Automated Login Test Documentation for DemoBlaze

# Table of Contents

1. Introduction

2. Setup and Installation

3. Test Cases

4. Page Object Model

5. Running the Tests

6. Conclusion

# 1. Introduction

This document provides detailed instructions and information regarding the automated tests for the login functionality on the DemoBlaze website. The tests are implemented using Selenium WebDriver, Java, and TestNG. The project follows the Page Object Model (POM) design pattern to enhance maintainability and readability.

# 2. Setup and Installation

To set up the project and run the tests, follow the steps below:

1. Clone the repository from the source control system (e.g., GitHub).  
2. Navigate to the project directory.  
3. Ensure that Java JDK 8 or higher is installed on your system.  
4. Install Apache Maven for managing project dependencies.  
5. Execute the command `mvn clean install` to build the project and download dependencies.

# 3. Test Cases

The following test cases are implemented to verify the login functionality on DemoBlaze:

## 3.1. Successful Login Test

This test verifies that a user can successfully log in with valid credentials.

Steps:

1. Navigate to the DemoBlaze website.  
2. Click on the "Log in" link.  
3. Enter a valid username and password.  
4. Click on the "Log in" button.  
5. Verify that the logout link is displayed, indicating a successful login.

## 3.2. Unsuccessful Login Test

This test verifies that the appropriate error message is displayed when a user attempts to log in with invalid credentials.

Steps:

1. Navigate to the DemoBlaze website.  
2. Click on the "Log in" link.  
3. Enter an invalid username and password.  
4. Click on the "Log in" button.  
5. Handle any alert that appears.  
6. Verify that the error message is displayed and the user remains on the login page.

# 4. Page Object Model

The project uses the Page Object Model (POM) design pattern to separate the test code from the page-specific code. This approach enhances the maintainability and readability of the test scripts.

## 4.1. LoginPage.java

The LoginPage class encapsulates the elements and actions related to the login page on the DemoBlaze website. It includes methods to perform login actions and verify the presence of specific elements.

```java  
package com.sparkrock.tausif.pages;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.WebElement;  
import org.openqa.selenium.support.FindBy;  
import org.openqa.selenium.support.ui.ExpectedConditions;  
import org.openqa.selenium.support.ui.WebDriverWait;  
import java.time.Duration;  
public class LoginPage extends BasePage {  
 public LoginPage(WebDriver driver) { super(driver); }  
 @FindBy(id = "login2") private WebElement loginLink;  
 @FindBy(id = "loginusername") private WebElement usernameField;  
 @FindBy(id = "loginpassword") private WebElement passwordField;  
 @FindBy(xpath = "//button[text()='Log in']") private WebElement loginButton;  
 @FindBy(css = "div#logInModal .alert") private WebElement errorMessage;  
 @FindBy(id = "logout2") private WebElement logoutLink;  
 public void clickLoginLink() {  
 new WebDriverWait(driver, Duration.ofSeconds(10)).until(ExpectedConditions.elementToBeClickable(loginLink));  
 loginLink.click();  
 }  
 public void enterUsername(String username) {  
 new WebDriverWait(driver, Duration.ofSeconds(10)).until(ExpectedConditions.visibilityOf(usernameField));  
 usernameField.sendKeys(username);  
 }  
 public void enterPassword(String password) {  
 new WebDriverWait(driver, Duration.ofSeconds(10)).until(ExpectedConditions.visibilityOf(passwordField));  
 passwordField.sendKeys(password);  
 }  
 public void clickLoginButton() {  
 new WebDriverWait(driver, Duration.ofSeconds(10)).until(ExpectedConditions.elementToBeClickable(loginButton));  
 loginButton.click();  
 }  
 public boolean isErrorMessageDisplayed() {  
 handleAlert();  
 new WebDriverWait(driver, Duration.ofSeconds(20)).until(ExpectedConditions.visibilityOf(errorMessage));  
 return errorMessage.isDisplayed();  
 }  
 public boolean isLogoutDisplayed() {  
 new WebDriverWait(driver, Duration.ofSeconds(10)).until(ExpectedConditions.visibilityOf(logoutLink));  
 return logoutLink.isDisplayed();  
 }  
 private void handleAlert() {  
 try {  
 WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(5));  
 Alert alert = wait.until(ExpectedConditions.alertIsPresent());  
 alert.accept();  
 } catch (Exception e) {  
 // No alert found  
 }  
 }  
}  
```

# 5. Running the Tests

To run the tests, use the following Maven commands:

1. Clean the project:  
 ```sh  
 mvn clean  
 ```

2. Install dependencies and build the project:  
 ```sh  
 mvn install  
 ```

3. Execute the tests:  
 ```sh  
 mvn test  
 ```

# 6. Conclusion

This document provides an overview of the automated tests for the login functionality on the DemoBlaze website. Following the steps and guidelines outlined will help you set up, run, and maintain these tests effectively.