Automate an E-Commerce Web Application writeup

The provided code is a simple test automation script written in Java using Selenium WebDriver and TestNG testing framework. The test script aims to automate interactions with an e-commerce web application (specifically, the "Flipkart" website). Let's break down the code and explain its functionality:

1. Maven Configuration (pom.xml):
The `pom.xml` file is a Maven Project Object Model (POM) file that defines the project's configuration, including its dependencies. In this case, the project has the following main dependencies:

- `selenium-java`: The Selenium Java library that provides core WebDriver functionality.
- `selenium-chrome-driver`: The WebDriver executable for Google Chrome browser.
- `selenium-firefox-driver`: The WebDriver executable for Mozilla Firefox browser.
- `logback-classic`: A logging framework for capturing logs generated during test execution.
- `testng`: The TestNG testing framework for test management and assertion capabilities.

2. Test Class (MyTest.java):
The `MyTest` class is a TestNG test class containing the test methods. It's responsible for setting up the WebDriver, performing the test, and tearing down the WebDriver after the test.

3. Test Setup (@BeforeClass):
The `setUp()` method annotated with `@BeforeClass` is executed before any test methods in the class. It sets up the WebDriver for Chrome browser by specifying the path to the `chromedriver.exe` using `System.setProperty()`. Alternatively, the code for Microsoft Edge browser setup is commented out in the example.

4. Test Method (@Test):
The `testExample()` method annotated with `@Test` represents the actual test scenario. In this example, the code navigates to the Flipkart website (`https://www.flipkart.com/`) and then sleeps for 80 seconds using `Thread.sleep(80000)`. The sleep is not a recommended way to wait for elements to appear; it's better to use explicit waits for synchronization.

5. Test Teardown (@AfterClass):
The `tearDown()` method annotated with `@AfterClass` is executed after all test methods in the class. It closes the WebDriver after the test execution is completed.

6. Test Execution:
To execute the test, TestNG will run the test methods in the `MyTest` class, which in turn will perform the specified actions on the Flipkart website. However, in the provided code, the test actions are commented out for demonstration purposes. Instead, there is a sleep statement, which waits for a specified time.

Please note that the commented-out test actions (lines 33 to 37) represent an example of how you can interact with elements on the web page, such as finding input fields by their IDs, entering text, and clicking buttons. You can uncomment and modify these lines to create meaningful test scenarios and assertions.

To run the test, you can use an IDE that supports TestNG integration (e.g., IntelliJ IDEA, Eclipse) or execute the test through the command line using Maven and TestNG plugins.