Test Plan

Overview

The goal of this document is to present the testing approach and objectives for testing a web application consisting on a form to collect answers from user.

Change History

Release N.	**Author**	**Date**
1.0	Matteo Orrù	14/02/2023

Introduction

The goal of this document is to present the testing approach and objectives for testing a web application consisting on a form (henceforth called either “LifeRay form” or “form under test”, on occasion shortened into FUT) available at the following url:

https://forms.liferay.com/web/forms/shared/-/form/122548

Objectives

The main objectives of this test plan are the following:

1. Verifying that the form loads correctly in the web browser.
2. Verifying that all form fields are present and functional.
3. Verify that the form correctly handles erroneous input by displaying error messages when mandatory fields are left blank or invalid data is entered.
4. Verifying that the input in the form field is properly collected.

In principle the overall testing activity should include also verifying that the data are stored property in the DB.

Reference

Documents and artifacts

- Test repository: https://github.com/M4tt30rru/AutomationTestingProject
- Test Strategy:
- Test Report:

Context of testing

Details of the project

The FUT consists on three fields to be used to collect specific information from users. Specifically the fields (all mandatory) are the following:

- Input field that collect the answer to the question “What is your favorite soccer player?”

- Input field to collect the answer to “what was the date that Liferay was founded?”
- Input field (text area) “Why did you join the testing area?”

A picture of the for is reported in Fig. 1.

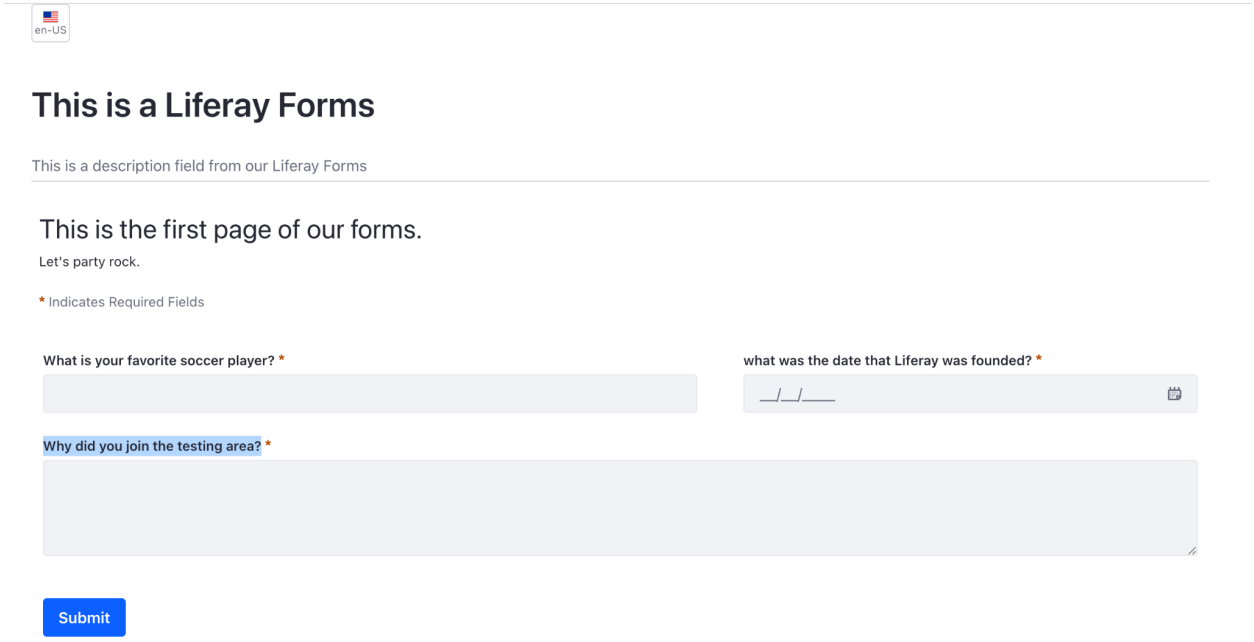

Fig 1. LifeRay form

All the fields in the form are mandatory. In case any of the field is missing the submission fails. Once the submission succeed, the user is directed to a different page where a success message is shown. This page is represented in Fig. 2.

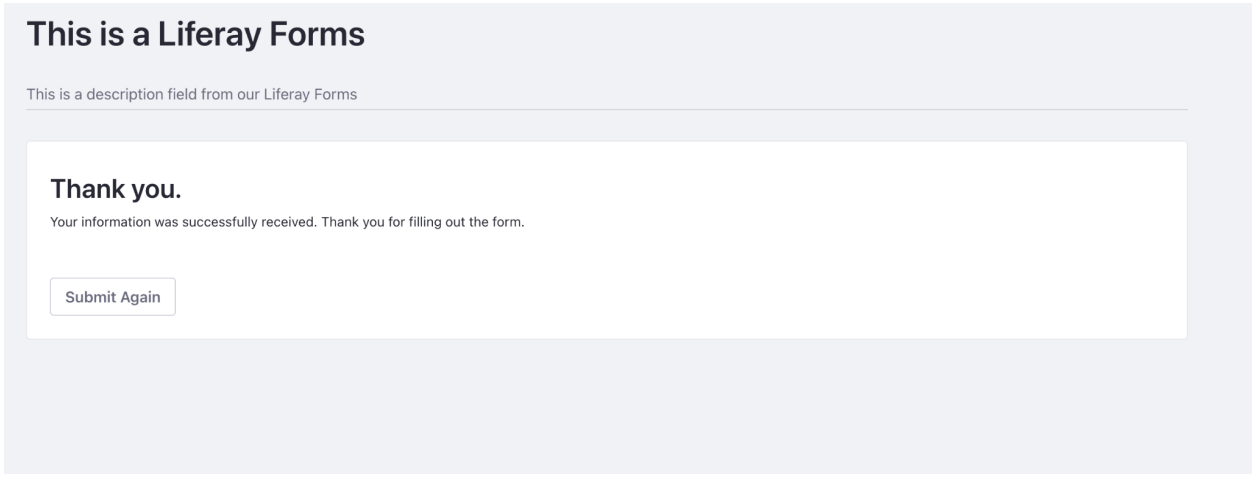

Fig 2. Thank you page

We assume that this information is getting stored in a database. No access to the database has been provided. The form is delivered in two languages: American English and Brazilian Portuguese.

Test scope

- The test performed will be functional e2e test to assure that all the functionality required by the form are correctly implemented.
- To collect the test case we relay on an preliminary exploratory testing activity.
- Combinatorial testing can be leverage in order to select the proper combination of data testing feature.

Assumptions and constraints

Since there is no access to a requirement document, the only way to devise test cases is by familiarize with the form through **exploratory testing**. It looks like the, with the exception of the second field, where the user is asked to reply with a date in a specific format to a **closed question**, no other constraint are required.
This lead to the assumption that the form is aimed at collecting user's answers in a free form, without providing the checks on the format that can influence, orientate or bias their answer.
The freedom left to the user has to be mitigated in orde to prevent the input or misleading or corrupted information that can clutter the DB.
A constraint for the testing activity is related to the unaccessibility of the DB, that prevent to verify the proper storage of the data.
Further tests such as performance testing to assure that the for will be working under different heavy traffic conditions in out of the scope of the current activity.
Security testing to prevent potential threats such as cross-site scripting and SQL injection can be also planned (targeting at least the main OWASP recommendation).

Stakeholders

Potential stakeholders are the following:

- Development Team with the ownership on the form and underlying services. Depending on the team structure the main reference could be the Product Owner, Team or Tech Lead.

- DBA with access to the DB where the data is stored.

- Platform or SREs to help with the automation process (in case of problems).

Testing communication lines inside and outside the test team

- Bugs found will be reported on Jira under Team dashboard, using the proper categorization (Epic, Team Involved field, etc.)
- The report generated by the test case will be available in the intranet.
- An alert is sent to a common channel of communication (i.e. slack channel, google chat bot) so that any stakeholder is kept in the loop. The alert will report the link to the report page, where the most recent history of the runs is also stored.

Risk register

This section outlines:

- The potential issues and risks that may arise during the testing process.
- The mitigation strategy to put in place to address them.

The following risks have been identified:

1. Data test incorrect, insufficient or not completed.
2. Browser's incompatibility.
3. Inadequate testing coverage.
4. Connectivity problems with the database.
5. Security risks (i.e. SQL injection).
6. Infrastructure issue. These can lead to automation failures, which would prevent the execution of the tests.

Mitigation strategies.

1. Ensure that the test data is correct and valid
2. Test the form on different browsers to ensure compatibility
3. Develop comprehensive test cases to ensure adequate testing coverage

4. Monitoring connectivity with the DB (for example with Prometheus or Grafite). Setup a monitoring dashboard (for example with Grafana). Implementing heartbeat probes to check the liveness of the DB server. Set alerts in Grafana to allow early detection of potential DB outages.
5. Implement security measures to protect the form and the data stored in the database.
6. Monitor the automation pipeline to identify and resolve any automation failures.

Risk management process steps will be the following:

1. Periodically review and update the risk assessment document.
2. Monitor the testing process to identify any new risks
3. Develop and implement mitigation strategies to address any new risks
4. Record and report any risks and their mitigation strategies

Test strategy

The testing approach will involve the following steps:

1. Identify all form fields and their respective data types
2. Develop test cases to verify
    a. The functionality of each form field.
    b. That the form can handle different types of user input.
    c. That the form data is properly stored in the database after submission.
    d. That the form displays error messages for invalid input.

Test deliverables

The following deliverables will be produced:

1. Test Plan document (this document)
2. Test Strategy document
3. Gherkin test scenarios (stored in the GitHub repository along with the code).
4. Java automation code (stored in the GitHub repository).
5. Test results

6. Test report for every run will be hosted in Gitlab pages.
7. Alert will be sent to a

Test design techniques

The test suite will be developed in Java, using Selenium Webdriver, adopting the Behavioural Driven Design approach. To implement the BDD approach the Cucumber framework will be used. The test design and implementation will be following the best programming practices (SOLID principle, clean code best practices) and programming patterns (i.e. Page Object pattern).

Test completion criteria

The test suite will be hosted in a CI/CD infrastructure (i.e. Gitlab) and will be triggered on a daily basis using a scheduled job. The test coverage has to be adequate.

Test environment requirements

The testing will be conducted in the following environment:

1. GitLab version: latest version
2. Operating System: Windows/Mac/Linux
3. Java version: latest version
4. Cucumber version: latest version
5. Selenium Webdriver: latest version

Staffing

No information about the staff is available.

Schedule

No information about the staff is available. Hence, without an idea of the effort required and the team velocity - computed for example, using story point over a consistent time span, it is hard to determine a schedule.