**Comprehensive Assignment**

**Answer1:**

class Account {

protected double interest;

public void calculateInterest() {

}

}

class SavingsAccount extends Account {

@Override

public void calculateInterest() {

}

}

class CurrentAccount extends Account {

@Override

public void calculateInterest() {

}

}

**Answer 2:**

class Student:

def \_\_init\_\_(self, name, grade, age):

self.name = name

self.grade = grade

self.age = age

def display(self):

print("Name:", self.name)

print("Grade:", self.grade)

print("Age:", self.age)

class School(Student):

def schoolStudentDisplay(self):

self.display()

school\_student = School("John", "10th", 15)

school\_student.schoolStudentDisplay()

**Answer 3:**

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

driver = webdriver.Firefox()

driver.get("https://www.makemytrip.com/")

wait = WebDriverWait(driver, 10)

logo\_element = wait.until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "a.logo img")))

if logo\_element.is\_displayed():

print("MakeMyTrip logo is present on the page.")

else:

print("MakeMyTrip logo is not present on the page.")

driver.quit()

**Answer 4:**

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

def launch\_browser():

driver = webdriver.Chrome()

return driver

def click\_element(driver, xpath):

element = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, xpath)))

element.click()

def enter\_text(driver, xpath, text):

element = WebDriverWait(driver, 10).until(EC.visibility\_of\_element\_located((By.XPATH, xpath)))

element.clear()

element.send\_keys(text)

driver = launch\_browser()

driver.get("https://www.makemytrip.com/")

click\_element(driver, "//a[@data-cy='menu\_Flights']")

click\_element(driver, "//li[@data-cy='oneWayTrip']")

enter\_text(driver, "//input[@id='fromCity']", "New York")

enter\_text(driver, "//input[@id='toCity']", "London")

driver.quit()

**Answer 5:**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class MakemytripLogoTest {

private WebDriver driver;

@BeforeMethod

public void setUp() {

System.setProperty("webdriver.gecko.driver", "path/to/geckodriver");

driver = new FirefoxDriver();

}

@Test

public void verifyMakemytripLogo() {

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

boolean logoPresent = driver.findElement(By.xpath("//img[@alt='MakeMyTrip']")).isDisplayed();

Assert.assertTrue(logoPresent, "Makemytrip logo is not present on the page");

}

}

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class FlightBookingTest {

private WebDriver driver;

@BeforeMethod

public void setUp() {

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

driver = new ChromeDriver();

}

@Test

public void bookOneWayFlight() {

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

driver.findElement(By.xpath("//span[text()='Flights']")).click();

driver.findElement(By.xpath("//li[text()='OneWay']")).click();

driver.findElement(By.id("fromCity")).sendKeys("FROM\_LOCATION");

driver.findElement(By.id("toCity")).sendKeys("TO\_LOCATION");

}

}

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

<suite name="Test Suite">

<test name="Test">

<classes>

<class name="com.example.MakemytripLogoTest"/>

<class name="com.example.FlightBookingTest"/>

</classes>

</test>

</suite>

mvn test

**Answer 6:**

**1.Create Maven Project:**

mvn archetype:generate -DgroupId=com.example -DartifactId=my-project -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

**2.Navigate:**

cd my-project

**3.Add dependencies:**

<dependencies>

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

</dependencies>

<dependencies>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.4.0</version>

</dependency>

</dependencies>

4.Create TestClasses:

MakeMyTripLogoVerificationTest

5.Run Tests using Maven:

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class LogoVerificationTest {

@Test

public void verifyMakemytripLogo() {

// Set the path to the geckodriver executable

System.setProperty("webdriver.gecko.driver", "path/to/geckodriver");

// Launch Firefox browser

WebDriver driver = new FirefoxDriver();

// Navigate to the page where the logo needs to be verified

driver.get("https://example.com");

// Check if the makemytrip logo is present on the page

boolean isLogoPresent = driver.findElement(By.id("makemytrip-logo")).isDisplayed();

// Close the browser

driver.quit();

// Assert the logo presence

org.junit.Assert.assertTrue("Makemytrip logo is not present on the page", isLogoPresent);

}

}

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class FlightBookingTest {

@Test

public void bookFlight() {

// Set the path to the chromedriver executable

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

// Launch Chrome browser

WebDriver driver = new ChromeDriver();

// Navigate to the MakeMyTrip website

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

// Click on the 'Flights' option

driver.findElement(By.xpath("//span[text()='Flights']")).click();

// Click on the 'OneWay' option

driver.findElement(By.xpath("//li[text()='One Way']")).click();

// Enter the 'FROM' location

driver.findElement(By.xpath("//input[@id='fromCity']"))

.sendKeys("Your FROM location");

// Enter the 'TO' location

driver.findElement(By.xpath("//input[@id='toCity']"))

.sendKeys("Your TO location");

// Close the browser

driver.quit();

}

}

mvn test

**Answer 7:**

**Answer8:**

// Positive Test Cases

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

pm.test("Field 'name' exists in response", function () {

pm.expect(pm.response.json()).to.have.property('name');

});

pm.test("Field 'web\_pages' is not empty", function () {

pm.expect(pm.response.json().web\_pages).to.not.be.empty;

});

pm.test("Field 'domains' is not empty", function () {

pm.expect(pm.response.json().domains).to.not.be.empty;

});

pm.test("Field 'country' is 'United States'", function () {

pm.expect(pm.response.json().country).to.eql('United States');

});

// Negative Test Cases

pm.test("Status code is not 404", function () {

pm.response.to.not.have.status(404);

});

pm.test("Field 'name' does not contain unexpected values", function () {

pm.expect(pm.response.json().name).to.not.be.oneOf([null, undefined, '']);

});

pm.test("Field 'web\_pages' does not contain unexpected URLs", function () {

pm.expect(pm.response.json().web\_pages).to.not.include.oneOf([null, undefined, '']);

});

pm.test("Field 'domains' does not contain unexpected domain names", function () {

pm.expect(pm.response.json().domains).to.not.include.oneOf([null, undefined, '']);

});

pm.test("Field 'country' is not empty or null", function () {

pm.expect(pm.response.json().country).to.not.be.oneOf([null, undefined, '']);

});

To validate the HTTP Status Codes and responses use Assertions concept in PostMan

<http://universities.hipolabs.com/search?country=United+States>

// Positive Test Cases

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

pm.test("Response body is not empty", function () {

pm.expect(pm.response.json()).to.not.be.empty;

});

pm.test("Response is in English", function () {

pm.expect(pm.response.headers.get("Content-Language")).to.eql("en");

});

// Negative Test Cases

pm.test("Status code is not 404", function () {

pm.response.to.not.have.status(404);

});

pm.test("Response body is empty or null", function () {

pm.expect(pm.response.json()).to.be.oneOf([null, undefined, []]);

});

pm.test("Response is not in a language other than English", function () {

pm.expect(pm.response.headers.get("Content-Language")).to.eql("en");

});

**Answer 9:**

To create a Thread Group in JMeter and use assertions to validate responses, follow these steps:

1. Open JMeter and create a new Test Plan.
2. Right-click on the Test Plan and select "Add" > "Threads (Users)" > "Thread Group".
3. In the Thread Group panel, you can configure the number of threads (users), ramp-up period, and loop count according to your requirements. For example, set the number of threads to 10, ramp-up period to 5 seconds, and loop count to 1.
4. Right-click on the Thread Group and select "Add" > "Sampler" > "HTTP Request".
5. In the HTTP Request panel, enter the URL "<https://www.makemytrip.com/>" in the "Server Name or IP" field.
6. Right-click on the Thread Group and select "Add" > "Listener" > "Assertion Results".
7. Run the test plan by clicking the green play button or pressing Ctrl+R.
8. After the test run, you can view the Assertion Results in the Assertion Results listener. It will show the success or failure of each assertion.

To add assertions to validate the responses:

1. Right-click on the HTTP Request sampler and select "Add" > "Assertions" > "Response Assertion".
2. In the Response Assertion panel, you can configure various options to validate the response. For example, you can choose to validate the response code, response message, or specific text in the response.
3. Run the test plan again, and the assertions will be applied to validate the responses.

**Answer 10:**

import pytest

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.common.by import By

@pytest.fixture

def browser():

# Set up Chrome WebDriver

webdriver\_service = Service('/path/to/chromedriver') # Replace with the actual path to chromedriver

driver = webdriver.Chrome(service=webdriver\_service)

yield driver

# Teardown - quit the browser after the test

driver.quit()

def test\_w3schools\_logo(browser):

# Launch the browser and navigate to W3Schools

browser.get('https://www.w3schools.com')

# Verify the presence of the W3Schools logo

logo\_element = browser.find\_element(By.CSS\_SELECTOR, 'a.w3schools-logo')

assert logo\_element.is\_displayed(), "W3Schools logo is not present on the page"

pytest test\_w3schools\_logo.py