BRELLA FORM

TEST PLAN

Document Change History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version Number** | **Date** | **Contributor** | **Description** |
| V1.0 | 08.04.2022 | Sibel Ulufer | General testing plan documentation |

April 9, 2022

BRELLA

Table of Contents

[1. Introduction 1](#_Toc98065713)

[1.1 Scope 1](#_Toc98065714)

[1.2 Quality Objective 1](#_Toc98065715)

[1.3 Roles & Responsibilities 1](#_Toc98065716)

[1.4 Definitions 2](#_Toc98065717)

[2. Test Methodology 2](#_Toc98065718)

[2.1 Purpose 2](#_Toc98065719)

[2.2 Test Approach 4](#_Toc98065720)

[2.3 Bug Fix Tests 4](#_Toc98065721)

[2.4 Test Completeness 5](#_Toc98065722)

[3. Test Deliverables 5](#_Toc98065723)

[3.1 Test Schedule 5](#_Toc98065724)

[3.2 Deliverables 6](#_Toc98065725)

[4. Resource Needs 7](#_Toc98065726)

[4.1 Tools 7](#_Toc98065727)

[4.2 Bug Severity and Priority Definition 7](#_Toc98065728)

# Introduction

This test approach document describes the appropriate strategies, processes, workflows, and methodologies used to plan, organize, execute and manage the testing of software projects within.

## Scope

Brella form is a web-based page used to save the form details which is filled. The test team is responsible for testing the product and ensuring it meets their needs. The test team is both the customer and the tester in this project.

## Quality Objective

The objective of testing application systems is to: assure that the system meets the full requirements, including quality requirements and fit metrics for each quality requirement and satisfies the use case scenarios, and maintains the quality of the product. At the end of the project development cycle, the user should find that the project has met or exceeded all of their expectations as detailed in the requirements.

Any changes, additions, or deletions to the requirements document, Functional Specification, or Design Specification will be documented and tested at the highest level of quality allowed within the remaining time of the project and within the ability of the test team.

## Roles & Responsibilities

Roles and responsibilities may differ based on the project. The below-listed functions are for the testing phase.

QA Team responsible to:

(a) Contribute to User Stories, test cases through review

(b) Contribute to development and execution of the development test scripts if it is the automated testing

(c) Conduct regression, smoke, sanity and end-to-end testing; this includes identifying testing scenarios, building the test scripts, executing scripts, and reporting test results

|  |  |
| --- | --- |
| **Resource Name** | **Role** |
| <Developer\_Name> | Developer |
| <PM\_\_Name> | Project Manager |
| Sibel Ulufer | QA Specialist |

## Definitions

**Bugs**: Any error or defect that causes the software/application or hardware to malfunction. That is also included in the requirements and does not meet the required workflow, process, or function point.

**Enhancement**:

1) Any alteration or modification to the existing system for better workflow and process.

2) An error or defect that causes the software/application or hardware to malfunction.

Where 1) **and** 2) is NOT included in the requirements can be categorized as an enhancement.

# Test Methodology

## 2.1 Purpose

The below list is not intended to limit the extent of the test plan and can be modified to become suitable for the particular project. The purpose of the Test Plan is to achieve the following:

* Define testing strategies for each area and sub-area to include all the functional and quality (nonfunctional) requirements.
* Define bug-tracking procedures.
* Identify testing risks.
* Identify required resources and related information.
* Provide testing schedule.

**Unit Testing**

Unit Testing is conducted by the Developer during the code development process to ensure that proper functionality and code coverage have been achieved by each developer both during coding and in preparation for acceptance into iterations testing.

**Integration Testing**

The project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

| **Test Case ID** | **Test Case Objective** | **Test Case Description** | **Expected Result** |
| --- | --- | --- | --- |
| **1** | Sending mail implementation | Fill the fields and Enter mail address and click on “Save” button and check the mail | All form details what is filled should be sent to the mail address |

**Sanity Testing**

It is a type of testing to prove that software application is working as per requirements as mentioned in the specified documents. It checks whether the application is built according to user needs. It is used to check after making minor changes to a part of the code; the application is still working as per the requirements of the user.

| **Test Case ID** | **Test Case Objective** | **Test Case Description** | **Expected Result** |
| --- | --- | --- | --- |
| **1** | Send all details as mail | Fill the fields and Enter mail address and click on “Save” button and check the mail | All form details what is filled should be sent to the mail address |
| **2** | Filter year | Select year option on filter and check the years | Only selected year records should be displayed |

**Smoke Testing**

Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities.

It is a mini and rapid regression test of major functionality. It is a simple test that shows the product is ready for testing.

| **Test Case ID** | **Test Case Objective** | **Test Case Description** | **Expected Result** |
| --- | --- | --- | --- |
| **1** | Fill and save the form | Fill the fields and click on “Save” button | Details in form should be saved |
| **2** | Sort action | Sort by name ASC | Should sort by name ASC |

**Regression Testing**

This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done. Regression testing is needed, when a new feature is added to the software application and for defect fixing as well as performance issue fixing.

| **Test Case ID** | **Test Case Objective** | **Test Case Description** | **Expected Result** |
| --- | --- | --- | --- |
| **1** | Fill and save the form | Fill the fields and click on “Save” button | Details in form should be saved |
| **2** | Sort action | Sort by name ASC | Should sort by name ASC |
| **3** | Filter year | Select year option on filter and check the years | Only selected year records should be displayed |

**End-to-End Testing**

The purpose of end-to-end testing is testing whole software for dependencies, data integrity and communication with other systems, interfaces and databases to exercise complete production like scenario.

| **Test Case ID** | **Test Case Objective** | **Test Case Description** | **Expected Result** |
| --- | --- | --- | --- |
| **1** | Fill the fields valid and invalid format | Fill all fields and delete email, see validation, and fill the email without “@” and click on “Save”, and see the error message, then fill valid format email and click on “Save” and see success message | Should see validation and messages and save |

**Performance Testing**

It is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under particular workload.

| **Test Case ID** | **Test Case Objective** |
| --- | --- |
| **1** | Verify response time is not more than 4 secs when 1000 records added simultaneously. |
| **2** | Verify response time of the Application Under Load is within an acceptable range when the network connectivity is slow |
| **3** | Check the maximum number of saved records that the application can handle before it crashes. |
| **4** | Check database execution time when 500 records are read/written simultaneously. |
| **5** | Verify response time of the application under low, normal, moderate, and heavy load conditions. |

**Usability Testing**

Mainly focuses on user’s ease of using application, flexibility of application to handle controls and ability of application to meet its objectives. Recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

**Acceptance Testing**

Manual acceptance testing is done once the features have been implemented to verify that they are working as expected.

## 2.2 Test Approach

The project is using an agile approach, with weekly iterations. At the end of each week the requirements identified for that iteration will be delivered to the team and will be tested.

Automated unit tests are part of the development process. End-to-End tests, smoke tests are planned in automation.

## 2.3 Bug Fix Tests

Bug fix will be a central tenant throughout all testing phases.

All bugs that are resolved as “Fixed, Needs Re-Testing” will be tested when the Testing team is notified of the new drop containing the fixes. When a bug passes tests, it will be considered “Closed, Fixed”. If a bug fails test, the testing team will notify the development team by entering notes into Jira. When a Severity 1 bug fails test, the Testing team should also put out an immediate inform to development team. The Test Lead will be responsible for tracking and reporting to development and product management the status of testing.

## 2.4 Test Completeness

Testing will be considered complete when the following conditions have been met:

**Standard Conditions**

1. When Testers and Developers, agree that testing is complete, the app is stable, and agree that the application meets functional requirements.
2. Script execution of all test cases in all areas has passed.
3. Automated test cases have in all areas have passed.
4. All priority bugs have been resolved and closed.
5. 50% of all resolved severity 1 and 2 bugs have been successfully re-regressed as final validation.
6. Ad hoc testing in all areas has been completed.

**Bug Reporting**

The testing team recognizes that the bug reporting process is a critical communication tool within the testing process. Without effective communication of bug information and other issues, the development and release process will be negatively impacted. The Test Lead will be responsible for managing the bug reporting process. Testing’s standard bug reporting tools and processes will be used. Jira is the company-wide standard Bug Logging / Tracking tool. Bug description, attachments, environment, data and priority will be indicated in the bug task.

# Test Deliverables

## Test Schedule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Start** | **Finish** | **Effort** | **Comments** |
| Test Planning | 08.04.2022 | 09.04.2022 |  |  |
| Review Requirements documents |  |  | 2 d |  |
| Create initial test estimates |  |  | 1 d |  |
| Staff and train new test resources |  |  |  |  |
| First deploy to QA test environment |  |  |  |  |
| Functional testing – Iteration 1 |  |  |  |  |
| Iteration 2 deploy to QA test environment |  |  |  |  |
| Functional testing – Iteration 2 |  |  |  |  |
| System testing |  |  |  |  |
| Regression testing |  |  |  |  |
| UAT |  |  |  |  |
| Resolution of final defects and final build testing |  |  |  |  |
| Deploy to Staging environment |  |  |  |  |
| Performance testing |  |  |  |  |
| Release to Production |  |  |  |  |

## Deliverables

|  |  |  |
| --- | --- | --- |
| **Deliverable** | **For** | **Date / Milestone** |
| Test Plan | Product Owner, QA Lead, QA Specialist |  |
| Traceability Matrix | Product Owner, QA Lead |  |
| Test Results | Product Owner |  |
| Test Status report | QA Lead |  |
| Metrics | All team members |  |

**Brella Form**

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version # | Date | Author | Changes Summary |
| 1.0 | 08/04/2022 | Sibel Ulufer | Initial test cases for Brella Form |
| 1.1 | 09/04/2022 | Sibel Ulufer | Updated test results |

|  |  |
| --- | --- |
| Number | Action |
| 1 | Fill and save the form |
| 2 | Sort action *(Hypothetical)* |
| 3 | Filter action *(Hypothetical)* |

**Test cases**

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Action | Expected result | Test result |
| **1** | **Fill and save the form** | | |
| **1.1** | **Fill all fields and save, as happy path** | | |
| 1.1.1 | Steps:  Land on Brella Form page  Fill name field  Activate the switch  Fill valid format email  Fill input number within a range  Select the year  Click on “Save” button | All form details what is filled should be saved and displayed in the box | **OK** |
| **1.2** | **Check all validations** | | |
| 1.2.1 | Steps:  Land on Brella Form page  Fill invalid format email  Fill invalid number  See number validation  Click on “Save” button  See pop-up error message  See email validation | All validations should be seen | **OK** |
| **1.3** | **Should work on mobile view for multiple saving** | | |
| 1.3.1 | Steps:  Land on Brella Form page in mobile view  Fill all fields and save  Fill all fields and save  Fill all fields and save | All form details what is filled should be saved and displayed in the box | **OK** |
| **1.4** | **Should not save without email** | | |
| 1.4.1 | Steps:  Land on Brella Form page  Fill name field  Fill input number within a range  Select the year  Can’t click on “Save” button | “Save” button should be displayed as disable | **OK** |
| **1.5** | **Should work with mixed order** | | |
| 1.5.1 | Steps:  Land on Brella Form page  Fill valid format email  Select the year  Fill name field  Click on “Save” button | All form details what is filled should be saved and displayed in the box | **OK** |

# Resource Needs

## Tools

Unit testing is done in Jest.

End-to-end tests are managed with Cypress.io.

Automated test results are created with mochawesome.html report.

Documentation is managed with Confluence.

## Bug Severity and Priority Definition

Bug Severity and Priority fields are both very important for categorizing bugs and prioritizing if and when the bugs will be fixed. The bug Severity and Priority levels will be defined as outlined in the following tables below. Testing will assign a severity level to all bugs.

**Severity List**

The tester entering a bug into Jira is also responsible for entering the bug Severity.

|  |  |  |
| --- | --- | --- |
| **Severity ID** | **Severity Level** | **Severity Description** |
| 1 | Critical | The module/product crashes or the bug causes non-recoverable conditions. System crashes, or database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a Sev. 1 bug. |
| 2 | High | Major system component unusable due to failure or incorrect functionality. Sev. 2 bugs cause serious problems such as a lack of functionality, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. Sev. 2 bugs can have a work around, but the work around is inconvenient or difficult. |
| 3 | Medium | Incorrect functionality of component or process. There is a simple work around for the bug if it is Sev. 3. |
| 4 | Minor | Documentation errors or signed off severity 3 bugs. |

**Priority List**

|  |  |  |
| --- | --- | --- |
| **Priority ID** | **Priority Level** | **Priority Description** |
| 5 | Must Fix | This bug must be fixed immediately; the product cannot ship with this bug. |
| 4 | Should Fix | These are important problems that should be fixed as soon as possible. It would be an embarrassment to the company if this bug shipped. |
| 3 | Fix When Have Time | The problem should be fixed within the time available. If the bug does not delay shipping date, then fix it. |
| 2 | Low Priority | It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed. |
| 1 | Trivial | Enhancements/ Good to have features incorporated- just are out of the current scope. |