



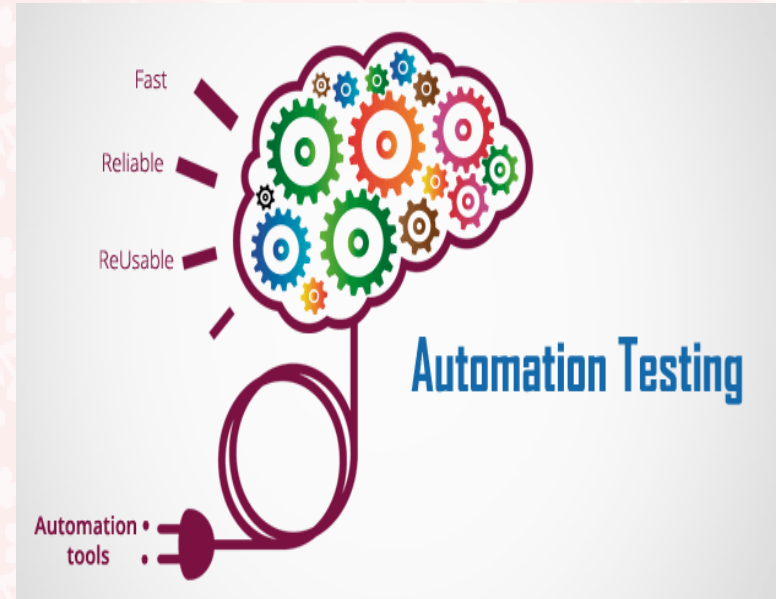
End-to-End Automation Testing of **Nykaa** Website

Capstone Project – Selenium Automation Testing:
Validating E-commerce functionality through
automated workflows.

Tools: Selenium | Java | TestNG | Maven

Name: Sneha Kumari, Ritika Ruhal , Ravindra
Kumar, Raghav Kaushal

Qualification : B.Tech in Computer Science and
Engineering



Introduction

This project automates the entire *end-to-end testing* process of the Nykaa e-commerce website. It focuses on validating key functionalities such as product selection, cart management, checkout, and navigation to the payment page. Using **Selenium WebDriver, TestNG, Maven, and Java**, the automation ensures reliability, faster test execution, and reduced manual effort. This presentation explains the project context and tools used.



01 Project Context: Nykaa E-commerce Platform Overview

Application Under Test:

Nykaa Website

Tools and Technologies Used:

- Programming Language: Java
- Automation Tool: Selenium WebDriver
- Test Framework: TestNG
- Build Tool: Maven



Automation Scope:

- Launch Nykaa website
- Search and select a product
- Add product to cart
- Proceed to checkout
- Navigate till payment page

Outcome:

- Automated critical user workflows
- Reduced dependency on manual testing
- Improved reliability of testing process
- Ensured smooth end-to-end functionality of the application



Automation Testing Objectives

Automate end-to-end functionality of the Nykaa website using Selenium and Java to ensure smooth and reliable user experience.

Key Activities:

- Designed automation framework using TestNG and Maven
- Automated test scenarios for product selection, cart validation, and checkout flow
- Executed tests across multiple scenarios
- Validated navigation till payment page





01

Automation Testing Workflow





Test Execution Console

The screenshot displays the Eclipse IDE interface. The Package Explorer on the left shows a project structure with a 'src/test/java' directory. The main editor shows the file 'TC_sneha.java' with the following code:

```
1 package SeleniumPac;
2
3 import org.testng.annotations.Test;
4
5 Run ALL
6 public class TC_sneha {
7     WebDriver driver;
8     Properties prop;
9
10    void sleep(long time) {
11        try { Thread.sleep(time); } catch (Exception ignored) {}
12    }
13 }
```

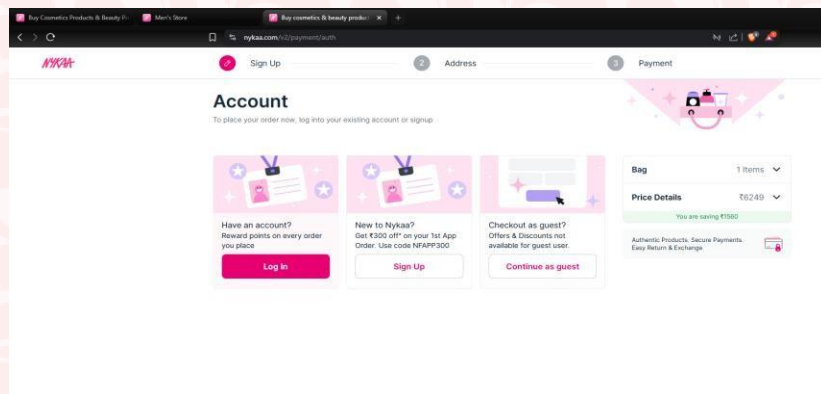
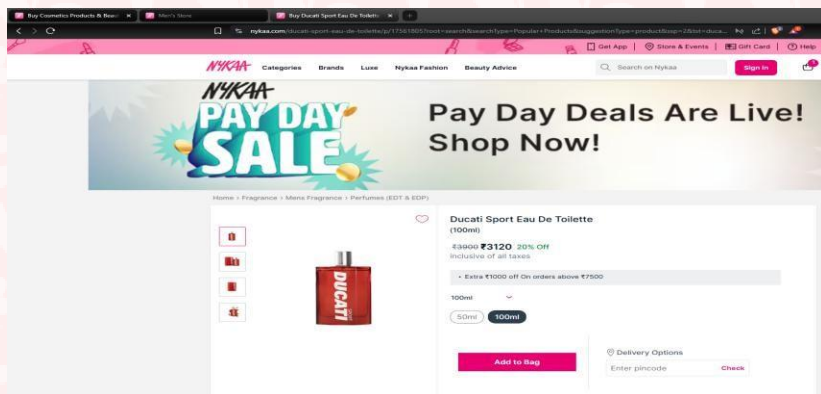
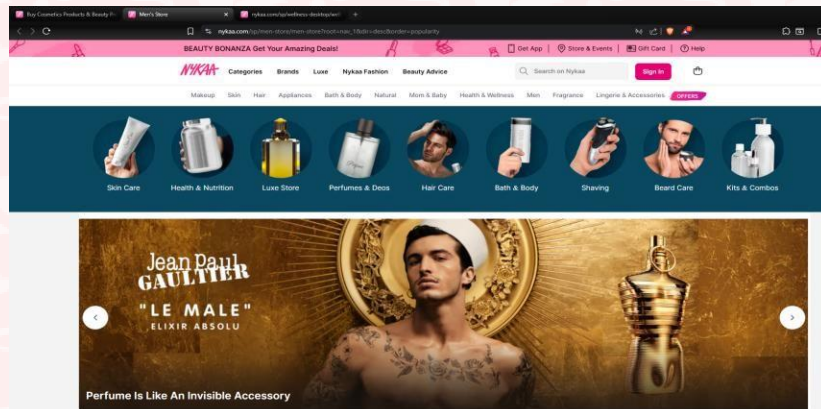
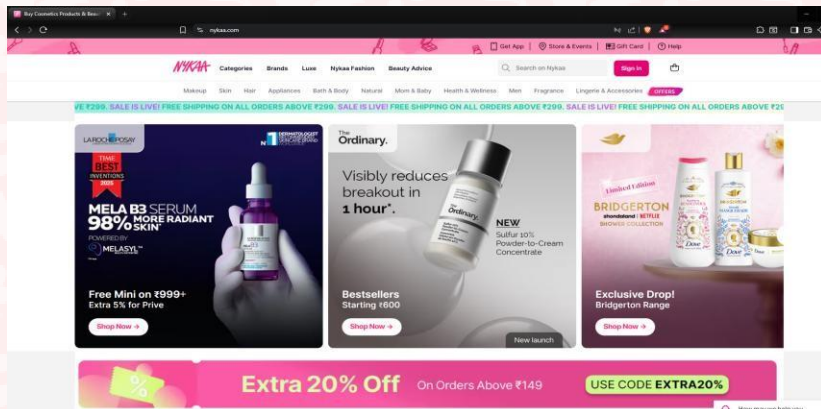
The Console window at the bottom shows the execution output for the test 'SeleniumPac.TC_sneha.tc1'. The output includes a warning about SLF4J providers and a successful test result:

```
<terminated> TC_sneha [TestNG] C:\Users\Sneha.3.Kumari\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\java.exe
[RemoteTestNG] detected TestNG version 7.11.0
SLF4J(W): No SLF4J providers were found.
SLF4J(W): Defaulting to no-operation (NOP) logger implementation
SLF4J(W): See https://www.slf4j.org/codes.html#noProviders for further details.
Feb 01, 2026 3:27:19 PM org.openqa.selenium.selenium.devtools.CdpVersionFinder findNearest
WARNING: Unable to find an exact match for CDP version 144, returning the closest
true
Price before: 396
Price after: 1170
PASSED: SeleniumPac.TC_sneha.tc1

=====
Default test
Tests run: 1, Failures: 0, Skips: 0
=====
```



Web Application Screenshot





Selenium Automation Testing

Selenium IDE - Nykaa

Project: Nykaa

Tests +

Search tests...

https://www.nykaa.com

	Command	Target	Value
TC01	open	/	
Untitled	set window size	1296x688	
	click	css=.MegaDropdownHeadingb ox:nth-child(9) > a	
	select window	handle=\${win1201}	
	mouse over	linkText=Health & Wellness	
	mouse out	linkText=Health & Wellness	
	click	css=.MegaDropdownHeadingb ox:nth-child(9) > a	
	select window	handle=\${win3552}	
	click	css=#\36 523d89c2c71bae399 07efa8 .outline-wrapper	
	run script	window.scrollTo(0,765.333312	

Command

Target





Selenium WebDriver and TestNG Usage

Selenium WebDriver is employed to simulate user actions in real web browsers, enabling **robust UI automation**. TestNG manages test execution, allows grouping and prioritization of test cases, and generates detailed reports. Together, they create a stable framework that facilitates effective testing, debugging, and maintenance of automation scripts.



Conclusions

- Successfully automated end-to-end testing of Nykaa website.
- Covered complete user journey from product selection to payment page.
- Reduced manual testing effort and improved execution speed.
- Gained hands-on experience with Selenium, TestNG, Maven, and Java.
- Enhanced understanding of real-world automation testing challenges.





Thank You

