**Table of Contents**

1 INTRODUCTION

2 SCOPE

3 QUALITY OBJECTIVES

3.1 Primary Objectives

3.2 Secondary Objectives

4 TEST APPROACH

4.1 Test Automation

5 ROLES AND RESPONSIBILITIES

6 ENTRY AND EXIT CRITERIA

6.1Entry Criteria

6.2 Exit Criteria

7 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

7.1 Suspension criteria

7.2 Resumption criteria

8 TEST STRATEGY

8.1 QA role in test process

8.2 Bug life cycle

8.3 Testing types

8.4 Bug Severity and Priority Definition

8.5 Severity List

8.6 Priority List

RESOURCE AND ENVIRONMENT NEEDS

9.1 Testing Tools

9.2 Configuration Management

9.3 Test Environment

10 TEST SCHEDULE

TERMS/ACRONYMS

**Test Plan**

**Front-End Automated Testing Application**

Document Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | Version | Description | Author | Reviewer | Approver |
| 12/01 | 2.0 | First Version of Media Organiser | Aaron Foster-Byrne | Aaron Foster-Byrne | Scott |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**1 INTRODUCTION**

The Test Plan has been created to communicate the test approach to team members. It includes the objectives, scope, schedule, risks and approach. This document will clearly identify what the test deliverables will be and what is deemed in and out of scope.

**2 SCOPE**

The initial Review will include all ‘must have’ requirements. These and any other requirements that get included must all be tested. At the end of Version 1, a tester must be able to:

1. Be able to access the website status information for MyTriathlon.
2. Be able to access past website status failures.

**3 QUALITY OBJECTIVES**

**3.1 Primary Objectives**

The First Version of the front-end automated testing application is used to allows MyTriathlon to see whether the official MyTriathlon website is operational. This software was written using HTML, SCSS, Javascript, SQL and PHP. The test team is responsible for testing the product and making sure it meets MyTriathlons requirements.

**4 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.

Exploratory testing will play a large part of the testing as Aaron has never used this type of tool and will be learning as he goes.

**4.1 Test Automation**

Test automation will be used through casperjs on the software application. CasperJS is a javascript automated testing framework.

**5 ROLES AND RESPONSIBILITIES**

|  |  |  |
| --- | --- | --- |
| Role | Staff Member | Responsibilities |
| Software Developer | Aaron Foster-Byrne | 1. Testing the Front-End Automated Testing Application.  2. Documenting the test plan, test case and test results. |
| Software Developer Manager | Scott Rigby-Mcgurk | 1.Approve of the test plan.  2. Approve the test case  3. Approve test results |

**6 ENTRY AND EXIT CRITERIA**

**6.1 Entry Criteria**

ξ All test hardware platforms must have been successfully installed, configured, and functioning properly.

ξ All the necessary documentation, design, and requirements information should be available that will allow testers to operate the system and judge the correct behavior. ξ All the standard software tools including the testing tools must have been successfully installed and functioning properly.

ξ Proper test data is available.

**6.2 Exit Criteria**

ξ A certain level of requirements coverage has been achieved.

ξ No high priority or severe bugs are left outstanding.

ξ All high-risk areas have been fully tested, with only minor residual risks left outstanding.

**7 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS 7.1 Suspension criteria**

• The build contains many serious defects which seriously or limit testing progress.

• Assigned resources are not available when needed by test team.

**7.2 Resumption criteria**

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

**8 TEST STRATEGY**

- Understanding Requirements:

• Requirement specifications will be given by MyTriathlon.

• Understanding of requirements will be done by Aaron Foster-Byrne.

- Preparing Test Cases:

Aaron will be preparing test cases based on the exploratory testing. This will cover all scenarios for requirements.

- Reviewing test cases:

Any comments or suggestions on test cases and test coverage will be

provided by reviewer respective Author of Test Case.

• Suggestions or improvements will be re-worked by author and will be

send for approval

• Re-worked improvements will be reviewed and approved by reviewer

Creating Test Data:

Test data will be created by Aaron on the Media Organiser based on scenarios and Test cases.

- Executing Test Cases:

• Test cases will be executed by Aaron on the Media Organiser based on designed scenarios, test cases and Test data.

• Test result (Actual Result, Pass/Fail) will be updated in the test case document Defect Logging and Reporting:

Aaron will be logging the defect/bugs in the Word document, found during the execution of test cases. After this, Aaron will inform the software development team about the defect/bugs.

- Retesting and Regression Testing:

Retesting for fixed bugs will be done by Aaron 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/defect reported after complete testing is fixed and no other bugs are found, report will be deployed to Whizzy Software by Aaron. .

**8.2 Bug life cycle:**

All the issues found while testing will be logged into the Word document.

**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.

Integration Testing:

Integration testing is a systematic technique for constructing the program structure while

conducting tests to uncover errors associated with interacting. In Report, integration

testing includes the testing Report from respective location(s).

Functional Testing:

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

System Testing:

System testing of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

Performance Testing:

- Check the optimal time the page is loaded

- Check the operation of the system under load

User acceptance testing:

The purpose behind user acceptance testing is to confirm that the system is developed according to the specified user requirements and is ready for operational use. Acceptance testing is carried out at two levels - Alpha and Beta Testing. User acceptance testing (UAT) will be done at the Client.

Alpha testing:

The alpha test is conducted at the developer's site by the client.

**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. Aaron will be responsible to see that a correct severity level is assigned to each bug.

Scott will participate in bug review meetings to assign the priority of all currently active bugs. Scott 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**

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

|  |  |  |
| --- | --- | --- |
| **Severity ID Severity** |  | **Severity Description** |
| 1 | Critical  **Level** | The module/product crashes or the bug causes non recoverable conditions. System crashes, GP Faults, 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** |
| 1 | Must Fix | This bug must be fixed immediately; the product cannot ship with this bug. |
| 2 | 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. |
| 4 | Low Priority | 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. |

11

**9 RESOURCE AND ENVIRONMENT NEEDS**

**9.1 Testing Tools**

|  |  |
| --- | --- |
| Process | Tool |
| Test case creation | Microsoft Word |
| Test case tracking | Microsoft Word |
| Test case execution | Manual |
| Test case management | Microsoft Word |
| Defect management | Microsoft Word |
| Test reporting | Microsoft Word |

**9.2 Configuration Management**

ξ Code CM: Git

**9.3 Test Environment**

ξ Support level 1 (browsers):

ξ Windows 8: Edge, Chrome (latest), Firefox (latest), Safari (latest)

ξ Mac OS X: Chrome (latest), Firefox (latest), Safari (latest)

12 ξ Linux Ubuntu: Chrome (latest), Firefox (latest) ξ Support level 1 (devices):

ξ iPhone 5 / 6, iPad 3, Nokia Lumia 910, Google Nexus 7, LG G3.

ξ Support level 2:

ξ Windows 7: IE 9+, Chrome (latest), Firefox (latest), Safari (latest)

ξ Windows XP: IE 8, Chrome (latest), Firefox (latest), Safari (latest)

ξ Support level 3:

ξ anything else

**10 TEST SCHEDULE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Start** | **Finish** | **Effort** | **Comments** |
| Test Planning | 10:00 | 12:00 | 1 |  |
| Review Requirements documents | 12:00 | 1:00 | 1 |  |
| Create test Case | 1:00 | 2:00 | 1 |  |
| System testing | 2:00 | 4:00 | 1 |  |
| Test report | 4:00 | 5:00 | 1 |  |

**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 |