Test Plan *"iBench"*

Document created by:
Andrew Usachev

Project Manager:
Sergey Efremov

Scram Master:
None

1. Introduction

Customer needs a perfect website, which has passed the full cycle of manual and automation testing. Given the specificity of the site it is very important to have the same quality of the site.

The Test Plan has been created to facilitate communication within the team members. This document describes approaches and methodologies that will apply to the unit, integration and system testing of the www.ibench.net. It includes the objectives, test responsibilities, entry and exit criteria, scope, schedule major milestones, entry and exit criteria and approach. This document has clearly identified what the test deliverables will be, and what is deemed in and out of scope.

2. Scope

The document targets End-to-End testing including the GUI testing, API, Database, security and performance testing of client's data in report output as it was previously discussed at the meeting .

2.1 Functions to be tested:

- GUI
- Search and Filters Logic
- Performance
- API Server response
- Data response of the backend

2.2 Functions not to be tested.

2.3. Not other than mentioned above in section 2.1.

3. Quality objectives

3.1 Primary Objectives

A primary objective of testing is to: assure that the system meets the full requirements, including quality requirements (functional and non-functional requirements) and fit metrics for each quality requirement and satisfies the use case scenarios and maintain 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.

3.2 Secondary Objectives

The secondary objectives of testing will be to: identify and expose all issues and associated risks, communicate all known issues to the project team, and ensure that all issues are addressed in an appropriate manner. As an objective, this requires careful and methodical testing of the application to first ensure all areas of the system are scrutinized and, consequently, all issues (bugs) found are dealt with appropriately.

4. Test approach

The approach that is used is Analytical therefore, in accordance with requirements-based strategy, where an analysis of the requirements specification forms the basis for planning, estimating and designing tests. Test cases will be created during exploratory testing even most of them already predetermined in Test Strategy.

Team also must use experience-based testing and error guessing to utilize testers' skills and intuition, along with their experience with similar applications or technologies.

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.

4.1 Test Automation

Automated unit tests are part of the development process, and UI smoke-tests must be also automated during which performance data must be captured

5. Roles and responsibilities

Role	Staff Member	Responsibilities
Project Manager, Scram Master	Sergey Efremov	1. Acts as a primary contact for development and QA team. 2. Responsible for Project schedule and the overall success of the project.

		3.Setting new tasks and reviewing bugs	
QA	Andrew Usachev	1	Understand requirements.
		2	Writing documentation (Test plans)
		3	Writing and executing Test cases.
		4	Preparing RTM.
		5	Reviewing Test cases, RTM.
		6	Defect reporting and tracking.
		7	Retesting and regression testing.
		8	Bug Review meeting.
		9	Preparation of Test Data.
		10	Coordinate with QA Lead for any issues or problems encountered during test preparation/execution/defect handling

6. Entry and exit criteria

6.1 Entry Criteria

- All test hardware platforms must have been successfully installed, configured, and functioning properly.
- All the necessary documentation, design, and requirements information should be available that will allow testers to operate the system and judge the correct behavior.
- All the standard software tools including the testing tools must have been successfully installed and functioning properly.
- Proper test data is available.
- The test environment such as lab, hardware, software, and system administration support should be ready.
- QA resources have completely understood the requirements.
- QA resources have shown knowledge of functionality.
- Reviewed test scenarios, test cases and RTM.

6.2 Exit Criteria

- A certain level of requirements coverage has been achieved.
- No high priority or severe bugs are left outstanding.
- All high-risk areas have been fully tested, with only minor residual risks left outstanding.
- The schedule has been achieved.

7. SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

7.1 Suspension criteria

- The build contains many serious defects which can seriously limit testing progress.
- Significant change in requirements suggested by client.
- Software/Hardware problems.
- Assigned resources are not available when needed by the test team.

7.2 Resumption criteria

Resumption will only occur when the problem(s) that caused the suspension have been resolved.

8. Test strategy

8.1 QA role in test process:

- Understanding Requirements.
- Requirement specifications will be sent by the client.
- Understanding of requirements will be done by QA.

- Preparing Test Cases:

QA will be preparing End-to-End test cases based on the exploratory testing. This will cover all scenarios for requirements.

- Preparing Test Matrix:

QA will be preparing a test matrix which maps test cases to respective requirements. This will ensure the coverage for requirements.

- Reviewing test cases and matrix:

- Peer review will be conducted for test cases and test matrix by QA Lead.
- Any comments or suggestions on test cases and test coverage will be provided by the reviewer respective Author of Test Case and Test Matrix.
- Suggestions or improvements will be re-worked by the author and will be sent for approval.
- Re-worked improvements will be reviewed and approved by the reviewer.

- Creating Test Data:

Test data will be created by respective QA on client's developments/test site based on scenarios and Test cases.

- Executing Test Cases:

- Test cases will be executed by respective QA on the client's development/test site based on designed scenarios, test cases and Test data.

- Test result (Actual Result, Pass/Fail) will be updated in test case document Defect Logging and Reporting: QA will be logging the defect/bugs in JIRA and Google Sheets found during execution of test cases. After this, QA will inform the respective developer about the defect/bugs.

- Retesting and Regression Testing:

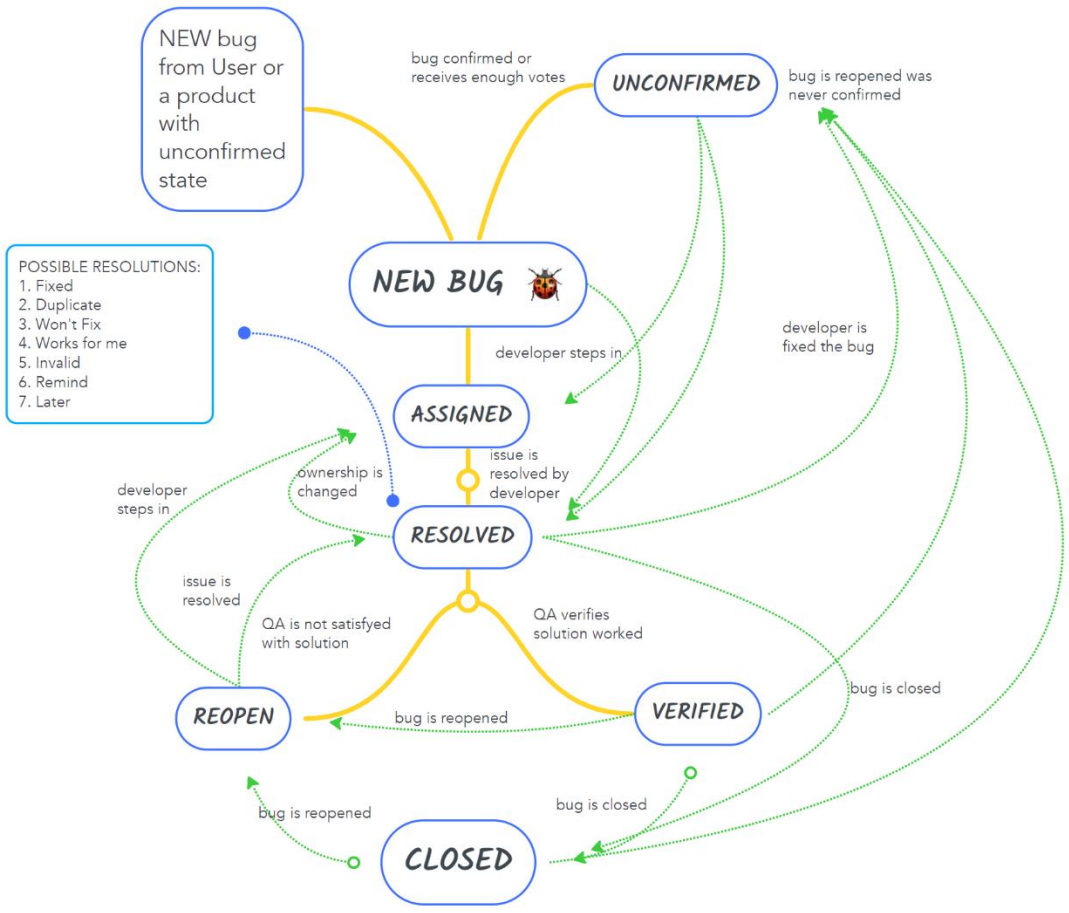
Retesting for fixed bugs will be done by respective QA once it is resolved by the respective developer and bug/defect status will be updated accordingly. In certain cases, regression testing will be done if required.

- Deployment/Delivery:

- Once all bugs/defects reported after complete testing are fixed and no other bugs are found, the report will be deployed to the client's test site or other requested by customer method.
- Once a round of testing will be done by QA on the client's test site if required Report will be delivered along with sample output by email to the respective lead and Report group.

8.2 Bug life cycle:

All the issues found while testing will be logged into JIRA.



8.3 Testing types

Black box testing:
It is sometimes called behavioral testing or Partition testing. This kind of testing focuses on the functional requirements of the software. It enables one to derive sets of input conditions that will fully exercise all functional requirements for a program.

GUI Testing:
GUI testing will include testing the UI part of the report. It covers users Report format, look and feel, error messages, spelling mistakes, GUI guideline violations.

Exploratory testing :
Exploratory testing will include a type of software testing where Test cases are not created in advance but QA check system on the fly. QA may note down ideas about what to test before test execution.

ADHOC testing:
ADHOC testing will include an informal testing type with an aim to break the system.

Positive testing:
Positive testing will include the type of testing that can be performed on the system by providing the valid data as input. It checks whether an application behaves as expected with positive inputs.

Negative testing:
Negative testing includes is a method of testing an application or system that ensures that the plot of the application is according to the requirements and can handle the unwanted input and user behavior. Invalid data is inserted to compare the output against the given input. Negative testing is also known as failure testing or error path testing. When performing negative testing exceptions are expected.

Functional Testing:
Functional testing is carried out in order to find out unexpected behavior of the report. The characteristics of functional testing are to provide correctness, reliability, testability and accuracy of the report output/data.

Boundary testing:
testing technique that focuses on testing the boundaries or limits of input values. It aims to uncover any issues or vulnerabilities that may occur at the edges of the input domain, such as minimum and maximum values, as well as values near the boundaries.

Performance API Testing:

- Check the optimal time the page is loaded
- Check form the optimal server response size
- Check for server code response
- Check for Header correct fields and values response
- Check validity of the response body
- Check validity of the body format
- Check equivalence of data stored in Environment and response body
- Set up monitor for consistent testing

Database Testing:

- Check if database exists
- Check correctness of the data in database
- Confirm data new data can be added and deleted
- Confirm that 100 users can be added simultaneously without problem
- Confirm Data can be filtered and slice with proper requested results
- Confirm Data can be join with other database for info extraction

8.4 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. The Test Lead will be responsible to see that a correct severity level is assigned to each bug.

The QA Lead, Development Lead and Project Manager will participate in bug review meetings to assign the priority of all currently active bugs. This meeting will be known as "Bug Triage Meetings". The QA Lead is responsible for setting up these meetings on a routine basis to address the current set of new and existing but unresolved bugs.

Severity List

Severity ID	Severity	Severity Description
1	Highest	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 Severity 1 bug.
2	High	Major system component unusable due to failure or incorrect functionality. Severity 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. Severity 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 Severity 3.
4	Low	Documentation errors or signed off Severity 3 bugs.

Priority List

Priority	Priority Level	Priority Description
1	Highest	This bug must be fixed immediately; the product cannot ship with this bug.
2	High	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	Medium	The problem should be fixed within the time available. If the bug does not delay the shipping date, then fix it.
4	Low	It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed. Enhancements/ Good to have features incorporated- just are out of the current scope.
5	Lowest	Documentation errors or signed off Low 4 bugs.

9. RESOURCE AND ENVIRONMENT NEEDS

9.1 Testing Tools

Process	Tool
Test case creation	Microsoft Word, JIRA, Google Sheets
Test case tracking	JIRA, Confluence
Test case execution	Manual, Selenium WebDriver, Visual Studio(SQL), Workbench, Mozilla Observatory, Lighthouse, GTMetrix, https://www.immuniweb.com, Postman, BrowserStock

Test case management	Google Sheets, JIRA, Confluence
Defect management	Microsoft Word, JIRA, Confluence, Swager
Test reporting	JIRA
Check list creating	Goggle Sheets, JIRA

9.2 Test Environment (browsers):

- Windows 10 : Edge(113.0.1774.50), Chrome (113.0.5672. 126), Firefox (113.0.1)
- Mac OS : Chrome (113.0.5672. 126), Firefox (113.0.1), Safari (16.4)

TEST SCHEDULE

Task Name	**Start**	**Finish**	**Effort**	**Comments**
Create test basis	02.09.2022		With team	flex
Test Planning	02.13.2022	02.13.2022	team	documentation
Black box testing	02.14.2022	02.16.2022	team	
Exploratory testing	02.14.2022	02.16.2022	team	
Boundary testing	02.14.2022	02.28.2022	team	
ADHOC testing	02.28.2022	03.12.2022	team	
Positive testing	02.14.2022	02.28.2022	team	
Negative testing	02.28.2022	02.12.2022	team	
Performance API testing	03.12.2022	03.26.2022	team	
Web Performance Testing	02.19.2022	02.23.2022	team	
Database Testing	03.24.2022	03.30.2022	team	
Security Testing (Web)	02.19.2022	02.23.2022	team	
Release to review	03.31.2022			

APPROVALS:

	Project Manager	Scram Master	Evaluation Manager
Name	Sergey Efremov	None	
Signature			

TERMS/ACRONYMS

The below terms are used as examples, please add/remove any terms relevant to the document.

TERM/ACRONYM	DEFINITION
API	Application Program Interface
GUI	Graphical user interface
PM	Project manager
UAT	User acceptance testing
CM	Configuration Management
QA	Quality Assurance
RTM	Requirements Traceability Matrix