Elite IT Assessment

Submitted By

Daniyal Zahid

Task 1:

a. Provide a test plan outline for testing the registration form. Include the following sections:

- Introduction
- Objectives
- Scope
- Test Environment
- Test Data
- Test Scenarios
- Test Execution Schedule
- Risks and Assumptions

Introduction

We will be developing a test plan basically outlining the strategy and approach we will be using for testing of the online registration form of the said website.

Objectives

- ➔ We will ensure that all the fields in our form accept valid input.
- ➔ If an erroneous input is made, we will verify that a suitable and appropriate prompt/ message is displayed for invalid inputs.
- ➔ At the end, if all fields have valid and correct data, we can confirm that the form can be submitted successfully.

Scope

- ➔ We will be performing functional testing, since we have to verify that the software (in this case, our website) performs its intended functions correctly. (i.e the fields in our form)
- ➔ We will be performing validation of error messages.
- ➔ A very crucial part is the submission of the form and afterwards, the server response handling.

Test Environment

- ➔ OS: Windows 11
- ➔ Chrome (Latest Version 126.0.6478.127)
- ➔ Framework: Selenium with pytest
- ➔ Python Version: 3.8 +

Test Data

➔ Valid data and invalid data for each respective field of the form.

➔ Different pairs of combinations of form data to test different scenarios.

➔ Example:

a. Full Name
b. Email Address
c. Password
d. Confirm Password
e. Date of Birth
f. Gender
g. Newsletter Subscription

Test Scenarios

1. **Full Name Field**
   a. Enter valid full name
   b. Enter empty full name
   c. Enter special characters and numbers in full name
2. **Email Address Field**
   a. Enter valid email address
   b. Enter invalid email address format
   c. Enter empty email address
3. **Password Field**
   a. Enter valid password (at least 8 characters)
   b. Enter password with less than 8 characters
   c. Enter empty password

4. **Confirm Password Field**
    a. Enter matching confirm password
    b. Enter non-matching confirm password
    c. Enter empty confirm password
5. **Date of Birth Field**
    a. Enter valid date of birth
    b. Enter future date of birth
    c. Enter empty date of birth
6. **Gender Field**
    a. Select each gender option
    b. Leave gender unselected
7. **Newsletter Subscription Field**
    a. Select "Yes"
    b. Select "No"
    c. Leave subscription unselected
8. **Submit Button**
    a. Click submit with all valid inputs
    b. Click submit with one or more invalid inputs
    c. Verify successful form submission and server response

Test Execution Schedule

The testing will be executed in the following phases:

- ➔ Phase 1: Preparation of test data and test environment setup.
- ➔ Phase 2: Execution of test scenarios and defect logging
- ➔ Phase 3: Retesting and regression testing
- ➔ Phase 4: Test closure and reporting

Risks and Assumptions

- ➔ Potential server downtime, browser compatibility issues.
- ➔ Changes in form design or functionality during testing may affect the test results.
- ➔ If the form has been implemented according to the specified requirements and is accessible in the test environment.

Task 2:

Write test cases for at least three scenarios covering different aspects of the registration form. Each test case should include:

- Test Case ID
- Test Case Description
- Preconditions
- Test Steps
- Expected Results
- Post-conditions

Test Case 1: Valid Registration

Test Case ID:

➔ TC_01

Test Case Description:

➔ Verify that the registration form accepts valid inputs and successfully submits the data.

Preconditions:

➔ The user is on the registration form page.

Test Steps:

1. Enter "John Doe" in the Full Name field.
2. Enter "johndoe@example.com" in the Email Address field.
3. Enter "Password123!" in the Password field.
4. Enter "Password123!" in the Confirm Password field.
5. Select a valid Date of Birth.
6. Select "Male" for Gender.
7. Select "Yes" for Newsletter Subscription.
8. Click the Submit button.

Expected Results:

➔ The form should be submitted successfully.

➔ A success message should be displayed (e.g., "Registration successful").

➔ The user should be redirected to a welcome or login page.

Post-conditions:

➔ The user account should be created in the database.

Test Case 2: Password Mismatch

Test Case ID:

➔ TC_02

Test Case Description:

➔ Verify that the registration form displays an error when the Password and Confirm Password fields do not match.

Preconditions:

➔ The user is on the registration form page.

Test Steps:

1. Enter "Jane Doe" in the Full Name field.
2. Enter "janedoe@example.com" in the Email Address field.
3. Enter "Password123!" in the Password field.
4. Enter "Password321!" in the Confirm Password field.
5. Select a valid Date of Birth.
6. Select "Female" for Gender.
7. Select "No" for Newsletter Subscription.
8. Click the Submit button.

Expected Results:

➔ An error message should be displayed near the Confirm Password field (e.g., "Passwords do not match").

➔ The form should not be submitted.

Post-conditions:

➔ The user should remain on the registration form page with previously entered data (excluding passwords).

Task 3: Invalid Email Format

Test Case ID:

➔ TC_03

Test Case Description:

➔ Verify that the registration form displays an error when the Email Address field contains an invalid email format.

Preconditions:

➔ The user is on the registration form page.

Test Steps:

1. Enter "Alice Doe" in the Full Name field.
2. Enter "alice.doe" in the Email Address field.
3. Enter "Password123!" in the Password field.
4. Enter "Password123!" in the Confirm Password field.
5. Select a valid Date of Birth.
6. Select "Other" for Gender.
7. Select "Yes" for Newsletter Subscription.
8. Click the Submit button.

Expected Results:

➔ An error message should be displayed near the Email Address field (e.g., "Please enter a valid email address").

➔ The form should not be submitted.

Post-conditions:

➔ The user should remain on the registration form page with previously entered data (excluding passwords).

Test Scenarios: Positive

Positive Scenarios:

1. Valid Input Data:
   a. Enter a valid full name (e.g., "John Doe").
   b. Enter a valid email address (e.g., "john.doe@example.com").
   c. Enter a password that meets the criteria (e.g., "Password123").
   d. Confirm the password by re-entering it.
   e. Select a valid date of birth.
   f. Choose a gender.
   g. Opt for or against the newsletter subscription.
   h. Click the "Submit" button.

   Expected Outcome: The form submits successfully, and the user receives a confirmation message indicating successful registration.

2. Successful Password Match:
   a. Enter a valid full name.
   b. Enter a valid email address.
   c. Enter a password that meets the criteria.
   d. Confirm the password by re-entering it.
   e. Select a valid date of birth.
   f. Choose a gender.
   g. Opt for or against the newsletter subscription.
   h. Click the "Submit" button.

   Expected Outcome: The form submits successfully without any error messages, and the user receives a confirmation message indicating successful registration.

Test Scenarios: Negative

Negative Scenarios:

1. Invalid Email Format:
    a. Enter a valid full name.
    b. Enter an invalid email address (e.g., "john.doe@com").
    c. Enter a password that meets the criteria.
    d. Confirm the password by re-entering it.
    e. Select a valid date of birth.
    f. Choose a gender.
    g. Opt for or against the newsletter subscription.
    h. Click the "Submit" button.

    Expected Outcome: The form displays an error message indicating an invalid email format and does not submit.

2. Password Mismatch:
    a. Enter a valid full name.
    b. Enter a valid email address.
    c. Enter a password that meets the criteria.
    d. Enter a different password in the confirm password field.
    e. Select a valid date of birth.
    f. Choose a gender.
    g. Opt for or against the newsletter subscription.
    h. Click the "Submit" button.

    Expected Outcome: The form displays an error message indicating that the passwords do not match and does not submit.

Scripts

```
import pytest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC


@pytest.fixture(scope="module")
def driver():
    driver = webdriver.Chrome() #installed chromedriver and then added in path
in system properties
    driver.get("https://demoqa.com/automation-practice-form")
    yield driver
    driver.quit()


def test_valid_input_data(driver):
    driver.find_element(By.ID, "firstName").send_keys("John")
    driver.find_element(By.ID, "lastName").send_keys("Doe")
    driver.find_element(By.ID, "userEmail").send_keys("john.doe@example.com")
    driver.find_element(By.ID, "userNumber").send_keys("0123456789")
    driver.find_element(By.ID, "dateOfBirthInput").click()
    driver.find_element(By.CLASS_NAME, "react-datepicker__year-
select").send_keys("1990")
    driver.find_element(By.CLASS_NAME, "react-datepicker__month-
select").send_keys("January")
    driver.find_element(By.CLASS_NAME, "react-datepicker__day--001").click()
    driver.find_element(By.CSS_SELECTOR, "label[for='gender-radio-
1']").click()
    driver.find_element(By.ID, "subjectsInput").send_keys("Maths")
    driver.find_element(By.CSS_SELECTOR, "label[for='hobbies-checkbox-
1']").click()
    driver.find_element(By.ID, "currentAddress").send_keys("123 Main St")
    driver.find_element(By.ID, "submit").click()

    confirmation_message = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.ID, "example-modal-sizes-title-
lg"))
    )
    assert "Thanks for submitting the form" in confirmation_message.text


def test_successful_password_match(driver):
    driver.find_element(By.ID, "firstName").send_keys("Jane")
    driver.find_element(By.ID, "lastName").send_keys("Doe")
    driver.find_element(By.ID, "userEmail").send_keys("jane.doe@example.com")
    driver.find_element(By.ID, "userNumber").send_keys("0123456789")
    driver.find_element(By.ID, "dateOfBirthInput").click()
    driver.find_element(By.CLASS_NAME, "react-datepicker__year-
select").send_keys("1992")
    driver.find_element(By.CLASS_NAME, "react-datepicker__month-
select").send_keys("February")
```

```
    driver.find_element(By.CLASS_NAME, "react-datepicker__day--002").click()
    driver.find_element(By.CSS_SELECTOR, "label[for='gender-radio-
2']").click()
    driver.find_element(By.ID, "subjectsInput").send_keys("Physics")
    driver.find_element(By.CSS_SELECTOR, "label[for='hobbies-checkbox-
2']").click()
    driver.find_element(By.ID, "currentAddress").send_keys("456 Elm St")
    driver.find_element(By.ID, "submit").click()

    confirmation_message = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.ID, "example-modal-sizes-title-
lg"))
    )
    assert "Thanks for submitting the form" in confirmation_message.text


def test_invalid_email_format(driver):
    driver.find_element(By.ID, "firstName").send_keys("John")
    driver.find_element(By.ID, "lastName").send_keys("Doe")
    driver.find_element(By.ID, "userEmail").send_keys("john.doe@com")
    driver.find_element(By.ID, "userNumber").send_keys("0123456789")
    driver.find_element(By.ID, "dateOfBirthInput").click()
    driver.find_element(By.CLASS_NAME, "react-datepicker__year-
select").send_keys("1990")
    driver.find_element(By.CLASS_NAME, "react-datepicker__month-
select").send_keys("January")
    driver.find_element(By.CLASS_NAME, "react-datepicker__day--001").click()
    driver.find_element(By.CSS_SELECTOR, "label[for='gender-radio-
1']").click()
    driver.find_element(By.ID, "subjectsInput").send_keys("Maths")
    driver.find_element(By.CSS_SELECTOR, "label[for='hobbies-checkbox-
1']").click()
    driver.find_element(By.ID, "currentAddress").send_keys("123 Main St")
    driver.find_element(By.ID, "submit").click()

    error_message = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.CSS_SELECTOR, ".was-validated
.form-control:invalid"))
    )
    assert "email" in error_message.get_attribute("validationMessage").lower()


def test_password_mismatch(driver):
    driver.find_element(By.ID, "firstName").send_keys("Jane")
    driver.find_element(By.ID, "lastName").send_keys("Doe")
    driver.find_element(By.ID, "userEmail").send_keys("jane.doe@example.com")
    driver.find_element(By.ID, "userNumber").send_keys("0123456789")
    driver.find_element(By.ID, "dateOfBirthInput").click()
    driver.find_element(By.CLASS_NAME, "react-datepicker__year-
select").send_keys("1992")
    driver.find_element(By.CLASS_NAME, "react-datepicker__month-
select").send_keys("February")
    driver.find_element(By.CLASS_NAME, "react-datepicker__day--002").click()
    driver.find_element(By.CSS_SELECTOR, "label[for='gender-radio-
2']").click()
    driver.find_element(By.ID, "subjectsInput").send_keys("Physics")
```

```
    driver.find_element(By.CSS_SELECTOR, "label[for='hobbies-checkbox-
2']").click()
    driver.find_element(By.ID, "currentAddress").send_keys("456 Elm St")
    driver.find_element(By.ID, "submit").click()

    error_message = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.CSS_SELECTOR, ".was-validated
.form-control:invalid"))
    )
    assert "match" in error_message.get_attribute("validationMessage").lower()
```