LifeRay Form Test Automation Project

Test Strategy

Revision History

Date	Version	Author	Description
15/2/2023	1.0	Matteo Orrù	Test Strategy for LifeRay form

Table of Contents

1. Scope 4

2. Test Approach 5

3. Test Environment 6

4. Risk Analysis 7

1. Scope

The scope of this document is to outline the testing strategy to test the LifeRay Form. This document will be reviewed by the QA Team and the Development Team who and the testing strategy will be implemented pending both Teams approval.

Different testing activities can be carried out:

- Functional E2E testing to verify the application under test (henceforth AUT) is working as expected and it fulfills the functional requirements.
- Security Testing to verify that no vulnerability are exposed to malicious users.
- Performance testing, and specifically Load Testing, to assure that the AUT stays available in different traffic conditions, including traffic bursts, without outages.

Due to time constraint the present Test Strategy will be focusing on the Functional E2E testing activities.

2. Test Approach

This paragraph outlines the test approach we'll be following to implement the testing automation strategy. As reported in the previous section, we'll be going to focus mostly on Functional E2E testing. The goal of this testing is to ensure that the form is functioning as expected and is able to handle all types of user input.

Objectives:

1. Testing that the form loads correctly.
2. Testing that all form fields are present and functional.
3. Testing error messages for invalid input are properly displayed.
4. Testing that the form is able to handle different types of input.

The testing approach will be articulated in the following steps:

1. Identification of all the form fields and their respective data types.
2. Development of the test scenarios in Gherkin syntax using Cucumber.
3. Coding the test scenarios using Java and Selenium WebDriver.
4. Implement a pipeline in order to run all test scenarios through GitLab CI/CD pipeline.
5. Record the results and generate test reports.

The following test data will be used:

1. Valid/Invalid data for all form fields.
2. Blank/edge case data for all the form fields.
3. Different types of user input.

Test Execution:

The test suite execution will be triggered at the of each delivery pipeline, and it will be run targeting the application deployed in a QA or Staging environment to be sure that no regression have been introduced. A report will be available in a dedicated space in Gitlab Pages. Here a recent history of the most recent run will be hosted as well.

3. Test Environment

The testing will be carried out in the following testing environment:

1. **Selenium Webdriver for Java (latest version).** Tests will be written in Java and will leverage the Selenium WebDriver API in order to automate browser actions and replicate users behavior (such as clicking buttons, filling forms, etc.) to verify that the expected results are produced. This allows to automate repetitive manual testing, saving time and enhancing reproducibility and reliability of the testing process.

2. **Cucumber (latest version).** We'll be leveraging on to apply a behavior-driven development (BDD) approach to our testing process. This will facilitate the communication between stakeholders (including business analysts, product owners, and developers) since it allows to write automated tests in a easly to understood plain language that is easily understood by stakeholders.

3. **GitLab (latest version).** The code will be stored in Gitlab CVS, and the CI/CD processes will be implemented using Gitlab CI features.

4. **External Testing platform.** tests will be conducted on a number of browsers, operating systems and devices by exploiting a cloud-based testing platform such as Browserstack.

4. Risk Analysis

A thorough risk analysis has been reported in the Test Plan.