Test Approach for Computer Entry System

Document Owner: Nick Hall
Document Name: BackBase Test Plan
Document Version: 1.0

Table of Contents

1. Introduction and purpose

This document aims to communicate the test approach of the BackBase Computer Entry system in terms of the following;

- Assumptions
- Define the testing to be performed
- Tools to be used
- Test Data

2. Assumptions

The following assumptions were are to be made during the testing

1. All text fields have a maximum of 250 alphanumerics

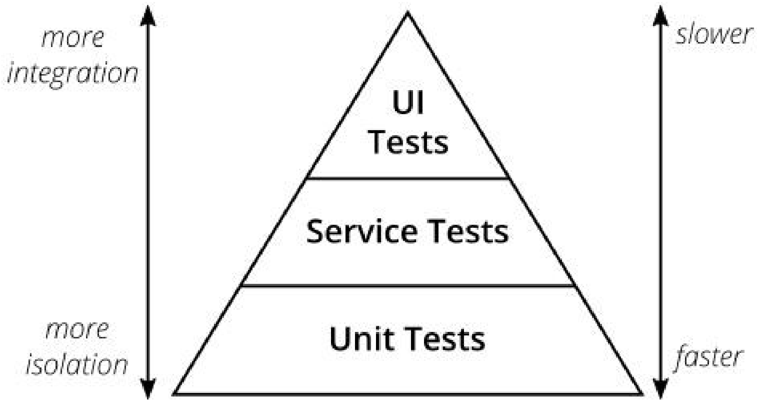
2. Duplicate entries are allowed

3. API testing completed or underway

4. There has been sufficient unit testing performed prior to commencement and will adhere to the Agile Testing Pyramid as detailed , much simplified below;

3. Testing to be performed

As mentioned in the assumptions, testing will be based on the Testing Pyramid which clearly defines, albeit simplistically, the stages of testing. It shows that Unit tests are both faster due to the lack of need for integration, which ultimately means they're more cost effective, the further up you progress the more integration is required, therefore the cost and time increase and taking longer to execute. In order to mitigate these issues, you can see that UI testing should be the least amount of testing performed.



The testing to be performed for the Service tests is a task for the Test Team but is out of scope for this document. The testing to be performed on the UI will be a mix of manual and automated user interface testing.

4. Tools to be used

The tools to be used to test the user interface will be Selenium Webdriver with Java, pico container, Maven and cucumber for writing gherkin syntax. Below is a simplified overview of the tools to be used.

4.1 Java

Is an open source software development language.

4.2 Selenium WebDriver

To complement Java, Selenium Webdriver is an open source collection of software api’s which are used to automate the testing of web applications.

4.3 Cucumber

Cucumber is a tool which compliments Selenium Webdriver which promotes clear communication between technical and non technical members of the engineering and product team. Cucumber is written with a syntax called Gherkin, and is used for behavioural driven development (BDD). Gherkin primarily consists of Given, When, Then, to bring clarity between the product and engineering when describing scenarios. Cucumber allows you to specify inputs using data within the gherkin sentences, alternatively, cucumber provides a powerful data-table mechanism, which brings transparency to the tests and data being used.

4.4 Pico Container

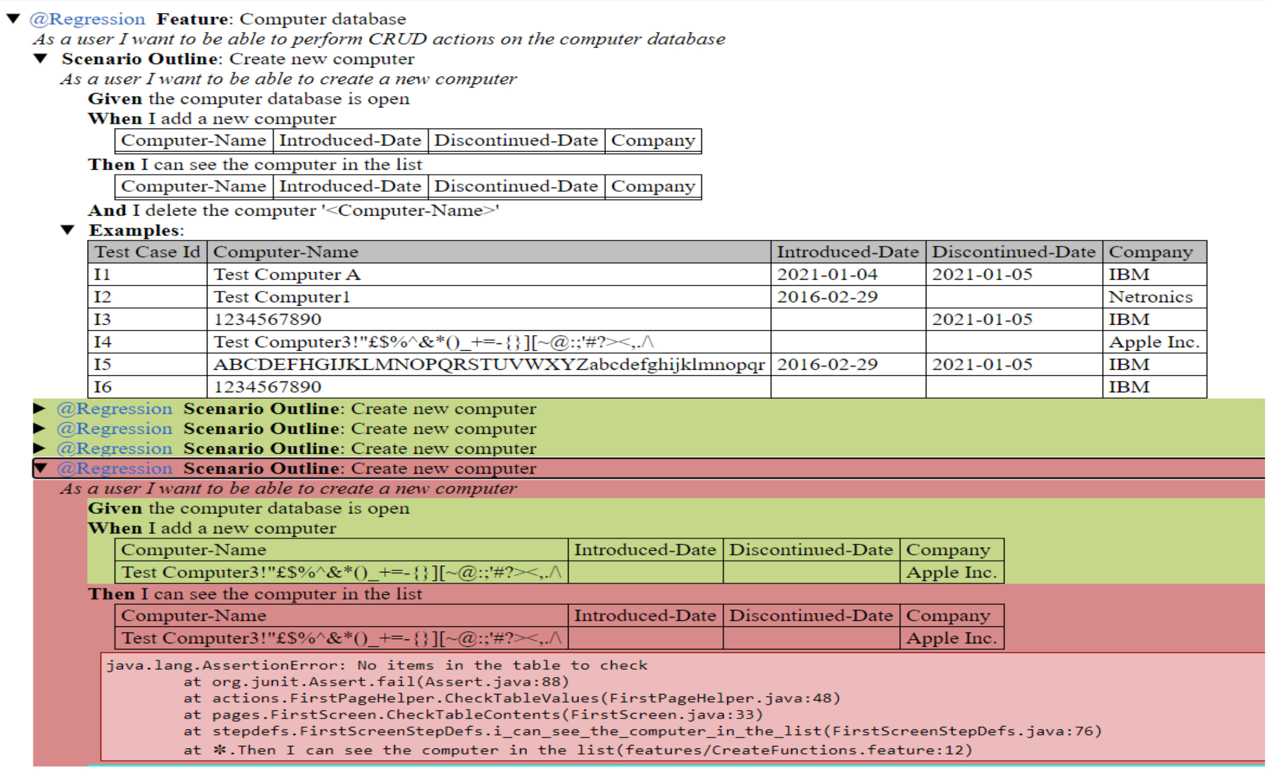
To complement Java, Selenium and Cucumber, the framework is utilising Pico container to leverage dependency injection allowing you to loosely couple classes meaning you can injection the webdriver instance into the required classes without the need of complicated code.

4.5 Maven

Maven is used for a complete lifecycle framework. Using Maven you are able to integrate this with the CI pipeline, allowing you to pass in runtime parameters. During the CI pipeline, Maven allows you to pass in parameters such that you can execute the test scripts against different environments, for example, Test, Dev or Pre prod environments. Additionally, you can pass in parameters that control which Browser the tests are to be executed against.

4.6 Surefire

This is a Maven plugin, which in the framework will be used to generate reports, as per the example below.



▼ @Regression **Feature**: Computer database

As a user I want to be able to perform CRUD actions on the computer database

▼ **Scenario Outline**: Create new computer

As a user I want to be able to create a new computer

Given the computer database is open

When I add a new computer

Computer-Name	Introduced-Date	Discontinued-Date	Company

Then I can see the computer in the list

Computer-Name	Introduced-Date	Discontinued-Date	Company

And I delete the computer '<Computer-Name>'

▼ **Examples**:

Test Case Id	Computer-Name	Introduced-Date	Discontinued-Date	Company
I1	Test Computer A	2021-01-04	2021-01-05	IBM
I2	Test Computer1	2016-02-29		Netronics
I3	1234567890		2021-01-05	IBM
I4	Test Computer3!"£$%^&*()_+=-{}][~@:;'#?><,./\			Apple Inc.
I5	ABCDEFHGIJKLMNOPQRSTUVWXYZabcdefghijklmnopqr	2016-02-29	2021-01-05	IBM
I6	1234567890			IBM

▶ @Regression **Scenario Outline**: Create new computer

▶ @Regression **Scenario Outline**: Create new computer

▶ @Regression **Scenario Outline**: Create new computer

▼ @Regression **Scenario Outline**: Create new computer

As a user I want to be able to create a new computer

Given the computer database is open

When I add a new computer

Computer-Name	Introduced-Date	Discontinued-Date	Company
Test Computer3!"£$%^&*()_+=-{}][~@:;'#?><,./\			Apple Inc.

Then I can see the computer in the list

Computer-Name	Introduced-Date	Discontinued-Date	Company
Test Computer3!"£$%^&*()_+=-{}][~@:;'#?><,./\			Apple Inc.

```
java.lang.AssertionError: No items in the table to check
	at org.junit.Assert.fail(Assert.java:88)
	at actions.FirstPageHelper.CheckTableValues(FirstPageHelper.java:48)
	at pages.FirstScreen.CheckTableContents(FirstScreen.java:33)
	at stepdefs.FirstScreenStepDefs.i_can_see_the_computer_in_the_list(FirstScreenStepDefs.java:76)
	at ✻.Then I can see the computer in the list(features/CreateFunctions.feature:12)
```

5. Test Data

The test data to be used will be the pre-existing data in the database. Any additional data will be specified in the gherkin tables.