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

# **Test Cases For GUI: -**

## Test Case 1: Dashboard Loading

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import org.junit.Assert;

public class DashboardLoadingTest {

WebDriver driver;

@Before

public void setup() {

System.setProperty("webdriver.chrome.driver", "path\_to\_chromedriver");

driver = new ChromeDriver();

}

@Test

public void testDashboardLoading() {

login("valid\_username", "valid\_password");

driver.get("login\_page\_url");

navigateToDashboard();

boolean isDashboardLoaded = !driver.getPageSource().contains("Error message");

WebElement dashboardElement = driver.findElement(By.id("https://www.metabase.com/docs/latest/dashboards/start"));

boolean isDashboardVisible = dashboardElement.isDisplayed();

boolean expectedDashboardLoad = true;

boolean expectedDashboardInteraction = true;

boolean actualDashboardLoad = isDashboardLoaded;

boolean actualDashboardInteraction = isDashboardVisible;

Assert.assertEquals("Dashboard loaded without errors", expectedDashboardLoad, actualDashboardLoad);

Assert.assertEquals("User can view and interact with the dashboard", expectedDashboardInteraction, actualDashboardInteraction);

}

@After

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

private void login(String username, String password) {

driver.get("https://www.metabase.com/docs/latest/dashboards/start");

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

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

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

usernameField.sendKeys(username);

passwordField.sendKeys(password);

loginButton.click();

}

private void navigateToDashboard() {

WebElement dashboardLink = driver.findElement(By.id("https://www.metabase.com/docs/latest/dashboards/start"));

dashboardLink.click();

}

}

## Test Case 2: Error Message on Invalid Credentials

package gui;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import org.junit.Assert;

public class InvalidCredentialsErrorMessageTest {

WebDriver driver;

@Before

public void setup() {

System.setProperty("webdriver.chrome.driver", "path\_to\_chromedriver");

driver = new ChromeDriver();

}

@Test

public void testErrorMessageOnInvalidCredentials() {

driver.get("login\_page\_url");

login("invalid\_username", "invalid\_password");

WebElement errorMessage = driver.findElement(By.className("error-message"));

boolean isErrorDisplayed = errorMessage.isDisplayed();

boolean isOnLoginPage = driver.getTitle().equals("Login Page");

boolean expectedErrorMessageDisplayed = true;

boolean expectedOnLoginPage = true;

boolean actualErrorMessageDisplayed = isErrorDisplayed;

boolean actualOnLoginPage = isOnLoginPage;

Assert.assertEquals("Error message displayed for invalid credentials", expectedErrorMessageDisplayed, actualErrorMessageDisplayed);

Assert.assertEquals("User remains on the login page", expectedOnLoginPage, actualOnLoginPage);

}

@After

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

private void login(String username, String password) {

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

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

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

usernameField.sendKeys(username);

passwordField.sendKeys(password);

loginButton.click();

}

}

## Test Case 3: Responsive Design Test (Desktop)

package gui;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import org.junit.Assert;

public class ResponsiveDesignTest {

WebDriver driver;

@Before

public void setup() {

System.setProperty("webdriver.chrome.driver", "path\_to\_chromedriver");

driver = new ChromeDriver();

}

@Test

public void testResponsiveDesignOnDesktop() {

driver.get("dashboard\_url");

WebElement dashboardElement = driver.findElement(By.id("dashboard"));

boolean isDashboardVisible = dashboardElement.isDisplayed();

WebElement dashboardInteractionElement = driver.findElement(By.className("dashboard-interaction"));

boolean canInteractWithDashboard = dashboardInteractionElement.isEnabled();

boolean expectedDashboardVisibility = true;

boolean expectedResponsiveDesign = true;

boolean expectedInteractWithDashboard = true;

boolean actualDashboardVisibility = isDashboardVisible;

boolean actualResponsiveDesign = /\* Logic to verify responsiveness \*/;

boolean actualInteractWithDashboard = canInteractWithDashboard;

Assert.assertEquals("Dashboard is visible", expectedDashboardVisibility, actualDashboardVisibility);

Assert.assertEquals("Dashboard is user-friendly and responsive", expectedResponsiveDesign, actualResponsiveDesign);

Assert.assertEquals("User can interact with the dashboard", expectedInteractWithDashboard, actualInteractWithDashboard);

}

@After

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

}

## Test Case 4: Performance Test with Minimal Widgets

package gui;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import org.junit.Assert;

public class MinimalWidgetsPerformanceTest {

WebDriver driver;

@Before

public void setup() {

System.setProperty("webdriver.chrome.driver", "path\_to\_chromedriver");

driver = new ChromeDriver();

}

@Test

public void testMinimalWidgetsPerformance() {

driver.get("dashboard\_url");

boolean isEfficientPerformance;/\* Logic to verify performance \*/;

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

boolean canInteractWithWidget = singleWidget.isEnabled();

boolean expectedEfficientPerformance = true;

boolean expectedInteractWithWidget = true;

boolean actualEfficientPerformance = isEfficientPerformance;

boolean actualInteractWithWidget = canInteractWithWidget;

Assert.assertEquals("Dashboard functions efficiently", expectedEfficientPerformance, actualEfficientPerformance);

Assert.assertEquals("User can interact with the single widget", expectedInteractWithWidget, actualInteractWithWidget);

}

@After

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

}

## Test Case 5: Performance Test with Maximum Widgets

package gui;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import org.junit.Assert;

public class MaximumWidgetsPerformanceTest {

WebDriver driver;

@Before

public void setup() {

System.setProperty("webdriver.chrome.driver", "path\_to\_chromedriver");

driver = new ChromeDriver();

}

@Test

public void testMaximumWidgetsPerformance() {

driver.get("dashboard\_url");

boolean isResponsiveAndUserFriendly; /\* Logic to verify responsiveness and user-friendliness \*/;

boolean canInteractWithAllWidgets; /\* Logic to interact with each widget and verify interaction \*/;

boolean expectedResponsiveAndUserFriendly = true;

boolean expectedInteractWithAllWidgets = true;

boolean actualResponsiveAndUserFriendly = isResponsiveAndUserFriendly;

boolean actualInteractWithAllWidgets = canInteractWithAllWidgets;

Assert.assertEquals("Dashboard remains responsive and user-friendly", expectedResponsiveAndUserFriendly, actualResponsiveAndUserFriendly);

Assert.assertEquals("User can interact with all the widgets", expectedInteractWithAllWidgets, actualInteractWithAllWidgets);

}

@After

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

}