

CCS366 SOFTWARE TESTING AND AUTOMATION LABORATORY

Experiment 1: Develop the Test Plan for Testing an E-commerce Web/Mobile Application (www.amazon.in)

Aim:

The aim of this experiment is to develop a comprehensive test plan for testing the functionality and usability of the e-commerce web/mobile application www.amazon.in.

Algorithm:

1. Identify the Scope: Determine the scope of testing, including the features and functionalities that need to be tested.
2. Define Test Objectives: Specify the primary objectives of testing, such as functional testing, usability testing, performance testing, security testing, etc.
3. Identify Test Environment: Define the platforms, browsers, devices, and operating systems on which the application will be tested.
4. Determine Test Deliverables: Decide on the documents and artifacts that will be generated during the testing process, such as test cases, test reports, and defect logs.
5. Create Test Strategy: Develop an overall approach for testing, including the testing techniques, entry and exit criteria, and the roles and responsibilities of the testing team.
6. Define Test Scope and Schedule: Specify the timeline for each testing phase and the scope of testing for each phase.
7. Risk Analysis: Identify potential risks and their impact on the testing process, and devise risk mitigation strategies.
8. Resource Planning: Allocate the necessary resources, including the testing team, hardware, and software required for testing.

9. Test Case Design: Prepare detailed test cases based on the requirements and functionalities of the e-commerce application.
10. Test Data Setup: Arrange test data required for executing the test cases effectively.
11. Test Execution: Execute the test cases and record the test results.
12. Defect Reporting: Document any defects encountered during testing and track their resolution.

Test Plan:

The test plan should cover the following sections:

1. Introduction: Briefly describe the purpose of the test plan and provide an overview of the e-commerce application to be tested.
2. Test Objectives: List the primary objectives of testing the application.
3. Test Scope: Specify the features and functionalities to be tested and any limitations on testing.
4. Test Environment: Describe the hardware, software, browsers, and devices to be used for testing.
5. Test Strategy: Explain the overall approach to be followed during testing.
6. Test Schedule: Provide a detailed timeline for each testing phase.
7. Risk Analysis: Identify potential risks and the strategies to mitigate them.
8. Resource Planning: Specify the resources required for testing.
9. Test Case Design: Include a summary of the test cases developed for the application.
10. Test Data Setup: Describe the process of arranging test data for testing.

11. Defect Reporting: Explain the procedure for reporting and tracking defects.

Test Case Table:

Processes	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
Test Plan	TC001	Scope of Testing	1. Review the test plan document.	Verify the scope of testing.	Done	The test plan includes all features.		
	TC002	Test Objectives	1. Review the test plan document.	Verify the test objectives.	Done	The test objectives are well-defined.		
	TC003	Test Environment	1. Review the test plan document.	Check the specified environments.	Done	Test environments are mentioned.		
	TC004	Test Deliverables	1. Review the test plan document.	Ensure all deliverables are listed.	Done	The test plan includes all deliverables.		
	TC005	Test Strategy	1. Review the test plan document.	Verify the overall approach.	Done	The test strategy is clearly stated.		
	TC006	Test Scope and Schedule	1. Review the test plan document.	Check the schedule and scope.	Done	The schedule and scope are defined.		

	TC007	Risk Analysis	1. Review the test plan document.	Ensure potential risks are identified.	Done	Risks and mitigation strategies are mentioned.		
	TC008	Resource Planning	1. Review the test plan document.	Check the required resources .	Done	Resources needed for testing are listed.		
	TC009	Test Case Design	1. Review and execute the test cases.	Validate the prepared test cases.	Done	Test cases are accurate and functional.		
	TC010	Test Data Setup	1. Review the test data setup process.	Verify the availability of test data.	Done	Test data is available for testing.		
	TC011	Test Execution	1. Run the test cases and document the outcomes.	Execute the test cases.	In Progress	Test results are recorded and documented .		
	TC012	Defect Reporting	1. Log defects with detailed information .	Ensure defects are reported correctly.	Not Started	Defects are reported with sufficient details.		
	TC013	Defect Tracking	1. Monitor defect status and updates.	Verify the tracking of defects.	Not Started	Defects are tracked until resolution.		

Explanation:

The test plan is a crucial document that outlines the entire testing process. It ensures that all aspects of the e-commerce application are thoroughly tested, and the results are systematically documented.

Result:

Upon completion of the experiment, you will have a well-structured test plan that provides a clear roadmap for testing the e-commerce web/mobile application www.amazon.in.

Experiment 2: Design the Test Cases for Testing the E-commerce Application

Aim:

The aim of this experiment is to design a set of comprehensive and effective test cases for testing the e-commerce application www.amazon.in.

Algorithm:

1. Understand Requirements: Familiarize yourself with the functional and non-functional requirements of the e-commerce application.
2. Identify Test Scenarios: Based on the requirements, identify different test scenarios that cover all aspects of the application.
3. Write Test Cases: Develop test cases for each identified scenario, including preconditions, steps to be executed, and expected outcomes.
4. Cover Edge Cases: Ensure that the test cases cover edge cases and boundary conditions to verify the robustness of the application.
5. Prioritize Test Cases: Prioritize the test cases based on their criticality and relevance to the application.
6. Review Test Cases: Conduct a peer review of the test cases to ensure their accuracy and completeness.
7. Optimize Test Cases: Optimize the test cases for reusability and maintainability.

Test Case Design:

The test case design should include the following components for each test case:

1. Test Case ID: A unique identifier for each test case.
2. Test Scenario: Description of the scenario being tested.

3. Test Case Description: Detailed steps to execute the test.
4. Precondition: The necessary conditions that must be satisfied before executing the test case.
5. Test Steps: The sequence of actions to be performed during the test.
6. Expected Result: The outcome that is expected from the test.

Test Case Table:

Process	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
Test Case Design	TC001	User Registration	1. Navigate to the registration page.	Verify user registration process.	Done	User can successfully register.		
	TC002	User Login	1. Navigate to the login page.	Verify user login process.	Done	User can successfully log in.		
	TC003	Search Functionality	1. Enter a keyword in the search bar.	Verify search functionality.	Done	Search results relevant to the keyword.		
	TC004	Add to Cart	1. Browse the product catalog.	Verify adding products to the cart.	Done	Product is added to the shopping cart.		
	TC005	Shopping Cart Validation	1. Click on the shopping cart icon.	Verify the shopping cart contents.	Done	Items in the shopping cart are displayed.		

	TC006	Checkout Process	1. Click on the "Checkout" button.	Verify the checkout process.	Not Started	Checkout process proceeds as expected.		
--	-------	------------------	------------------------------------	------------------------------	-------------	--	--	--

Explanation:

Test cases are designed to validate the functionality and behaviour of the e-commerce application. They ensure that the application performs as intended and meets the specified requirements.

Result:

Upon completion of the experiment, you will have a set of well-defined test cases ready for testing the e-commerce application www.amazon.in.

Experiment 3: Test the E-commerce Application and Report the Defects in It

Aim:

The aim of this experiment is to execute the designed test cases and identify defects or issues in the e-commerce application www.amazon.in.

Algorithm:

1. Test Environment Setup: Set up the testing environment with the required hardware, software, and test data.
2. Test Case Execution: Execute the test cases designed in Experiment 2, following the specified steps.
3. Defect Identification: During test execution, record any discrepancies or issues encountered.
4. Defect Reporting: Log the identified defects with detailed information, including steps to reproduce, severity, and priority.
5. Defect Tracking: Track the progress of defect resolution and verify fixes as they are implemented.
6. Retesting: After defect fixes, retest the affected areas to ensure the issues are resolved.
7. Regression Testing: Conduct regression testing to ensure new changes do not introduce new defects.

Test Case Table:

Process	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
Test Case Design	TC001	User Registration	1. Navigate to the registration page.	Verify user registration process.	Done	User can successfully register.		
	TC002	User Login	1. Navigate to the login page.	Verify user login process.	Done	User can successfully log in.		
	TC003	Search Functionality	1. Enter a keyword in the search bar.	Verify search functionality.	Done	Search results relevant to the keyword.		
	TC004	Add to Cart	1. Browse the product catalog.	Verify adding products to the cart.	Done	Product is added to the shopping cart.		
	TC005	Shopping Cart Validation	1. Click on the shopping cart icon.	Verify the shopping cart contents.	Done	Items in the shopping cart are displayed.		
	TC006	Checkout Process	1. Click on the "Checkout" button.	Verify the checkout process.	Not Started	Checkout process proceeds as expected.		

Explanation:

Testing the e-commerce application aims to validate its functionality and usability. By identifying and reporting defects, you ensure the application's quality and reliability.

Result:

Upon completion of the experiment, you will have a list of identified defects and their status after resolution.

Experiment 4: Develop the Test Plan and Design the Test Cases for an Inventory Control System

Aim:

The aim of this experiment is to create a comprehensive test plan and design test cases for an Inventory Control System.

Algorithm:

Follow the same algorithm as described in Experiment 1 for developing the test plan for an inventory control system.

Follow the same algorithm as described in Experiment 2 for designing test cases for an inventory control system.

Test Plan:

Processes	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
Test Plan	TC001	Scope of Testing	1. Review the requirements and project documentation .	Verify the scope of testing.	Done	The test plan includes all essential features.		
			2. Identify the modules to be tested.					
			3. Determine the out-of-scope items.					

Processes	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
	TC002	Test Objectives	1. Review the requirements and project documentation.	Verify the test objectives.	Done	The test objectives are clearly defined.		
			2. Discuss with stakeholders to understand expectations.					
	TC003	Test Environment	1. Identify the hardware and software requirements.	Verify the required environments.	Not Started	The test environment is defined.		
			2. Set up the required hardware and software.					
	TC004	Test Deliverables	1. Determine the documents and artifacts to be produced.	Verify the required deliverables.	Not Started	All necessary documents are listed.		
			2. Create templates for test reports, defect logs, etc.					
	TC005	Test Strategy	1. Decide on the testing approach and techniques.	Verify the overall approach for testing.	Not Started	The test strategy is defined.		

Processes	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comments
			2. Determine the entry and exit criteria.					
	TC006	Test Scope and Schedule	1. Define the timeline for each testing phase.	Verify the schedule for testing.	Not Started	The schedule is established.		
			2. Determine the scope of testing for each phase.					
	TC007	Risk Analysis	1. Identify potential risks in the testing process.	Verify risk analysis and mitigation strategies.	Not Started	Potential risks are identified with mitigation plans.		
			2. Discuss risk mitigation strategies with the team.					
	TC008	Resource Planning	1. Allocate the required resources for testing.	Verify the availability of resources.	Not Started	Resources needed for testing are allocated.		
			2. Determine the roles and responsibilities of the team.					

Test Case Design:

Process	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
Test Case Design	TC001	Module A - Functionality Test	1. Review the requirements related to Module A.	Verify the functionality of Module A.	Not Started	All functionalities of Module A are tested.		
			2. Identify test scenarios for Module A.					
			3. Develop detailed test cases for Module A.					
	TC002	Module B - Integration Test	1. Review the requirements related to Module B.	Verify the integration of Module B with others.	Not Started	Module B is successfully integrated.		
			2. Identify integration points with other modules.					
			3. Design test cases for testing integration scenarios.					

Processes	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
	TC003	Module C - Performance Test	1. Review the performance requirements for Module C.	Verify the performance of Module C.	Not Started	Module C performs optimally under load.		
			2. Determine performance metrics to be measured.					
			3. Develop performance test cases for Module C.					
	TC004	Module D - Usability Test	1. Review the usability requirements for Module D.	Verify the usability of Module D.	Not Started	Module D is user-friendly and intuitive.		
			2. Identify usability aspects to be tested.					
			3. Create test cases for evaluating Module D's usability.					
	TC005	Module E - Security Test	1. Review the security requirements for Module E.	Verify the security of Module E.	Not Started	Module E is protected against security threats.		
			2. Identify potential					

Processes	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
			security vulnerabilities					
			3. Design security test cases to assess Module E.					

Explanation:

An inventory control system is critical for managing stock and supplies. Proper testing ensures the system functions accurately and efficiently.

Result:

Upon completion of the experiment, you will have a well-structured test plan and a set of test cases ready for testing the Inventory Control System.

Experiment 5: Execute the Test Cases against a Client-Server or Desktop Application and Identify the Defects

Aim:

The aim of this experiment is to execute the test cases against a client-server or desktop application and identify defects.

Algorithm:

1. Test Environment Setup: Set up the testing environment, including the client-server or desktop application, required hardware, and test data.
2. Test Case Execution: Execute the test cases designed in Experiment 2 against the application.
3. Defect Identification: During test execution, record any discrepancies or issues encountered.
4. Defect Reporting: Log the identified defects with detailed information, including steps to reproduce, severity, and priority.
5. Defect Tracking: Track the progress of defect resolution and verify fixes as they are implemented.
6. Retesting: After defect fixes, retest the affected areas to ensure the issues are resolved.
7. Regression Testing: Conduct regression testing to ensure new changes do not introduce new defects.

Test Case Table:

Process	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
Test Case Execution	TC001	User Login	1. Launch the application.	Verify user login process.	Not Started	User can successfully log in.		
			2. Enter valid login credentials.					
			3. Click on the "Login" button.					
	TC002	Data Validation	1. Access a data input form.	Verify data validation on the form.	Not Started	Invalid data shows appropriate error messages.		
			2. Enter invalid data in the form fields.					
			3. Submit the form.					
	TC003	File Upload	1. Access the file upload feature.	Verify file upload functionality.	Not Started	File is uploaded successfully.		
			2. Select a file from the system.					
			3. Click on the "Upload" button.					

	TC004	Network Connectivity	1. Disconnect the network.	Verify the application's response.	Not Started	Application gracefully handles disconnection.		
			2. Attempt to perform an action requiring network access.					
	TC005	Concurrent Users	1. Simulate concurrent user sessions.	Verify application performance.	Not Started	Application performs well under load.		
			2. Perform actions simultaneously.					
	TC006	Compatibility	1. Test the application on different platforms.	Verify cross-platform compatibility.	Not Started	Application works on all specified platforms.		
			2. Execute tests on various browsers.					
	TC007	Client-Server Communication	1. Monitor network traffic between client and server.	Verify communication integrity.	Not Started	Data is correctly transmitted and received.		

Explanation:

Testing a client-server or desktop application ensures its functionality across different platforms and environments.

Result:

Upon completion of the experiment, you will have a list of identified defects and their status after resolution for the client-server or desktop application.

Experiment 6: Test the Performance of the E-commerce Application

Aim:

The aim of this experiment is to test the performance of the e-commerce application www.amazon.in.

Algorithm:

1. Identify Performance Metrics: Determine the performance metrics to be measured, such as response time, throughput, and resource utilization.
2. Define Test Scenarios: Create test scenarios that simulate various user interactions and loads on the application.
3. Performance Test Setup: Set up the performance testing environment with appropriate hardware and software.
4. Execute Performance Tests: Run the performance tests using the defined scenarios and collect performance data.
5. Analyze Performance Data: Analyze the collected data to identify any performance bottlenecks or issues.
6. Performance Tuning: Implement necessary optimizations to improve the application's performance.

Performance Table:

Process	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
Performance		Response Time for	1. Access the home page of the e-commerce	Measure the response	Not	The home page loads within the specified response time		

Testing	TC001	Home Page	application.	time.	Started	threshold.		
---------	-------	-----------	--------------	-------	---------	------------	--	--

Process	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
			2. Use a performance testing tool to record the time.					
			3. Analyze the recorded data to determine response time.					
	TC002	Throughput during Peak Hours	1. Simulate peak-hour traffic on the application.	Measure the throughput.	Not Started	The application can handle peak-hour traffic without significant delays.		
			2. Execute performance tests during peak hours.					
			3. Analyze the data to determine the throughput.					
	TC003	Resource Utilization	1. Monitor CPU, memory, and network usage during testing.	Measure resource utilization.	Not Started	Resource utilization remains within acceptable limits.		

			2. Execute performan ce tests while monitoring resources.					
--	--	--	--	--	--	--	--	--

Process	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
			3. Analyze the data to assess resource utilization.					
	TC004	Concurrent Users	1. Simulate multiple concurrent users accessing the app.	Measure app performance under load.	Not Started	The application remains stable and responsive under load.		
			2. Increase the number of concurrent users gradually.					
			3. Record the application's behavior with increased load.					
	TC005	Stress Testing	1. Apply maximum load to test the system's breaking point.	Measure system behavior under extreme load.	Not Started	The system recovers gracefully after stress is removed.		
			2. Apply the maximum user load the application can handle.					
			3. Observe the application's response under stress.					

Process	No.	Test Case	Steps	Description	Status	Expected Result	Actual Result	Comment
	TC006	Performance Tuning	1. Identify performance bottlenecks and areas of improvement.	Improve application performance.	Not Started	Performance bottlenecks are addressed and application performs better.		
			2. Analyze the performance test results.					
			3. Implement necessary optimizations.					

Explanation:

Performance testing helps to identify bottlenecks in the e-commerce application, ensuring it can handle real-world user loads effectively.

Result:

Upon completion of the experiment, you will have performance test results and any optimizations made to improve the application's performance.

Experiment 7: Automate the testing of e-commerce applications using Selenium.

Aim:

The aim of this task is to automate the testing of an e-commerce web application (www.amazon.in) using Selenium WebDriver, which will help improve testing efficiency and reliability.

Algorithm:

1. Set up the environment:

- Install Java Development Kit (JDK) and configure the Java environment variables.
- Install an Integrated Development Environment (IDE) like Eclipse or IntelliJ.
- Download Selenium WebDriver and the required web drivers for the browsers you intend to test (e.g., ChromeDriver, GeckoDriver for Firefox).

2. Create a new Java project in the IDE:

- Set up a new Java project in the IDE and include the Selenium WebDriver library.

3. Develop test cases:

- Identify the key functionalities and scenarios to test in the e-commerce application.
- Design test cases covering various aspects like login, search, product details, add to cart, checkout, etc.

4. Implement Selenium automation scripts:

- Write Java code using Selenium WebDriver to automate the identified test cases.
- Utilize different Selenium commands to interact with the web elements, navigate through pages, and perform various actions.

5. Execute the automated test cases:

- Run the automated test scripts against the e-commerce application.
- Observe the test execution and identify any failures or defects.

6. Analyze the test results:

- Review the test execution results to identify any failed test cases.
- Debug and fix any issues with the automation scripts if necessary.

7. Report defects:

- Document any defects found during the automated testing process.
- Provide detailed information about each defect, including steps to reproduce and expected results.

Program:

```
package program;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import
org.openqa.selenium.chrome.ChromeDriver;

public class selenium {
public static void main(String[] args)
{
System.setProperty("webdriver.chrome.driver","C:\\Users\\Admin\\Downloads\\
chromedriver- win64\\chromedriver-win64\\chromedriver.exe");

WebDriver d=new
ChromeDriver();

d.get("https://www.amazon.in");

d.findElement(By.xpath("//*[@id=\"nav-link-accountList\"]/span/span")).click();
d.findElement(By.id("ap_email")).sendKeys("youremail@gmail.com");
d.findElement(By.xpath("//*[@id=\"continue\"]")).click();
d.findElement(By.id("ap_password")).sendKeys("your password");
d.findElement(By.xpath("//*[@id=\"signInSubmit\"]")).click();
String u=d.getCurrentUrl(); if(u.equals("https://www.amazon.in/?
ref_=nav_ya_signin"))
{
```

```
        System.out.println("Test Case Passed");  
    }
```

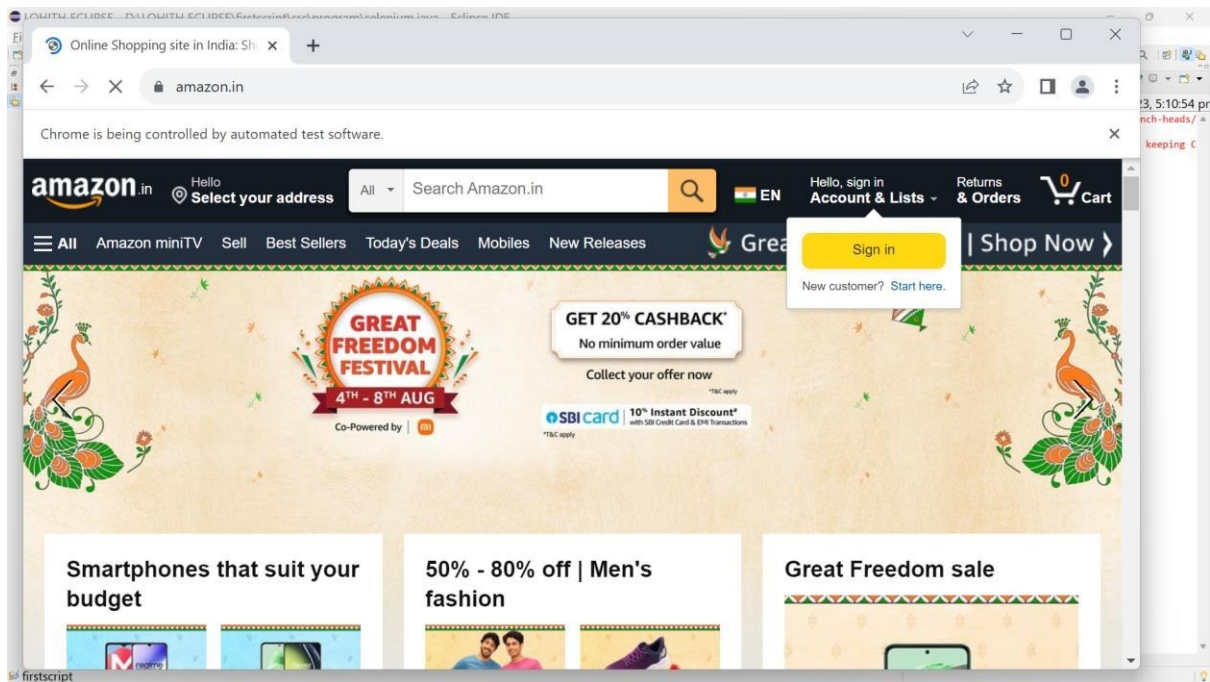


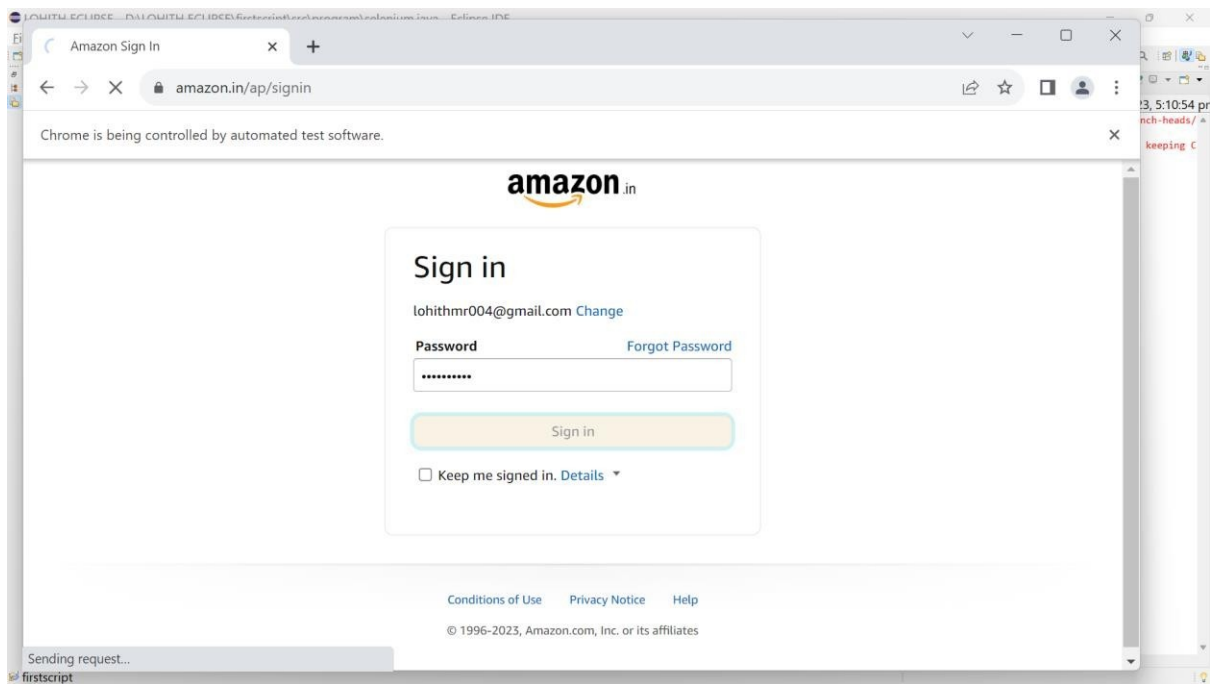
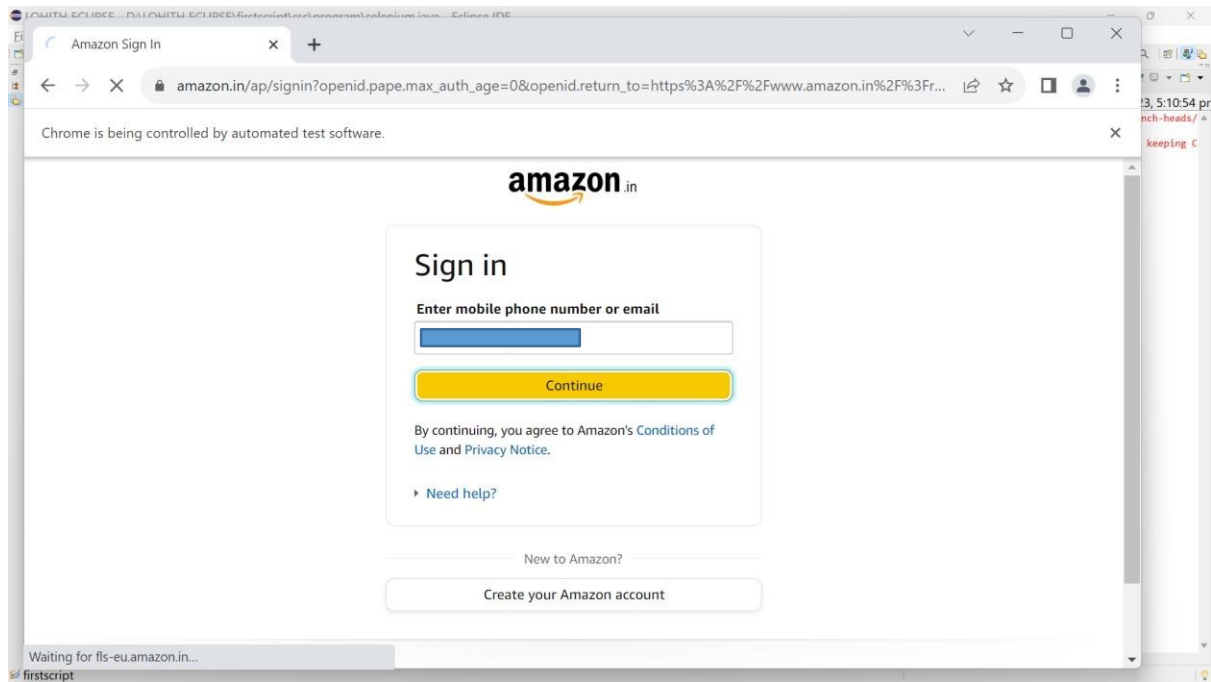
```

els
e
{
    System.out.println("Test Case Failed");
}
d.close();
}
}

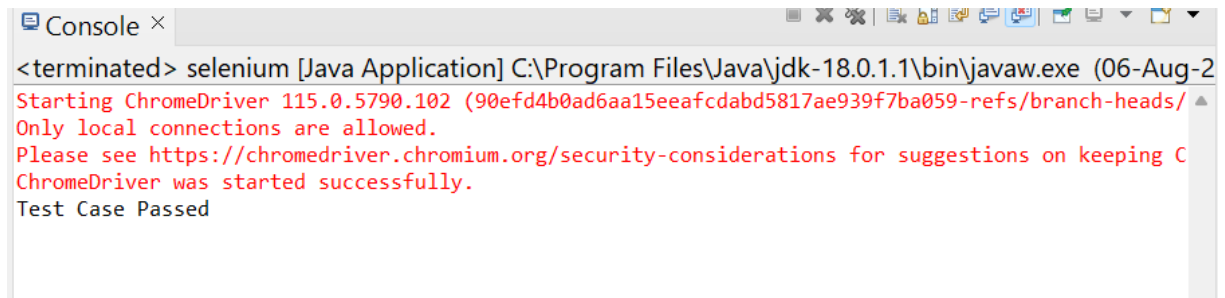
```

Automation Process:





Console output:

A screenshot of a Java console window titled "Console x". The window shows the output of a Selenium WebDriver test. The text is as follows:

```
<terminated> selenium [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.exe (06-Aug-2024)
Starting ChromeDriver 115.0.5790.102 (90efd4b0ad6aa15eeafcdabd5817ae939f7ba059-refs/branch-heads/115@{#})
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping C
ChromeDriver was started successfully.
Test Case Passed
```

Result:

The successful completion of this task will yield:

- Automated test scripts for the e-commerce application using Selenium WebDriver.
- Identification of defects, if any, in the application.

Experiment 8: Integrate TestNG with the above test automation.

Aim:

The aim of this task is to integrate TestNG with the existing Selenium automation scripts for the e-commerce application, enhancing test management, parallel execution, and reporting capabilities.

Algorithm:

1. Set up TestNG in the project:
 - Add TestNG library to the existing Java project.
2. Organize test cases using TestNG annotations:
 - Add TestNG annotations (@Test, @BeforeTest, @AfterTest, etc.) to the existing test cases.
 - Group similar test cases using TestNG's grouping mechanism.
3. Implement data-driven testing (optional):
 - Utilize TestNG's data providers to implement data-driven testing if required.
4. Configure TestNG test suite:
 - Create an XML configuration file for TestNG to define test suites, test groups, and other configurations.
5. Execute the automated test cases using TestNG:
 - Run the automated test suite using TestNG.
 - Observe the test execution and identify any failures or defects.
6. Analyze the test results:
 - Review the TestNG-generated test reports to identify any failed test cases.
 - Utilize TestNG's reporting capabilities to understand the test execution status.
7. Report defects (if any):
 - Document any defects found during the automated testing process.

- Provide detailed information about each defect, including steps to reproduce and expected results.

Program Code (Program1.java) :

```
package mytest;

import java.time.Duration;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class Program1 {

    WebDriver driver;

    @BeforeMethod

    public void setUp()

    {

        System.setProperty("webdriver.chrome.driver","C:\\selenium\\chromedriver_win32\\chromedriver.exe");

        driver=new ChromeDriver();

        driver.get("https://amazon.in");
```

```
driver.manage().window().maximize();  
  
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));  
  
}
```

```

    @Test

    public void verifyTitle()

    {

        String actualTitle=driver.getTitle();

        String expectedTitle="Online Shopping site in India: Shop Online for Mobiles, Books,
        Watches, Shoes and More - Amazon.in";

        Assert.assertEquals(actualTitle, expectedTitle);

    }

    @Test

    public void verifyLogo()

    {

        boolean flag=driver.findElement(By.xpath("//a[@id='nav-logo-sprites']")).isDisplayed();

        Assert.assertTrue(flag);

    }

    @AfterMethod

    public void tearDown()

    {

        driver.quit();

    }

}

```

Program Code (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 http://maven.apache.org/xsd/maven-
4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>MiniProject2</groupId>

```

```
<artifactId>MiniProject2</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

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

<dependency>

  <groupId>org.seleniumhq.selenium</groupId>

  <artifactId>selenium-java</artifactId>

  <version>4.3.0</version>

</dependency>

</dependencies>

<build>

  <sourceDirectory>src</sourceDirectory>

  <plugins>

    <plugin>

      <artifactId>maven-compiler-plugin</artifactId>

      <version>3.8.1</version>

      <configuration>

        <release>16</release>

      </configuration>

    </plugin>

  </plugins>

</build>
```


</project>

Program Code (testng.xml) :

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

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Suite">

<test name="Test">

<classes>

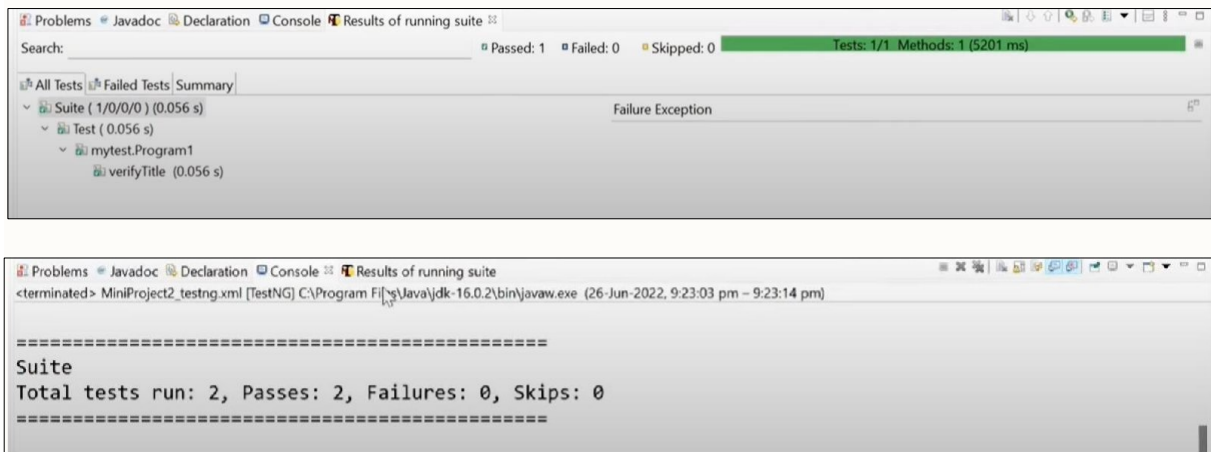
<class name="mytest.Program1"></class>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->
```

Output:



Result:

The successful completion of this task will yield:

- Integration of TestNG with the existing Selenium automation scripts.
- Enhanced test management and reporting capabilities.
- Identification of defects, if any, in the application and improved efficiency in handling test scenarios.