SDLMS Login Page Test Automation Documentation

Table of Contents

1. Introduction

This documentation includes a full overview of the TestCafe script designed for automated testing of the SDLMS (Sample Deep Learning Management System) login page. The major goal of this script is to assure the functioning and stability of the login procedure.

2. Setting up the Environment

2.1 Node.js Environment

Before beginning with TestCafe, it's vital to confirm that your Node.js environment is fully set up. Follow these steps:

Node.js Installation: Download and install the current version of Node.js from the official website: https://nodejs.org/

Verification: Confirm the successful installation by performing the following commands in your terminal or command prompt:

```
node -v
npm -v
```

2.2 TestCafe Installation

The next step is to install TestCafe worldwide. Use the following command:

```
npm install -g testcafe
```

3. Automated Test Cases

In this part, we explain the automated test scenarios aimed to validate the SDLMS login page.

3.1 Test Case 1: Successful Login

Objective: To validate that a user can successfully log in using valid credentials.

Test Code:

```
import { Selector } from 'testcafe';

// Define selectors for the username input, password input, and login button
const nameInput = Selector('#username');
const passwordInput = Selector('#password');
const loginButton = Selector('#login');

fixture('Login Tests')
  .page('https://dev.deepthought.education/login');

test('Successful Login', async (t) => {
  await t
    .maximizeWindow()
    .typeText(nameInput, 'mr_vishesh') // Replace with a valid username
    .typeText(passwordInput, 'vishesh@123') // Replace with a valid password
    .click(loginButton)
});
```

3.2 Test Case 2: Unsuccessful Login

Objective: To check that users cannot log in using incorrect credentials.

Test Code:

```
import { Selector } from 'testcafe';

// Define selectors for the username input, password input, and login button
const nameInput = Selector('#username');
const passwordInput = Selector('#password');
const loginButton = Selector('#login');

fixture('Login Tests')
  .page('https://dev.deepthought.education/login');

test('Unsuccessful Login', async (t) => {
  await t
    .maximizeWindow()
    .typeText(nameInput, 'invalid_user') // Replace with an invalid username
    .typeText(passwordInput, 'invalid_password') // Replace with an invalid password
    .click(loginButton)
});
```

4. Cross-Browser Testing

To assure the interoperability of the SDLMS login page across different browsers, we do cross-browser testing.

4.1 Configuration

- The test script is configured to run on Chrome and Firefox.

4.2 Test Execution

To perform cross-browser tests, enter your terminal or command prompt and type the following command:

```
testcafe chrome,firefox loginTests.js
```

5. Documentation

Documentation is crucial for understanding the testing procedure and outcomes.

5.1 Reflecting on Testing

Reflecting on our testing method helps us obtain deeper insights into our learning. Here's a perspective on our testing journey:

Learning: Through this testing procedure, we got significant knowledge in building up automated tests and utilising TestCafe.

issues: We experienced issues in verifying error messages and ensuring tests perform consistently across browsers.

5.2 Testing Approach

Our testing strategy comprised of:

- Writing automated test cases for both successful and failing login scenarios.

- Conducting cross-browser testing to verify compatibility.

- Documenting our method and issues.

5.3 Test Cases

We wrote two major test cases:

Successful Login: This case confirms that users can log in successfully with proper credentials.

Unsuccessful Login: This case validates that users cannot log in with incorrect credentials and confirms the error message.

5.4 Challenges Faced During Testing

Challenge: Ensuring that error messages are accurately presented for failing login attempts.

Solution: Utilized a particular identifier for the error message and checked its content.

6. Conclusion

This detailed documentation explains the process of setting up the environment, building automated test cases, doing cross-browser testing, and reporting on the testing strategy for the SDLMS login page.