Automate an E-Commerce Web Application

FlipkartTest.java:

```
package com.ecommerce.test;

import java.time.Duration;
import java.util.List;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.edge.EdgeOptions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

public class FlipkartTest {

private WebDriver driver;


@BeforeMethod
public void setUp() {
ChromeOptions options = new ChromeOptions();
options.addArguments("--start-maximized");
driver = new ChromeDriver(options);
driver.manage().timeouts().~~implicitlyWait~~(10, TimeUnit.SECONDS);


}
```

```
@Test(priority = 1) public void
testPageLoadTime() { long startTime =
System.currentTimeMillis();
driver.get("https://www.flipkart.com/");
long endTime = System.currentTimeMillis();
long pageLoadTime = endTime - startTime;
System.out.println("Page load time: " + pageLoadTime + " milliseconds");
}


@Test(priority = 2) public void
testSearchProduct() {
driver.get("https://www.flipkart.com/");
WebElement searchBox = driver.findElement(By.name("q"));
searchBox.sendKeys("iPhone 13 Mobile"); searchBox.sendKeys(Keys.ENTER);
// You can add some assertions to verify the search results page
System.out.println("Search performed for 'iPhone 13 Mobile'.");
}


@Test(priority = 3)
public void testImagesLoadedAndVisible() { driver.get("https://www.flipkart.com/");


WebElement searchBox = driver.findElement(By.name("q"));
searchBox.sendKeys("iPhone 13 Mobile"); searchBox.sendKeys(Keys.ENTER);


// Wait for images to load
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.presenceOfAllElementsLocatedBy(By.tagName("img")));


// Find all image elements on the page
List<WebElement> imageElements = driver.findElements(By.tagName("img"));


// Get the screen height
Number screenHeight = (Number) ((JavascriptExecutor) driver).executeScript("return
window.innerHeight");

for (WebElement imageElement : imageElements) {
// Get the image position relative to the viewport's bottom
Number imagePositionBottom = (Number) ((JavascriptExecutor) driver)
.executeScript("return arguments[0].getBoundingClientRect().bottom", imageElement);


// Check if the bottom position of the image is within the screen height if
(imagePositionBottom.doubleValue() >= 0
&& imagePositionBottom.doubleValue() <= screenHeight.doubleValue()) {
System.out.println("Image is loaded and visible within the screen height."); }
else {
System.out.println("Image is not visible within the screen height.");
}
}
}


@Test(priority = 4) public void
testScrollFeature() {
driver.get("https://www.flipkart.com/");


// Get the initial page height
Long initialPageHeight = (Long) ((JavascriptExecutor) driver)
.executeScript("return document.body.scrollHeight");


// Scroll down to the bottom of the page to load more content JavascriptExecutor
jsExecutor = (JavascriptExecutor) driver;
jsExecutor.executeScript("window.scrollTo(0, document.body.scrollHeight)");


// Wait for a short time to allow more content to load try
{
Thread.sleep(5000);
} catch (InterruptedException e) { e.printStackTrace();
}


// Get the page height after scrolling
Long scrolledPageHeight = (Long) jsExecutor.executeScript("return
document.body.scrollHeight");
```

```
// Check if the page height has increased after scrolling if
(scrolledPageHeight > initialPageHeight) {
System.out.println("The page has a scroll feature.");
} else {
System.out.println("The page does not have a scroll feature.");
}
}


@Test(priority = 5)
public void testRefreshFrequency() { driver.get("https://www.flipkart.com/");


// Scroll down to the bottom of the page to load more content JavascriptExecutor
jsExecutor = (JavascriptExecutor) driver;
jsExecutor.executeScript("window.scrollTo(0, document.body.scrollHeight)");


// Wait for a short time to allow more content to load try
{
Thread.sleep(5000);
} catch (InterruptedException e) { e.printStackTrace();
}


// Get the current page source after waiting
String pageSourceAfterWait = driver.getPageSource();


// Scroll down again to the bottom to load more content
jsExecutor.executeScript("window.scrollTo(0, document.body.scrollHeight)");


// Wait for a short time again try
{
Thread.sleep(5000);
} catch (InterruptedException e) {
e.printStackTrace();
}


// Get the page source again after the second wait
String pageSourceAfterSecondWait = driver.getPageSource();
```

```
// Compare the two page sources to check if the content was refreshed if
(pageSourceAfterWait.equals(pageSourceAfterSecondWait)) {
System.out.println("The content is not refreshed while scrolling."); }
else {
System.out.println("The content is refreshed while scrolling.");
}
}


@Test(priority = 6) public void
testLazyLoading() {
driver.get("https://www.flipkart.com/");


// Scroll down to trigger lazy loading of images
JavascriptExecutor jsExecutor = (JavascriptExecutor) driver;
jsExecutor.executeScript("window.scrollTo(0, document.body.scrollHeight)");


// Wait for a short time to allow images to load try
{
Thread.sleep(5000);
} catch (InterruptedException e) { e.printStackTrace();
}


// Find all image elements on the page
List<WebElement> imageElements = driver.findElements(By.tagName("img"));

for (WebElement imageElement : imageElements) { //
Check if the image is displayed on the page
if (imageElement.isDisplayed()) {
System.out.println("Image is lazy loaded and displayed on the page.");
} else {
System.out.println("Image is not lazy loaded and displayed on the page.");
}
}
}


@Test(priority = 7)
```

```
public void testScrollToBottom() { driver.get("https://www.flipkart.com/");


// Scroll to the bottom of the page
JavascriptExecutor jsExecutor = (JavascriptExecutor) driver;
jsExecutor.executeScript("window.scrollTo(0, document.body.scrollHeight)");


// Wait for a short time to allow scrolling to complete try
{
Thread.sleep(2000);
} catch (InterruptedException e) { e.printStackTrace();
}


// Check if the page is scrolled to the bottom
long pageHeight = (Long) jsExecutor.executeScript("return document.body.scrollHeight");
long windowHeight = (Long) jsExecutor.executeScript("return window.innerHeight");

if (pageHeight == windowHeight) {
System.out.println("The page has scrolled to the bottom.");
} else {
System.out.println("The page has not scrolled to the bottom.");
}
}


@Test(priority = 8)
public void testDifferentBrowsersAndResolutions() {
// Open the website with default browser and screen resolution
// Microsoft Edge
EdgeOptions edgeOptions = new EdgeOptions(); driver
= new EdgeDriver(edgeOptions);
driver.get("https://www.flipkart.com/");


// Your test logic here to perform actions and assertions on the website


// Now, let's test different screen resolutions
testWithScreenResolution(1366, 768);
testWithScreenResolution(1920, 1080); //
Add more screen resolutions if needed
```

```
// Test with different browsers (Firefox and Edge) testWithEdge();
}

private void testWithScreenResolution(int width, int height) {
Dimension resolution = new Dimension(width, height);
driver.manage().window().setSize(resolution);


 // Open the website with the specified screen resolution
driver.get("https://www.flipkart.com/");


 // Your test logic here to perform actions and assertions on the website


 // Print the current screen resolution being tested
 System.out.println("Testing with screen resolution: " + width + "x" + height);


}


private void testWithEdge() {


// Open the website with the Edge browser driver.get("https://www.flipkart.com/");


// Your test logic here to perform actions and assertions on the website


// Example: Check if the search box is displayed
WebElement searchBox = driver.findElement(By.name("q"));
if (searchBox.isDisplayed()) {
System.out.println("Search box is displayed in Microsoft Edge."); }
else {
System.out.println("Search box is not displayed in Microsoft Edge.");
}
```

```
// Example: Perform a search and verify search results
searchBox.sendKeys("iPhone 13 Mobile"); searchBox.sendKeys(Keys.ENTER);


// ... Perform further assertions or actions specific to Edge ...
}


@AfterMethod
public void tearDown() { // Close
the browser after each test
if (driver != null) {
driver.quit();
}
}
}
```

Pom.xml :

```
<?xml version="1.0" encoding="UTF-8"?>


<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion> <groupId>hello-selenium</groupId>
<artifactId>hello-selenium</artifactId>
<version>0.0.1-SNAPSHOT</version>


<name>hello-selenium</name>
<!-- FIXME change it to the project's website -->
<url>http://www.example.com</url>


<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven.compiler.source>1.7</maven.compiler.source>
```

```
<maven.compiler.target>1.7</maven.compiler.target>
</properties>


<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.11</version>
<scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
<groupId>org.seleniumhq.selenium</groupId>
<artifactId>selenium-java</artifactId>
<version>4.10.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-chrome-driver -
>
<dependency>
<groupId>org.seleniumhq.selenium</groupId>
<artifactId>selenium-chrome-driver</artifactId>
<version>4.10.0</version>
</dependency>
<dependency>
<groupId>org.testng</groupId>
<artifactId>testng</artifactId>
<version>7.8.0</version> <scope>compile</scope>
</dependency>
<dependency>
<groupId>com.quiz</groupId>
<artifactId>Online_Quiz</artifactId>
<version>0.0.1-SNAPSHOT</version>
</dependency>
</dependencies>


<build>
<pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be
moved to parent pom) -->
<plugins>
<!-- clean lifecycle, see
https://maven.apache.org/ref/current/mavencore/lifecycles.html#clean_Lifecycle -->
```

```
<plugin>
<artifactId>maven-clean-plugin</artifactId>
<version>3.1.0</version>
</plugin>
<!-- default lifecycle, jar packaging: see
https://maven.apache.org/ref/current/mavencore/default-
bindings.html#Plugin_bindings_for_jar_packaging -->
<plugin>
<artifactId>maven-resources-plugin</artifactId>
<version>3.0.2</version>
</plugin>
<plugin>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.8.0</version>
</plugin>
<plugin>
<artifactId>maven-surefire-plugin</artifactId>
<version>2.22.1</version>
</plugin>
<plugin>
<artifactId>maven-jar-plugin</artifactId>
<version>3.0.2</version>
</plugin>
<plugin>
<artifactId>maven-install-plugin</artifactId>
<version>2.5.2</version>
</plugin>
<plugin>
<artifactId>maven-deploy-plugin</artifactId>
<version>2.8.2</version>
</plugin>
<!-- site lifecycle, see
https://maven.apache.org/ref/current/mavencore/lifecycles.html#site_Lifecycle -->
<plugin>
<artifactId>maven-site-plugin</artifactId>
<version>3.7.1</version>
</plugin>
<plugin>
<artifactId>maven-project-info-reports-plugin</artifactId>
<version>3.0.0</version>
</plugin>
</plugins>
</pluginManagement>
```

```
</build>
</project>
```