SE 3XA3

REST Assured

December 6, 2017

Test Report

Team 31

Dawson Myers	myersd1	400005616
Yang Liu	liuy136	400038517
Brandon Roberts	roberb1	400018117

Contents

List of Tables

¡¡¡¡¡¡¡ HEAD ======= ¿¿¿¿¿¿¿ e50dce788878d8d0b510ed27237c61e0c32419e8

1 Revision History

Table 1: **Revision History**

Date	**Developer(s)**	**Change Revision**	
Nov 29, 2017	Dawson Myers, Yang Liu, Brandon Roberts	Revision 1 Draft	1
Dec 6, 2017	Dawson Myers, Yang Liu, Brandon Roberts	Revision 1 Completion	1

2 Project Drivers

3 General Information

3.1 Purpose

The purpose of this document is to outline the testing, validation, verification results that were carried out on the reconstruction of the Sails Live Chrome app, named the REST Assured Test Client. Through testing, the REST Assured team used rigorous unit testing as well as manual testing to improve the products correctness and build confidence. The testing helped provide proof that the project adheres to the requirements specified in the Software Requirements Specification document. Types of testing included structural, static and dynamic, functional and nonfunctional, manual and automated unit testing. Various testing tools were used to achieve these test results.

3.2 Naming Conventions and Terminology

Term	Definition
HTTP	Hypertext Transfer Protocol.
REST	Representational state transfer (REST) or RESTful web services is a way of providing interoperability between computer systems on the Internet.
JSON	JavaScript Object Notation. An open-standard file format that uses human-readable text to transmit data objects consisting of attribute-value pairs and array data types (or any other serializable value).
API	Application Program Interface. A document detailing the name of each function the client may call in their software and the purpose of those functions.
FR	Functional requirements that describes what the product will do.
User	A person who will be using the final product.
App	The application being designed; the system-to-be.

Table 2: **Table of Definitions**

3.3 Overview of Document

This document begins with a general overview of the application and tests, including sections on the software description, introduction of the test team and tools used for testing. Next, detailed system test results for functional and nonfunctional requirements, as well as unit tests. After that, the requirement and module traceability matrices are displayed

3.4 Software Description

The REST Assured Test Client provides software developers with a tool for web API building and testing. The application provides tests endpoints and the capability to diagnose bugs in applications featuring RESTful interfaces.

3.5 Test Team

The REST Assured team members that were responsible for all testing procedures are Dawson Myers, Brandon Roberts, and Yang Liu. These responsibilities included test writing and execution for various types of testing outlined in this document.

3.6 Automated Testing Approach

Automated testing for the REST Assured Test Client was done in using Jasmine.

3.7 Testing Tools

The majority of the project code is JavaScript front-end code. The following testing tools were used:

- PhantomJS (UI Testing)
- Jasmine (Unit Testing)

4 System Test Description

The software will allow users to test their REST servers responses to GET/POST/PUT/DELETE requests. It will be implemented with common front end languages (HTML, javascript, css) and libraries (react, jQuery, bootstrap).

4.1 Tests for Functional Requirements

4.1.1 User Input

FRT-UI-1

Type	Manual
Initial State	Request form has input data, and response form has response information
Input	clear button clicked
Output	Request form and response form are cleared, leaving no characters in field
Procedure	The function clearing the request form and response form will run, the tester will manually verify if both forms have been cleared
Result	Pass

FRT-UI-2

Type	Functional, Dynamic, Manual, Static etc.
Initial State	Input text fields empty
Input	clear button clicked
Output	Field remains cleared, no characters in field
Procedure	Manually perform test to verify if field has been cleared
Result	Pass

FRT-UI-3

Type	Manual
Initial State	Input text fields cleared by clear button
Input	HTTP POST/GET/DELETE/PUT requests to test url
Output	HTTP request returns output fitting to request criteria
Procedure	Manually perform test to verify whether field clearing action will interfere with HTTP request functionalities
Result	Pass

FRT-UI-4

Type	Manual
Initial State	The selected test stub is open
Input	The user clicks another test stub in the test selection menu
Output	The test stub viewer will update to display information about the newly selected test stub
Procedure	The test will manually be performed by a tester, and the program will pass the test if the wanted behaviour is reflected
Result	Pass

FRT-UI-5

Type	Manual
Initial State	Test stub view is displayed
Input	HTTP POST/GET/DELETE/PUT requests to test url
Output	Test stub will change colour to corresponding request colour in the test selection menu
Procedure	The test will manually be performed by a tester, the functions corresponding to the HTTP requests will be run, we check the response and the program will pass the test if the desired behaviour is reflected
Result	Pass

FRT-UI-6

Type	Manual
Initial State	Request form is awaiting input data
Input	User inputs request data into request form
Output	Program will format data for HTTP request with parameters, fit for browser entry
Procedure	The test will manually be performed by a tester, and the program will pass the test if the wanted behaviour is reflected
Result	Pass

FRT-UI-7

Type	Manual
Initial State	A valid request entry has been entered in the request form as input data
Input	The save request entry button is clicked
Output	The saved request entry is added to the list of saved entries and appears in the saved entry selection window
Procedure	The save request entry function will be called for the input request entry, the test will manually verify that the input request is added to the saved list of requests and appears in the saved entry selection window
Result	Pass

FRT-UI-8

Type	Manual
Initial State	Request form has been cleared of input data
Input	A previously saved request entry is selected, submit button is clicked
Output	Request form has been loaded with the selected previously saved request entry as input data
Procedure	The load saved request entry function will be called for the selected request entry, the test will manually verify that the request form has been populated with the selected request
Result	Pass

FRT-UI-9

Type	Manual
Initial State	The program is open
Input	The user clicks the new test button
Output	A new test stub is created underneath the lowest test stub
Procedure	The test will manually be performed by a tester, the functions corresponding to the HTTP requests will be run, we check the response and the program will pass the test if the desired behaviour is reflected
Result	Pass

FRT-UI-10

Type	Manual
Initial State	The program is openThe program is openThe program is open
Input	The user clicks and drags a test stub
Output	The test stub will follow the cursor users cursor until the let go by the user
Procedure	The test will manually be performed by a tester, the functions corresponding to the HTTP requests will be run, we check the response and the program will pass the test if the desired behaviour is reflected
Result	Pass

4.1.2 Protocol Tests

FRT-PT-1

Type	Functional
Initial State	At main window
Input	Properly formatted JSON
Output	Should return true
Procedure	How test will be performed: REST query string validator function is called with a JSON request object
Result	Pass

FRT-PT-2

Type	Functional
Initial State	At main window
Input	Improperly formatted JSON
Output	Should return false
Procedure	How test will be performed: REST query string validator function is called with a JSON request object
Result	Pass

4.1.3 HTTP Communications

FRT-CM-1

Type	Functional
Initial State	At main window
Input	JSON request object
Output	JSON response object containing the correct set of data from the resource URL
Procedure	Test is run that will call the sendMsg function with a JSON request object. The function should return a JSON object with a data set from the server. The data will be validated to verify it is correct
Result	Pass

4.2 Tests for Nonfunctional Requirements

4.2.1 Performance

NRT-P-1

Type	Functional
Initial State	At main window
Input	100,000 requests are enqueued
Output	JSON responses
Procedure	A test will add 100,000 request objects to the send message queue. The app should be able to process the responses without becoming unresponsive. The response text box should only store the previous 1000 rows of text
Result	Pass

NRT-P-2

Type	Functional
Initial State	At main window
Input	JSON request for a very large data set
Output	JSON response
Procedure	A test will run that will make a request for a very large data set. The app should not become unresponsive while processing the response
Result	Pass

5 Tests for Proof of Concept

5.1 User Input

PCT-UI-1	
Type	Functional
Initial State	Main window waiting for request information
Input	User inputs request information
Output	The program unfolds the request information into a JSON object
Procedure	The test will manually be performed by a test member, and the program will pass the test if the wanted behaviour is reflected
Result	Pass

PCT-UI-2	
Type	Functional
Initial State	Request form has input data, and response form has response information
Input	User clicks the clear button
Output	The Request form, and response form should be cleared of all information
Procedure	The test will manually be performed by a tester, and the program will pass the test if the wanted behaviour is reflected
Result	Pass

6 Comparison to Existing Implementation

The existing project had very few test cases. Thus, the team has had to develop tests from scratch.

7 Unit Testing

Jasmine was used for test unit testing internal functions.

UT-1	
Type	Automated
Module	JsonComparer
Suit	When comparing a JSON object with
Case	An Identical JSON object
Expectation	compare returns true compareExact returns true compareWithTolerance returns true
Result	Pass

UT-2	
Type	Automated
Module	JsonComparer
Suit	When comparing a JSON object with
Case	A JSON object with 1 difference
Expectation	compare returns false compareExact returns false compareWithTolerance(1) returns true
Result	Pass

UT-3	
Type	Automated
Module	JsonComparer
Suit	When comparing a JSON object with
Case	A JSON object with 2 difference
Expectation	compare returns false compareExact returns false compareWithTolerance(2) returns false
Result	Pass

UT-4

Type	Automated
Module	JsonComparer
Suit	When comparing a JSON object with
Case	A JSON object that is a subset
Expectation	compare returns true
	compareExact returns false
	compareWithTolerance(1) returns true
Result	Pass

UT-5

Type	Automated
Module	JsonParser
Suit	None
Case	When prettifying a JSON object
Expectation	it becomes a string
Result	Pass

UT-6

Type	Automated
Module	JsonParser
Suit	None
Case	When paramaterizing a JSON object
Expectation	It is accurate
Result	Pass

UT-7

Type	Automated
Module	RestStub
Suit	None
Case	When creating a RestStub
Expectation	It should be empty
	It should be a copy of data if entered
Result	Pass

UT-8

Type	Automated
Module	RestStub
Suit	None
Case	When setting the request type
Expectation	It should accept GET/PUSH/POST/DELETE
	It should not accept non GET/PUSH/POST/DELETE
Result	Pass

UT-9

Type	Automated
Module	RestChain
Suit	None
Case	When creating a RestChain
Expectation	It should be empty
	It should be a copy of data if entered
Result	Pass

UT-10

Type	Automated
Module	RestChain
Suit	None
Case	When modifying the list
Expectation	You can add tests to the chain
	You can remove tests from the chain
	You can change positions of tests in the chain
Result	Pass

UT-11

Type	Automated
Module	ProfileStore
Suit	None
Case	When creating a ProfileStore
Expectation	It should be empty
Result	Pass

UT-12

Type	Automated
Module	ProfileStore
Suit	None
Case	After creating a RestChain and RestStub
Expectation	The RestStub can be copied
Result	Pass

UT-13

Type	Automated
Module	ProfileStore
Suit	None
Case	After creating a RestChain and RestStub
Expectation	It can add RestStubs to the RestChain
Result	Pass

UT-14

Type	Automated
Module	ProfileStore
Suit	None
Case	After creating a RestChain and RestStub
Expectation	It can remove RestStubs from the RestChain
Result	Pass

UT-15

Type	Automated
Module	ProfileStore
Suit	None
Case	After creating a RestChain and RestStub
Expectation	The RestChain can be removed
Result	Pass

UT-16

Type	Automated
Module	ProfileStore
Suit	None
Case	After creating a RestChain and RestStub
Expectation	The RestChain can be copied
Result	Pass

8 Trace to Requirements

Test	Requirements
Functional Requirements Testing	
FRT-UI-1	FR2
FRT-UI-2	FR1, FR2
FRT-UI-3	FR3
FRT-UI-4	FR2, FR2
FRT-UI-5	FR2
FRT-UI-6	FR1
FRT-UI-7	FR3
FRT-UI-8	FR1, FR2
FRT-UI-9	FR1
FRT-UI-10	FR3
FRT-PT-1	FR3
FRT-PT-2	FR1, FR2, FR3
Non-functional Requirements Testing	
NRT-P-1	NFR1, NFR3
NRT-P-2	NFR7
PCT-UI-1	NFR2, NR9
PCT-UI-2	NFR2
Automated Testing	
UT-1	NFR2
UT-2	NFR4
UT-3	NFR9
UT-4	NFR1
UT-5	NFR2
UT-6	NFR6
UT-7	NFR9
UT-8	NFR8
UT-9	NFR8
UT-10	NFR6
UT-11	NFR7

Table 3: Trace Between Tests and Requirements

9 Trace to Modules

Test	Requirements
Functional Requirements Testing	
FRT-UI-1	M3
FRT-UI-2	M1
FRT-UI-3	M3, M4, M7, M8
FRT-UI-4	M7
FRT-UI-5	M2, M7
FRT-UI-6	M3, M8
FRT-UI-7	M2
FRT-UI-8	M6, M8
FRT-UI-9	M3
FRT-UI-10	M1
FRT-PT-1	M6
FRT-PT-2	M2, M3, M5, M7
Non-functional Requirements Testing	
NRT-P-1	M8
NRT-P-2	M2, M5, M8
PCT-UI-1	M3
PCT-UI-2	M5
Automated Testing	
UT-1	M3
UT-2	M4, M8
UT-3	M4, M7, M8
UT-4	M3, M5
UT-5	M2
UT-6	M2
UT-7	M1, M2
UT-8	M5, M8
UT-9	M2
UT-10	M1, M2, M3, M5, M8
UT-11	M5

Table 4: Trace Between Tests and Modules

10 Code Coverage Metrics

The test for *RestAssured* have approximately covered 90% of the project code.

None

11 Appendix

Additional information

11.1 Symbolic Parameters

Symbolic Parameters The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

Term	**Definition**
RESOURCE_ROOT_URL	https://jsonplaceholder.typicode.com
RESOURCE_POSTS	/posts
RESOURCE_COMMENTS	/comments

Table 5: **Table of Symbols**