EcommerceAutomationTest.java

```
package com.example;

import static org.testng.Assert.assertEquals;

import java.io.File;
import java.io.IOException;
import java.util.List;
import java.util.concurrent.TimeUnit;

import org.apache.commons.io.FileUtils;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Reporter;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class EcommerceAutomationTest {
```

```
	private int countOfScreenshots = 0;
	private WebDriver webDriver;
	private String homeTitle = "Online Shopping Site for Mobiles, Electronics, Furniture, Grocery, Lifestyle, Books & More. Best Offers!";

	@Parameters({ "driverName", "driverPath" })
	@BeforeTest
	public void setup(String driverName, String driverPath) {
		System.setProperty("webdriver." + driverName + ".driver", driverPath);
		switch (driverName) {
		case "edge":
			webDriver = new EdgeDriver();
			break;
		case "chrome":
			webDriver = new ChromeDriver();
			break;
		case "gecko":
			webDriver = new FirefoxDriver();
			break;
		}
		webDriver.manage().window().maximize();
		webDriver.manage().timeouts().implicitlyWait(10000, TimeUnit.MILLISECONDS);
	}

	@Parameters({ "websiteURL", "driverName", "fileLoc" })
	@Test
	public void HomePageTest(String websiteURL, String driverName, String fileLoc) throws IOException {
		long startTime = System.currentTimeMillis();
		webDriver.get(websiteURL);
		long endTime = System.currentTimeMillis();
		Reporter.log("Page load time: " + (endTime - startTime) + " milliseconds");
```

```
		Reporter.log("Reached Flipkart Home page with " + driverName + " driver");
		assertEquals(webDriver.getTitle(), homeTitle);
		Reporter.log("Home title matched");
		takeScreenshot(webDriver, countOfScreenshots++, driverName, fileLoc);
		try {
			WebElement popupElement =
webDriver.findElement(By.xpath("//button[@class='_2KpZ6l _2doB4z']"));
			if (popupElement != null) {
				takeScreenshot(webDriver, countOfScreenshots++, driverName,
fileLoc);
				popupElement.click();
				Reporter.log("Pop-up has appeared, and it has been closed");
			}
		} catch (NoSuchElementException exc) {
			takeScreenshot(webDriver, countOfScreenshots++, driverName, fileLoc);
			Reporter.log("Pop-up has not appeared.");
		}
	}

	@Parameters({ "driverName", "fileLoc" })
	@Test(priority = 2)
	public void MobileSearchTest(String driverName, String fileLoc) throws IOException {
		Reporter.log("Clicking on Mobiles link");
		WebElement mobilesLink = webDriver.findElement(By.linkText("Mobiles"));
		mobilesLink.click();

		Reporter.log("Reached mobiles Page");
		takeScreenshot(webDriver, countOfScreenshots++, driverName, fileLoc);

		WebElement searchInput = webDriver.findElement(By.xpath("//input[@type='text'
and @name='q']"));
```

```
            searchInput.sendKeys("iPhone 13");
            Reporter.log("Sending Keys \"iPhone 13\"");
            takeScreenshot(webDriver, countOfScreenshots++, driverName, fileLoc);

            WebElement searchButton =
webDriver.findElement(By.xpath("//button[@type='submit']"));
            searchButton.click();
            Reporter.log("Clicking on search option");
        }

        @Parameters({ "driverName", "fileLoc" })
        @Test(priority = 3)

        public void ScrollFeatureTest(String driverName, String fileLoc) throws IOException {

            Reporter.log("Reached iPhone 13 Page");
            Reporter.log("Testing scroll feature");
            boolean isBottomReached = scrollToBottom(webDriver);
            takeScreenshot(webDriver, countOfScreenshots++, driverName, fileLoc);
            if (isBottomReached) {
                Reporter.log("Successfully navigated to the bottom of the page.");
            } else {
                Reporter.log("Navigation to the bottom of the page failed.");
            }
        }

        @Parameters({ "driverName", "fileLoc" })
        @Test(priority = 4)
        public void ImageVisibilityTest(String driverName, String fileLoc) throws IOException {
            Reporter.log("Checking image visibility...");
```

```
JavascriptExecutor js = (JavascriptExecutor) webDriver;
js.executeScript("window.scrollTo(0, 0);");



try {
		Thread.sleep(2000);
} catch (InterruptedException e) {
		e.printStackTrace();
}


Reporter.log("Verify the frequency of content refresh while scrolling");
  String jsScript = "let imageCount = 0;" +
 "const observer = new MutationObserver((mutationsList) => {" +
 "    for (const mutation of mutationsList) {" +
 "        if (mutation.type === 'childList') {" +
 "            for (const node of mutation.addedNodes) {" +
 "                if (node.tagName === 'IMG' && node.className.includes('_396cs4')) {" +
 "                    imageCount++;" +
 "                }" +
 "            }" +
 "        }" +
 "    }" +
 "});" +
 "observer.observe(document, { childList: true, subtree: true });" +
 "setTimeout(() => {" +
 "    observer.disconnect();" +
 "    console.log('Images displayed: ' + imageCount);" +
 "}, 2000);";



Reporter.log("Executing the JavaScript code...");
```

```
                js.executeScript(jsScript);


                Reporter.log("Checking if images are visible on the screen...");

                Reporter.log("Verifying that the image is downloaded just before the user scrolls to its\n"

                                + "position and gets displayed in time...");

                List<WebElement> productImages = webDriver.findElements(By.cssSelector("img._396cs4"));

      for (WebElement image : productImages) {

         boolean isInViewport = (boolean) js.executeScript("const rect = arguments[0].getBoundingClientRect();"

               + "return rect.top >= 0 && rect.left >= 0 && rect.bottom <= (window.innerHeight || document.documentElement.clientHeight) && rect.right <= (window.innerWidth || document.documentElement.clientWidth);",

               image);


         if (isInViewport) {

            long startTime = System.currentTimeMillis();

            image.getLocation();

            long endTime = System.currentTimeMillis();

            long timeTaken = endTime - startTime;

            Reporter.log("Time taken to display image: " + timeTaken + " milliseconds.");


            Reporter.log("Image is visible within the screen height.");

         } else {

             long startTime = System.currentTimeMillis();

              image.getLocation();

              long endTime = System.currentTimeMillis();

              long timeTaken = endTime - startTime;

              Reporter.log("Time taken to load image: " + timeTaken + " milliseconds.");


            Reporter.log("Image is not visible within the screen height.");

         }
```

```
    }
            takeScreenshot(webDriver, countOfScreenshots++, driverName, fileLoc);
        }

        @AfterTest
        public void tearDown() {
            webDriver.quit();
        }

        private boolean scrollToBottom(WebDriver webDriver) {
            JavascriptExecutor js = (JavascriptExecutor) webDriver;
            long initialHeight = (Long) js.executeScript("return window.innerHeight;");
            long pageHeight = (Long) js.executeScript("return document.body.scrollHeight;");

            int maxScrollAttempts = 10;
            int currentScrollAttempts = 0;

            while (currentScrollAttempts < maxScrollAttempts) {
                js.executeScript("window.scrollTo(0, document.body.scrollHeight);");
                try {
                    Thread.sleep(2000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

                long newPageHeight = (Long) js.executeScript("return
document.body.scrollHeight;");
                if (newPageHeight == pageHeight) {
                    break;
                }
```

```
                pageHeight = newPageHeight;

                currentScrollAttempts++;

            }


            boolean isBottomReached = (initialHeight <= pageHeight);

            return isBottomReached;

        }


        public static void takeScreenshot(WebDriver wd, int count, String driverName, String fileLoc)
throws IOException {

            File file = ((TakesScreenshot) wd).getScreenshotAs(OutputType.FILE);

            FileUtils.copyFile(file, new File(fileLoc + driverName + "_test_" + count + ".png"));

        }

}
```

```
<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>Flipkart-Automation</groupId>
        <artifactId>Flipkart-Automation</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>Flipkart-Automation</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <maven.compiler.source>1.8</maven.compiler.source>
                <maven.compiler.target>1.8</maven.compiler.target>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>3.8.1</version>
                        <scope>test</scope>
                </dependency>
                <dependency>
```

```
                <groupId>org.slf4j</groupId>
                <artifactId>slf4j-api</artifactId>
                <version>2.0.6</version>
            </dependency>
            <dependency>
                <groupId>AppAutoWithTestNG</groupId>
                <artifactId>Web-App-Automation-and-Testing</artifactId>
                <version>0.0.1-SNAPSHOT</version>
            </dependency>
        </dependencies>
</project>
```

Testing.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Flipkart-Automation">
        <parameter name="fileLoc"
                value="C:\\Users\\DELL\\Desktop\\Java_Practicing\\Flipkart-
Automation\\Screenshots"></parameter>
        <parameter name="websiteURL" value="https://www.flipkart.com"></parameter>
        <test name="Testing on Chrome Browser" parallel="true">
                <parameter name="driverName" value="chrome"></parameter>
                <parameter name="driverPath"
                        value="C:\Users\DELL\Downloads\chromedriver.exe"></parameter>
                <classes>
                        <class name="com.example.EcommerceAutomationTest" />
                </classes>
        </test>
        <test name="Testing on Edge Browser" parallel="true">
                <parameter name="driverName" value="edge"></parameter>
                <parameter name="driverPath"
                        value="C:\Users\DELL\Downloads\msedgedriver.exe"></parameter>
                <classes>
                        <class name="com.example.EcommerceAutomationTest" />
                </classes>
        </test>
        <test name="Testing on Firefox Browser" parallel="true">
                <parameter name="driverName" value="gecko"></parameter>
                <parameter name="driverPath"
                        value="C:\Users\DELL\Downloads\geckodriver.exe"></parameter>
                <classes>
                        <class name="com.example.EcommerceAutomationTest" />
                </classes>
        </test>
</suite>
```