Test Plan Outline for Testing the Registration Form

1. Introduction

The online registration form is a critical feature of any website that allows new users to create an account. This form collects personal information from users and validates the input data before submission. The registration process ensures that users can securely sign up and access the services provided by the website.

2. Objectives

- To verify that the registration form collects all necessary information.
- To ensure that all form fields validate user inputs correctly.
- To confirm that error messages are displayed for invalid inputs.
- To check that the form data is successfully submitted to the server when all inputs are valid.
- To validate the user’s ability to subscribe to the newsletter.

3. Scope

The scope of this test plan includes:

- Functional testing of each form field.
- Validation of input data.
- Testing the submission process.
- Verifying the display of error messages.
- Checking that the form’s functionality aligns with the requirements.

4. Test Environment

- **Operating System:** Windows, macOS, Linux
- **Browsers:** Chrome, Firefox, Edge
- **Test Tools:** Puppeteer, Cucumber
- **Server:** Staging environment for testing purposes
- **Database:** Test database for form submissions

5. Test Data

- **Full Name:** "John Doe"
- **Email Address:** "john.doe@example.com"
- **Password:** "Password123"
- **Confirm Password:** "Password123"
- **Date of Birth:** "01/01/1990"
- **Gender:** "Male"
- **Newsletter Subscription:** "Yes"

Invalid Test Data Examples:

- **Email Address:** "john.doe@example"
- **Password:** "short"
- **Confirm Password:** "Password12"
- **Date of Birth:** "31/02/1990"

6. Test Scenarios

Scenario ID	**Description**	**Test Data**
TC01	Submit the form with valid data	All fields are filled with valid data
TC02	Enter invalid email address	"john.doe@example"
TC03	Enter mismatched passwords	"Password123" and "Password124"
TC04	Leave the password field empty	Leave the password field blank
TC05	Leave required fields blank	Leave Full Name and Email Address blank

7. Test Execution Schedule

Task	**Start Date**	**End Date**
Test Planning	2024-07-15	2024-07-16
Test Case Development	2024-07-17	2024-07-18
Test Environment Setup	2024-07-19	2024-07-19
Test Execution	2024-07-22	2024-07-24
Test Reporting	2024-07-25	2024-07-26
Review and Feedback	2024-07-29	2024-07-30

8. Risks and Assumptions

Risks:

- Changes in the form layout or requirements.
- Unavailability of the test environment.
- Browser compatibility issues.

Assumptions:

- The development team has provided the final version of the form.
- The test environment mirrors the production environment.
- Test data will be available for use.

Test Cases

1. Test Case for Valid Data Submission

Test Case ID: TC01

Test Case Description: Verify that the form successfully submits valid data.

Preconditions: The form is loaded and accessible.

Test Steps:

1. Enter a valid Full Name.
2. Enter a valid Email Address.
3. Enter a valid Password (at least 8 characters).
4. Confirm the Password.
5. Select Date of Birth.
6. Select Gender.
7. Choose Newsletter Subscription (Yes).
8. Click the Submit Button.

Expected Results: The form should successfully submit the data and display a confirmation message.

Post-conditions: User data should be sent to the server for registration.

---

2. Test Case for Invalid Email Address

Test Case ID: TC02

Test Case Description: Verify that an invalid email address displays the correct error message.

Preconditions: The form is loaded and accessible.

Test Steps:

1. Enter a valid Full Name.
2. Enter an invalid Email Address (e.g., "john.doe@example").
3. Enter a valid Password.
4. Confirm the Password.
5. Select Date of Birth.
6. Select Gender.
7. Choose Newsletter Subscription (Yes).
8. Click the Submit Button.

Expected Results: The form should display an error message indicating that the email address is invalid.

Post-conditions: The form should not submit data and remain on the registration page.

---

3. Test Case for Mismatched Passwords

Test Case ID: TC03

Test Case Description: Verify that mismatched passwords display the correct error message.

Preconditions: The form is loaded and accessible.

Test Steps:

1. Enter a valid Full Name.
2. Enter a valid Email Address.
3. Enter a valid Password (e.g., "Password123").
4. Enter a different Confirm Password (e.g., "Password124").
5. Select Date of Birth.
6. Select Gender.
7. Choose Newsletter Subscription (Yes).
8. Click the Submit Button.

Expected Results: The form should display an error message indicating that the passwords do not match.

Post-conditions: The form should not submit data and remain on the registration page.

C. Scenario Identification

Positive Scenarios

1. **Scenario: Successful Registration**

   Steps:

   1. Enter valid Full Name, Email Address, Password, Confirm Password, Date of Birth, and Gender.
   2. Select “Yes” for Newsletter Subscription.
   3. Click the Submit Button.

   Expected Outcome: The form is successfully submitted, and the user receives a confirmation message.

2. **Scenario: Newsletter Subscription Selection**

   Steps:

   1. Enter valid details in all fields.
   2. Select “Yes” for Newsletter Subscription.
   3. Click the Submit Button.

   Expected Outcome: The form is submitted, and the user is subscribed to the newsletter.

Negative Scenarios

1. **Scenario: Invalid Email Address**

   Steps:

   1. Enter valid Full Name, Password, Confirm Password, Date of Birth, and Gender.
   2. Enter an invalid Email Address.
   3. Click the Submit Button.

   Expected Outcome: The form displays an error message about the invalid email address.

2. **Scenario: Password Field Empty**

   Steps:

   1. Enter valid Full Name, Email Address, Confirm Password, Date of Birth, and Gender.
   2. Leave the Password field empty.
   3. Click the Submit Button.

   Expected Outcome: The form displays an error message about the empty Password field.

Automated Test Scripts Using TypeScript, Puppeteer, and Cucumber

Below are TypeScript test scripts for the positive scenarios using Puppeteer and Cucumber.

1. Feature File

`registration.feature`

```
Feature: Registration Form

  Scenario: Successful Registration
    Given I am on the registration page
    When I enter valid details for all fields
    And I click the Submit button
    Then I should see a registration confirmation message

  Scenario: Newsletter Subscription Selection
    Given I am on the registration page
    When I enter valid details for all fields
    And I select "Yes" for Newsletter Subscription
    And I click the Submit button
    Then I should see a registration confirmation message
```

2. Step Definitions

`registrationSteps.ts`

```
import { Given, When, Then } from '@cucumber/cucumber';
import puppeteer from 'puppeteer';

let page: puppeteer.Page;
let browser: puppeteer.Browser;

Given('I am on the registration page', async () => {
  browser = await puppeteer.launch({ headless: false });
  page = await browser.newPage();
  await page.goto('https://example.com/registration');
});

When('I enter valid details for all fields', async () => {
  await page.type('#fullName', 'John Doe');
  await page.type('#email', 'john.doe@example.com');
  await page.type('#password', 'Password123');
  await page.type('#confirmPassword', 'Password123');
  await page.type('#dob', '01/01/1990');
  await page.click('#genderMale');
  await page.click('#newsletterYes');
});

When('I click the Submit button', async () => {
  await page.click('#submitButton');
});

Then('I should see a registration confirmation message', async () => {
  await page.waitForSelector('#confirmationMessage');
  const message = await page.$eval('#confirmationMessage', el =>
el.textContent);
  if (message !== 'Registration successful!') {
    throw new Error('Expected registration confirmation message but did not
see it.');
  }
  await browser.close();
});
```