Spotted Zebra SDET Assignment API Test Strategy

Table of Contents

[Revision History 3](#_Toc138968390)

[Definitions and Terminology 4](#_Toc138968391)

[Test Process Overview 5](#_Toc138968392)

[Issue Levels 5](#_Toc138968393)

[Issue Levels 6](#_Toc138968394)

[Test Report 6](#_Toc138968395)

[Test Conditions 6](#_Toc138968396)

[Role Endpoint 6](#_Toc138968397)

[Skills Endpoint 6](#_Toc138968398)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Description** | **Author** | **Date** |
| **0.1** | Initial version | J.Adams | 30/06/2023 |

# Definitions and Terminology

|  |  |
| --- | --- |
| **Name** | **Description** |
| SDET | Software Development Engineer in Test |
| API | Application Programming Interface |

# Test Process Overview

This exercise cannot cover the entirety of a development process, so this section can only define the process which will be followed for test development.

With limited access to information (the only access being the document on the GraphQL playground), a large percentage of test development will be exploratory, construction queries and seeing how the API responds. From these exploratory tests, more formal automation tests will be developed, and these will be in the associated test harness.

# Issue Levels

(These levels are for information only as each product, and indeed each issue tracking system, has their own definition of how severe an issue is.)

|  |  |
| --- | --- |
| **Defect Level** | **Description** |
| Blocker | A blocker issue represents the highest level of severity in software testing. It refers to a critical problem that completely blocks or prevents users from utilizing essential functionalities or renders the software unusable. Blocker issues can cause widespread system failures, data loss, or security breaches, severely impacting the software's intended purpose. These issues require immediate resolution and are given the highest priority to ensure the software can be used effectively without any major hindrance. |
| Critical | A critical issue denotes a severe problem within the software that significantly impacts its functionality, stability, or security. These issues can lead to major system failures, data corruption, or security vulnerabilities that could compromise the software or the underlying system. Immediate attention and resolution are crucial to prevent further damage or disruption. Critical issues demand swift action to restore the software to a stable and secure state. |
| Normal | A normal issue signifies a moderate problem within the software that can potentially affect its functionality or user experience to a noticeable degree. These issues are not critical, but they require attention and resolution to ensure the software performs optimally. Normal issues may cause unexpected behavior, result in minor data loss, or impede certain features, and addressing them is important for delivering a stable and reliable software product. |
| Minor | A minor issue indicates a relatively small problem within the software that may cause some inconvenience or slightly affect the user experience. While these issues may not be critical, they are noticeable and can impact the smooth operation of certain features or functionalities. Although they do not pose a significant threat, addressing these minor issues can enhance the overall quality and user satisfaction with the software. |
| Trivial | A trivial issue refers to a minor defect or flaw in the software that has a negligible impact on its functionality or usability. These issues are typically cosmetic in nature and do not significantly affect the overall performance or user experience. They can be considered as minor annoyances or inconveniences that do not hinder the core functionality of the software. |

As a general rule, no software should be considered for release if there are Blocker or Critical issues which have not been resolved.

# Issue Levels

For this test, the issues will be presented in the README file which will be provided with this strategy.

# Test Report

The automated harness will produce a report which will fulfil this requirement. It is hoped to use Allure for the report (as this will also allow tests to be assigned severity levels as per the previous page.)

# Test Conditions

## Role Endpoint

|  |
| --- |
| **Test Condition** |
| The API should allow a role to be created |
| The API should NOT allow a role to be created if the role exists |
| The API should NOT allow a role to be created if there is no name |
| The API should allow a role to be searched for by name |
| The API should allow a role to be searched for by id |
| The API should allow a role to be deleted by id |
| The API should not allow a role to be deleted by id which does not exist |
| The API should allow a role to be updated |
| The API should NOT allow a role to be updated with an empty name |

## Skills Endpoint

|  |
| --- |
| **Test Condition** |
| The API should allow a skill to be created |
| The API should NOT allow a skill to be created which is already existing |
| The API should allow a skill to be deleted |
| The API should allow a deleted skill to be recreated |
| The API should allow a skill to be updated |
| The API should NOT allow a skill to be updated which does not exist |