SOURCE CODE FOR PHASE END PROJECT AUTOMATE AN ECOMMERCE WEB APPLICATION

BasePage.java: (src/main/java/pages)

```
package pages;

import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import utils.DriverManager;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.concurrent.atomic.AtomicInteger;

public class BasePage {
  protected WebDriver driver;
  private static AtomicInteger screenshotCounter = new
AtomicInteger(0);

  /**
   * This is the constructor
   */
  public BasePage() {
    this.driver = DriverManager.getDriver();
  }

  /**
   * This method captures a screenshot and saves it with a
unique name
   *
   * @param screenshotName The name of the screenshot
   */
  public void captureScreenshot(String screenshotName) {
    TakesScreenshot ts = (TakesScreenshot) driver;
    File source = ts.getScreenshotAs(OutputType.FILE);

    Path destination = Paths.get("screenshots",
        "screenshot_" +
screenshotCounter.getAndIncrement() + "_" + screenshotName
+ ".png");

    try {
      Files.createDirectories(destination.getParent());
      Files.copy(source.toPath(), destination);
      System.out.println("Screenshot captured and saved: "
+ destination);
    } catch (IOException e) {
      System.out.println("Failed to capture screenshot: "
+ e.getMessage());
    }
  }
}
```

FlipkartHomePage.java: (src/main/java/pages)

```
package pages;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import utils.DriverManager;

public class FlipkartHomePage {
  private WebDriver driver;

  private final String baseUrl =
"https://www.flipkart.com/";
  private final By searchInput =
By.cssSelector("input[name='q']");
  private final By searchButton =
By.cssSelector("button[type='submit']");
  String searchOptions = "(//div[contains(text(),'" +
"PRODUCT_SEARCH_OPTION" + "')]/parent::a/parent::div)[1]";

  /**
   * This is the constructor of the current page
   */
  public FlipkartHomePage() {
    this.driver = DriverManager.getDriver();
  }

  /**
   * This method gets the home page loading time
   *
   * @return load time in long
   */
  public long getHomepageLoadTime() {
    driver.get(baseUrl);
    System.out.println("Application is launched. ");
    System.out.println("Application URL is: " + baseUrl);

    return (Long) ((JavascriptExecutor)
driver).executeScript(
        "return (window.performance.timing.loadEventEnd -
window.performance.timing.navigationStart);");
  }

  /**
   * This method is used to search for a product
   *
   * @param productName
   * @throws InterruptedException
   */
  public void searchForProduct(String productName) throws
InterruptedException {
    driver.findElement(searchInput).sendKeys(Keys.chord(Ke
ys.ESCAPE));
    System.out.println("Pressed escape key to close the
login popup if it is displayed");

    driver.findElement(searchInput).sendKeys(productName);
    System.out.println("Search text is entered as : " +
productName);

    String searchOptionsProductXpathString =
searchOptions.replaceAll("PRODUCT_SEARCH_OPTION",
        productName.toLowerCase());
    Thread.sleep(5000);
    String searchProduct =
driver.findElement(By.xpath(searchOptionsProductXpathStrin
g)).getText();
    driver.findElement(By.xpath(searchOptionsProductXpathS
tring)).click();
    System.out.println("Searched product is selected form
list option as: " + searchProduct);

    Thread.sleep(4000);
    driver.findElement(searchButton).click();
    System.out.println("Search button is clicked on to
find the product results");
  }

}
```

SearchResultsPage.java : (src/main/java/pages)

```
package pages;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import utils.DriverManager;

import java.util.List;

public class SearchResultsPage {
  private WebDriver driver;
  private final By productResults =
By.cssSelector("div[data-id]");
  private final By imageSelector = By.xpath(
      "//span[contains(text(),'results
for')]/ancestor::div[4]/div[starts-
with(@class,'_1AtVbE')]/descendant::img[@loading='eager']"
);

  /**
   * This is the constructor of the current page
   */
  public SearchResultsPage() {
    this.driver = DriverManager.getDriver();
  }

  /**
   * This method is used to verify if search results are
displayed or not
   *
   * @return !isEmpty - boolean value
   */
  public boolean isSearchResultsDisplayed() {
    boolean isEmpty =
driver.findElements(productResults).isEmpty();
    return !isEmpty;
  }

  /**
   * This method is used to get the search results
   *
   * @return productResultsList
   * @throws InterruptedException
   */
  public List<WebElement> getSearchResults() throws
InterruptedException {
    List<WebElement> productResultsList =
driver.findElements(productResults);
    Thread.sleep(5000);
    return productResultsList;
  }

  /**
   * This method is used to scroll to the element
   *
   * @param element
   * @throws InterruptedException
   */
  public void scrollToElement(WebElement element) throws
InterruptedException {
    Thread.sleep(500);
    ((JavascriptExecutor)
driver).executeScript("arguments[0].scrollIntoView(true);"
, element);
```

```
  }

  /**
   * This method is used to check if the product image is
displayed in the search
   * results
   *
   * @param product
   * @return
   * @throws InterruptedException
   */
  public boolean isImageDisplayed(WebElement product)
throws InterruptedException {
    Thread.sleep(1000);
    return (Boolean) ((JavascriptExecutor)
driver).executeScript(
        "return arguments[0].complete && typeof
arguments[0].naturalWidth != 'undefined' &&
arguments[0].naturalWidth > 0;",
        product.findElement(imageSelector));
  }

  /**
   * This method is used to check if the page has scroll
feature once the results
   * are displayed upon search
   *
   * @return
   */
  public boolean hasScrollFeature() {
    return (boolean) ((JavascriptExecutor) driver)
        .executeScript("return
document.documentElement.scrollHeight >
document.documentElement.clientHeight;");
  }

  /**
   * This method is used to scroll to the bottom of the
page
   */
  public void scrollToBottom() {
    ((JavascriptExecutor)
driver).executeScript("window.scrollTo(0,
document.body.scrollHeight);");
  }

  /**
   * This method is used to get the number of visible
products
   *
   * @return product size
   */
  public int getNumberOfVisibleProducts() {
    return driver.findElements(productResults).size();
  }

  /**
   * This method is used to get the number of visible
images
   *
   * @return images count
   */
  public int getNumberOfVisibleImages() {
    List<WebElement> visibleImages =
driver.findElements(imageSelector);
    return (int)
visibleImages.stream().filter(WebElement::isDisplayed).cou
nt();
  }

  /**
   * This method is used to get the total number of images
   *
   * @return images size
   */
  public int getTotalNumberOfImages() {
    return driver.findElements(imageSelector).size();
  }

  /**
   * This method is used to verify if the cursor at the
bottom page
   *
```

```
    * @return
    */
  public boolean isAtBottomOfPage() {
    double scrollY = (Double) ((JavascriptExecutor)
driver).executeScript("return window.scrollY;");
    long innerHeight = (Long) ((JavascriptExecutor)
driver).executeScript("return window.innerHeight;");
    long scrollHeight = (Long) ((JavascriptExecutor)
driver)
        .executeScript("return
document.documentElement.scrollHeight;");
    return scrollY + innerHeight >= scrollHeight;
  }

}
```

CreateExcelFile.java : (src/main/java/utils)

```
package utils;

import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.FileOutputStream;
import java.io.IOException;

public class CreateExcelFile {
    public static void main(String[] args) {
        try (Workbook workbook = new XSSFWorkbook()) {
            Sheet sheet = workbook.createSheet("Sheet1");
            Row headerRow = sheet.createRow(0);

            String[] headers = {"Name", "Age",
"Occupation"};

            for (int i = 0; i < headers.length; i++) {
                Cell cell = headerRow.createCell(i);
                cell.setCellValue(headers[i]);
            }

            String[][] data = {
                    {"John Doe", "30", "Engineer"},
```

```java
                    {"Jane Smith", "25", "Teacher"},
                    {"Michael Brown", "40", "Doctor"}
            };

            for (int i = 0; i < data.length; i++) {
                Row dataRow = sheet.createRow(i + 1);

                for (int j = 0; j < data[i].length; j++) {
                    Cell cell = dataRow.createCell(j);
                    cell.setCellValue(data[i][j]);
                }
            }

            try (FileOutputStream fileOut = new
FileOutputStream("testdata.xlsx")) {
                workbook.write(fileOut);
            }

            System.out.println("Excel file created
successfully!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

DriverManager.java : (src/main/java/utils)

```java
package utils;

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.CapabilityType;
import org.openqa.selenium.remote.DesiredCapabilities;

public class DriverManager {

  private static WebDriver driver;

  /**
```

```
   * This method is used to initialize the driver for
chrome driver
   */
  public static void initializeDriver() {
    ChromeOptions options = new ChromeOptions();
    options.addArguments("--incognito");
    options.addArguments("--disable-popup-blocking");
    options.setAcceptInsecureCerts(true);
    options.addArguments("--remote-allow-origins=*");
    options.addArguments("--start-maximized");

    DesiredCapabilities capabilities = new
DesiredCapabilities();
    capabilities.setJavascriptEnabled(true);
    capabilities.setCapability(CapabilityType.SUPPORTS_ALE
RTS, true);
    capabilities.setCapability(CapabilityType.SUPPORTS_LOC
ATION_CONTEXT, true);
    capabilities.setCapability(CapabilityType.SUPPORTS_APP
LICATION_CACHE, true);
    capabilities.setCapability(ChromeOptions.CAPABILITY,
options);

    driver =
WebDriverManager.chromedriver().capabilities(options).crea
te();
    System.out.println("Chrome Browser is launched");

    driver.manage().window().maximize();

  }

  public static WebDriver getDriver() {
    if (driver == null) {
      initializeDriver();
    }
    return driver;
  }

  /**
   * This method quits/close the driver if it is not null
   */
  public static void quitDriver() {
    if (driver != null) {
      driver.quit();
      driver = null;
    }
  }

}
```

FileUtils.java : (src/main/java/utils)

```
package utils;

import java.io.FileInputStream;
import java.io.IOException;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class FileUtils {
  public static Object[][] readTestData(String filePath,
String sheetName) {
    Object[][] testData = null;
    try (FileInputStream fis = new
FileInputStream(filePath); Workbook workbook = new
XSSFWorkbook(fis)) {

      Sheet sheet = workbook.getSheet(sheetName);
      int rowCount = sheet.getLastRowNum();
      int colCount = sheet.getRow(0).getLastCellNum();
      testData = new Object[rowCount][colCount];

      for (int i = 0; i < rowCount; i++) {
        for (int j = 0; j < colCount; j++) {
          testData[i][j] = sheet.getRow(i +
1).getCell(j).toString();
        }
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
    return testData;
  }
}
```

FlipkartLazyLoadingTest.java : (src/test/java/tests)

```
package tests;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.firefox.FirefoxOptions;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
import pages.*;
import utils.DriverManager;

import java.net.URL;
import java.util.List;

public class FlipkartLazyLoadingTest {
  private FlipkartHomePage homePage;
  private SearchResultsPage searchResultsPage;

  @BeforeClass
  public void setup() {
    DriverManager.initializeDriver();
    homePage = new FlipkartHomePage();
    searchResultsPage = new SearchResultsPage();
  }

  private void captureScreenshot(String screenshotName) {
    BasePage basePage = new BasePage();
    basePage.captureScreenshot(screenshotName);
  }

  @Test(priority = 1, enabled = true, description =
"Verify the flipkart hoomepage load time", groups = {
      "regression" })
  public void testFlipkartHomepageLoadTime() {
```

```java
    long loadTime = homePage.getHomepageLoadTime();
    Assert.assertTrue(loadTime < 8000, "Homepage load time
is within acceptable range.");
  }

  @Test(priority = 2, enabled = true, description =
"Verify the functionality of search product in mobile
category", groups = {
      "regression" })
  public void testSearchProductInMobileCategory() throws
InterruptedException {
    String product = "iPhone 13";
//    homePage.searchForProduct(product, "Mobile");
    homePage.searchForProduct(product);
    Assert.assertTrue(searchResultsPage.isSearchResultsDis
played(), "Search results are displayed.");
  }

  @Test(priority = 3, enabled = true, description =
"Verify the functionality of image loading and
visibility", groups = {
      "regression" })
  public void testImageLoadingAndVisibility() throws
InterruptedException {
    List<WebElement> products =
searchResultsPage.getSearchResults();
    for (WebElement product : products) {
      try {
        searchResultsPage.scrollToElement(product);
        Thread.sleep(3000);
        boolean isImageDisplayed =
searchResultsPage.isImageDisplayed(product);
        Assert.assertTrue(isImageDisplayed, "Image is
loaded and visible.");

        // Capture screenshot after verifying the image
visibility
        captureScreenshot("image_visible_" +
products.indexOf(product));
      } catch (Exception e) {
        // Handle any exceptions that might occur during
scrolling or image verification
```

```
        e.printStackTrace();
      }
    }
  }

  @Test(priority = 4, enabled = true, description =
"Verify the flipkart hoomepage load time", groups = {
      "regression" })
  public void testScrollFeatureAndContentRefresh() {
    boolean hasScrollFeature =
searchResultsPage.hasScrollFeature();
    Assert.assertTrue(hasScrollFeature, "The page has a
scroll feature.");

    int initialProductCount =
searchResultsPage.getNumberOfVisibleProducts();
    searchResultsPage.scrollToBottom();
    int finalProductCount =
searchResultsPage.getNumberOfVisibleProducts();

    Assert.assertTrue(finalProductCount ==
initialProductCount, "Content is refreshed while
scrolling.");
  }

  @Test(priority = 5, enabled = true, description =
"Verify the lazy loading functionality", groups = {
      "regression" })
  public void testLazyLoading() {
    int visibleImageCount =
searchResultsPage.getNumberOfVisibleImages();
    System.out.println("visibleImageCount is: " +
visibleImageCount);

    int totalImageCount =
searchResultsPage.getTotalNumberOfImages();
    System.out.println("totalImageCount is: " +
totalImageCount);

    Assert.assertTrue((visibleImageCount > 0) &&
(visibleImageCount == totalImageCount),
        "Images are lazily loaded as the user scrolls
down.");
  }

  @Test(priority = 6, enabled = true, description =
"Verify the funcitonality of navigating to bottom", groups
= {
      "regression" })
  public void testNavigationToBottom() {
    searchResultsPage.scrollToBottom();
    boolean isAtBottom =
searchResultsPage.isAtBottomOfPage();

    Assert.assertTrue(isAtBottom, "Successfully navigated
to the bottom of the page.");
  }

  @Test(priority = 7, enabled = true, description =
"Verify the functionality of cross browser and screen
resolution", groups = {
      "regression" })
  public void testCrossBrowserAndScreenResolution() {
    String[] browsers = { "chrome", "firefox" };
    String[] screenResolutions = { "1920x1080", "1366x768"
};

    for (String browser : browsers) {
      for (String resolution : screenResolutions) {
        try {
          // Set the URL of the Selenium Grid hub
          URL gridUrl = new URL("http://your-grid-hub-
url:4444/wd/hub");

          // Create the WebDriver instance with the
desired capabilities
          WebDriver driver =
createRemoteWebDriver(browser, resolution, gridUrl);

          // Execute your test steps here with the
'driver' instance
          // You may reuse the existing methods from the
SearchResultsPage class
```

```
          // Close the browser after the test is done
          driver.quit();
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    }
  }

  private WebDriver createRemoteWebDriver(String browser,
String resolution, URL gridUrl) {
    WebDriver driver = null;

    if ("chrome".equalsIgnoreCase(browser)) {
      ChromeOptions options = new ChromeOptions();
      options.addArguments("--window-size=" + resolution);
      driver = new RemoteWebDriver(gridUrl, options);
    } else if ("firefox".equalsIgnoreCase(browser)) {
      FirefoxOptions options = new FirefoxOptions();
      options.addArguments("--window-size=" + resolution);
      driver = new RemoteWebDriver(gridUrl, options);
    }

    return driver;
  }

  @AfterClass
  public void tearDown() {
    DriverManager.quitDriver();
  }
}
```

log4j.properties : (src/test/resources)

```
# Root logger option
log4j.rootLogger=INFO, file

# Define the appender
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.File=logs/app.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd
HH:mm:ss} %-5p %c{1}:%L - %m%n
```

pom.xml :

```
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>FlipkartLazyLoadingProject</artifactI
d>
    <version>1.0-SNAPSHOT</version>
    <name>FlipkartLazyLoadingProject</name>
    <description>To automate a real-world web
application</description>
    <properties>

    <maven.compiler.source>1.8</maven.compiler.source
>

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

    https://mvnrepository.com/artifact/io.github.boni
garcia/webdrivermanager -->
        <dependency>
            <groupId>io.github.bonigarcia</groupId>

    <artifactId>webdrivermanager</artifactId>
            <version>5.3.2</version>
        </dependency>

        <!-- Selenium -->
        <!--
    https://mvnrepository.com/artifact/org.seleniumhq
.selenium/selenium-java -->
        <dependency>

    <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <!--<version>4.10.0</version>-->
            <version>4.4.0</version>

        </dependency>

        <!-- TestNG -->
        <!--
https://mvnrepository.com/artifact/org.testng/testng
-->
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <!--<version>7.7.1</version>-->
            <version>7.5</version>
            <!--<scope>test</scope>-->
        </dependency>

        <!-- Apache POI -->
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi</artifactId>
            <version>5.0.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml</artifactId>
            <version>5.0.0</version>
        </dependency>
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>1.2.6</version>
        </dependency>

        <dependency>
            <groupId>com.automation</groupId>
            <artifactId>saucelabs</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="Suite">
  <test thread-count="5" name="Test">
    <classes>
      <class name="tests.FlipkartLazyLoadingTest" />
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```