**import** java.time.Duration;

**import** java.util.Set;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.Keys;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**public** **class** assessment {

**public** **static** **void** main(String[] args) {

WebDriver driver = **new** ChromeDriver();

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

driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(20));

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

driver.findElement(By.*xpath*("//input[contains(@placeholder,'Search for anything')]")).sendKeys("book",

Keys.***ENTER***);

driver.findElement(By.*xpath*("(//div[contains(@class,'item\_\_wrapper clearfix')])[3]")).click();

Set<String> allWindowsIds = driver.getWindowHandles();

**for** (String id : allWindowsIds) {

driver.switchTo().window(id);

**if** (driver.getCurrentUrl().contains("skw=book&itmmeta"))

**break**;

}

**try** {

driver.findElement(By.*xpath*("//a[contains(@id,'atcBtn\_btn')]")).click();

} **catch** (Exception e) {

System.***out***.println("Item cannot be added to addToCart");

}

driver.findElement(By.*xpath*("//button[contains(@aria-label,'Close overlay')]")).click();

WebElement element1 = driver

.findElement(By.*xpath*("(//span[contains(@aria-label,'Your shopping cart contains')])[2]"));

System.***out***.println(element1.getText());

}

}

**API Automation**

import io.restassured.RestAssured;

import io.restassured.http.ContentType;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

import static io.restassured.RestAssured.given;

import static org.hamcrest.Matchers.\*; // Hamcrest matchers for assertions

public class BitcoinApiTest {

private static final String BASE\_URI = “https://api.coingecko.com";

private static final String BITCOIN\_ENDPOINT = "/api/v3/coins/bitcoin";

@BeforeClass

public void setup() {

RestAssured.baseURI = BASE\_URI;

}

@Test(description = "Automate GET request and verify Bitcoin API response details")

public void testBitcoinDetailsApi() {

given()

.when()

.get(BITCOIN\_ENDPOINT)

.then()

.statusCode(200)

.contentType(ContentType.JSON)

.log().all()

// a. Verify the response contains 3 BPIs (USD, GBP, EUR)

.body("bpi.USD", notNullValue()) // Assert that USD BPI exists

.body("bpi.GBP", notNullValue()) // Assert that GBP BPI exists

.body("bpi.EUR", notNullValue()) // Assert that EUR BPI exists

// b. Verify each cryptocurrency (USD, GBP, EUR) has a market\_cap and total\_volume

// For USD

.body("bpi.USD.market\_cap", notNullValue()) // Assert USD market\_cap exists

.body("bpi.USD.market\_cap", isA(Number.class)) // Assert USD market\_cap is a number

.body("bpi.USD.total\_volume", notNullValue()) // Assert USD total\_volume exists

.body("bpi.USD.total\_volume", isA(Number.class)) // Assert USD total\_volume is a number

// For GBP

.body("bpi.GBP.market\_cap", notNullValue()) // Assert GBP market\_cap exists

.body("bpi.GBP.market\_cap", isA(Number.class)) // Assert GBP market\_cap is a number

.body("bpi.GBP.total\_volume", notNullValue()) // Assert GBP total\_volume exists

.body("bpi.GBP.total\_volume", isA(Number.class)) // Assert GBP total\_volume is a number

// For EUR

.body("bpi.EUR.market\_cap", notNullValue()) // Assert EUR market\_cap exists

.body("bpi.EUR.market\_cap", isA(Number.class)) // Assert EUR market\_cap is a number

.body("bpi.EUR.total\_volume", notNullValue()) // Assert EUR total\_volume exists

.body("bpi.EUR.total\_volume", isA(Number.class)) // Assert EUR total\_volume is a number

// c. Verify the price change percentage over the last 24 hours

.body("market\_data.price\_change\_percentage\_24h", notNullValue()) // Assert price\_change\_percentage\_24h exists

.body("market\_data.price\_change\_percentage\_24h", isA(Number.class)) // Assert it's a number

// d. Verify homepage URL is not empty

// The homepage is an array, we check the first element

.body("links.homepage[0]", notNullValue()) // Assert the first homepage URL exists

.body("links.homepage[0]", not(emptyOrNullString())); // Assert the first homepage URL is not empty or null

}

}