LAMBDATEST
Playground

# Project Name: LambdaTest

(Selenium Webdriver)

| Document Author | Document Version | Revised Date | |
|---|---|---|---|
| Fizza javed | 1.0 | 13 oct 2025 | |

## 1. Introduction

This project focuses on automating key user functionalities of an eCommerce web application using **Selenium WebDriver** and **TestNG**.
The purpose of this automation testing project is to validate the website's critical workflows, including **user registration, login, and add-to-cart** features , to ensure they perform as expected across various user actions.

Automation was implemented using the **Page Object Model (POM)** design pattern, which improves script reusability and maintainability.
The project helps demonstrate the efficiency of automated testing in improving testing accuracy, reducing manual effort, and identifying bugs in early stages of development.

This project is part of my QA learning journey, aimed at building hands-on experience with real-world web automation using open-source tools.

## 2. Scope

### 2.1. In Scope

- **Register**
  - Validates that new users can successfully create an account by filling in all required fields and complying with form validations such as email format, password confirmation, and privacy policy agreement.

- **Login**

  Confirms that registered users can log into their accounts using valid credentials and verifies that appropriate error messages are displayed for invalid login attempts.

- **ADDTOCART**
  - Ensures that users can view product details, select available options (such as size or quantity), and add products to their shopping cart successfully. The system's response and confirmation messages are also validated.

## 3. Quality Objective

The primary goal of this project is to ensure the **functionality, reliability, and usability** of key eCommerce features (Registration, Login, and Add to Cart) through systematic and automated testing.

**Quality objectives include:**

1. **Accuracy:**
   Verify that all features work as expected and that the website correctly handles valid and invalid user actions.
2. **Consistency:**
   Ensure that the same inputs consistently produce the same outputs across multiple test runs and browsers.
3. **Usability & Accessibility:**
   Confirm that user interactions, such as filling forms or adding items to the cart, are smooth and intuitive.
4. **Error Handling:**
   Validate that the application displays clear and user-friendly error messages for incorrect or missing inputs.
5. **Regression Prevention:**
   Automate repetitive test cases to quickly detect bugs introduced by future changes in the application.
6. **Performance Validation:**
   Ensure that core functions respond within acceptable time limits and do not lead to crashes or freezes.

## 4. Testing Methodologies

### 4.1. Manual testing

Manual testing was performed to validate the core functionalities such as:

- User registration with valid and invalid data.
- Login verification with correct and incorrect credentials.
- Add-to-cart feature behavior (including product selection, size dropdown validation, and confirmation message).

Manual testing ensured usability, layout correctness, and smooth navigation from a real user's perspective.

### 1.1. Automation testing

Automation was implemented using:

- **Selenium WebDriver** for UI automation.
- **TestNG** for test case organization, execution, and reporting.

Automation scripts focused on **repetitive regression tests** like login, registration, and cart functionality, reducing human error and increasing efficiency.

## 1.2. Black Box Testing

- Black box testing was applied, focusing on verifying system behavior **without accessing internal code logic**.
  Test cases were based solely on:
- Functional requirements.
- Expected input and output validation

## 1.3. Regression Testing

- After adding new scripts or fixing bugs, regression tests were rerun to ensure existing functionalities were not broken by changes.

## 1.4. Functional testing

∉ Each feature (Register, Login, Add to Cart) was tested against its **functional specifications** to confirm that it performs as intended under different input scenarios.

## 2. Resources & Environment Needs

**Operating System:** Windows 10 or higher

**Browser:** Google Chrome (latest version)

**Java Development Kit (JDK):** Version 11 or above

**Eclipse IDE:** For coding and test script execution

**Maven:** For dependency management

**TestNG Plugin:** For structured test execution and reporting

**ChromeDriver:** For Selenium WebDriver automation

## 2.1. Testing Tools

### *Selenium WebDriver*

Used for **automating web-based test cases**, including registration, login, and add-to-cart flows. It interacts directly with browser elements to perform user-like actions.

### *2. TestNG*

Used for **test management and reporting**. It helped structure test cases with annotations (`@Test`, `@BeforeClass`, etc.), define priorities, and generate readable HTML test reports.

### *3. Eclipse IDE*

Used as the **integrated development environment** for writing, managing, and executing test scripts.

### *4. GitHub*

Used for **version control and project hosting**, allowing easy collaboration and showcasing of the project repository.

### *5. Chrome Developer Tools*

Used for **inspecting web elements, debugging, and identifying locators (IDs, XPaths, CSS selectors)** used in Selenium scripts.