Exp No: **Automate the testing of e-commerce applications using Selenium**
Date:

AIM:

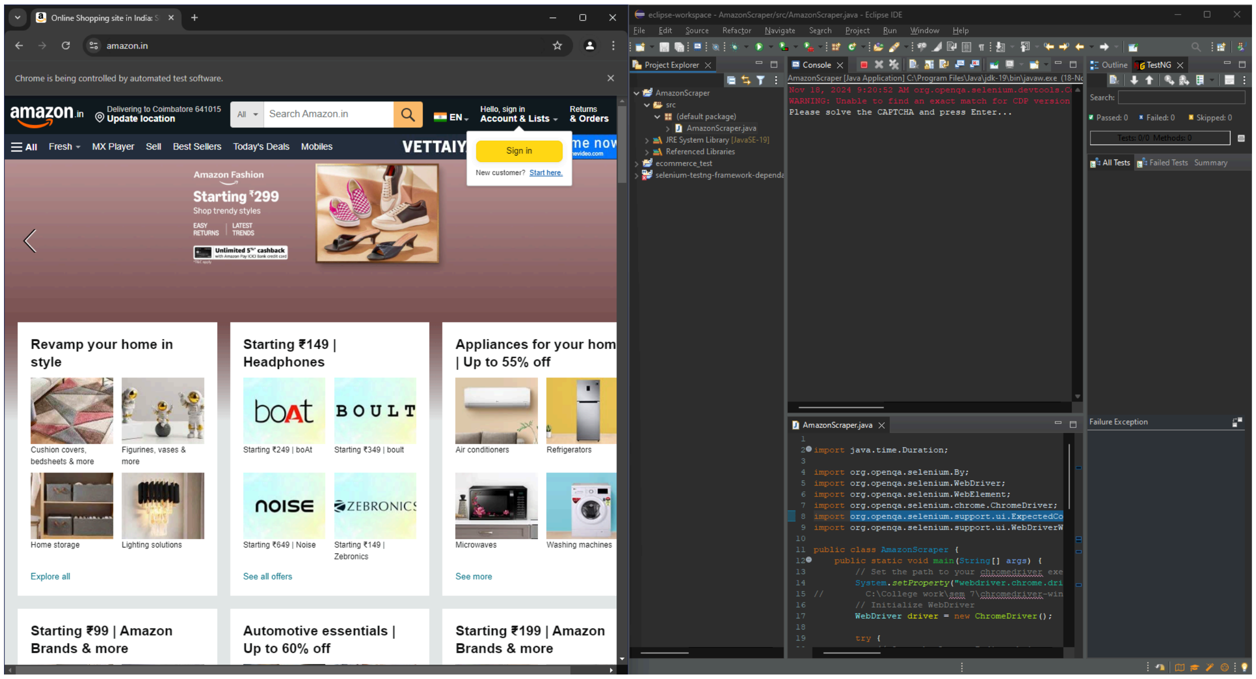
To test an e-commerce application (amazon.in) and automate the testing using selenium.

PROCEDURE:

1. Download Selenium via Maven or from the official site.
2. Download the appropriate WebDriver (e.g., ChromeDriver) and set its path.
3. Set WebDriver path with System.setProperty("webdriver.chrome.driver", "path_to_chromedriver");.
4. Create a new Java project and add the Selenium library.
5. Write code to initialize WebDriver and automate tasks.
6. Run the Java app in your IDE or from the command line.
7. Handle Captchas with a pause using input("Resolve Captcha and press Enter...");.
8. Close the browser using driver.quit();

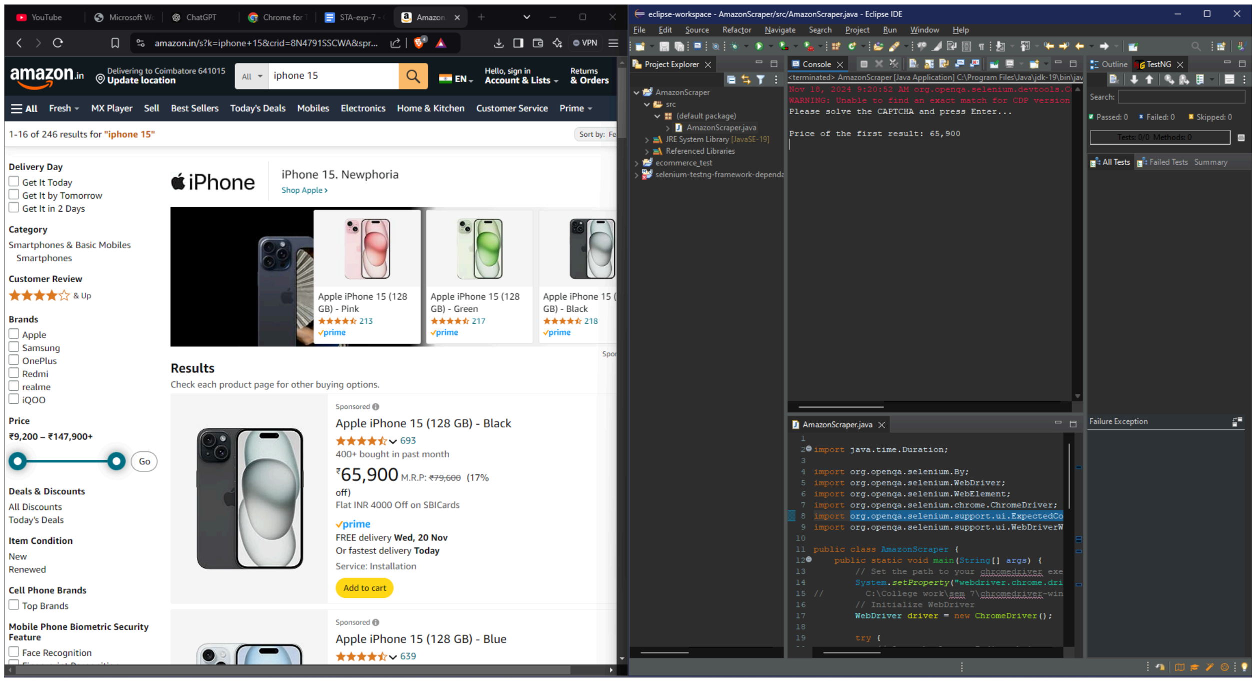
PROGRAM:

```
import java.time.Duration;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class AmazonScraper {
   public static void main(String[] args) {
       // Set the path to your chromedriver executable
       System.setProperty("webdriver.chrome.driver", "C:/College work/sem
7/chromedriver-win64/chromedriver-win64/chromedriver.exe");
//        C:\College work\sem 7\chromedriver-win64\chromedriver-win64
       // Initialize WebDriver
       WebDriver driver = new ChromeDriver();

       try {
            // Open the Amazon India website
            driver.get("http://www.amazon.in/");

            // Pause execution to manually handle CAPTCHA
            System.out.println("Please solve the CAPTCHA and press Enter...");
            System.in.read();

            // Wait for the search button to be clickable
            WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(5));

wait.until(ExpectedConditions.elementToBeClickable(By.id("nav-search-submit-button")));

            // Enter the search keyword
            String keyword = "iphone 15";
            WebElement searchBox = driver.findElement(By.id("twotabsearchtextbox"));
            searchBox.sendKeys(keyword);

            // Click the search button
            WebElement searchButton = driver.findElement(By.id("nav-search-submit-button"));
            searchButton.click();

            // Wait for the search results to load
            wait = new WebDriverWait(driver, Duration.ofSeconds(10));
            wait.until(ExpectedConditions.presenceOfAllElementsLocatedBy(
                    By.xpath("//div[contains(@class, 's-widget-container s-spacing-small
s-widget-container-height-small celwidget')]")));

            // Get the price of the first search result
            WebElement firstResultPrice = driver.findElement(By.xpath(

"//*[@id='search']/div[1]/div[1]/div/span[1]/div[1]/div[3]/div/div/div/div/span/div/div/div/div[2]/div/div
/div[3]/div[1]/div/div[1]/div[1]/div[1]/a/span/span[2]/span[2]"));
            System.out.println("Price of the first result: " + firstResultPrice.getText());
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // Close the driver
            driver.quit();
        }
    }
}
```

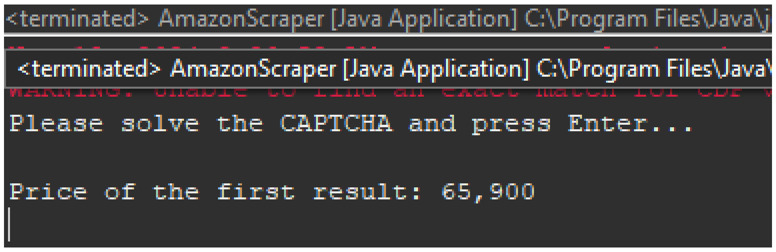


```
AmazonScraper [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe  (18-No
Nov 18, 2024 9:20:52 AM org.openqa.selenium.devtools.C
WARNING: Unable to find an exact match for CDP version
Please solve the CAPTCHA and press Enter...
```

```java

import java.time.Duration;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedCo
import org.openqa.selenium.support.ui.WebDriverW

public class AmazonScraper {
    public static void main(String[] args) {
        // Set the path to your chromedriver exe
        System.setProperty("webdriver.chrome.dri
//          C:\College work\sem 7\chromedriver-win
        // Initialize WebDriver
        WebDriver driver = new ChromeDriver();

        try {
```



```
<terminated> AmazonScraper [Java Application] C:\Program Files\Java\jdk-19\bin\jav
Nov 18, 2024 9:20:52 AM org.openqa.selenium.devtools.C
WARNING: Unable to find an exact match for CDP version
Please solve the CAPTCHA and press Enter...

Price of the first result: 65,900
```



```
<terminated> AmazonScraper [Java Application] C:\Program Files\Java\j
<terminated> AmazonScraper [Java Application] C:\Program Files\Java\
Please solve the CAPTCHA and press Enter...

Price of the first result: 65,900
```

Evaluation Parameter	**Max. Marks**	**Marks Awarded**
Observation	20	
Implementation	40	
Output	10	
Viva	10	
Record	20	
Total Marks	**100**	

Result

e-commerce application (amazon.in) was tested successfully the automation was implemented using Selenium.

Exp No: Integrate TestNG with ecommerce testing automation

Date:

AIM:

To integrate TestNG with the amazon.in ecommerce application testing automation module.

PROCEDURE:

1. Download TestNG via Maven or from the official site.
2. Add TestNG dependency to pom.xml or manually include JARs in the project.
3. Create a new Java project and configure TestNG in the project settings.
4. Write test methods with the @Test annotation.
5. Initialize WebDriver and other setup code in @BeforeMethod or @BeforeClass.
6. Write cleanup code in @AfterMethod or @AfterClass to close WebDriver.
7. Run TestNG tests using your IDE's TestNG plugin or through the command line with testng.xml.
8. Handle Captchas with a pause using input("Resolve Captcha and press Enter..."); inside test methods.
9. Use assertions like assertEquals() or assertTrue() to validate test results.
10. Organize tests in groups and run them using the TestNG XML suite.

PROGRAM:

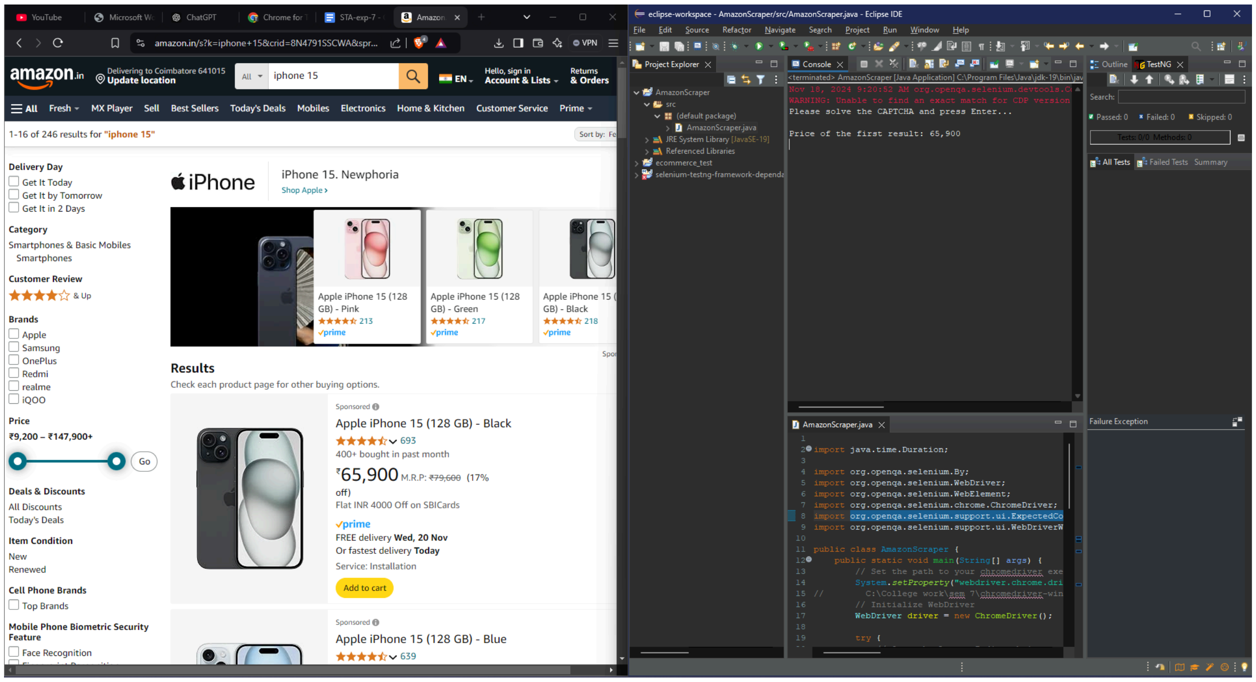
AmazonScraper.java

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class AmazonScraper {
    WebDriver driver;

    @BeforeClass
    public void setUp() {
        System.setProperty("webdriver.chrome.driver", "C:/College work/sem 
7/chromedriver-win64/chromedriver-win64/chromedriver.exe");
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.amazon.in");
    }

    @Test
    public void searchForProduct() {
        WebElement searchBox = driver.findElement(By.id("twotabsearchtextbox"));
        searchBox.sendKeys("iPhone 15");

        WebElement searchButton = driver.findElement(By.id("nav-search-submit-button"));
        searchButton.click();

        WebElement firstResultPrice = driver.findElement(By.xpath(
            "(//span[contains(@class, 'a-price-whole')])[1]"));
        System.out.println("Price of first result: " + firstResultPrice.getText());
    }

    @AfterClass
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```

Testng.xml

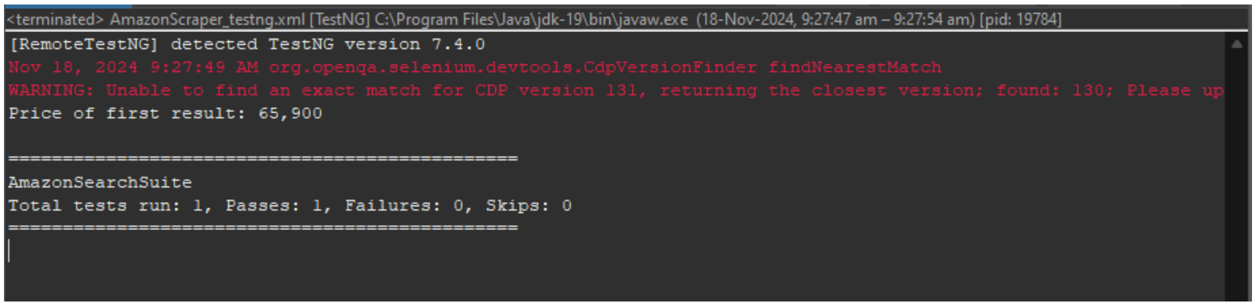
```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="AmazonSearchSuite">
    <test name="AmazonScraper">
        <classes>
            <class name="AmazonScraper"/>
        </classes>
    </test>
</suite>
```

OUTPUT:

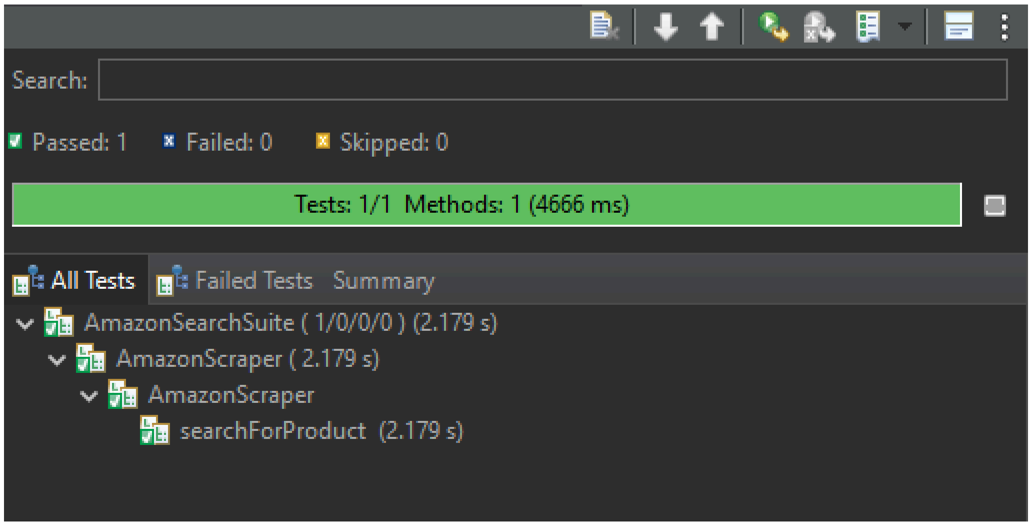


```
<terminated> AmazonScraper [Java Application] C:\Program Files\Java\jdk-19\bin\jav
Nov 18, 2024 9:20:52 AM org.openqa.selenium.devtools.C
WARNING: Unable to find an exact match for CDP version
Please solve the CAPTCHA and press Enter...

Price of the first result: 65,900
```

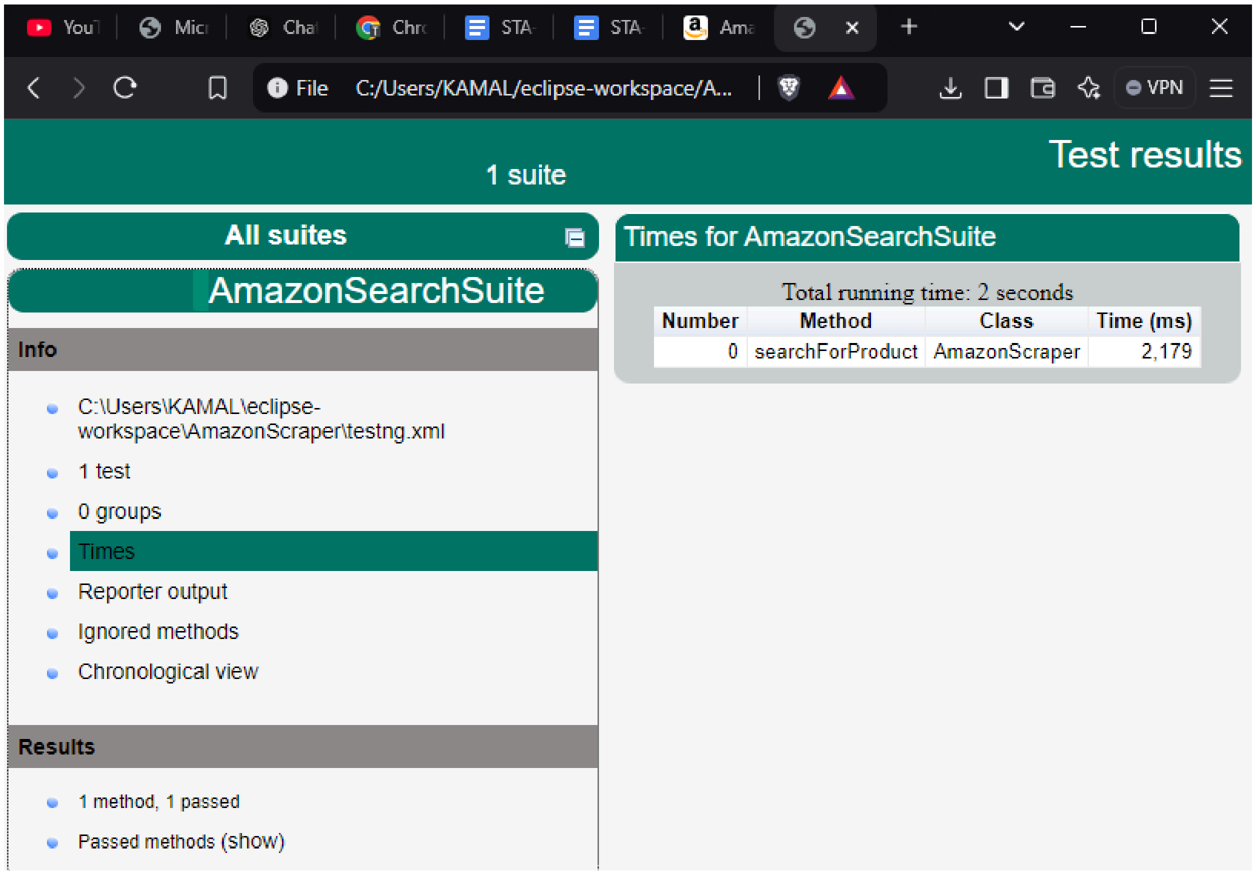
```java

import java.time.Duration;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedCo
import org.openqa.selenium.support.ui.WebDriverW

public class AmazonScraper {
    public static void main(String[] args) {
        // Set the path to your chromedriver exe
        System.setProperty("webdriver.chrome.dri
//        C:\College work\sem 7\chromedriver-win
        // Initialize WebDriver
        WebDriver driver = new ChromeDriver();

        try {
```



```
<terminated> AmazonScraper_testng.xml [TestNG] C:\Program Files\Java\jdk-19\bin\javaw.exe  (18-Nov-2024, 9:27:47 am – 9:27:54 am) [pid: 19784]
[RemoteTestNG] detected TestNG version 7.4.0
Nov 18, 2024 9:27:49 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 131, returning the closest version; found: 130; Please up
Price of first result: 65,900

===============================================
AmazonSearchSuite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```



```
Search:

Passed: 1    Failed: 0    Skipped: 0

Tests: 1/1  Methods: 1 (4666 ms)

All Tests   Failed Tests   Summary
AmazonSearchSuite ( 1/0/0/0 ) (2.179 s)
  AmazonScraper ( 2.179 s)
    AmazonScraper
      searchForProduct  (2.179 s)
```



Test results

1 suite

All suites

AmazonSearchSuite

Info

- C:\Users\KAMAL\eclipse-workspace\AmazonScraper\testng.xml
- 1 test
- 0 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

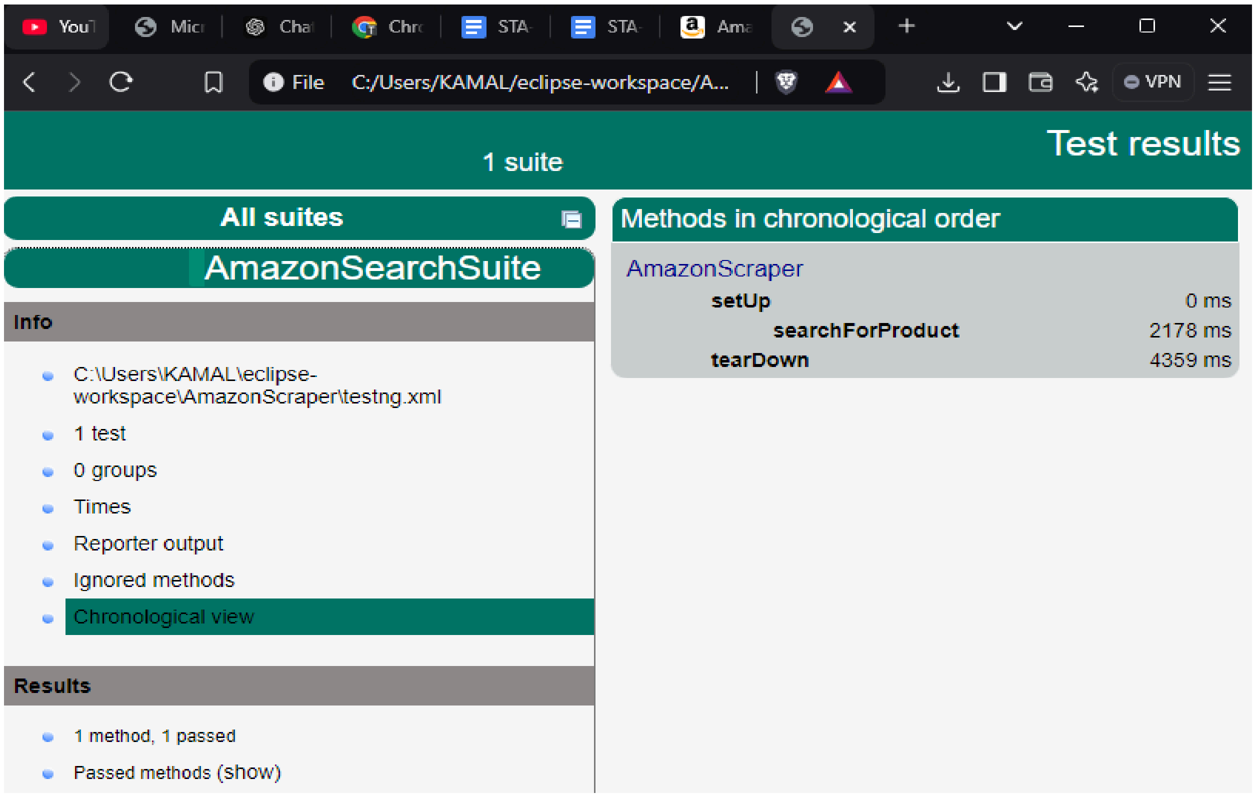
Results

- 1 method, 1 passed
- Passed methods (show)

Times for AmazonSearchSuite

Total running time: 2 seconds

Number	Method	Class	Time (ms)
0	searchForProduct	AmazonScraper	2,179



Test results

1 suite

All suites

AmazonSearchSuite

Info

- C:\Users\KAMAL\eclipse-workspace\AmazonScraper\testng.xml
- 1 test
- 0 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

Results

- 1 method, 1 passed
- Passed methods (show)

Methods in chronological order

```
AmazonScraper
        setUp                          0 ms
              searchForProduct      2178 ms
        tearDown                    4359 ms
```

Evaluation Parameter	**Max. Marks**	**Marks Awarded**
Observation	20	
Implementation	40	
Output	10	
Viva	10	
Record	20	
Total Marks	**100**	

Result

The e-commerce application (amazon.in) was tested successfully and the automation was implemented using Selenium and testNG.

Exp no : 5 **Execute test cases against client server application and**
Date: **identify defects**

Aim

To Execute the test cases against a client server or desktop application and identify the defects

Procedure

1. Create an HTML file with product display, including missing data and broken images.
2. Write JavaScript to simulate missing product names, prices, and broken images.
3. Initialize the WebDriver in Python Selenium and open the HTML file.
4. Automate checks for product name, price, and image using Selenium.
5. Verify the presence of product name and price.
6. Trigger a custom alert for missing price in product 3.
7. Check if product images are not broken.
8. Keep the browser open for manual inspection after script execution.

Code

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Product Display</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="product-container">
        <div class="product" id="product-1">
            <img src="https://pixlr.com/images/index/ai-image-generator-one.webp"
alt="Product Image" class="product-img" id="img-1">
```

```
            <h2 class="product-name" id="name-1">Awesome Product</h2>
            <p class="product-price" id="price-1">$100</p>
            <button class="buy-now-btn" id="buy-1">Buy Now</button>
        </div>

        <div class="product" id="product-2">
            <img src="https://pixlr.com/images/generator/how-to-generate.webp"
alt="Product Image" class="product-img" id="img-2">
            <h2 class="product-name" id="name-2">fvbfkdvbdf</h2> <!-- Simulated
missing name -->
            <p class="product-price" id="price-2">$200</p>
            <button class="buy-now-btn" id="buy-2">Buy Now</button>
        </div>

        <div class="product" id="product-3">
            <img src="https://cdn.eso.org/images/thumb300y/eso2008a.jpg"
alt="Product Image" class="product-img" id="img-3">
            <h2 class="product-name" id="name-3">Yet Another Product</h2>
            <p class="product-price" id="price-3"></p> <!-- Simulated missing price
-->
            <button class="buy-now-btn" id="buy-3">Buy Now</button>
        </div>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

script.js

```
document.addEventListener("DOMContentLoaded", function() {
    // Simulate missing product name in product 2
    const productName2 = document.getElementById("name-2");
    if (!productName2.innerText) {
        console.error("Product name missing for Product 2");
    }

    // Simulate missing product price in product 3
    const productPrice3 = document.getElementById("price-3");
    if (!productPrice3.innerText) {
        console.error("Product price missing for Product 3");
    }

    // Simulate broken image in product 1
    const productImg1 = document.getElementById("img-1");
    productImg1.onerror = function() {
        console.error("Broken image for Product 1");
    };

    // Simulate button click without price for product 3
    const buyButton3 = document.getElementById("buy-3");
    buyButton3.addEventListener("click", function() {
        if (!productPrice3.innerText) {
            console.error("Cannot buy Product 3 because the price is missing.");
            alert("Price is missing! Cannot proceed with purchase.");
        }
    });
});
```

Styles.css

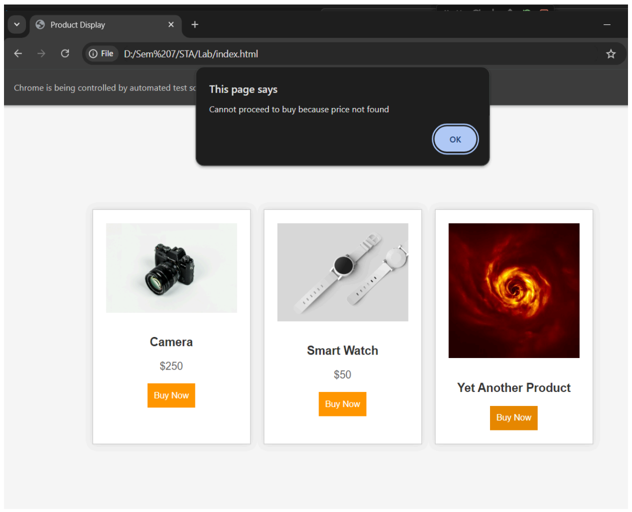
```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f5f5f5;
}

.product-container {
    display: flex;
    flex-direction: row;
    gap: 20px;
```

```css
}

.product {
   border: 1px solid #ccc;
   padding: 20px;
   width: 200px;
   background-color: white;
   text-align: center;
   box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

.product img {
   width: 100%;
   height: auto;
   margin-bottom: 15px;
}

.product h2 {
   font-size: 18px;
   color: #333;
}
.product p {
   font-size: 16px;
   color: #666;
}

.buy-now-btn {
   background-color: #ff9900;
   color: white;
   padding: 10px;
   border: none;
   cursor: pointer;
}

.buy-now-btn:hover {
   background-color: #e68a00;
}
```

test.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time

# Initialize the Chrome WebDriver
driver = webdriver.Chrome()
# Open the product display page with local file path
file_path = "file:///D:/Sem%207/STA/Lab/index.html"
driver.get(file_path)
time.sleep(3)
# Find all products on the page
products = driver.find_elements(By.CLASS_NAME, "product")
print(f"Total products found: {len(products)}")

# Loop through each product to check required elements
for i, product in enumerate(products, start=1):
    print(f"\nChecking Product {i}...")
    # Check for missing product name
    try:
        product_name = product.find_element(By.CLASS_NAME, "product-name")
        if not product_name.text.strip():
            print(f"Product {i} Failed: Name is missing.")
        else:
            print(f"Product {i} Passed: Name is present.")
    except Exception:
        print(f"Product {i} Failed: Could not find the product name.")

    # Check for missing price
    try:
        product_price = product.find_element(By.CLASS_NAME, "product-price")
        if not product_price.text.strip():
            print(f"Product {i} Failed: Price is missing.")
            if i == 3:
                print(f"Product {i}: Custom alert for missing price.")
                # Trigger a custom alert for the third product
                driver.execute_script("alert('Cannot proceed to buy because price not found');")
                time.sleep(2)  # Wait for the alert to display
                alert = driver.switch_to.alert
                alert.accept()  # Close the alert after displaying the message
        else:
            print(f"Product {i} Passed: Price is present.")
    except Exception:
        print(f"Product {i} Failed: Could not find the product price.")
    # Check for broken image
    try:
        product_img = product.find_element(By.CLASS_NAME, "product-img")
        if product_img.get_attribute("naturalWidth") == "0":
            print(f"Product {i} Failed: Image is broken.")
        else:
            print(f"Product {i} Passed: Image is present.")
    except Exception:
        print(f"Product {i} Failed: Could not find the product image.")
time.sleep(20)
# Keep the browser open for manual inspection
#print("Testing complete. The browser will remain open until you manually close it.")
```

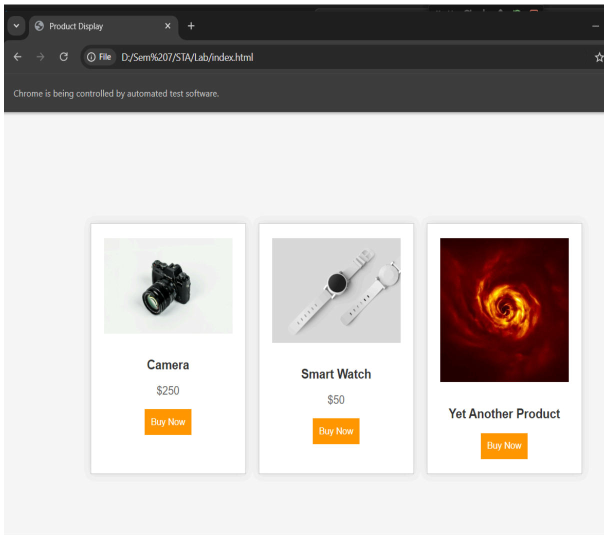
Output



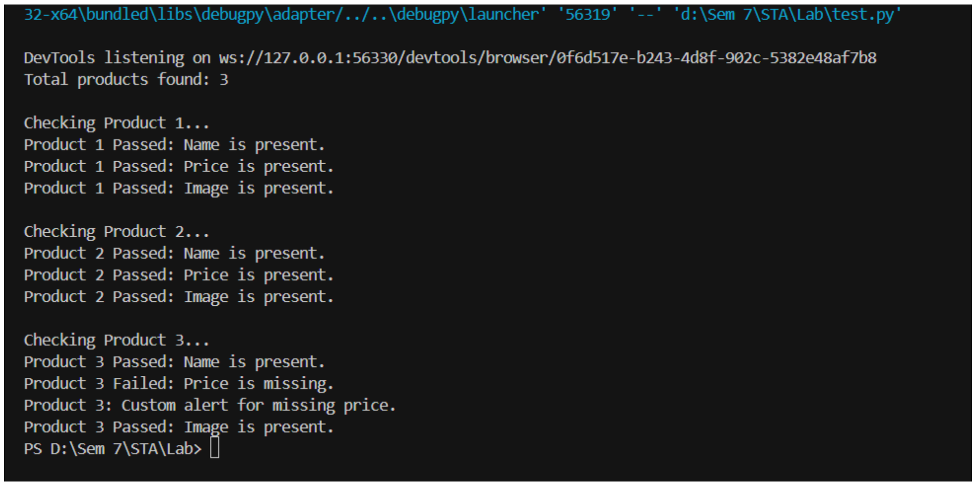
Product Display
D:/Sem%207/STA/Lab/index.html
Chrome is being controlled by automated test so

This page says

Cannot proceed to buy because price not found



Product Display
D:/Sem%207/STA/Lab/index.html
Chrome is being controlled by automated test software.
Camera
$250
Buy Now
Smart Watch
$50
Buy Now
Yet Another Product
Buy Now



```
32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '56319' '--' 'd:\Sem 7\STA\Lab\test.py'

DevTools listening on ws://127.0.0.1:56330/devtools/browser/0f6d517e-b243-4d8f-902c-5382e48af7b8
Total products found: 3

Checking Product 1...
Product 1 Passed: Name is present.
Product 1 Passed: Price is present.
Product 1 Passed: Image is present.

Checking Product 2...
Product 2 Passed: Name is present.
Product 2 Passed: Price is present.
Product 2 Passed: Image is present.

Checking Product 3...
Product 3 Passed: Name is present.
Product 3 Failed: Price is missing.
Product 3: Custom alert for missing price.
Product 3 Passed: Image is present.
PS D:\Sem 7\STA\Lab>
```

Evaluation Parameter	**Max. Marks**	**Marks Awarded**
Observation	20	
Implementation	40	
Output	10	
Viva	10	
Record	20	
Total Marks	**100**	

RESULT

Thus, the test cases against a client server or desktop application were identified successfully and its defects reported.

Exp no : **Test the performance of the e-commerce application**
Date:

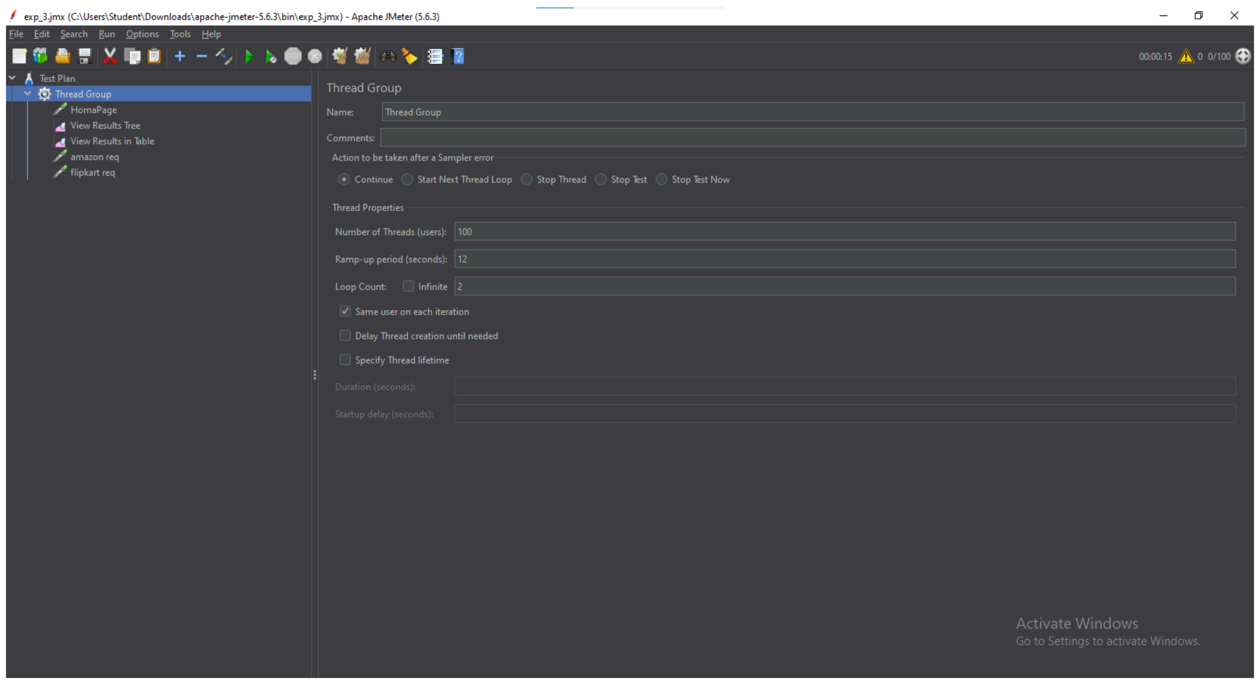
Aim

To test the performance of the e-commerce application using Apache Jmeter.

Procedure

1. Download JMeter from the official website.
2. Extract the files to a known location.
3. Start JMeter by running jmeter.bat or ./jmeter.
4. Create a new Test Plan in JMeter.
5. Configure the Thread Group with user settings and add 100 threads.
6. Add an HTTP Request sampler to the Thread Group for e-commerce application homepages.
7. Fill in the server name, path, and method for the http request pages.
8. Add listeners like View Results Tree and View Results Table for analysis.
9. Click the start button to execute your test and view results

Output


exp_3.jmx (C:\Users\Student\Downloads\apache-jmeter-5.6.3\bin\exp_3.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
00:00:15 0 0/100
Test Plan
Thread Group
HomaPage
View Results Tree
View Results in Table
amazon req
flipkart req
Activate Windows
Go to Settings to activate Windows.

Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 100

Ramp-up period (seconds): 12

Loop Count: Infinite 2

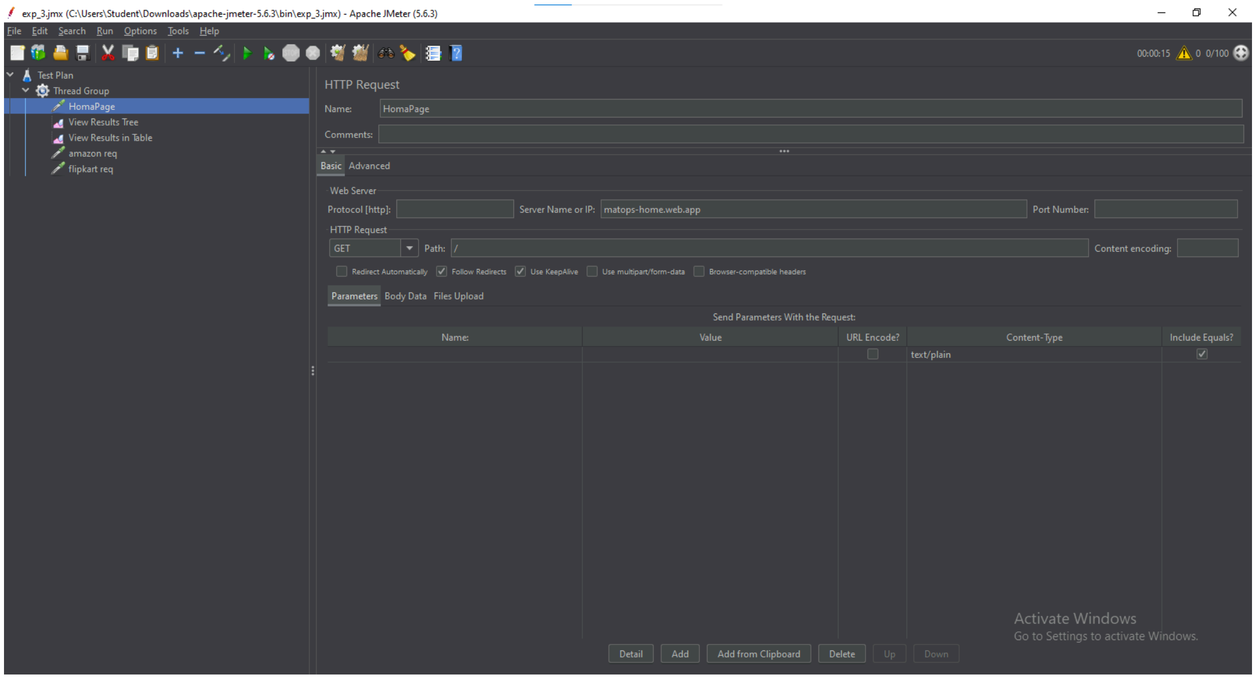
[✓ Same user on each iteration]

Delay Thread creation until needed

Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):


File Edit Search Run Options Tools Help
00:00:15 0 0/100
Detail
Add
Add from Clipboard
Delete
Up
Down
Activate Windows
Go to Settings to activate Windows.

exp_3.jmx (C:\Users\Student\Downloads\apache-jmeter-5.6.3\bin\exp_3.jmx) - Apache JMeter (5.6.3)

- Test Plan
  - Thread Group
    - HomaPage
    - View Results Tree
    - View Results in Table
    - amazon req
    - flipkart req

HTTP Request

Name: HomaPage

Comments:

Basic | Advanced

Web Server

Protocol [http]: | Server Name or IP: matops-home.web.app | Port Number:

HTTP Request

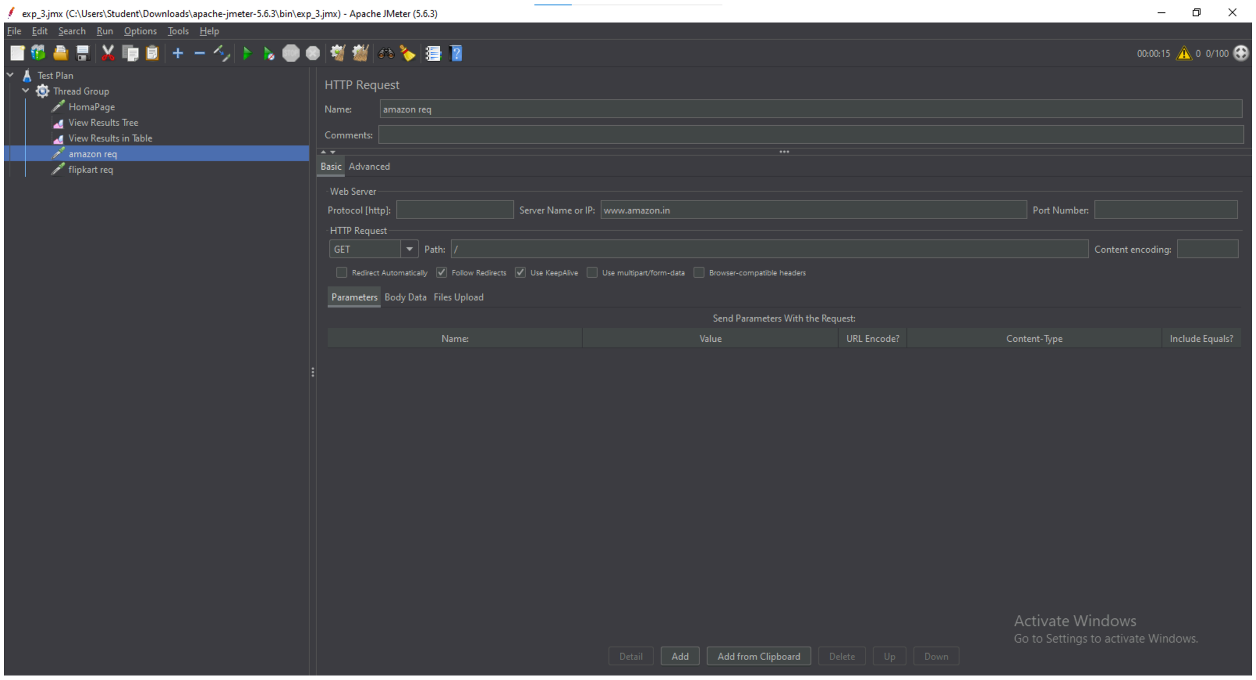
GET | Path: / | Content encoding:

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
		☐	text/plain	☑


File Edit Search Run Options Tools Help
00:00:15 0 0/100
Detail
Add
Add from Clipboard
Delete
Up
Down
Activate Windows
Go to Settings to activate Windows.

exp_3.jmx (C:\Users\Student\Downloads\apache-jmeter-5.6.3\bin\exp_3.jmx) - Apache JMeter (5.6.3)

- Test Plan
  - Thread Group
    - HomaPage
    - View Results Tree
    - View Results in Table
    - amazon req
    - flipkart req

HTTP Request

Name: amazon req

Comments:

Basic | Advanced

Web Server

Protocol [http]: | Server Name or IP: www.amazon.in | Port Number:

HTTP Request

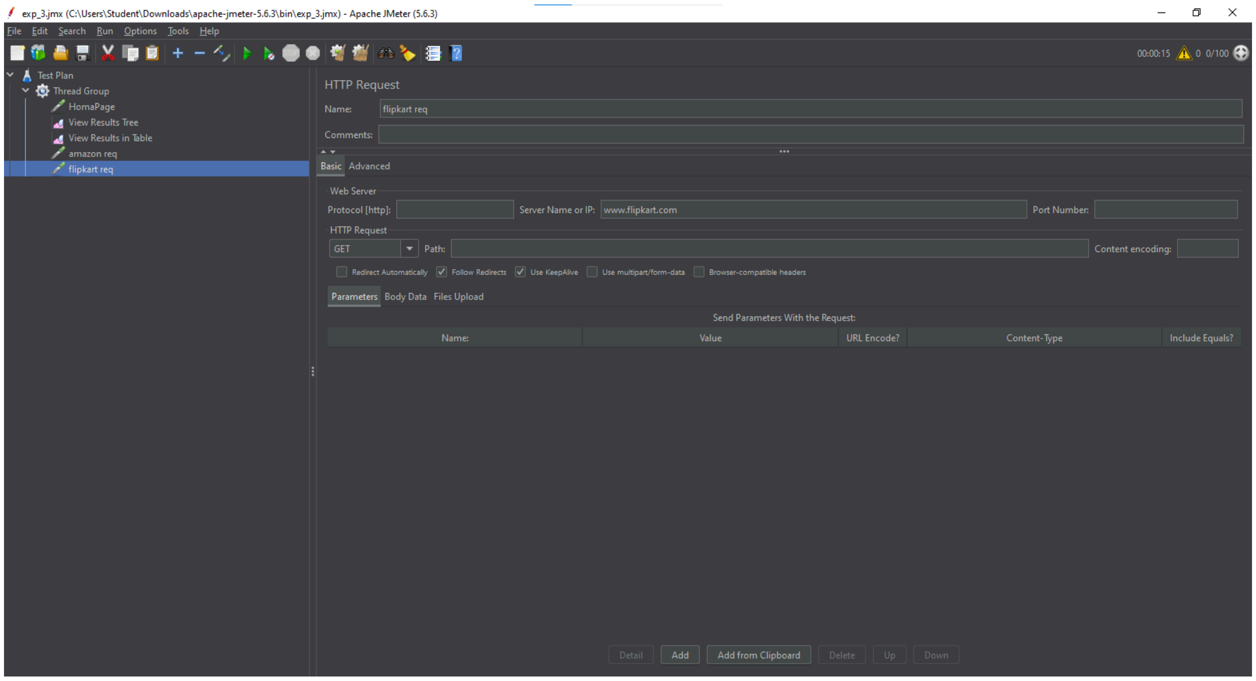
GET | Path: / | Content encoding:

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?


exp_3.jmx (C:\Users\Student\Downloads\apache-jmeter-5.6.3\bin\exp_3.jmx) - Apache JMeter (5.6.3)

- Test Plan
  - Thread Group
    - HomaPage
    - View Results Tree
    - View Results in Table
    - amazon req
    - flipkart req

HTTP Request

Name: flipkart req

Comments:

Basic | Advanced

Web Server

Protocol [http]:   Server Name or IP: www.flipkart.com   Port Number:

HTTP Request

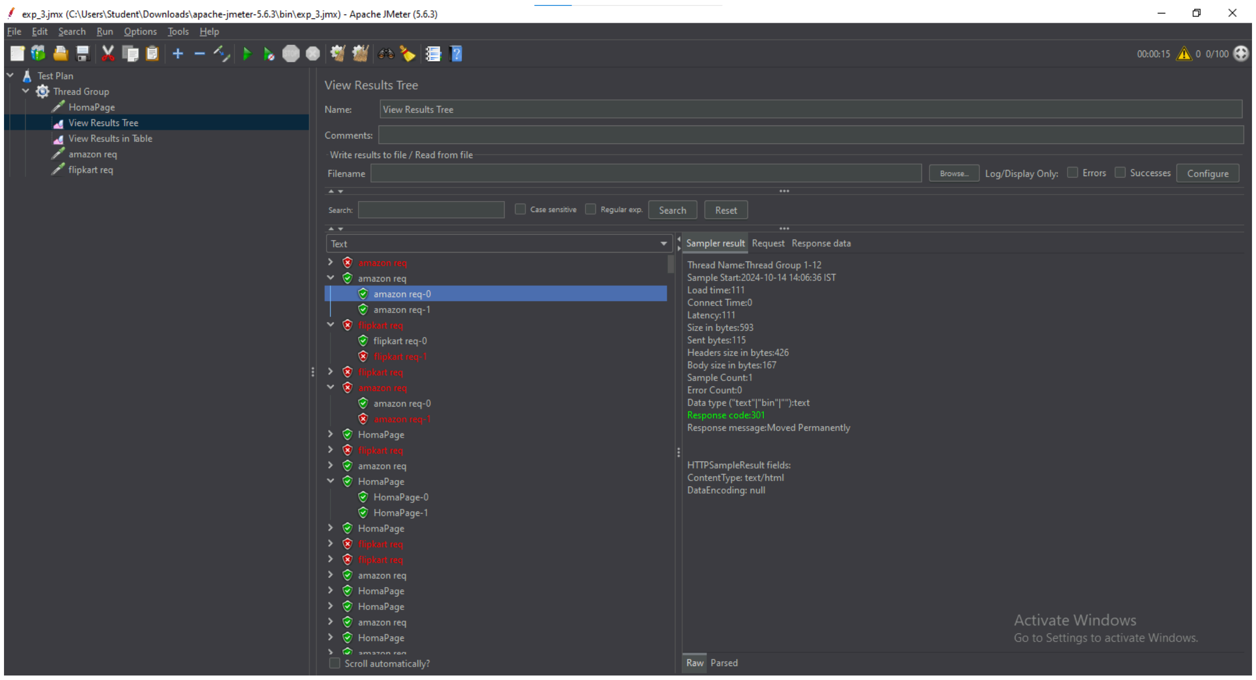
GET   Path: Content encoding:

[ ] Redirect Automatically [✓] Follow Redirects [✓] Use KeepAlive [ ] Use multipart/form-data [ ] Browser-compatible headers

Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?


exp_3.jmx (C:\Users\Student\Downloads\apache-jmeter-5.6.3\bin\exp_3.jmx) - Apache JMeter (5.6.3)

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename   Log/Display Only: [ ] Errors [ ] Successes

Search: [ ] Case sensitive [ ] Regular exp.

Text

```
amazon req
amazon req
    amazon req-0
    amazon req-1
flipkart req
    flipkart req-0
    flipkart req-1
flipkart req
amazon req
    amazon req-0
    amazon req-1
HomaPage
flipkart req
amazon req
HomaPage
    HomaPage-0
    HomaPage-1
HomaPage
flipkart req
flipkart req
amazon req
HomaPage
HomaPage
amazon req
HomaPage
```

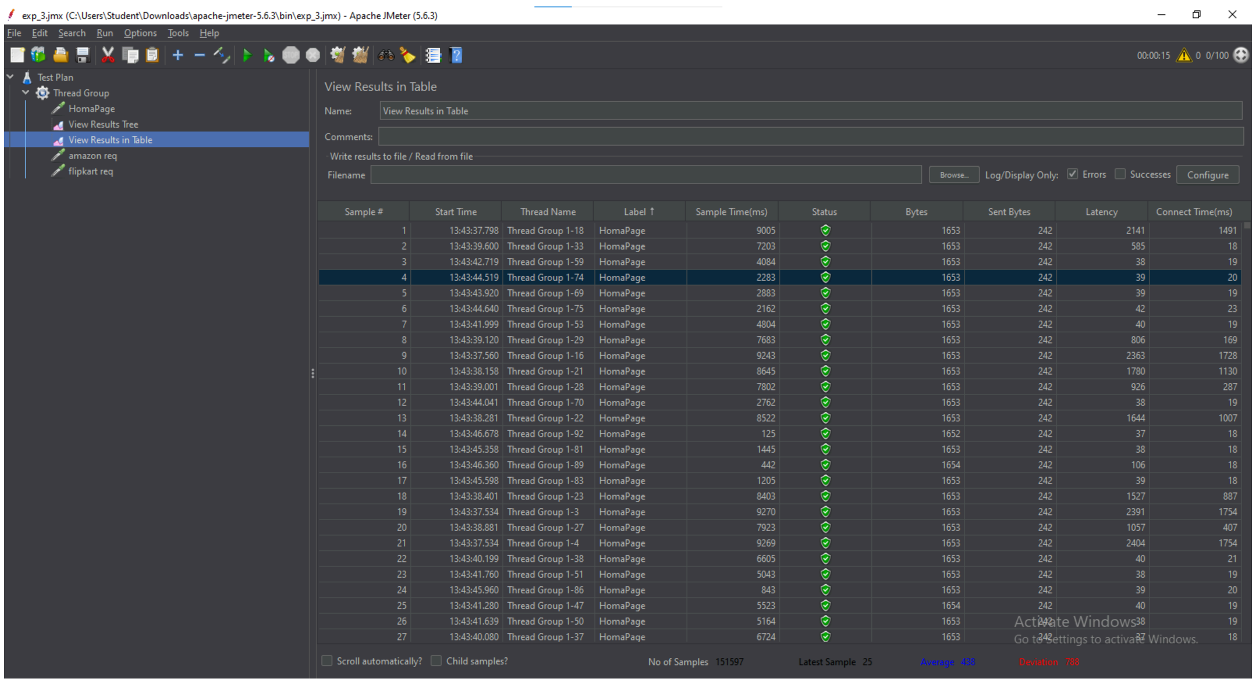
[ ] Scroll automatically?

Sampler result | Request | Response data

```
Thread Name:Thread Group 1-12
Sample Start:2024-10-14 14:06:36 IST
Load time:111
Connect Time:0
Latency:111
Size in bytes:593
Sent bytes:115
Headers size in bytes:426
Body size in bytes:167
Sample Count:1
Error Count:0
Data type ("text"|"bin"|""):text
Response code:301
Response message:Moved Permanently


HTTPSampleResult fields:
ContentType: text/html
DataEncoding: null
```

Raw | Parsed

Activate Windows
Go to Settings to activate Windows.



exp_3.jmx (C:\Users\Student\Downloads\apache-jmeter-5.6.3\bin\exp_3.jmx) - Apache JMeter (5.6.3)

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename | Log/Display Only: ✓ Errors | Successes

Sample #	Start Time	Thread Name	Label ↑	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	13:43:37.798	Thread Group 1-18	HomaPage	9005	✓	1653	242	2141	1491
2	13:43:39.600	Thread Group 1-33	HomaPage	7203	✓	1653	242	585	18
3	13:43:42.719	Thread Group 1-59	HomaPage	4084	✓	1653	242	38	19
4	13:43:44.519	Thread Group 1-74	HomaPage	2283	✓	1653	242	39	20
5	13:43:43.920	Thread Group 1-69	HomaPage	2883	✓	1653	242	39	19
6	13:43:44.640	Thread Group 1-75	HomaPage	2162	✓	1653	242	42	23
7	13:43:41.999	Thread Group 1-53	HomaPage	4804	✓	1653	242	40	19
8	13:43:39.120	Thread Group 1-29	HomaPage	7683	✓	1653	242	806	169
9	13:43:37.560	Thread Group 1-16	HomaPage	9243	✓	1653	242	2363	1728
10	13:43:38.158	Thread Group 1-21	HomaPage	8645	✓	1653	242	1780	1130
11	13:43:39.001	Thread Group 1-28	HomaPage	7802	✓	1653	242	926	287
12	13:43:44.041	Thread Group 1-70	HomaPage	2762	✓	1653	242	38	19
13	13:43:38.281	Thread Group 1-22	HomaPage	8522	✓	1653	242	1644	1007
14	13:43:46.678	Thread Group 1-92	HomaPage	125	✓	1652	242	37	18
15	13:43:45.358	Thread Group 1-81	HomaPage	1445	✓	1653	242	38	18
16	13:43:46.360	Thread Group 1-89	HomaPage	442	✓	1654	242	106	18
17	13:43:45.598	Thread Group 1-83	HomaPage	1205	✓	1653	242	39	18
18	13:43:38.401	Thread Group 1-23	HomaPage	8403	✓	1653	242	1527	887
19	13:43:37.534	Thread Group 1-3	HomaPage	9270	✓	1653	242	2391	1754
20	13:43:38.881	Thread Group 1-27	HomaPage	7923	✓	1653	242	1057	407
21	13:43:37.534	Thread Group 1-4	HomaPage	9269	✓	1653	242	2404	1754
22	13:43:40.199	Thread Group 1-38	HomaPage	6605	✓	1653	242	40	21
23	13:43:41.760	Thread Group 1-51	HomaPage	5043	✓	1653	242	38	19
24	13:43:45.960	Thread Group 1-86	HomaPage	843	✓	1653	242	39	20
25	13:43:41.280	Thread Group 1-47	HomaPage	5523	✓	1654	242	40	19
26	13:43:41.639	Thread Group 1-50	HomaPage	5164	✓	1653	242	38	19
27	13:43:40.080	Thread Group 1-37	HomaPage	6724	✓	1653	242	37	18

Scroll automatically? Child samples? No of Samples 151597 Latest Sample 25 Average 438 Deviation 788

Evaluation Parameter	**Max. Marks**	**Marks Awarded**
Observation	20	
Implementation	40	
Output	10	
Viva	10	
Record	20	
Total Marks	**100**	

RESULT

Thus the performance e-commerce application was successfully tested using Apache Jmeter and output verified.

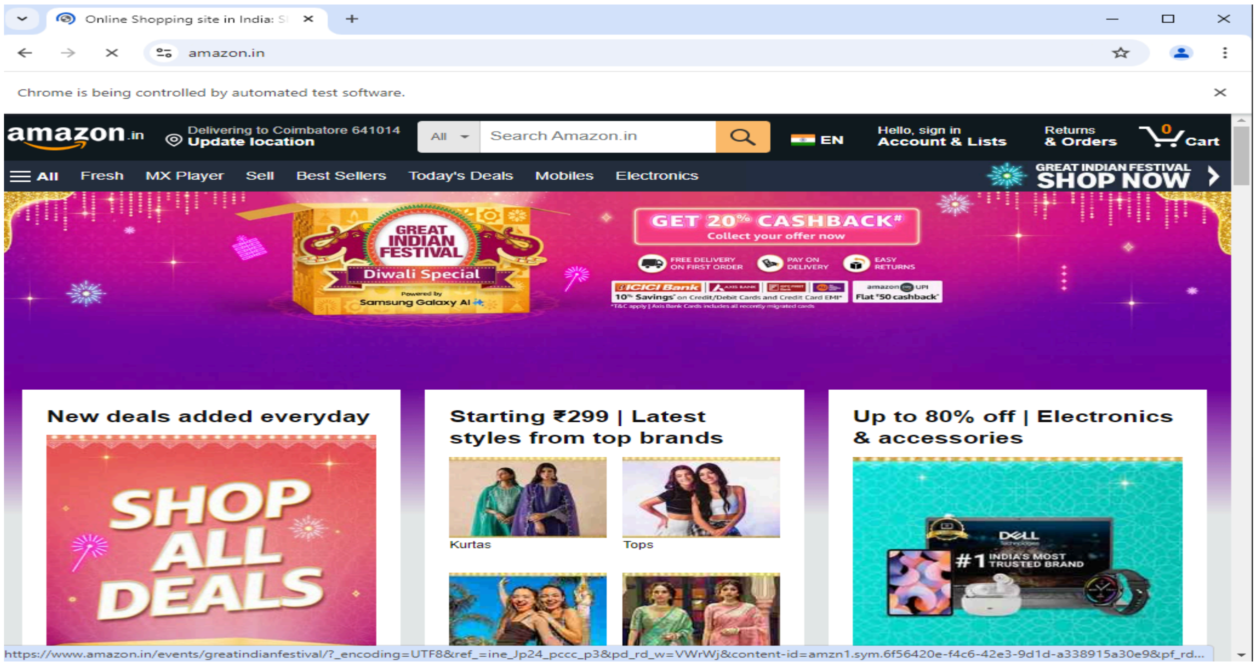
Exp no : **Test the performance of the e-commerce application**
Date:

Aim
To test the performance of the e-commerce application using Selenium.

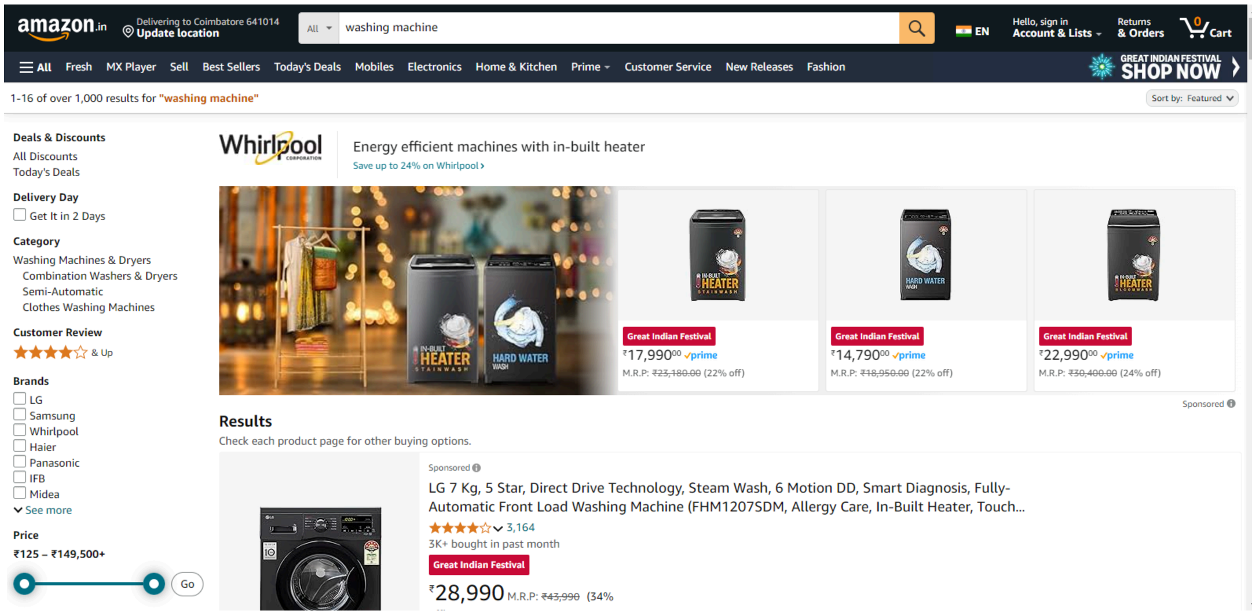
Procedure
1. Install Selenium by running the pip command in your terminal.
2. Download the appropriate version of ChromeDriver and configure it.
3. Import the necessary libraries from the Selenium package for automation.
4. Initialize the Chrome WebDriver to launch the browser session.
5. Navigate to the Amazon homepage using the `get` method.
6. Measure the time taken during the search submission process.
7. Ensure to close the browser after the script execution completes.

Code
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
import time

# Initialize the Chrome driver
driver = webdriver.Chrome()  # Specify the path to your chromedriver

try:
    # Navigate to Amazon homepage
    driver.get("https://www.amazon.in")
    start_time = time.time()

    # Wait for the search box to be clickable
    WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.ID,
'twotabsearchtextbox')))

    # Enter search keyword and submit
    keyword = "washing machine"
    search_box = driver.find_element(By.ID, "twotabsearchtextbox")
    search_box.send_keys(keyword)
    search_button = driver.find_element(By.ID, "nav-search-submit-button")
    search_button.click()

    # Measure the page load time
    end_time = time.time()
    page_load_time = (end_time - start_time) * 1000
    print(f"Page Load Time: {page_load_time:.2f} milliseconds")

finally:
    # Close the browser
    driver.quit()
```

Output



Online Shopping site in India: S
amazon.in
Chrome is being controlled by automated test software.
amazon.in
Delivering to Coimbatore 641014
Update location
All
Search Amazon.in
EN
Hello, sign in
Account & Lists
Returns
& Orders
0
Cart
All
Fresh
MX Player
Sell
Best Sellers
Today's Deals
Mobiles
Electronics
GREAT INDIAN FESTIVAL
SHOP NOW
GREAT INDIAN FESTIVAL
Diwali Special
Samsung Galaxy AI
GET 20% CASHBACK*
Collect your offer now
FREE DELIVERY ON FIRST ORDER
PAY ON DELIVERY
EASY RETURNS
New deals added everyday
SHOP ALL DEALS
Starting ₹299 | Latest styles from top brands
Kurtas
Tops
Up to 80% off | Electronics & accessories
DELL
#1 INDIA'S MOST TRUSTED BRAND
https://www.amazon.in/events/greatindianfestival/?_encoding=UTF8&ref_=ine_Jp24_pccc_p3&pd_rd_w=VWrWj&content-id=amzn1.sym.6f56420e-f4c6-42e3-9d1d-a338915a30e9&pf_rd...



1-16 of over 1,000 results for "washing machine"

Energy efficient machines with in-built heater

₹17,990⁰⁰ prime M.R.P: ~~₹23,180.00~~ (22% off)

₹14,790⁰⁰ prime M.R.P: ~~₹18,950.00~~ (22% off)

₹22,990⁰⁰ prime M.R.P: ~~₹30,400.00~~ (24% off)

Results

Check each product page for other buying options.

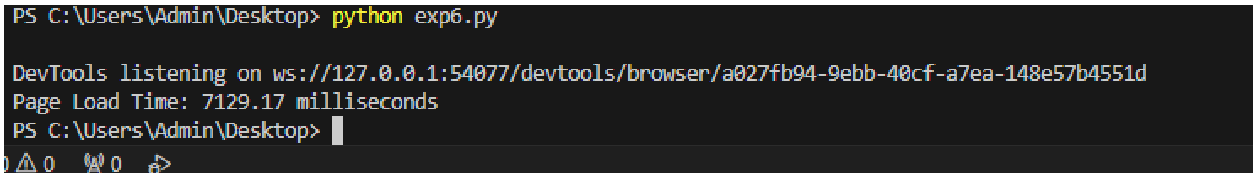
Sponsored

LG 7 Kg, 5 Star, Direct Drive Technology, Steam Wash, 6 Motion DD, Smart Diagnosis, Fully-Automatic Front Load Washing Machine (FHM1207SDM, Allergy Care, In-Built Heater, Touch...

3,164

3K+ bought in past month

₹28,990 M.R.P: ~~₹43,990~~ (34%



```
PS C:\Users\Admin\Desktop> python exp6.py

DevTools listening on ws://127.0.0.1:54077/devtools/browser/a027fb94-9ebb-40cf-a7ea-148e57b4551d
Page Load Time: 7129.17 milliseconds
PS C:\Users\Admin\Desktop> 
```

Evaluation Parameter	**Max. Marks**	**Marks Awarded**
Observation	20	
Implementation	40	
Output	10	
Viva	10	
Record	20	
Total Marks	**100**	

RESULT

Thus the performance e-commerce application was successfully tested using selenium and output verified.