User Guide for Demo OpenCart Testing Project

1. Introduction

This document provides a detailed guide on how to use the Selenium + TestNG automation framework designed for testing the **OpenCart demo site** (https://demo.opencart.com).
It covers setup instructions, framework structure, execution steps, and guidelines for adding new test cases.

2. Objective

The purpose of this framework is to automate functional testing of OpenCart's key features such as:

- User registration and login
- Product search and add-to-cart
- Checkout process
- Product comparison and wishlist

3. Tools & Technologies

Component	Description
Language	Java
Automation Tool	Selenium WebDriver
Testing Framework	TestNG
Build Tool	Maven
IDE	IntelliJ IDEA
Reporting	Extent Reports / TestNG HTML reports
Browser Drivers	ChromeDriver, EdgeDriver, Firefox
Target Website	demo.opencart.com

4. Framework Architecture

```
OpenCart_Automation/
│
├── src/
│   ├── main/
│   │   └── java/
│   │       └── utilities/
│   │           ├── ConfigReader.java
│   │           ├── ExcelUtils.java
│   │           └── BrowserFactory.java
│   │
│   └── test/
│       └── java/
│           ├── base/
│           │   └── BaseTest.java
│           ├── pages/
│           │   ├── HomePage.java
│           │   ├── LoginPage.java
│           │   ├── ProductPage.java
│           │   └── CartPage.java
│           ├── testcases/
│           │   ├── TC_LoginTest.java
│           │   ├── TC_AddToCartTest.java
│           │   └── TC_SearchProductTest.java
│           └── listeners/
│               └── TestListener.java
│
```

├── testng.xml

├── pom.xml

└── config.properties

5. Framework Components Overview

5.1 BaseTest.java

- Initializes WebDriver before each test.
- Loads configuration from config.properties.
- Manages test setup and teardown.

5.2 Page Object Classes (POM)

Each page on OpenCart has a corresponding Java class that defines:

- Web elements using @FindBy
- Page actions (e.g., login(), searchProduct(), addToCart())

5.3 Test Cases (TestNG Classes)

Each test case extends BaseTest and uses page objects to perform steps.

5.4 Utilities

Utility classes for:

- Reading configuration and test data.
- Handling browser setup.
- Capturing screenshots on failure.

5.5 TestNG.xml

Defines suite configuration, parallel execution, and test grouping.

6. Setup Instructions

Step 1: Clone or Download the Project

git clone https://github.com/yourrepo/OpenCart_Automation.git

Step 2: Open in IDE

- Open the project in **Eclipse** or **IntelliJ IDEA**.

Step 3: Update Configuration

Edit config.properties:

browser = chrome

url = https://demo.opencart.com/

email = testuser@mail.com

password = 12345

Step 4: Install Dependencies

In terminal:

mvn clean install

Step 5: Run Tests

- Using TestNG XML: Right-click testng.xml → Run as **TestNG Suite**
- Or using Maven command:

mvn test

7. Reporting

After execution, reports will be available at:

/test-output/index.html

or

/ExtentReports/ExtentReport.html

8. Adding New Test Cases

Step 1: Create a new class

In testcases/, create a new test class (e.g., TC_RegisterTest.java).

Step 2: Extend BaseTest

Ensure the test inherits WebDriver setup from BaseTest.

Step 3: Use Page Objects

Step 4: Add to testng.xml

Add the class reference inside <classes>:

<class name="testcases.TC_RegisterTest"/>

Step 5: Run the suite again

Execute through IDE or Maven to validate new test addition.

9. Best Practices

- Use **Page Object Model (POM)** for maintainability.
- Keep locators in page classes, not in test scripts.
- Reuse utility functions for browser setup, waits, and reporting.
- Follow naming conventions:
  - Pages → *Page.java
  - Tests → TC_*Test.java
- Use **assertions** for clear pass/fail validation.
- Maintain test data externally (Excel, JSON, or Properties file).

10. Troubleshooting

Issue	Possible Cause	Solution
WebDriver not launching	**Wrong path or outdated driver**	**Update driver in BrowserFactory**
Element not found	**Page not loaded fully**	**Use explicit waits**
Report not generating	**Missing report path**	**Check ExtentReport initialization**
Config not loaded	**Wrong file path**	**Verify config.properties location**

11. Future Enhancements

- Integrate with **Jenkins** for CI/CD
- Add **parallel execution** on multiple browsers
- Include **data-driven** testing using Excel/JSON
- Integrate **Allure Reports** for detailed visualization
- Use **Docker + Selenium Grid** for distributed testing

12. Conclusion

This framework provides a modular, reusable, and scalable approach to automate **OpenCart** functional testing.
By following this guide, testers can:

- Understand framework structure

- Configure and execute tests easily
- Extend coverage by adding new test cases with minimal effort