**nopCommerce Checkout Flow Automation**

This project automates the checkout process on [nopCommerce](https://practicesoftwaretesting.com/) using Selenium WebDriver, Java, and TestNG. The objective is to verify that the checkout process runs smoothly, from selecting a product to completing an order.

**Project Overview**

The automation script performs the following steps using a data-driven approach:

1. Select a product and adds it to the cart.
2. Proceeds to checkout.
3. Enters shipping and billing information.
4. Completes the order.

**Project Structure**

* **Programming Language**: Java
* **Frameworks & Libraries**:
  + **Selenium WebDriver** for browser automation.
  + **TestNG** for test execution, organization, and reporting.
  + **Page Object Model (POM)** to maintain a clear separation between test scripts and the UI layer.
  + **Allure Reporting** for comprehensive test reports.
  + **ITestListener** from TestNG for custom reporting and test tracking.
* **Patterns Applied**:
  + **Fluent Pattern** for clean, chainable methods in POM classes.
  + **Anonymous Objects** to keep the code concise and readable.

**Prerequisites**

* **Java** (JDK 8 or above)
* **IntelliJ IDEA** (or any Java-compatible IDE)
* **Maven** for dependency management
* **Allure** for test reporting
* **WebDriverManager** for the target browser (e.g., ChromeDriver)

**Project Components**

* **src/main/java/pages/**: Contains POM classes for each page involved in the checkout process.
* **src/test/java/tests/**: Contains TestNG test classes for executing the checkout automation flow.
  + **ITestListener implementation** for enhanced logging and tracking.
* **src/main/java/utils/**: Contains helper utilities, including:
  + **DataProvider** to supply data-driven inputs for various fields.

**Test Cases**

**Checkout Flow Test**

1. **Add Product to Cart**: Select a product and validates that it is successfully added to the cart.
2. **Proceed to Checkout**: Navigates to the checkout page and ensures all elements are correctly loaded.
3. **Enter Shipping and Billing Information**: Populates checkout fields with data-driven values.
4. **Complete Order**: Verifies successful order placement and checks for order confirmation.

**Design Patterns**

1. **Page Object Model (POM)**: Separate classes for each page reduce redundancy and maintainability issues.
2. **Fluent Pattern**: Uses chainable methods for better readability and flow.
3. **Anonymous Objects**: Simplifies code by reducing boilerplate.

**Reporting and Logging**

* **Allure Reporting**: Provides detailed test execution reports.
* **ITestListener**: Implements custom logging to capture test success, failure, and skip events.