National University of Computer and Emerging Sciences Chiniot-Faisalabad Campus



**metabase**

Software Quality Engineering

### Semester Project

### Phase 2

### BS (SE) 5A

## Team Name: zzwave-testing

## Team Member’s:

## 21F-9516 -> Suresh Kumar

## 21F-9519 -> Rai umer farooq

## 21f-9510 -> mian fahad akhtar

GUI Automated Test Cases in java

# XML

<dependencies>

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>3.141.59</version> <!-- Use the latest version available -->

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.1</version> <!-- Use the latest version available -->

</dependency>

</dependencies>

**Login Validation:**

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import static org.junit.Assert.assertTrue;

public class LoginValidationTest {

private WebDriver driver;

@Before

public void setUp() {

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

driver = new ChromeDriver();

}

@Test

public void testLoginScreenDisplayed() {

// Navigate to the login page

driver.get("your\_login\_page\_url");

WebElement usernameField = driver.findElement(By.id("username")); // Replace with the actual ID of the username field

WebElement passwordField = driver.findElement(By.id("password")); // Replace with the actual ID of the password field

assertTrue(usernameField.isDisplayed());

assertTrue(passwordField.isDisplayed());

}

@After

public void tearDown() {

// Close the browser

if (driver != null) {

driver.quit();

}

}

}°

**Successful Login:**

**Invalid Login:**

**Logout Functionality:**

**Action List Display:**

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import static org.junit.Assert.assertEquals;

import static org.junit.Assert.assertTrue;

public class LoginTests {

private WebDriver driver;

@Before

public void setUp() {

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

driver = new ChromeDriver();

}

@Test

public void testSuccessfulLogin() {

driver.get("your\_login\_page\_url");

WebElement usernameField = driver.findElement(By.id("username"));

WebElement passwordField = driver.findElement(By.id("password"));

WebElement loginButton = driver.findElement(By.id("loginButton"));

usernameField.sendKeys("valid\_username");

passwordField.sendKeys("valid\_password");

loginButton.click();

String expectedDashboardUrl = "expected\_dashboard\_url";

assertEquals(expectedDashboardUrl, driver.getCurrentUrl());

}

@Test

public void testInvalidLogin() {

driver.get("your\_login\_page\_url");

WebElement usernameField = driver.findElement(By.id("username"));

WebElement passwordField = driver.findElement(By.id("password"));

WebElement loginButton = driver.findElement(By.id("loginButton"));

usernameField.sendKeys("invalid\_username");

passwordField.sendKeys("invalid\_password");

loginButton.click();

WebElement errorMessage = driver.findElement(By.id("errorMessage"));

assertTrue(errorMessage.isDisplayed());

}

@Test

public void testLogoutFunctionality() {

driver.get("your\_dashboard\_url");

WebElement logoutButton = driver.findElement(By.id("logoutButton"));

logoutButton.click();

String expectedLoginUrl = "expected\_login\_url";

assertEquals(expectedLoginUrl, driver.getCurrentUrl());

}

@Test

public void testActionListDisplay() {

driver.get("your\_dashboard\_url");

WebElement actionList = driver.findElement(By.id("actionList"));

assertTrue(actionList.isDisplayed());

}

@After

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

}

API Automated Test Cases in java

**Authentication Success:**

**Authentication Failure:**

**Get Action List:**

**Execute Action:**

**Execute Critical Action with Confirmation:**

import io.restassured.RestAssured;

import io.restassured.response.Response;

import org.junit.Test;

import static io.restassured.RestAssured.given;

import static org.junit.Assert.assertEquals;

public class ApiTests {

private static final String BASE\_URL = "your\_base\_url";

@Test

public void testAuthenticationSuccess() {

RestAssured.baseURI = BASE\_URL;

Response response = given()

.param("username", "valid\_username")

.param("password", "valid\_password")

.when()

.post("/authenticate");

assertEquals(200, response.getStatusCode());

}

@Test

public void testAuthenticationFailure() {

RestAssured.baseURI = BASE\_URL;

Response response = given()

.param("username", "invalid\_username")

.param("password", "invalid\_password")

.when()

.post("/authenticate");

assertEquals(401, response.getStatusCode());

}

@Test

public void testGetActionList() {

RestAssured.baseURI = BASE\_URL;

Response response = given()

.header("Authorization", "valid\_token")

.when()

.get("/actions");

assertEquals(200, response.getStatusCode());

}

@Test

public void testExecuteAction() {

RestAssured.baseURI = BASE\_URL;

Response response = given()

.header("Authorization", "valid\_token")

.param("actionId", "valid\_action\_id")

.when()

.post("/executeAction");

assertEquals(200, response.getStatusCode());

}

@Test

public void testExecuteCriticalActionWithConfirmation() {

RestAssured.baseURI = BASE\_URL;

Response response = given()

.header("Authorization", "valid\_token")

.param("actionId", "critical\_action\_id")

.param("confirmation", true)

.when()

.post("/executeCriticalAction");

assertEquals(200, response.getStatusCode());

}

}