

/\* 11. Scenario: Login Page Automation

You have a web application with a login page. Can you demonstrate how you would use Selenium

to automate the login process, including entering valid credentials and handling a successful login? \*/

package as;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class P11 {

public static void main(String[] args) {

// Initialize WebDriver for Chrome browser

WebDriver driver = new ChromeDriver();

// Open the login page

driver.get("http://lms.nmit.ac.in/moodle/");

// Navigate to the login page

driver.findElement(By.*linkText*("Log in")).click();

// Enter the username and password

driver.findElement(By.*name*("username")).sendKeys("your\_lms\_userid");

driver.findElement(By.*name*("password")).sendKeys("your\_lms\_password");

// Submit the login form

driver.findElement(By.*id*("loginbtn")).submit();

// Close the browser after login

driver.quit();

}

}

/\* 12. Scenario: Negative Testing for Login

Extend the previous scenario to include negative testing. How would you use Selenium to automate

the login page to handle scenarios such as entering invalid credentials and verifying appropriate

error messages? \*/

package as;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.WebElement;

public class P12 {

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

// Navigate to the Facebook login page

driver.get("https://www.facebook.com/");

// Enter invalid credentials

driver.findElement(By.*name*("email")).sendKeys("invalid\_email@example.com");

driver.findElement(By.*name*("pass")).sendKeys("incorrect\_password");

// Submit the login form

driver.findElement(By.*name*("login")).click();

// Wait for the error message to appear (this could be replaced by explicit waits for better synchronization)

WebElement errorMessage = driver.findElement(By.*xpath*("//div[contains(text(),'The email or mobile number you entered isn’t connected to an account.')]"));

// Verify the error message appears

if (errorMessage.isDisplayed()) {

System.***out***.println("Error message displayed: " + errorMessage.getText());

} else {

System.***out***.println("Error message not displayed.");

}

// Close the browser after the test

driver.quit();

}

}

/\* 13. Scenario: Navigation Testing

Imagine a scenario where you need to navigate through different pages of a web application using

Selenium. How would you automate the navigation flow, including clicking on links or buttons to

move between pages? \*/

package as;

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 P13 {

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

// Open Google and perform a search

driver.get("https://www.google.com/");

// Find the search box and input a query

WebElement searchBox = driver.findElement(By.*name*("q"));

searchBox.sendKeys("NMIT");

// Wait until the search button is clickable, then click it

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

WebElement searchButton = wait.until(ExpectedConditions.*elementToBeClickable*(By.*name*("btnK")));

searchButton.click();

// Wait until a result link appears, click the first result (for demonstration purposes)

WebElement firstResult = wait.until(ExpectedConditions.*elementToBeClickable*(By.*xpath*("//h3")));

firstResult.click();

// Wait for the page to load, and navigate back

wait.until(ExpectedConditions.*titleContains*("NMIT"));

driver.navigate().back();

// Wait again for the Google home page to reload and click the search button again

wait.until(ExpectedConditions.*titleContains*("Google"));

WebElement searchBox2 = driver.findElement(By.*name*("q"));

searchBox2.sendKeys("Selenium WebDriver");

searchButton.click();

// Finally, quit the browser

driver.quit();

}

}

/\*14.Scenario: Form Filling Automation

The application has a multi-step form. Demonstrate how you would automate the process of filling

out the form using Selenium, including handling various form elements such as text fields,

checkboxes, and radio buttons. \*/

package as; // Only one package declaration at the top

// Import required Selenium libraries

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.Select;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.openqa.selenium.support.ui.ExpectedConditions; // Add this import

import java.time.Duration;

public class P14 {

public static void main(String[] args) {

// Initialize the WebDriver (Chrome)

WebDriver driver = new ChromeDriver();

try {

// Navigate to the form page

driver.get("https://www.jotform.com/build/242983020566458?s=templates2");

// Fill in text fields

driver.findElement(By.*id*("first\_4")).sendKeys("Dhananjay");

driver.findElement(By.*id*("middle\_4")).sendKeys("N");

driver.findElement(By.*id*("last\_4")).sendKeys("R");

// Select checkboxes

driver.findElement(By.*id*("label\_input\_10\_0")).click();

driver.findElement(By.*id*("label\_input\_10\_1")).click();

driver.findElement(By.*id*("label\_input\_10\_2")).click();

driver.findElement(By.*id*("label\_input\_10\_3")).click();

driver.findElement(By.*id*("label\_input\_11\_0")).click();

// Wait for the page to update and click on the next checkboxes

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

wait.until(ExpectedConditions.*elementToBeClickable*(By.*id*("label\_input\_11\_1"))).click();

wait.until(ExpectedConditions.*elementToBeClickable*(By.*id*("label\_input\_11\_2"))).click();

// Selecting values from drop-down menus (radio buttons or select options)

WebElement genderSelect = driver.findElement(By.*id*("input\_3"));

Select gender = new Select(genderSelect);

gender.selectByValue("Male"); // Assuming "Male" is a valid value

// Fill in additional fields

driver.findElement(By.*id*("input\_6")).sendKeys("dhananjaynr10@gmail.com");

driver.findElement(By.*id*("input\_5")).sendKeys("27626");

// Select a course from the drop-down menu

WebElement courseSelect = driver.findElement(By.*id*("input\_7"));

Select course = new Select(courseSelect);

course.selectByValue("Math 101"); // Assuming "Math 101" is a valid value

// Submit the form

driver.findElement(By.*id*("input\_9")).submit();

} catch (Exception e) {

e.printStackTrace();

} finally {

// Close the browser at the end of the test

driver.quit();

}

}

}

package as;

/\* 17. Scenario: File Upload Automation

The application requires users to upload a file. How would you automate the file upload process

using Selenium? Provide steps to locate the file input element, interact with it, and verify the

successful file upload. \*/

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;

public class P17 {

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

try {

// Step 1: Navigate to the file upload page

driver.get("https://the-internet.herokuapp.com/upload");

// Step 2: Maximize the browser window

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

// Step 3: Set an implicit wait for elements to be located

driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(10));

// Step 4: Locate the file input element and upload the file

WebElement fileInput = driver.findElement(By.*id*("file-upload"));

// Use the file path you provided

fileInput.sendKeys("C:\\Users\\yash\\Downloads\\txt.txt");

// Step 5: Locate and click the "Upload" button

WebElement uploadButton = driver.findElement(By.*id*("file-submit"));

uploadButton.click();

// Step 6: Verify if the file upload was successful

WebElement successMessage = driver.findElement(By.*id*("uploaded-files"));

if (successMessage.getText().equals("txt.txt")) {

System.***out***.println("File upload successful and verified.");

} else {

System.***out***.println("File upload verification failed.");

}

} catch (Exception e) {

// Handle any exceptions

System.***out***.println("An error occurred: " + e.getMessage());

} finally {

// Step 7: Close the browser after the test

driver.quit();

}

}

}

package as;

/\* 19. Scenario: Handling Alerts and Pop-ups

The application displays an alert or pop-up during certain interactions.

How would you handle alerts and pop-ups using Selenium?

Provide an example scenario and demonstrate your approach. \*/

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.Alert;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

public class P19 {

public static void main(String[] args) {

WebDriver driver = new ChromeDriver();

try {

// Open the JavaScript Alerts page

driver.get("https://the-internet.herokuapp.com/javascript\_alerts");

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

driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(10));

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

// Handle Simple Alert

WebElement alertButton = driver.findElement(By.*xpath*("//button[text()='Click for JS Alert']"));

alertButton.click();

// Explicit wait for alert to be present and accept it

Alert alert = wait.until(ExpectedConditions.*alertIsPresent*());

alert.accept();

// Verify the result after handling the alert

WebElement resultText = driver.findElement(By.*id*("result"));

if (resultText.getText().equals("you successfully clicked an alert")) {

System.***out***.println("Simple alert handled successfully");

} else {

System.***out***.println("Simple alert handling failed");

}

// Handle Confirm Alert (Dismiss)

WebElement confirmButton = driver.findElement(By.*xpath*("//button[text()='Click for JS Confirm']"));

confirmButton.click();

// Explicit wait for confirm alert to be present and dismiss it

alert = wait.until(ExpectedConditions.*alertIsPresent*());

alert.dismiss();

// Verify the result after handling the confirm alert

WebElement resultText1 = driver.findElement(By.*id*("result"));

if (resultText1.getText().equals("you clicked: cancel")) {

System.***out***.println("Confirmation alert handled successfully");

} else {

System.***out***.println("Confirmation alert handling failed");

}

// Handle Prompt Alert (Send text and accept)

WebElement promptButton = driver.findElement(By.*xpath*("//button[text()='Click for JS Prompt']"));

promptButton.click();

// Explicit wait for prompt alert to be present, send text, and accept the alert

alert = wait.until(ExpectedConditions.*alertIsPresent*());

alert.sendKeys("Hello, Selenium!");

alert.accept();

// Verify the result after handling the prompt alert

WebElement resultText2 = driver.findElement(By.*id*("result"));

if (resultText2.getText().equals("you entered: Hello, Selenium!")) {

System.***out***.println("Prompt alert handled successfully");

} else {

System.***out***.println("Prompt alert handling failed");

}

} catch (Exception e) {

// Handle any exceptions that occur during the test execution

e.printStackTrace();

} finally {

// Quit the browser at the end of the test

driver.quit();

}

}

}