AUTOMATION & TESTING

Automated User Login Processes for SDLMS Application
Introduction

This document provides the details for automation of user login processes for the SDLMS (Sample Digital Learning Management System) application using TestCafe. The process involves setting up a local Node.js environment, writing test cases for successful and unsuccessful logins, and configuring TestCafe to run tests on different browsers.

Prerequisites

Before starting the automation process,we need to ensure we have:-:

i) Node.js and npm installed on your local environment.
ii) Chrome and Firefox browsers installed.

Approach

1) Set up a local Node.js environment

Install Node.js and npm:

Download and install Node.js from the official website: Node.js.

Install TestCafe globally:

:
Run ->npm install -g testcafe in Command Prompt.

2) Test Successful and Unsuccessful Logins

Create Test Files:
Create separate test files (e.g., loginTest.js) for testing successful and unsuccessful logins.

Write Test Cases:

Write test cases to simulate successful login with valid credentials and unsuccessful login attempts with invalid credentials.
Use TestCafe's API to interact with the SDLMS application, enter credentials, click buttons, and verify outcomes.

Sample Test Cases:

Successful Login:

```
import { fixture } from "testcafe";

fixture `Login Tests`
    .page `https://dev.deepthought.education/login`;

test('Login user with valid username and password', async t =>{
    await t
        .typeText('#username', 'soumyajit56340@gmail.com')
        .typeText('#password', 'Hacker@56340')
        .click(Selector('.btn'))

        await t.expect(Selector('H5').withExactText('Welcome to
DeepThought').exists).ok();
});
```

Unsuccessful Login (Incorrect Username):

```
fixture `Login Tests`
    .page `https://dev.deepthought.education/login`;

test('Login user with Invalid Username', async t =>{
    await t
        .typeText(Selector('#username'), 'soumyajit56@gmail.com')
        .typeText(Selector('#password'), 'Hacker@56340')
        .click(Selector('.btn'))

        await t.expect(Selector('#login-error-notify').exists).ok();
});
```

Unsuccessful Login (Incorrect Password):

```
fixture `Login Tests`
    .page `https://dev.deepthought.education/login`;

test('Invalid password', async t =>{
    await t
        .typeText(Selector('#username'), 'soumyajit56340@gmail.com')
        .typeText(Selector('#password'), 'Hac6340')
        .click(Selector('.btn'))

        await t.expect(Selector('#login-error-notify').exists).ok();
});
```

Unsuccessful Login (Empty Fields):

```
fixture `Login Tests`
    .page `https://dev.deepthought.education/login`;

    test('Unsuccessful Login - Empty Fields', async t => {
        await t
            .click(Selector('.btn'))

            await t.expect(Selector('#login-error-notify').exists).ok();
    });
```

Step 3: Configure TestCafe to Run on Different Browsers

Update Test Configuration:

Specify the browsers you want to test against.

```
  fixture `Login Tests`
  .page `https://dev.deepthought.education/login`;

test('Chrome Browser', async t => {
  await t.useRole(chromeUser).run();
});

test('Firefox Browser', async t => {
  await t.useRole(firefoxUser).run();
});
```

Create Roles:
Define roles for different users and specify the browser they will use.

Step 4: Validate Results

Open a command prompt or terminal, navigate to the directory containing your test files, and run the tests:

testcafe chrome loginTest.js
testcafe firefox loginTest.js

Review the test output to ensure that the processes work consistently across the chosen browsers.

Conclusion

By following these steps, you have successfully automated the user login processes for the SDLMS application using TestCafe. This approach ensures that login functionalities are thoroughly tested across different browsers, providing a robust and reliable testing framework for your application.