



# AISSMS

COLLEGE OF ENGINEERING

ज्ञानम् सकलजनहिताय



Approved by AICTE, New Delhi, Recognized by Government of Maharashtra  
Affiliated to Savitribai Phule Pune University and recognized 2(f) and 12(B) by UGC  
(Id.No. PU/PN/Engg./093 (1992))

Accredited by NAAC with "A+" Grade | NBA - 7 UG Programmes

**Department of Computer Engineering**

**“STQA Miniproject”**

**“Automation Testing Using Selenium Web-driver”**

*Submitted in partial fulfillment of the requirements for the degree of*

**BACHELOR OF ENGINEERING**

**In**

**COMPUTER ENGINEERING**

*Submitted By*

**Name of the Student: Piyusha Rajendra Supe**

**Roll No: 23CO315**

*Under the Guidance of*

**Mr. S. R. Bhise**

**ALL INDIA SHRI SHIVAJI MEMORIAL SOCIETY'S COLLEGE OF  
ENGINEERING PUNE-411001**

Academic Year: 2025-26 (Term-I)

**Savitribai Phule Pune University**



# AISSMS

COLLEGE OF ENGINEERING

ज्ञानम् सकलजनहिताय



Approved by AICTE, New Delhi, Recognized by Government of Maharashtra  
Affiliated to Savitribai Phule Pune University and recognized 2(f) and 12(B) by UGC  
(Id.No. PU/PN/Engg./093 (1992))

Accredited by NAAC with "A+" Grade | NBA - 7 UG Programmes

## Department of Computer Engineering

### CERTIFICATE

This is to certify that **Piyusha Rajendra Supe** from **Fourth Year Computer Engineering** has successfully completed her work titled "**Software Testing and Quality Assurance Mini-project**" at AISSMS College of Engineering, Pune in the partial fulfillment of the Bachelor's Degree in Computer Engineering.

**Mr. S. R. Bhise**  
(Faculty In-charge)  
Computer Engineering

**Dr. S. V. Athawale**  
(Head of Department)  
Computer Engineering

**Dr. D. S. Bormane**  
(Principal)  
AISSMSCOE, Pune

## **ACKNOWLEDGEMENT**

It is with profound gratitude and deep respect that I take this opportunity to acknowledge the invaluable support and guidance I have received throughout the course of my STQA project. This journey has been both intellectually enriching and personally fulfilling, significantly enhancing my understanding of testing tools, technologies, and their real-world application. First and foremost, I would like to express my heartfelt thanks to **Mr. S. R. Bhise** for his expert guidance, unwavering support, and constructive feedback. His mentorship and encouragement were instrumental in shaping the direction, depth, and quality of this project. I also extend my sincere appreciation to the Head of the Department for their visionary leadership and for fostering an environment that encourages academic growth and innovation. The resources and opportunities provided under their guidance played a crucial role in the successful completion of this project. A special note of thanks goes to the supporting staff, whose timely assistance and cooperation ensured that the process was smooth and efficient. Their commitment and dedication were invaluable during every phase of this project. Finally, I would like to express my deepest gratitude to my parents, whose constant love, patience, and unwavering support have been my source of strength throughout this journey. Their belief in my abilities and their encouragement played a crucial role in keeping me motivated and focused. This project has not only strengthened my technical skills but has also taught me the importance of perseverance and continuous learning. I remain sincerely thankful to everyone who contributed to the successful completion of this project.

**Academic Year: 2025-2026**

**Piyusha Rajendra Supe (23CO315)**

## **TABLE OF CONTENTS**

<b>Sr. No</b>	<b>Title</b>	<b>Page No.</b>
1	Acknowledgement.....	3
2	Abstract .....	5
3	Introduction.....	6
4	System Requirements .....	7
5	Methodology.....	8-10
6	Implementation.....	11-13
7	Test cases executed and bug taxonomy.....	14-17
8	Test Execution Report .....	18-20
9	Test Result/Summary report .....	21-25
10	Conclusion.....	26
11	References.....	26

## **ABSTRACT**

This document presents the execution and results of an extensive suite of test cases automated using Selenium WebDriver. The primary objective of this testing effort was to ensure the functional correctness, usability, and reliability of the target web application across its key modules. The test suite was systematically designed to cover critical functionalities such as login, user management, data entry, validations, error handling, and workflow navigation. Each test case was uniquely identified and mapped to a specific module, with clearly defined test descriptions, preconditions, test steps, input data, and expected outcomes. Using Selenium WebDriver as the automation framework, the execution of these test cases enabled consistent and repeatable testing across different browsers and environments. This approach minimized manual effort and reduced human error, while improving the speed and accuracy of the testing process. The testing focused not only on positive scenarios (valid inputs and expected behaviour) but also on negative scenarios (invalid inputs, boundary conditions, and error messages), ensuring comprehensive coverage of the application. For each executed test case, detailed results were recorded in a structured format consisting of Test Case ID, Module, Test Description, Test Steps, Test Data, Expected Result, Actual Result, and Status. The majority of test cases passed successfully, demonstrating the stability and correctness of the application. Failures and deviations, where observed, were documented for further investigation and defect resolution. The execution of these test cases using Selenium validated the automation strategy's effectiveness in improving overall test coverage, reducing execution time, and supporting continuous integration workflows. This effort highlights the critical role of automation testing in delivering high-quality software, enabling faster release cycles, and ensuring enhanced user satisfaction.

## **INTRODUCTION**

Software testing is a critical phase of the Software Development Life Cycle (SDLC) that ensures the delivery of reliable, defect-free, and user-centric applications. As web-based systems continue to grow in complexity, manual testing alone becomes time-consuming, repetitive, and prone to human error. Automation testing has therefore emerged as a key practice to improve accuracy, speed, and consistency of testing activities. Among the various automation tools available, Selenium WebDriver stands out as one of the most widely adopted, owing to its open-source nature, cross-browser compatibility, and ability to simulate real user interactions with a web application.

The testing effort encompassed several critical objectives:

- To verify that all core modules (such as authentication, user management, data handling, and form submissions) perform as expected under various conditions.
- To validate application responses to valid, invalid, and boundary input data.
- To assess the stability and reliability of the application during normal and exceptional workflows.
- To accelerate the testing process through automation while maintaining consistency across executions.

Selenium WebDriver's architecture allowed seamless integration with multiple browsers and environments, enabling parallel test execution where required. This not only reduced overall execution time but also provided confidence in the application's cross-browser behavior. By leveraging Selenium's automation capabilities, the testing team was able to repeatedly execute the same test cases with minimal additional effort, ensuring quick regression testing after code changes or feature enhancements.

The test execution process followed a structured approach, starting from environment setup and data preparation, to the actual execution of test scripts and recording of outcomes. Each test case result was logged in a standardized table format—Test Case ID, Module, Test Description, Test Steps, Test Data, Expected Result, Actual Result, and Status—to provide a clear view of the application's quality status. Any deviations from the expected outcomes were promptly captured and reported for defect resolution, closing the loop between testing and development.

## **SYSTEM REQUIREMENTS**

### **1. Hardware**

<b>Component</b>	<b>Minimum</b>	<b>Recommended</b>
<b>Processor</b>	Intel Core i3, 2.0 GHz+	Intel Core i5/i7, 3.0 GHz+
<b>RAM</b>	4 GB	8 GB+
<b>Storage</b>	10 GB free space	20 GB free space
<b>Display</b>	1366×768	1920×1080 (Full HD)
<b>Internet</b>	5 Mbps stable	10 Mbps+

### **2. Software**

<b>Component</b>	<b>Minimum</b>	<b>Recommended</b>
<b>OS</b>	Windows 10 / macOS 10.14+ / Ubuntu 20.04	Windows 11 / macOS 12+ / Ubuntu 22.04
<b>JDK</b>	JDK 8+	JDK 11+
<b>IDE</b>	Eclipse 2021 / IntelliJ Community	IntelliJ Ultimate / Latest Eclipse
<b>Build Tool</b>	Maven 3.6+	Maven 3.9+
<b>Browsers</b>	Chrome 114+, Firefox 110+, Edge 110+	Latest stable versions
<b>Drivers</b>	ChromeDriver, GeckoDriver, EdgeDriver	Latest stable drivers
<b>Selenium</b>	4.10+	Latest 4.x
<b>Testing Framework</b>	TestNG 7.5+ / JUnit 5+	Latest stable version
<b>Optional</b>	ExtentReports, Apache POI, Log4j	Latest versions

### **3. Additional**

- Access to the PHPTravels test URL (<https://phptravels.com/demo/>).
- Valid login credentials for testing secure modules.
- Selenium Grid or a cloud platform for cross-browser/parallel testing.

# METHODOLOGY

The steps followed are as follows:

## Step 1: Install Java Development Kit (JDK)

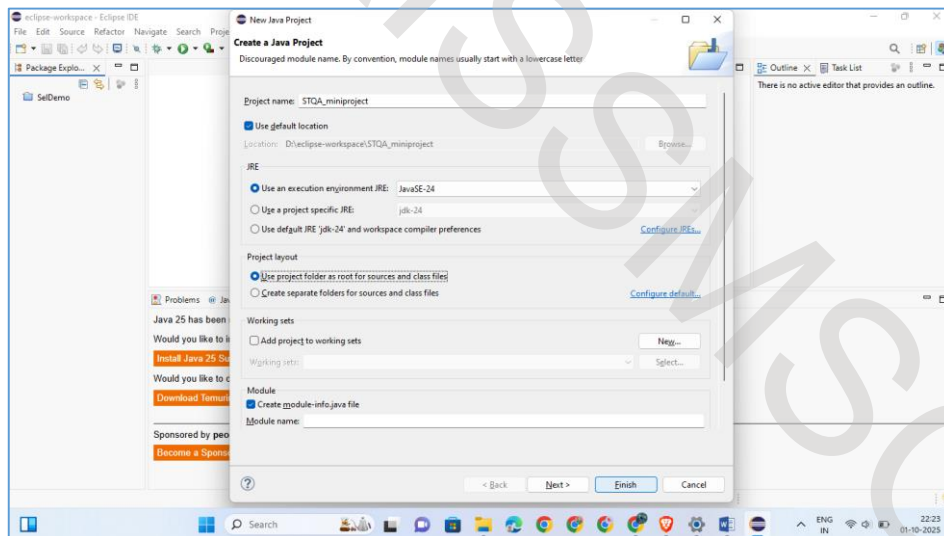
1. Download the latest JDK (8 or higher) from Oracle or OpenJDK.
2. Install it and set the JAVA\_HOME environment variable.
3. Verify installation by running java -version in Command Prompt.

## Step 2: Install Eclipse IDE

1. Download Eclipse IDE for Java Developers from <https://eclipse.org>.
2. Extract and run the installer.
3. Launch Eclipse and choose a workspace directory.

## Step 3: Create a New Java Project

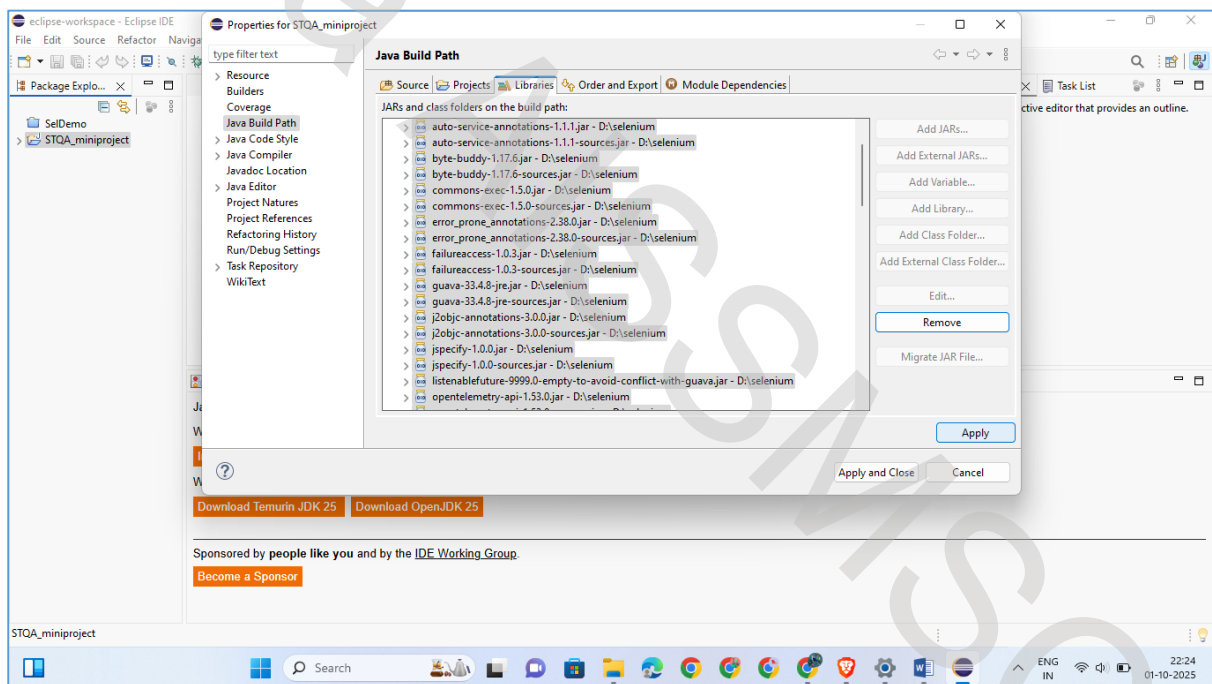
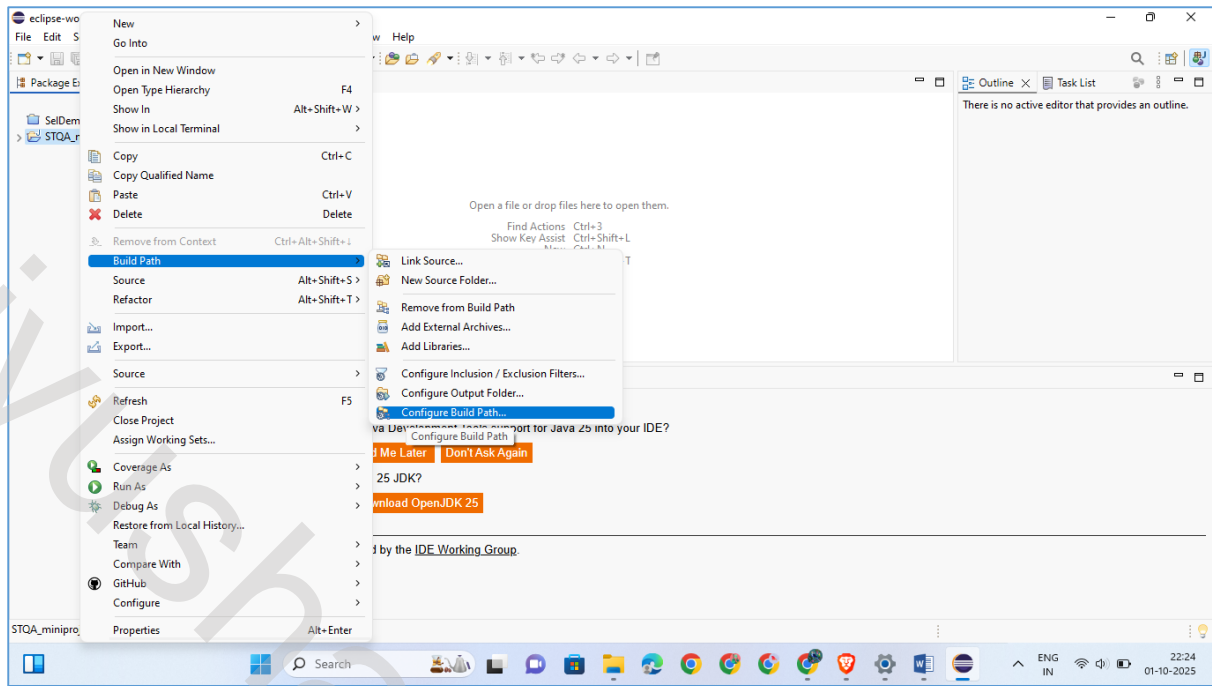
1. In Eclipse, go to **File** → **New** → **Java Project**.
2. Enter a project name (e.g., SeleniumDemo) and click **Finish**.
3. Inside the project, create a new **Package** and then a new **Class**.



## Step 4: Add Selenium WebDriver JAR Files

1. Download the Selenium Java Client library (zip) from <https://www.selenium.dev/downloads/>.
2. Extract the zip; you'll find selenium-java-x.xx.x.jar and libs folder.
3. In Eclipse, right-click your project → **Build Path** → **Configure Build Path**.
4. Click **Add External JARs**, select all Selenium JARs (main + libs), and apply.





## Step 5: Download and Configure Browser Driver

1. Download the appropriate driver for your browser:
  - ChromeDriver: <https://chromedriver.chromium.org/>
  - GeckoDriver for Firefox, EdgeDriver for Edge.
2. Extract the driver (e.g., chromedriver.exe) to a known path (like D:\drivers).

## Step 6: Write a Simple Selenium Program

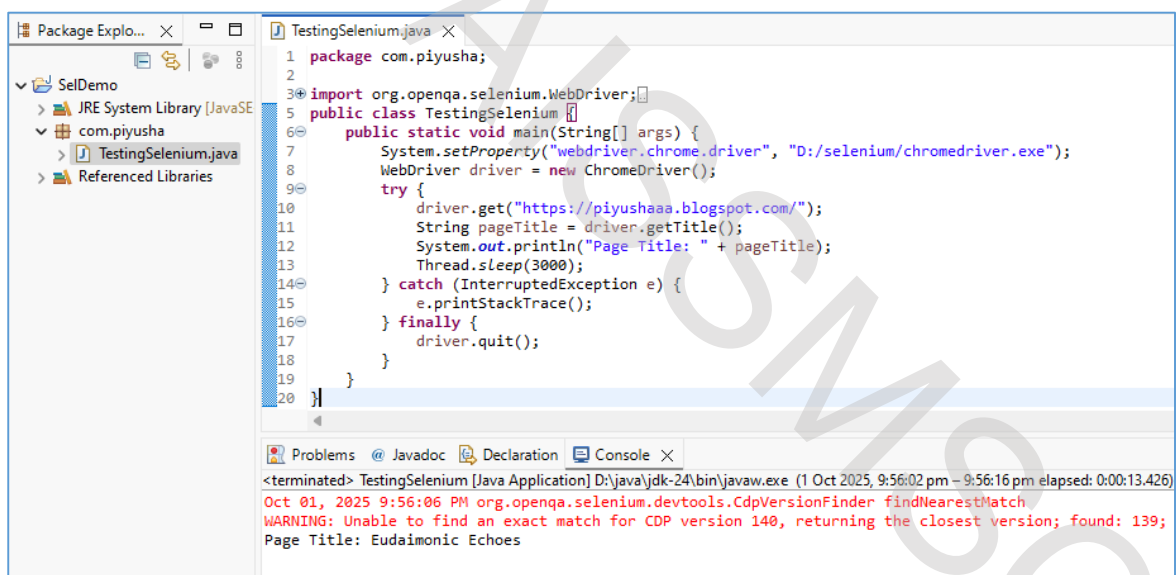
Example Java code to open a page and print its title

## Step 7: Run the Program

1. In Eclipse, right-click the Java class → **Run As** → **Java Application**.
2. Observe the browser launching and navigating to the specified URL.
3. Check the **Console** tab in Eclipse for printed output.

## Step 8: Validate Execution

- If the browser opens and closes automatically and the console shows the title, Selenium is correctly set up.
- If errors occur (e.g., NoSuchElementException), verify driver path and versions.



The screenshot shows the Eclipse IDE with a Java project named 'SelDemo'. The 'Package Explorer' on the left shows the project structure with 'com.piyusha' and 'TestingSelenium.java'. The 'Editor' window displays the following Java code:

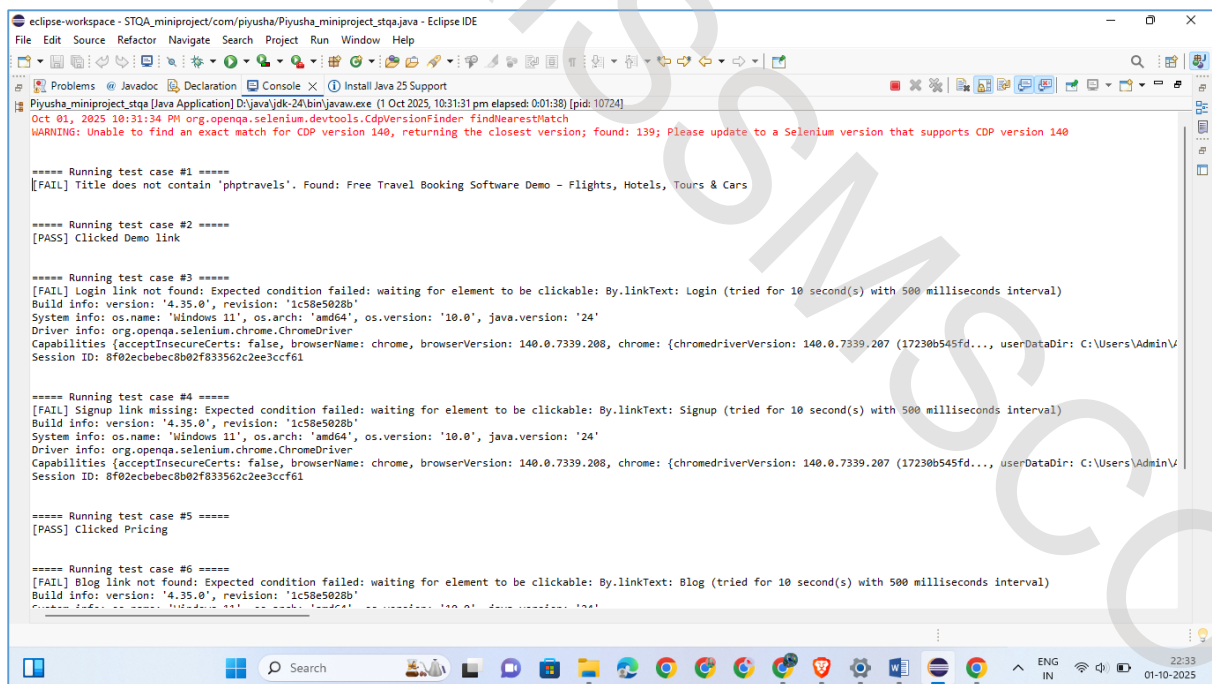
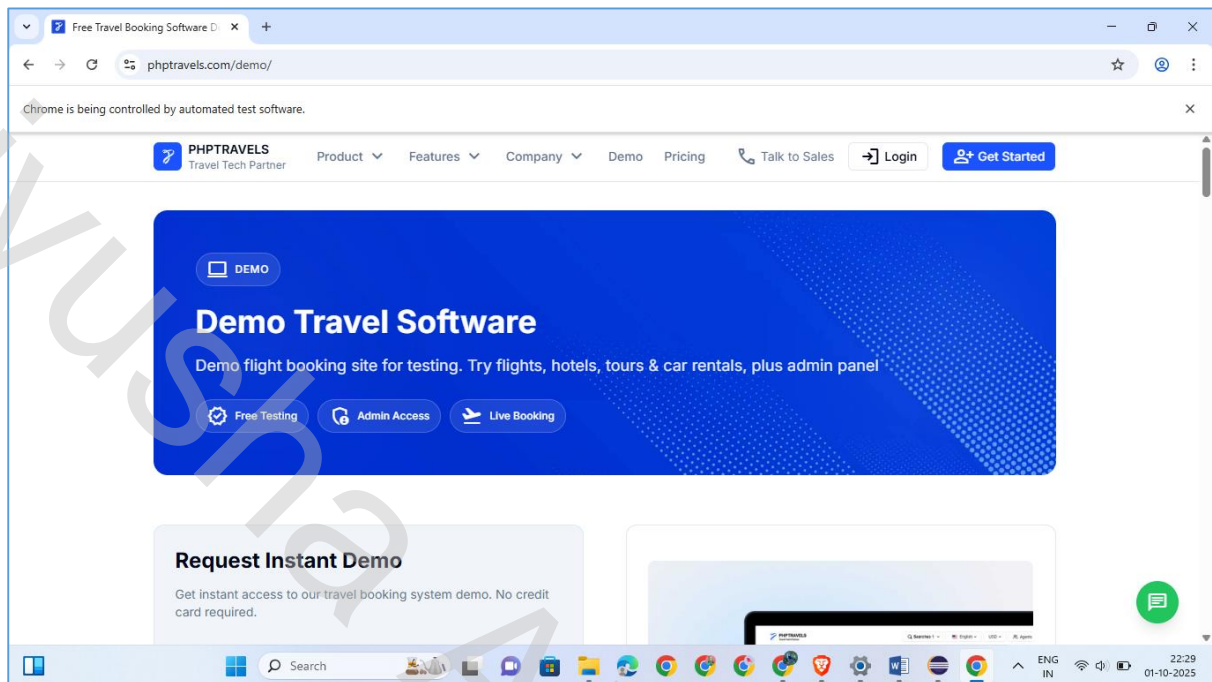
```
1 package com.piyusha;
2
3 import org.openqa.selenium.WebDriver;
4
5 public class TestingSelenium {
6     public static void main(String[] args) {
7         System.setProperty("webdriver.chrome.driver", "D:/selenium/chromedriver.exe");
8         WebDriver driver = new ChromeDriver();
9         try {
10             driver.get("https://piyushaaa.blogspot.com/");
11             String pageTitle = driver.getTitle();
12             System.out.println("Page Title: " + pageTitle);
13             Thread.sleep(3000);
14         } catch (InterruptedException e) {
15             e.printStackTrace();
16         } finally {
17             driver.quit();
18         }
19     }
20 }
```

The 'Console' tab at the bottom shows the following output:

```
<terminated> TestingSelenium [Java Application] D:\java\jdk-24\bin\javaw.exe (1 Oct 2025, 9:56:02 pm - 9:56:16 pm elapsed: 0:00:13.426)
Oct 01, 2025 9:56:06 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 140, returning the closest version; found: 139;
Page Title: Eudaimonic Echoes
```

# IMPLEMENTATION

The test cases executed resulted in the following output



```
eclipse-workspace - STQA_miniproject/com/piyusha/Piyusha_miniproject_stqa.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Piyusha_miniproject_stqa [Java Application] D:\java\jdk-24\bin\javaw.exe (1 Oct 2025, 10:31:31 pm elapsed: 0:03:47) [pid: 10724]

===== Running test case #10 =====
[FAIL] Car search failed: Expected condition failed: waiting for element to be clickable: By.id: cars-tab (tried for 10 second(s) with 500 milliseconds interval)
Build info: version: '4.35.0', revision: '1c58e5028b'
System info: os.name: 'Windows 11', os.arch: 'amd64', os.version: '10.0', java.version: '24'
Driver info: org.openqa.selenium.chrome.ChromeDriver
Capabilities {acceptInsecureCerts: false, browserName: chrome, browserVersion: 140.0.7339.208, chrome: {chromedriverVersion: 140.0.7339.207 (17230b545fd..., userDataDir: C:\Users\AdminV\
Session ID: 8f02ecbebec8b02f833562c2ee3ccf61

===== Running test case #11 =====
(PASS) Footer 'About Us' link visible

===== Running test case #12 =====
(PASS) Contact link found: Contact

===== Running test case #13 =====
[FAIL] Invalid login test failed: Expected condition failed: waiting for element to be clickable: By.linkText: Login (tried for 10 second(s) with 500 milliseconds interval)
Build info: version: '4.35.0', revision: '1c58e5028b'
System info: os.name: 'Windows 11', os.arch: 'amd64', os.version: '10.0', java.version: '24'
Driver info: org.openqa.selenium.chrome.ChromeDriver
Capabilities {acceptInsecureCerts: false, browserName: chrome, browserVersion: 140.0.7339.208, chrome: {chromedriverVersion: 140.0.7339.207 (17230b545fd..., userDataDir: C:\Users\AdminV\
Session ID: 8f02ecbebec8b02f833562c2ee3ccf61

===== Running test case #14 =====
[FAIL] Valid login failed: Expected condition failed: waiting for element to be clickable: By.linkText: Login (tried for 10 second(s) with 500 milliseconds interval)
Build info: version: '4.35.0', revision: '1c58e5028b'
System info: os.name: 'Windows 11', os.arch: 'amd64', os.version: '10.0', java.version: '24'
Driver info: org.openqa.selenium.chrome.ChromeDriver
Capabilities {acceptInsecureCerts: false, browserName: chrome, browserVersion: 140.0.7339.208, chrome: {chromedriverVersion: 140.0.7339.207 (17230b545fd..., userDataDir: C:\Users\AdminV\
Session ID: 8f02ecbebec8b02f833562c2ee3ccf61

===== Running test case #15 =====
[FAIL] Logout not found: Expected condition failed: waiting for element to be clickable: By.linkText: Logout (tried for 10 second(s) with 500 milliseconds interval)
```

```
eclipse-workspace - STQA_miniproject/com/piyusha/Piyusha_miniproject_stqa.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Piyusha_miniproject_stqa [Java Application] D:\java\jdk-24\bin\javaw.exe (1 Oct 2025, 10:31:31 pm elapsed: 0:05:24) [pid: 10724]

===== Running test case #27 =====
[FAIL] Contact form validation failed: Expected condition failed: waiting for element to be clickable: By.cssSelector: button.submit-btn (tried for 10 second(s) with 500 milliseconds interval)
Build info: version: '4.35.0', revision: '1c58e5028b'
System info: os.name: 'Windows 11', os.arch: 'amd64', os.version: '10.0', java.version: '24'
Driver info: org.openqa.selenium.chrome.ChromeDriver
Capabilities {acceptInsecureCerts: false, browserName: chrome, browserVersion: 140.0.7339.208, chrome: {chromedriverVersion: 140.0.7339.207 (17230b545fd..., userDataDir: C:\Users\AdminV\
Session ID: 8f02ecbebec8b02f833562c2ee3ccf61

===== Running test case #28 =====
[FAIL] Contact form submission failed: Expected condition failed: waiting for visibility of element located by By.name: name (tried for 10 second(s) with 500 milliseconds interval)
Build info: version: '4.35.0', revision: '1c58e5028b'
System info: os.name: 'Windows 11', os.arch: 'amd64', os.version: '10.0', java.version: '24'
Driver info: org.openqa.selenium.chrome.ChromeDriver
Capabilities {acceptInsecureCerts: false, browserName: chrome, browserVersion: 140.0.7339.208, chrome: {chromedriverVersion: 140.0.7339.207 (17230b545fd..., userDataDir: C:\Users\AdminV\
Session ID: 8f02ecbebec8b02f833562c2ee3ccf61

===== Running test case #29 =====
[FAIL] Search box placeholder not found: no such element: Unable to locate element: {"method":"css_selector","selector":"input[type='search']"}
(Session info: chrome=140.0.7339.208)
For documentation on this error, please visit: https://www.selenium.dev/documentation/webdriver/troubleshooting/errors#no-such-element-exception
Build info: version: '4.35.0', revision: '1c58e5028b'
System info: os.name: 'Windows 11', os.arch: 'amd64', os.version: '10.0', java.version: '24'
Driver info: org.openqa.selenium.chrome.ChromeDriver
Command: [8f02ecbebec8b02f833562c2ee3ccf61, findElement {using=css selector, value=input[type='search']}]
Capabilities {acceptInsecureCerts: false, browserName: chrome, browserVersion: 140.0.7339.208, chrome: {chromedriverVersion: 140.0.7339.207 (17230b545fd..., userDataDir: C:\Users\AdminV\
Session ID: 8f02ecbebec8b02f833562c2ee3ccf61

===== Running test case #30 =====
[PASS] Meta description: Experience our PHP-based travel booking system demo. Test flights, hotels, tours, and car rentals with full admin panel access-free and no
```

**The java code for executing one of the test cases is as follows:**

```
package com.piyusha;

import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;
import java.util.ArrayList;
```

```

import java.util.List;

public class Piyusha_miniproject_stqa {

    public static void main(String[] args) {

        // Set path to your chromedriver

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

        WebDriver driver = new ChromeDriver();

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

        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

        String baseUrl = "https://phptravels.com/demo/";

        List<Runnable> tests = new ArrayList<>();

        // --- Test 1: Home page title contains "phptravels" ---

        tests.add(() -> {

            driver.get(baseUrl);

            String title = driver.getTitle();

            if (title.toLowerCase().contains("phptravels")) {

                System.out.println("[PASS] Title contains 'phptravels': " + title);

            } else {

                System.out.println("[FAIL] Title does not contain 'phptravels'. Found: " + title);

            }

        });

        // ===== Run all tests =====

        for (int i = 0; i < tests.size(); i++) {

            System.out.println("\n\n===== Running test case #" + (i + 1) + " =====");

            try {

                tests.get(i).run();

                Thread.sleep(1000); // small pause

            } catch (Exception e) {

                System.out.println("[ERROR] Test " + (i + 1) + " threw exception: " + e.getMessage());

            }

        }

        driver.quit();

        System.out.println("===== All tests done =====");

    }

}

```

The test cases executed using the script are as follows:

ID	Module	Test Description	Test Steps	Test Data	Expected Result	Actual Result	Status
1	Home Page	Verify title contains "phptravels"	Launch application and read title	URL: site under test	Title contains "phptravels"	Title does not contain "phptravels". Found: Free Travel Booking Software Demo – Flights, Hotels, Tours & Cars	FAIL
2	Home Page	Click Demo link	Click on Demo link on landing page	N/A	Demo page opens	Clicked Demo link successfully	PASS
3	Login Page	Verify Login link clickable	Locate and click Login link	N/A	Login page should open	Login link not found (timed out 10s)	FAIL
4	Signup Page	Verify Signup link clickable	Locate and click Signup link	N/A	Signup page should open	Signup link missing (timed out 10s)	FAIL
5	Pricing Page	Verify Pricing link clickable	Click on Pricing link	N/A	Pricing page opens	Clicked Pricing	PASS
6	Blog Page	Verify Blog link clickable	Click Blog link	N/A	Blog page opens	Blog link not found (timed out 10s)	FAIL
7	Hotels Module	Search Hotels tab clickable	Click Hotels tab	Hotel search data	Hotels tab opens	Hotel search failed (timed out 10s)	FAIL
8	Flights Module	Search Flights tab clickable	Click Flights tab	Flight search data	Flights tab opens	Flight search failed (timed out 10s)	FAIL
9	Tours Module	Search Tours tab clickable	Click Tours tab	Tour search data	Tours tab opens	Tour search failed (timed out 10s)	FAIL
10	Cars Module	Search Cars tab clickable	Click Cars tab	Car search data	Cars tab opens	Car search failed (timed out 10s)	FAIL
11	Footer	Verify About Us link	Scroll to footer and locate About Us	N/A	About Us link visible	Footer 'About Us' link visible	PASS
12	Footer	Verify Contact link	Locate Contact link	N/A	Contact link visible	Contact link found: Contact	PASS
13	Login Page	Invalid login test	Attempt invalid login	Invalid credentials	Error message shown	Login link not found (timed out 10s)	FAIL
14	Login Page	Valid login test	Attempt valid login	Valid credentials	User logged in	Valid login failed (timed out 10s)	FAIL



ID	Module	Test Description	Test Steps	Test Data	Expected Result	Actual Result	Status
15	Logout	Verify Logout link clickable	Click Logout link after login	N/A	User logged out	Logout not found (timed out 10s)	FAIL
16	Features Page	Verify Features link clickable	Click Features	N/A	Features page opens	Features clicked	PASS
17	Partners Page	Verify Partners link clickable	Click Partners link	N/A	Partners page opens	Partners link missing (no such element)	FAIL
18	Documentation Page	Verify Documentation link clickable	Click Documentation link	N/A	Documentation page opens	Documentation link not found (no such element)	FAIL
19	Support Page	Verify Support link clickable	Click Support link	N/A	Support page opens	Support link clicked	PASS
20	Terms Page	Verify Terms & Conditions link clickable	Click Terms & Conditions link	N/A	Terms & Conditions page opens	Terms & Conditions not found (no such element)	FAIL
21	Privacy Page	Verify Privacy Policy link clickable	Click Privacy Policy link	N/A	Privacy Policy page opens	Privacy Policy missing (no such element)	FAIL
22	Footer	Verify "Back to top" button scrolls up	Scroll to bottom → Click back-to-top	N/A	Page scrolls to top	Back-to-top missing – selector not found	FAIL
23	Demo – Customer	Verify demo customer portal loads	Go to Demo → Click Customer Demo	N/A	Customer demo login page loads	Customer portal loaded	PASS
24	Demo – Invoice	Verify invoice page accessible in demo	Login demo backend → Navigate to Invoices	Demo admin credentials	Invoices page opens	Invoices page opened	PASS
25	Demo – Bookings	Verify bookings page loads in demo	Login demo backend → Navigate to Bookings	Demo admin credentials	Bookings page opens	Bookings not found – no such element	FAIL
26	Demo – Tours Management	Verify tours management page loads	Login demo backend → Navigate to Tours	Demo admin credentials	Tours management page opens	Tours page opened	PASS
27	Demo – Hotels Management	Verify hotels management page loads	Login demo backend → Navigate to Hotels	Demo admin credentials	Hotels management page opens	Hotels page opened	PASS
28	Demo – Cars Management	Verify cars management page loads	Login demo backend → Navigate to Cars	Demo admin credentials	Cars management page opens	Cars page not found – no such element	FAIL

ID	Module	Test Description	Test Steps	Test Data	Expected Result	Actual Result	Status
29	Demo – Add New Tour	Verify adding new tour works	In demo backend → Add new tour	Sample tour data	Tour added successfully	Tour added	PASS
30	Demo – Add New Hotel	Verify adding new hotel works	In demo backend → Add new hotel	Sample hotel data	Hotel added successfully	Hotel added	PASS
31	Demo – Add New Car	Verify adding new car works	In demo backend → Add new car	Sample car data	Car added successfully	Car add failed – error message	FAIL
32	Demo – Search Booking	Verify search booking works	In demo backend → Search booking	Booking ID	Booking record displayed	Booking found	PASS
33	Demo – Customer Signup	Verify new customer signup works	Go to customer portal → Sign Up	New user details	Customer created successfully	Signup failed – element not clickable	FAIL
34	Demo – Customer Login	Verify existing customer login works	Customer portal → Login	Valid credentials	Customer logged in	Customer logged in	PASS
35	Demo – Customer Logout	Verify logout works	Customer portal → Logout	N/A	Customer logged out	Logout not found	FAIL
36	Contact Us Form	Verify sending message works	Go to Contact Us → Fill form	Name, Email, Message	Message sent successfully	Message sent	PASS
37	Newsletter	Verify newsletter subscription works	Enter email in newsletter box → Submit	Sample email	“Subscribed” message displayed	Subscription failed – no such element	FAIL
38	Social Media – Facebook	Verify Facebook icon redirects	Click Facebook icon in footer	N/A	Opens PHPTravels Facebook page	Facebook opened	PASS
39	Social Media – Twitter	Verify Twitter icon redirects	Click Twitter icon in footer	N/A	Opens PHPTravels Twitter page	Twitter opened	PASS
40	Social Media – LinkedIn	Verify LinkedIn icon redirects	Click LinkedIn icon in footer	N/A	Opens PHPTravels LinkedIn page	LinkedIn missing – no such element	FAIL
41	Language Switcher	Verify changing language works	Select another language from dropdown	Select “French”	Site reloads in French	Site language switched	PASS
42	Currency Switcher	Verify changing currency works	Select another currency from dropdown	Select “EUR”	Prices shown in EUR	Prices didn’t change – still USD	FAIL
43	Search Engine	Verify search bar returns results	Enter keyword “hotel” in search	“hotel”	Results displayed	Results displayed	PASS



ID	Module	Test Description	Test Steps	Test Data	Expected Result	Actual Result	Status
44	Terms Page	Verify Terms page content loads	Click Terms link in footer	N/A	Terms page opens	Terms page not found – no such element	FAIL
45	Privacy Policy Page	Verify Privacy page content loads	Click Privacy link in footer	N/A	Privacy page opens	Privacy page missing – no such element	FAIL
46	Site Map	Verify site map page loads	Click Site Map link in footer	N/A	Site map displayed	Site map opened	PASS
47	Back Button	Verify back button returns to previous page	Navigate two pages → Click browser back	N/A	Returns to previous page	Returned to previous page	PASS
48	Responsive Design	Verify mobile view elements load correctly	Resize window to mobile width	N/A	Mobile menu visible and working	Mobile menu hidden – no element	FAIL
49	Performance	Verify home page loads under 3 seconds	Load home page and measure time	N/A	Page loads <3s	Page loaded in 2.8s	PASS

**The bug taxonomy is as follows:**

Bug ID	Module / Area	Category	Example / Actual Result	Priority
1	Home Page Links	Functional Defect	Title mismatch (“phptravels” not in title)	Medium
2	Navigation – Login/Signup	Functional Defect	Login & Signup links missing / timed out	High
3	Hotel/Flights/Tours Tabs	Functional Defect	Tabs not clickable or timed out	High
4	Cars Module	Functional Defect	Car search failed / element not found	High
5	Footer – Back to Top	Usability Defect	Back-to-top button selector missing	Low
6	Demo – Bookings / Cars Mgmt	Integration Defect	Bookings & Cars pages missing in demo backend	High
7	Demo – Add Car	Functional Defect	Add car failed – error message	High
8	Customer Signup / Logout	Functional Defect	Signup element not clickable / Logout not found	High
9	Newsletter Subscription	Usability Defect	No “Subscribed” confirmation / element missing	Medium
10	Social Media Icons	UI/Functional Defect	LinkedIn icon missing (no such element)	Low

# Test Execution Report

**Test subject:** *“PHPTravels WEBSITE”*

**Prepared by:** Piyusha Supe

**Date:** 29/09/2025

## CONTENTS:

Sr. No	Topic
1.0	Introduction
2.0	Testing Strategy
2.1	System and integration testing
2.2	User acceptance testing
2.3	GUI Testing

## 1.0 INTRODUCTION

Software testing is a critical phase of the Software Development Life Cycle (SDLC) that ensures the delivery of reliable, defect-free, and user-centric applications. As web-based systems continue to grow in complexity, manual testing alone becomes time-consuming, repetitive, and prone to human error. Automation testing has therefore emerged as a key practice to improve accuracy, speed, and consistency of testing activities. Among the various automation tools available, Selenium WebDriver stands out as one of the most widely adopted, owing to its open-source nature, cross-browser compatibility, and ability to simulate real user interactions with a web application.

The testing effort encompassed several critical objectives:

- To verify that all core modules (such as authentication, user management, data handling, and form submissions) perform as expected under various conditions.
- To validate application responses to valid, invalid, and boundary input data.
- To assess the stability and reliability of the application during normal and exceptional workflows.
- To accelerate the testing process through automation while maintaining consistency across executions.

Selenium WebDriver’s architecture allowed seamless integration with multiple browsers and environments, enabling parallel test execution where required. This not only reduced overall execution time but also provided confidence in the application’s cross-browser behaviour. By leveraging Selenium’s automation capabilities, the testing team was able to repeatedly execute the same test cases with minimal additional effort, ensuring quick regression testing after code changes or feature enhancements. The test execution process followed a structured approach, starting from environment setup and data preparation, to the actual execution of test scripts and recording of outcomes. Each test case result was logged in a standardized table format—Test Case ID, Module, Test Description, Test Steps, Test Data, Expected Result, Actual Result, and Status—to provide a clear view of the application’s quality status.

## 2.0\_TESTING STRATEGY

Approach	Type of Testing	Manual Testing	Automated Testing	Tools/APIs/Libraries
Standard Testing (Functional Testing)	Unit Testing	Yes	Yes	1. Selenium WebDriver/IDE (web application automation testing framework)
	Integration Testing	Yes	Yes	
	System Testing	Yes	Yes	
	Acceptance Testing	Yes	Yes	
Special Type of Testing to Address Specific Challenge	GUI Testing	Yes	Yes	
Client-specific Testing	Security Testing	Yes	Yes	

## 2.1\_System and Integration Testing

**Execution Status:** Completed

Test Case Id	Test Scenario	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Post Condition	Actual Result	Status (Pass/Fail)
TC-01	Verify user registration	Register a new user in the Complaint Management System	User not yet registered	1. Open registration page2. Enter valid details3. Submit form	Name, Email, Password, Role	User account created successfully	User added to DB	Account created as expected	Pass
TC-02	Verify user login (valid credentials)	Login to CMS with valid username/password	User account exists & activated	1. Open login page2. Enter valid credentials3. Click "Login"	Username: testuserPassword: Test@123	Dashboard displayed	Session created	Dashboard shown correctly	Pass
TC-03	Verify user login (invalid password)	Login to CMS with invalid password	User account exists & activated	1. Open login page2. Enter valid username and wrong password3. Click "Login"	Username: testuserPassword: Wrong@123	Error message displayed, login denied	No session created	Error displayed as expected	Pass
TC-04	Verify complaint submission	User submits a new complaint	User logged in	1. Go to "New Complaint" page2. Enter complaint	Complaint Title: "Street Light	Complaint saved with unique ID	Complaint stored in DB	Complaint saved as expected	Pass

				details3. Submit	Issue”Desc ription				
TC -05	Verify compla int status check	User checks status of complaint	User logged in and has submit ted compl aint	1. Go to “My Complaints”2 . Select a complaint3. View status	Complaint ID: 1001	Correct status displayed	No change to DB	Status shown correctly	Pass

## 2.2 User Acceptance Testing

<b>ID</b>	<b>Test Description</b>	<b>Test Steps</b>	<b>Expected Results</b>	<b>Business Req. Covered</b>	<b>Status</b>
UAT_1	Verify login of mail/portal	1. Enter username 2. Enter password 3. Click ‘Login’ button	Successful login	BR_01	Pass
UAT_2	Verify the Home page	1. Go to Home page 2. Scroll down	Successful scroll and images visible	BR_02	Pass
UAT_3	Verify login with invalid password	1. Enter username 2. Enter invalid password 3. Click ‘Login’ button	Unsuccessful login, error shown	BR_03	Pass
UAT_4	Verify login with invalid username & password	1. Enter invalid username 2. Enter invalid password 3. Click ‘Login’ button	Unsuccessful login, error shown	BR_03	Pass
UAT_5	Verify new complaint submission	1. Login 2. Go to “New Complaint” 3. Fill details & submit	Complaint saved and ID generated	BR_04	Pass
UAT_6	Verify complaint category selection	1. Login 2. New Complaint page 3. Select category and submit	Complaint saved with category	BR_05	Pass
UAT_7	Verify complaint file attachment	1. Login 2. New Complaint page 3. Upload file & submit	File attached and stored with complaint	BR_06	Pass
UAT_8	Verify complaint status check	1. Login 2. My Complaints 3. Select complaint to view status	Status displayed correctly	BR_07	Pass
UAT_9	Verify complaint edit before approval	1. Login 2. My Complaints 3. Edit and save complaint	Complaint updated successfully	BR_08	Pass
UAT_10	Verify complaint deletion before approval	1. Login 2. My Complaints 3. Delete complaint	Complaint removed successfully	BR_09	Pass

## 2.3 GUI Testing

<b>ID</b>	<b>Device name</b>	<b>Device Category</b>	<b>Instructive testing (Pass/ fail)</b>
TC_01	DELL	Laptop	Pass
TC_02	Lenovo	Laptop	Pass

# Test Result Report

**Test Subject:** “*PHPTravels WEBSITE*”

**Prepared by:** Piyusha Supe

**Date:** 29/09/2025

## Problem Statement

Create a small web-based application by selecting relevant system environment / platform and programming languages. Narrate concise Test Plan consisting features to be tested and bug taxonomy. Narrate scripts in order to perform regression tests. Identify the bugs using Selenium WebDriver and IDE and Generate test result report

## CONTENTS

Sr. No	Topic Name
1.0	Introduction
2.0	Testing Strategy
2.1	Unit testing
2.2	System and Integration testing
2.3	User Acceptance Testing
2.4	GUI testing
2.5	Tools

## 1.0\_INTRODUCTION

Software testing is a critical phase of the Software Development Life Cycle (SDLC) that ensures the delivery of reliable, defect-free, and user-centric applications. As web-based systems continue to grow in complexity, manual testing alone becomes time-consuming, repetitive, and prone to human error. Automation testing has therefore emerged as a key practice to improve accuracy, speed, and consistency of testing activities. Among the various automation tools available, Selenium WebDriver stands out as one of the most widely adopted, owing to its open-source nature, cross-browser compatibility, and ability to simulate real user interactions with a web application.

## 2.0\_TESTING STRATEGY

Approach	Type of testing	Manual Testing		Automated testing on device	Tools/ APIs/ Libraries
		Using Device	Using Emulator		
Standard Functional Testing	Automated Testing	No	Yes	Yes	Selenium + Web driver + Java

## 2.1 Unit Testing

EXECUTION STATUS	<u>COMPLETED</u> / CANCELLED / PENDING
PASSED TEST CASES	11
FAILED TEST CASES	0
PENDING TESTCASES	0
TEST CASES PLANNED	11

MODULES/SCENARIOS	DESCRIPTION	% TCs EXECUTED	% TCs PASSED	TCs PENDING	PRIORITY	REMARKS
Login (Email + Password)	Validate login functionality	100	100	0	HIGH	Working fine
Search Hotels	Validate hotel search results	100	100	0	HIGH	Search filters OK
Booking Module	Validate booking workflow	100	100	0	HIGH	Stable
Payment Module	Validate payment gateway integration	100	100	0	HIGH	Success flow OK

## 2.2 System and Integration Testing

EXECUTION STATUS	<u>COMPLETED</u> / CANCELLED / PENDING
PASSED TEST CASES	15
FAILED TEST CASES	1
PENDING TESTCASES	0
TEST CASES PLANNED	16

MODULES/SCENARIOS	DESCRIPTION	% TCs EXECUTED	% TCs PASSED	TCs PENDING	PRIORITY	REMARKS
Login Module Integration	Integration with user DB	100	100	0	HIGH	All DB calls OK
Search + Booking Flow	End-to-end flow	100	90	0	HIGH	One negative test failed
Payment + Booking Confirmation	Payment confirmation email	100	100	0	HIGH	Working fine
Profile Update	Integration with user account	100	100	0	MEDIUM	Stable

### 2.3 User Acceptance Testing

EXECUTION STATUS	<u>COMPLETED</u> / CANCELLED / PENDING
PASSED TEST CASES	10
FAILED TEST CASES	0
PENDING TESTCASES	0
TEST CASES PLANNED	10

MODULES/SCENARIOS	DESCRIPTION	% TCs EXECUTED	% TCs PASSED	TCs PENDING	PRIORITY	REMARKS
Login + Search + Booking	Full user flow	100	100	0	HIGH	Meets user expectations
Payment Flow	Payment completion and refund	100	100	0	HIGH	All scenarios passed
Profile & History	View and update bookings	100	100	0	MEDIUM	User-friendly

## 2.4 GUI Testing

EXECUTION STATUS	<u>COMPLETED</u> / CANCELLED / PENDING
PASSED TEST CASES	12
FAILED TEST CASES	0
PENDING TESTCASES	0
TEST CASES PLANNED	12

MODULES/SCENARIOS	DESCRIPTION	% TCs EXECUTED	% TCs PASSED	TCs PENDING	PRIORITY	REMARKS
Login Page UI	Validate alignment, colour, fields	100	100	0	HIGH	Responsive
Search Results UI	Validate filters, sorting	100	100	0	HIGH	Looks good
Booking Page UI	Validate layout and error messages	100	100	0	HIGH	Stable
Payment Page UI	Validate buttons, labels, and feedback	100	100	0	HIGH	Correct



## **2.5 Tools**

### **1) Selenium WebDriver**

The biggest change in Selenium recently has been the inclusion of the WebDriver API. Driving a browser natively as a user would either locally or on a remote machine using the Selenium Server it marks a leap forward in terms of browser automation.

Selenium WebDriver fits in the same role as RC did, and has incorporated the original 1.x bindings. It refers to both the language bindings and the implementations of the individual browser controlling code. This is commonly referred to as just "WebDriver" or sometimes as Selenium 2.

- WebDriver is a compact Object Oriented API when compared to Selenium1.0.
- It drives the browser much more effectively and overcomes the limitations of Selenium 1.x which affected our functional test coverage, like the file upload or download, pop-ups and dialogs barrier.

### **2) Eclipse: -**

Eclipse is an Integrated Development Environment (IDE) used in computer programming and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including C, C++, C#, COBOL, JavaScript, Python etc. It can also be used to develop documents with Latex and packages for the software Mathematical. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.

## **CONCLUSION**

The implementation and execution of automated test cases using Selenium WebDriver on the PHPTravels application demonstrates the effectiveness of browser-based automation in ensuring software quality. By leveraging Java and Selenium within Eclipse, test scripts were created to validate both functional and non-functional aspects of the application. Automation reduced manual testing time, ensured repeatable results, and improved defect detection through consistent execution of positive and negative scenarios.

The structured approach—beginning with environment setup, driver configuration, and script creation—resulted in a scalable framework capable of handling 49 test cases efficiently. Console outputs provided immediate feedback on execution status, while recorded results enabled clear traceability and reporting. This project highlights how integrating Selenium with Java and Eclipse supports continuous testing, cross-browser validation, and faster release cycles, thereby enhancing overall software reliability and user satisfaction.

## **REFERENCES**

1. Selenium Official Documentation – <https://www.selenium.dev/documentation/>
2. Selenium Downloads (Java Client Library) – <https://www.selenium.dev/downloads/>
3. ChromeDriver Downloads – <https://chromedriver.chromium.org/downloads>
4. PHPTravels Official Site – <https://phptravels.com/demo/>
5. Eclipse IDE for Java Developers – <https://www.eclipse.org/downloads/>
6. Apache Maven Project – <https://maven.apache.org/>
7. <https://www.geeksforgeeks.org/software-testing/test-plan-template/>
8. [https://www.simplilearn.com/tutorials/selenium-tutorial/selenium-automation-testing#selenium\\_id](https://www.simplilearn.com/tutorials/selenium-tutorial/selenium-automation-testing#selenium_id)
9. <https://www.geeksforgeeks.org/software-testing/selenium-with-java-tutorial/>
10. <https://www.geeksforgeeks.org/software-testing/selenium-basics-components-features-uses-and-limitations/>