com/index.php";

        driver.get(URL);
        driver.manage().window().maximize();
        // Initialise Actions class object
        Actions actions=new Actions(driver);
        driver.manage().timeouts().implicitlyWait(2000, TimeUnit.MILLISECONDS);
        WebElement womenTab=driver.findElement(By.linkText("WOMEN"));
        WebElement
        TshirtTab=driver.findElement(By.xpath("//div[@id='block_top_menu']/ul/li[1]/ul/li[1]/ul//a[
        @title='T-shirts']"));
        actions.moveToElement(womenTab).moveToElement(TshirtTab).click().perform();
        Thread.sleep(2000);
        // Get Product Name
        String
        ProductName=driver.findElement(By.xpath("/html[1]/body[1]/div[1]/div[2]/div[1]/div[3]/div
        [2]/ul[1]/li[1]/div[1]/div[2]/h5[1]/a[1]")).getText();
        System.out.println(ProductName);
        driver.findElement(By.id("search_query_top")).sendKeys(ProductName);
        driver.findElement(By.name("submit_search")).click();
        // Get Name of Searched Product
        String
        SearchResultProductname=driver.findElement(By.xpath("/html[1]/body[1]/div[1]/div[2]/div[
        1]/div[3]/div[2]/ul[1]/li[1]/div[1]/div[2]/h5[1]/a[1]")).getText();
        // Verify that correct Product is displaying after search
        if(ProductName.equalsIgnoreCase(SearchResultProductname)) {
            System.out.println("Results Matched;Test Case Passed");
        }else{
            System.out.println("Results NotMatched;Test Case Failed");
        }
        // Close the browser
        driver.close();
    }
}

```

6. Automate the 'Buy Product' feature of Amazon like an e-commerce website with Selenium  
 The most important function of an e-commerce website is buying a product, which includes various steps like selecting a product, selecting size/color, adding to the cart, checkout, etc. You will find every test scenario along with the automation code in the following section.



(1) Test Case - Automate the end-to-end "Buy Product" feature of the e-commerce website.

**Steps to Automate:**

- i. Open link <http://automationpractice.com/index.php>
- ii. log in to the website.
- iii. Move your cursor over the Women's link.
- iv. Click on the sub-menu 'T-shirts'.
- v. Mouse hover on the second product displayed.
- vi. 'More' button will be displayed, click on the 'More' button.
- vii. Increase quantity to 2.
- viii. Select size 'L'
- ix. Select color.
- x. Click the 'Add to Cart' button.
- xi. Click the 'Proceed to checkout' button.
- xii. Complete the buy order process till payment.
- xiii. Make sure that the Product is ordered.

**The following code for Purchase Product is contributed by Uday:**

```
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.Select;
import io.github.bonigarcia.wdm.WebDriverManager;

public class EcomExpert {
    public static void main(String[] args){
        WebDriverManager.chromedriver().setup();
        WebDriver driver=new ChromeDriver();
        String URL="http://automationpractice.com/index.php";
        // Open URL and Maximize browser window
        driver.get(URL);
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(3000, TimeUnit.MILLISECONDS);
        //Click on Sign in
        driver.findElement(By.linkText("Sign in")).click();
        //Login
        driver.findElement(By.id("email")).sendKeys("test1249@test.com");
        driver.findElement(By.id("passwd")).sendKeys("PKR@PKR");
        driver.findElement(By.id("SubmitLogin")).click();
        //Click on Women
        driver.findElement(By.linkText("WOMEN")).click();
        WebElement
        SecondImg=driver.findElement(By.xpath("/html/body/div[1]/div[2]/div/div[3]/div[2]/ul/li[2]
```

```

/div/div[1]/div/a[1]/img"));
WebElement
MoreBtn=driver.findElement(By.xpath("/html/body[1]/div[1]/div[2]/div[1]/div[3]/div[2]/ul/li
[2]/div[1]/div[2]/div[2]/a[2]"));
Actions actions=new Actions(driver);
actions.moveToElement(SecondImg).moveToElement(MoreBtn).click().perform();
//Change quantity by 2
driver.findElement(By.id("quantity_wanted")).clear();
driver.findElement(By.id("quantity_wanted")).sendKeys("2");
//Select size as L
WebElement Sizedrpdwn=driver.findElement(By.xpath("//*[@id='group_1']"));
Select oSelect=new Select(Sizedrpdwn);
oSelect.selectByVisibleText("M");
//Select Color
driver.findElement(By.id("color_11")).click();

//Click on add to cart
driver.findElement(By.xpath("//p[@id='add_to_cart']/span[.='Add to cart']")).click();
//Click on proceed
driver.findElement(By.xpath("/html//div[@id='layer_cart']/a[@title='Proceed to
checkout']/span")).click();
//Checkout page Proceed

driver.findElement(By.xpath("/html/body/div[1]/div[2]/div/div[3]/div/p[2]/a[1]/span")).click(
);
driver.findElement(By.xpath("/html/body/div[1]/div[2]/div/div[3]/div/form/p/button/span")).
click();
//Agree terms&Conditions
driver.findElement(By.xpath("//*[@id='cgv']")).click();
driver.findElement(By.xpath("/html/body/div[1]/div[2]/div/div[3]/div/div/form/p/button/span
")).click();
//Click on Payby Check
driver.findElement(By.xpath("/html/body/div[1]/div[2]/div/div[3]/div/div/div[3]/div[2]/div/p
/a")).click();
//Confirm the order
driver.findElement(By.xpath("/html/body/div[1]/div[2]/div/div[3]/div/form/p/button/span")).
click();
//Get Text
String
ConfirmationText=driver.findElement(By.xpath("//div[@id='center_column']/p[@class='aler
t alert-success']")).getText();
// Verify that Product is ordered
if(ConfirmationText.contains("complete")) {
System.out.println("Order Completed: Test Case Passed");
}
else {
System.out.println("Order Not Successfull: Test Case Failed");
}
}
}
}

```

(2) Test Case - Verify that 'Add to Wishlist' only works after login.

### **Steps to Automate:**

- i. Open link <http://automationpractice.com/index.php>
- ii. Move your cursor over the Women's link.
- iii. Click on the sub-menu 'T-shirts'.
- iv. Mouse hover on the second product displayed.
- v. 'Add to Wishlist' will appear on the bottom of that product, click on it.
- vi. Verify that the error message is displayed 'You must be logged in to manage your wish list.'

(3) Test Case - Verify that Total Price is reflecting correctly if the user changes quantity on the 'Shopping Cart Summary' Page.

### **Steps to Automate:**

- i. Open link <http://automationpractice.com/index.php>
- ii. Log in to the website.
- iii. Move your cursor over the Women's link.
- iv. Click on the sub-menu 'T-shirts'.
- v. Mouse hover on the second product displayed.
- vi. 'More' button will be displayed, click on the 'More' button.
- vii. Make sure the quantity is set to 1.
- viii. Select size 'M'
- ix. Select the color of your choice.
- x. Click the 'Add to Cart' button.
- xi. Click the 'Proceed to checkout' button.
- xii. Change the quantity to 2.
- xiii. Verify that the Total price is changing and reflecting the correct price.
- xiv. Similar way you can add a few more test cases.

**Result :**

Thus the Automate the testing of e-commerce applications using Selenium was done and these output was verified successfully.

Ex.no : 08

Date :

Integrate TestNG with the above test automation.

### Aim :

To Integrate TestNG with the above test automation.

### Procedures :

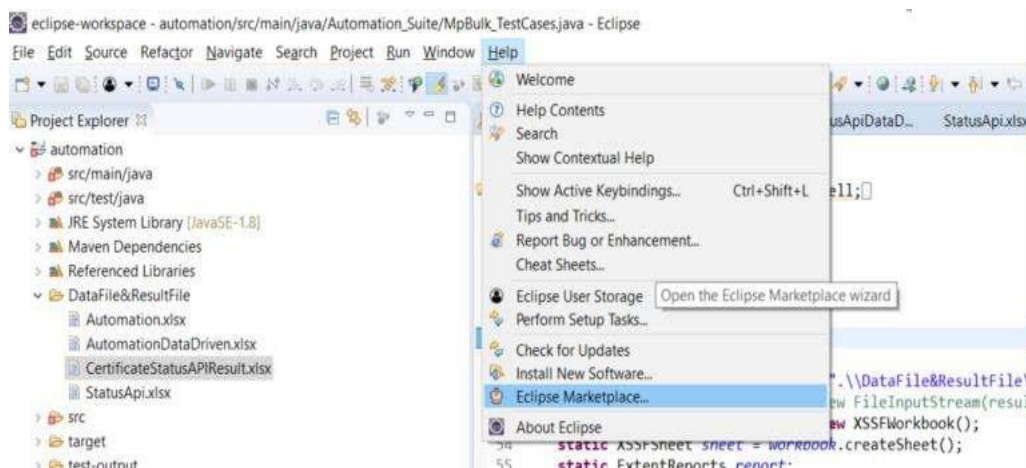
#### TestNG Framework?

TestNG is an open-source test automation framework for Java. It is developed on the same lines as JUnit and NUnit. A few advanced and useful features provided by TestNG make it a more robust framework than its peers. The NG in TestNG stands for 'Next Generation.'

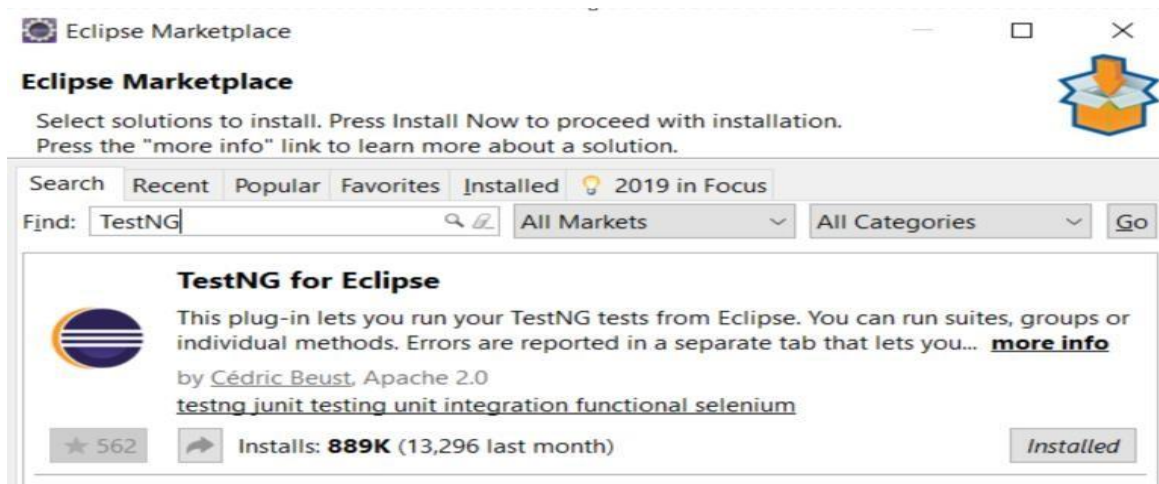
#### Installing and Setting up TestNG

Below are the steps to install TestNG:

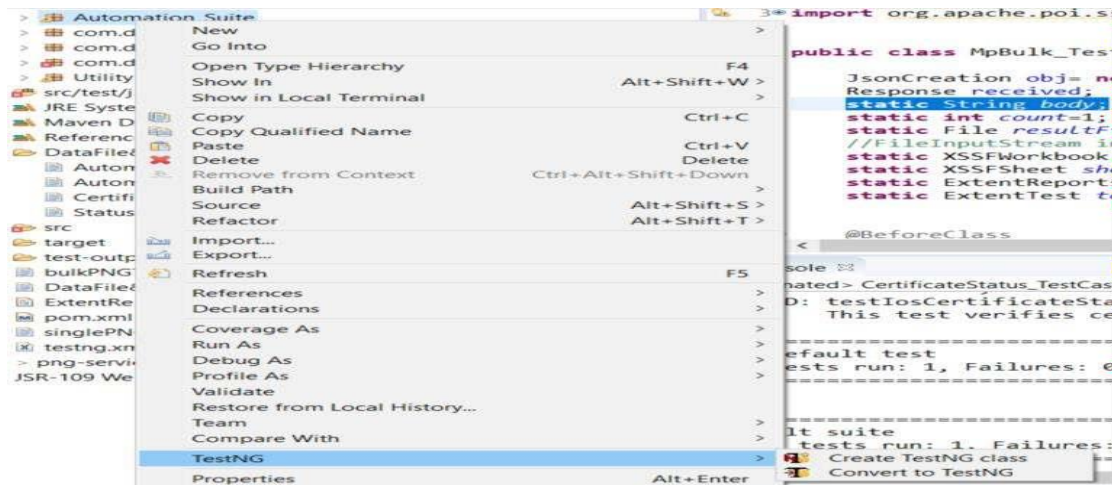
1. Install Eclipse IDE from the Eclipse website. It serves as a platform for you to code on and write your test cases
2. Once installed, go to help and navigate to the 'Eclipse Marketplace'. The referenced snapshot is below:



3. Click on 'Eclipse Marketplace'. You will be directed to the marketplace modal. Type TestNG in the search keyword and hit 'Go'. The referenced snapshot is below:

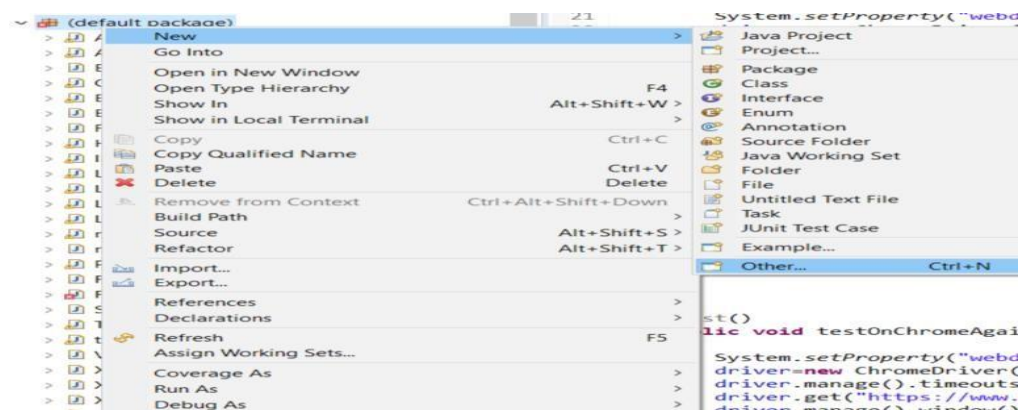


4. If TestNG is not installed in your Eclipse, rather than the 'Installed' button you would see 'install'. Click on install and your TestNG framework will be installed in your Eclipse. As a good practice, Eclipse would recommend you to restart to use the features of the installed plugin
5. Post-restarting your Eclipse, re-verify whether you can see options for creating a TestNG class or not as below:

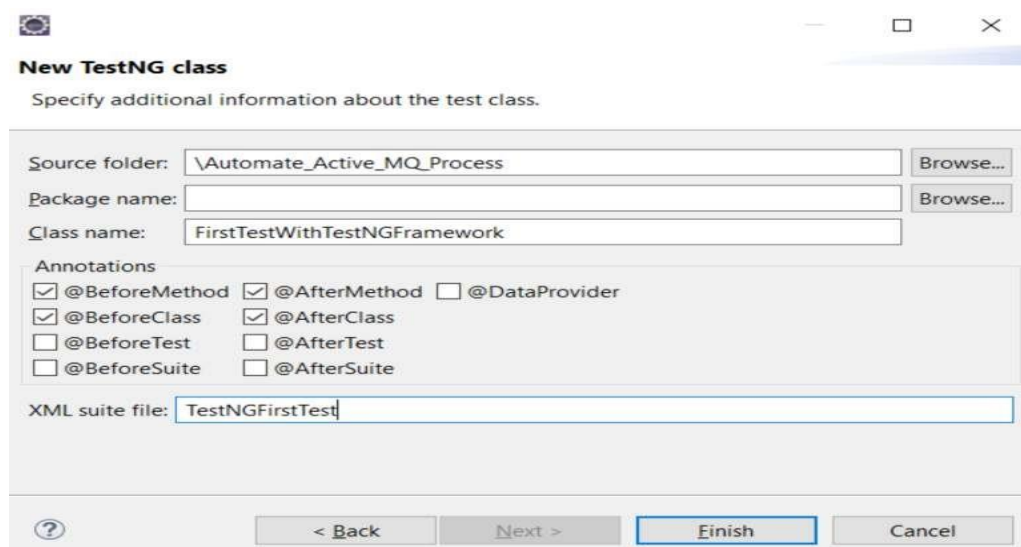


## How to Write your First TestNG Test

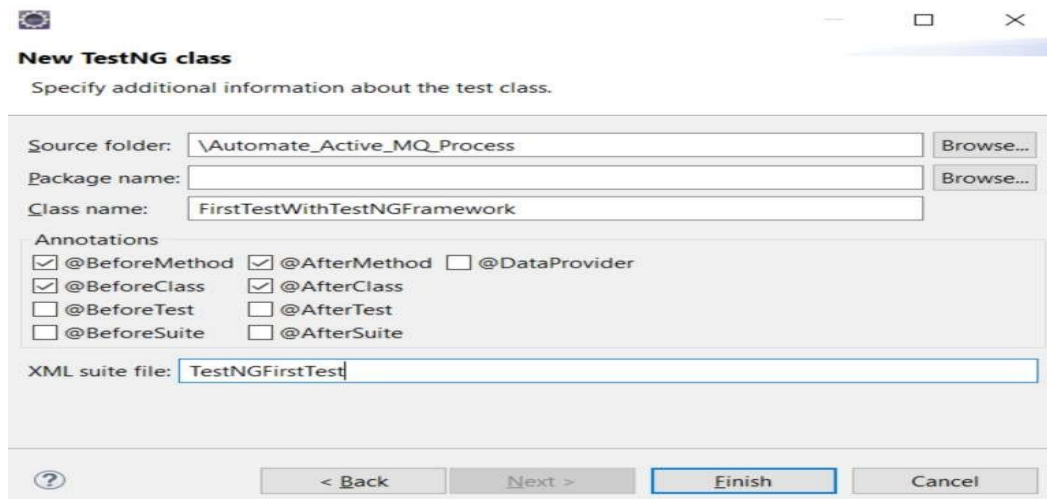
**Step #1** – Create your first TestNG class by right-clicking on the package and selecting the 'Other' folder from the 'New' option.



**Step #2** – From the wizard modal, select the TestNG folder and select the TestNG class as below:



**Step #3** – Click the ‘New’ button and choose any predefined annotations you wish to have in your class as below:



**New TestNG class**  
Specify additional information about the test class.

Source folder: \Automate\_Active\_MQ\_Process Browse...

Package name: Browse...

Class name: FirstTestWithTestNGFramework

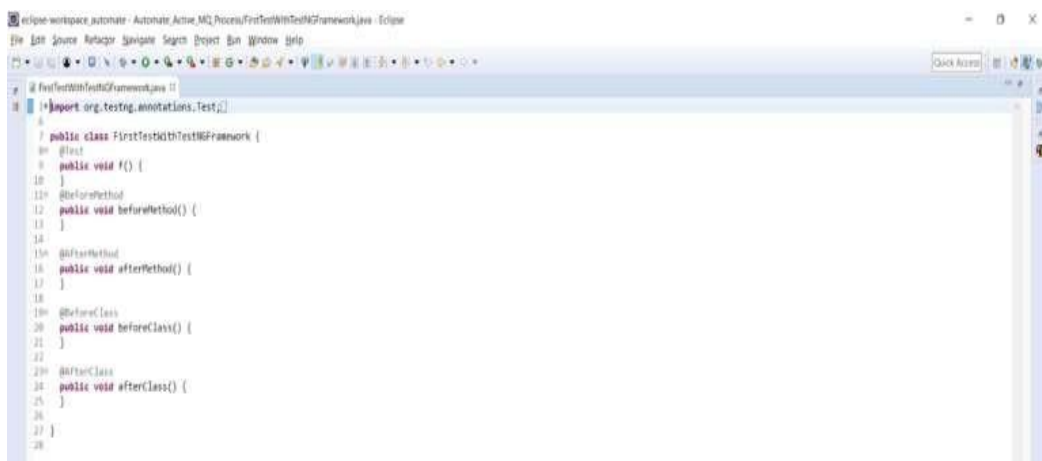
Annotations

<input checked="" type="checkbox"/> @BeforeMethod	<input checked="" type="checkbox"/> @AfterMethod	<input type="checkbox"/> @DataProvider
<input checked="" type="checkbox"/> @BeforeClass	<input checked="" type="checkbox"/> @AfterClass	
<input type="checkbox"/> @BeforeTest	<input type="checkbox"/> @AfterTest	
<input type="checkbox"/> @BeforeSuite	<input type="checkbox"/> @AfterSuite	

XML suite file: TestNGFirstTest

? < Back Next > Finish Cancel

**Step #4** – Click on finish, and you are ready to start writing your first TestNG class. Reference screenshot below, containing the defined methods with the annotations chosen in the step above:



```
1 import org.testng.annotations.Test;
2
3 public class FirstTestWithTestNGFramework {
4     @Test
5     public void f() {
6     }
7
8     @BeforeMethod
9     public void beforeMethod() {
10    }
11
12     @AfterMethod
13     public void afterMethod() {
14    }
15
16     @BeforeClass
17     public void beforeClass() {
18    }
19
20     @AfterClass
21     public void afterClass() {
22    }
23 }
24
25
26
27
28
```

**Step #5** – We will automate the sign-up flow using these annotations in the code snippet below.

```
import org.testng.annotations.Test;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
public class FirstTestWithTestNGFramework {
```

```
WebDriver driver;

@BeforeClass
public void testSetup()
{
    System.setProperty("webdriver.chrome.driver", ".\\Driver\\chromedriver.exe");
    driver=new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.manage().window().maximize();
}

@BeforeMethod
public void openBrowser()
{
    driver.get("https://www.browserstack.com/");
    driver.findElement(By.id("signupModalButton")).click();
    System.out.println("We are currently on the following URL" +driver.getCurrentUrl());
}

@Test(description="This method validates the sign up functionality")
public void signUp()
{
    driver.findElement(By.id("user_full_name")).sendKeys("user_name");
    driver.findElement(By.id("user_email_login")).sendKeys("email_id");
    driver.findElement(By.id("user_password")).sendKeys("password");
    driver.findElement(By.xpath("//input[@name='terms_and_conditions']")).click();
    driver.findElement(By.id("user_submit")).click();
}

@AfterMethod
public void postSignUp()
{
    System.out.println(driver.getCurrentUrl());
}
```



@AfterClass

```
public void afterClass()

{

driver.quit();

}

}
```

**Step #6** – To execute your report, you can either choose to run directly as a TestNG class or run the XML file created which contains the class name and details. Below is the auto-created TestNG XML file:

```
<?XML version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Suite">

<test name="Test">

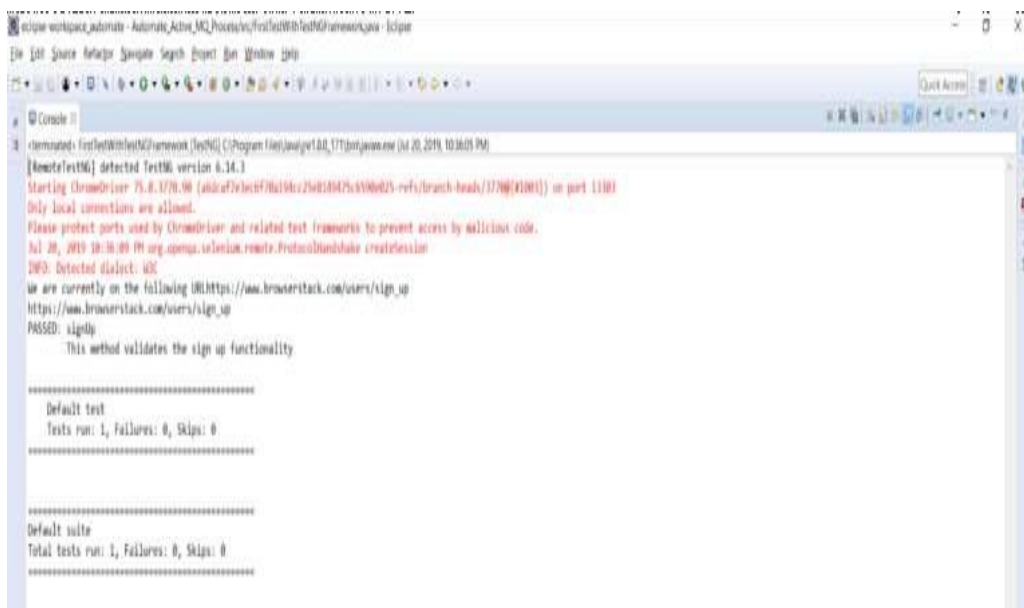
<classes>

<class name=".FirstTestWithTestNGFramework"/>

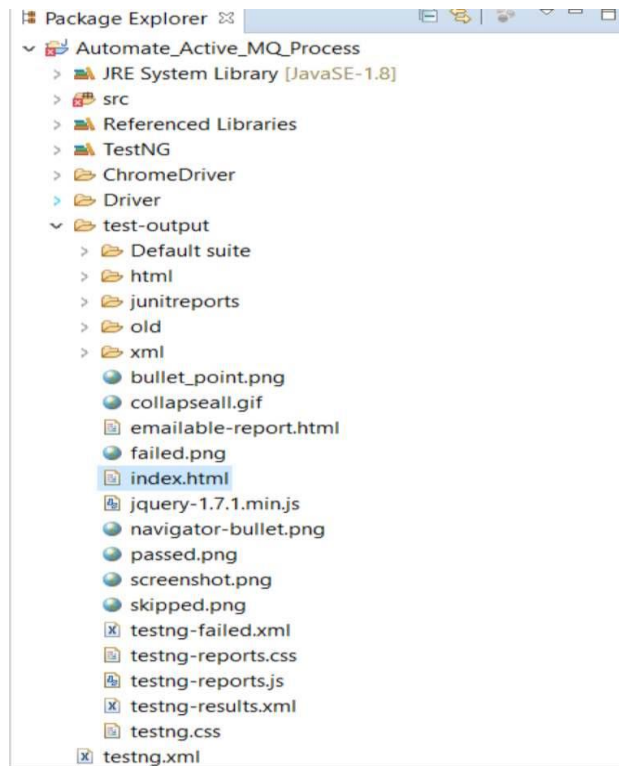
</classes>

</test> <!-- Test -->

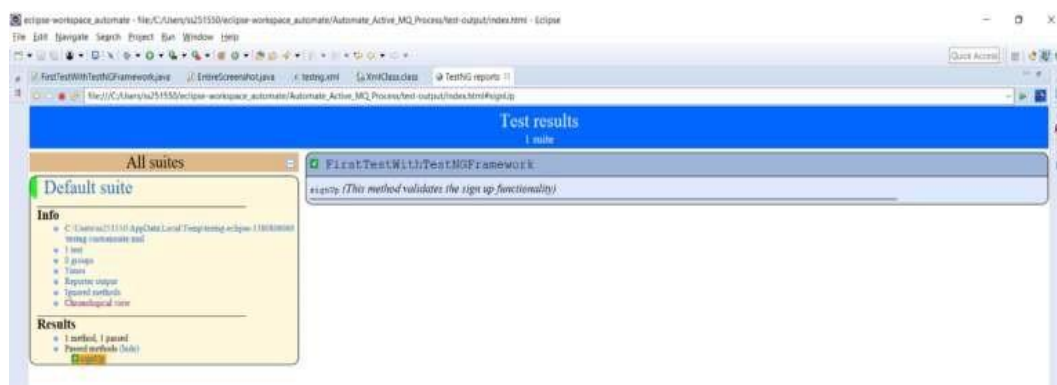
</suite> <!-- Suite -->
```



**Step #7** – To access the TestNG report, you only need to refresh your project folder. A folder as ‘test output’ will be auto-generated. Within this folder, you shall see a file ‘index.html’ as below:



**Step #8** – Double click on it and you can view your TestNG reports, showing the passed and failed methods of the execution as below.



**Step #9** – As a QA or tester, the next step is to validate your tests. If you did not validate your test methods, it basically means your testing process is incomplete.

**Result :**

Thus the Integrate TestNG with the above test automation was done and these output was verified successfully.

Ex.no : 09

Date :

## Build a data-driven framework using Selenium and TestNG

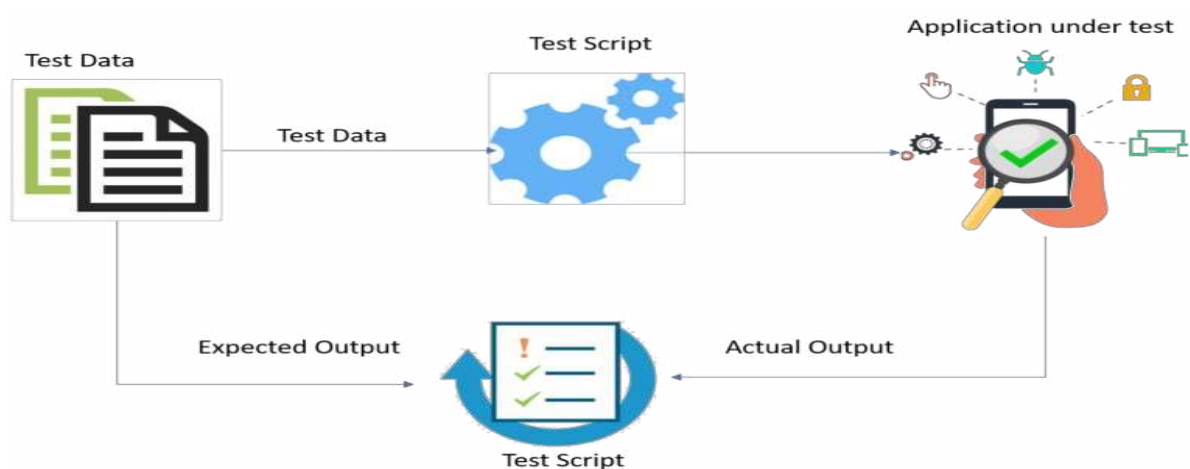
### Aim :

To Build a data-driven framework using Selenium and TestNG.

### Procedures :

### Data Driven Testing Framework in Selenium?

Data Driven framework is used to drive test cases and suites from an external data feed. The data feed can be data sheets like xls, xlsx, and csv files.



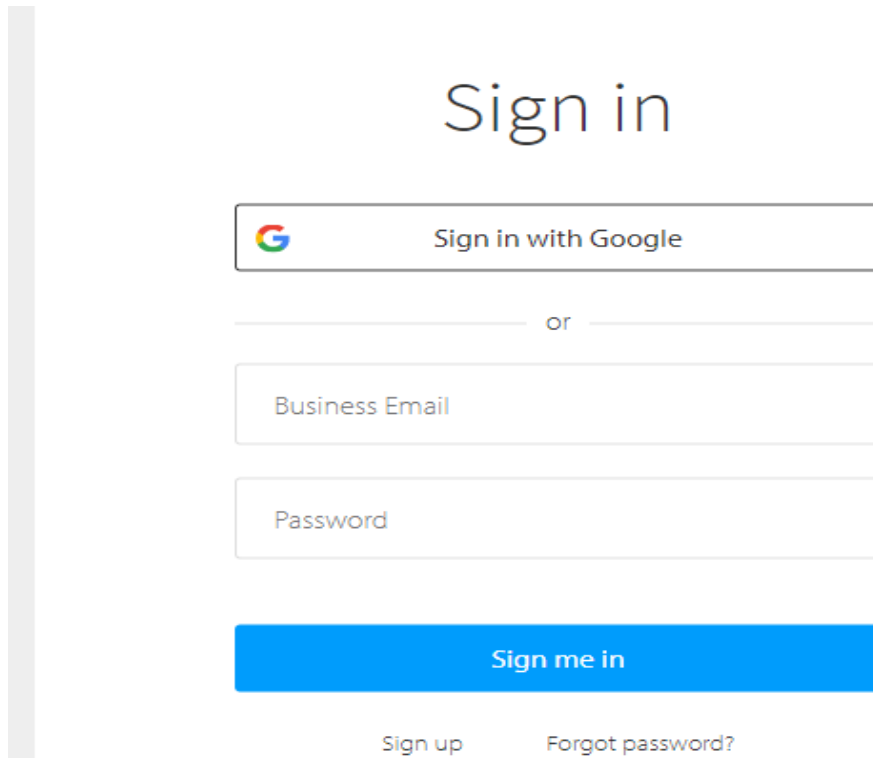
### Data Driven Testing Example

- To download Apache POI Jar files click [here](#). Download the zip file or tar file as per requirement and place them along with the set of Selenium JARs and configure your build path.
- Now let's understand how to write the first test case. An excel file to read the data from the sheet. The user has entered different combinations of username and password in the sheet.

The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E
1	<a href="#">Username1@xyz.com</a>	Password 1			
2	<a href="#">Username2@xyz.com</a>	Password 2			
3	<a href="#">Username3@abc.com</a>	Password 3			
4	<a href="#">Username 4@abc.cm</a>	Password 4			
5	<a href="#">Username 5@qwe.com</a>	Password 5			
6	<a href="#">Username 6@qwe.com</a>	Password 6			
7					

- Here, the target is to enter all these combinations of username and password into the Browser stack Sign in page as shown below.



Let's write a code snippet to read the data files.

**Step 1:** Go to the Eclipse IDE and create a project. Add all the dependencies for TestNG, Selenium and Apache POI.

**Step 2:** Create a class file to write the functionality

**Code :**

```
import org.openqa.selenium.By;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
public class ExcelExample{
    @Test(dataProvider="testdata")
    public void demoClass(String username, String password) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "Path of Chrome Driver");
        Webdriver driver = new ChromeDriver();
        driver.get("<a href='\"https://www.browserstack.com/users/sign_in\"'/>");
```

```

driver.findElement(By.name("user[login]")).sendKeys(username);
driver.findElement(By.name("user[password]")).sendKeys(password);
driver.findElement(By.name("commit")).click();
Thread.sleep(5000);

Assert.assertTrue(driver.getTitle().matches("BrowserStack Login | Sign Into The Best Mobile &
    Browser Testing Tool"), "Invalid credentials");

System.out.println("Login successful");
}

@AfterMethod

void ProgramTermination() {
driver.quit();
}

@DataProvider(name="testdata")
public Object[][] testDataExample(){
ReadExcelFile configuration = new ReadExcelFile("Path_of_Your_Excel_File");
int rows = configuration.getRowCount(0);
Object[][]signin_credentials = new Object[rows][2];

for(int i=0;i<rows;i++)
{
signin_credentials[i][0] = config.getData(0, i, 0);
signin_credentials[i][1] = config.getData(0, i, 1);
}
return signin_credentials;
}
}

```

- In the above code, there is a “TestDataExample() method” in which the user has created an object instance of another class named “ReadExcelFile”. The user has mentioned the path to the excel file. The user has further defined a for loop to retrieve the text from the excel workbook. But to fetch the data from the excel file, one needs to write a class file for the same.

**Code :**

```
import java.io.File;
import java.io.FileInputStream;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public class ReadExcelFile{
XSSFWorkbook work_book;
XSSFSheet sheet;
public ReadExcelFile(String excelfilePath) {
try {
File s = new File(excelfilePath);
FileInputStream stream = new FileInputStream(s);
work_book = new XSSFWorkbook(stream);
}
catch(Exception e) {
System.out.println(e.getMessage());
}
}
public String getData(int sheetnumber, int row, int column){
sheet = work_book.getSheetAt(sheetnumber);
String data = sheet.getRow(row).getCell(column).getStringCellValue();
return data;
}
public int getRowCount(int sheetIndex){
int row = work_book.getSheetAt(sheetIndex).getLastRowNum();
row = row + 1;
return row;
}
}
```

- In the code above, the user has used Apache POI libraries to fetch the data from the excel file. Next, it will point to the data present in the excel file and then enter the relevant username and password to the sign in page.

**Result :**

Thus the mini project of Build a data-driven framework using Selenium and TestNG was done and these output was verified successfully.