Linnéuniversitetet
Kalmar Växjö

Test Report

Linnéuniversitetet
Kalmar Växjö

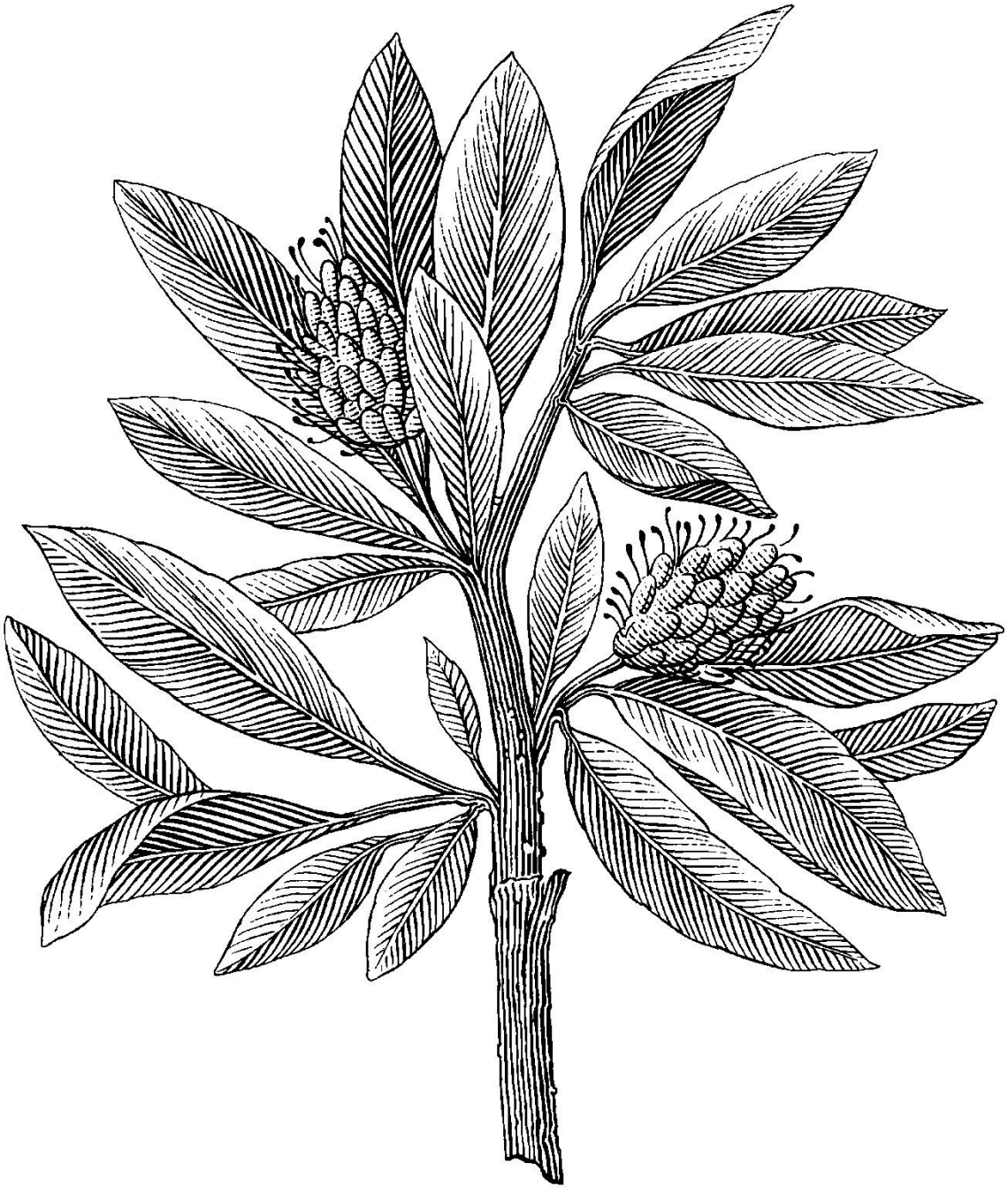

Table of Contents

Contents

1.Introduction

In this document, the results of manual tests and automation tests demonstrate. The manual tests result will first be displayed in a table, then in another table, traceability of test cases and requirements would be demonstrated. Finally, the reasons why some test cases failed would be discussed, then the results of automated tests would be displayed with their coverage.

2. Manual Test Cases Results

Test case number	Result	Possible reason
1.1.1	Passed	
1.1.2	Failed	No access log file exists
1.1.3	Passed	
1.1.4	Failed	The error provided is different from what expected
1.1.5	Passed	
1.1.6	Passed	
1.1.7	Passed	
1.1.8	Passed	
1.1.9	Passed	
1.1.10	Passed	
1.1.11	Passed	
1.1.12	Passed	
1.1.13	Passed	
1.1.14	Failed	Error message provided in the source code should be fixed as server can **only** start with 2 arguments not" one or two"
1.2.1	Passed	
1.2.2	Passed	
1.2.3	Passed	
1.2.4	Failed	No access log file exists
1.3.1	Passed	
1.3.2	Passed	
1.3.3	Passed	
1.3.4	Passed	
1.4.1	Failed	
1.6.1	passed	
1.8.1	Failed	No access log file exists

it can be seen that most of the test cases failed only because the "access log" file was missing, and it could be fixed only by providing a text file and record required logs there.

3. Traceability Matrix

Requirements		Req1	Req 1.1	Req 1.2	Req 1.3	Req 1.4	Req 1.5	Req 1.6	Req 1.7	Req 1.8	Req 2	Req 3	Req 4	Req 5	Req 6
Test Cases	Total	25	14	4	4	1	5	1	24	1					
TC1.1.1	3	☑	☑						☑						
TC1.1.2	3	☑	☑						☑						
TC1.1.3	3	☑	☑						☑						
TC1.1.4	3	☑	☑						☑						
TC1.1.5	3	☑	☑						☑						
TC1.1.6	3	☑	☑						☑						
TC1.1.7	4	☑	☑				☑		☑						
TC1.1.8	3	☑	☑						☑						
TC1.1.9	3	☑	☑						☑						
TC1.1.10	3	☑	☑						☑						
TC1.1.11	3	☑	☑						☑						
TC1.1.12	3	☑	☑						☑						
TC1.1.13	3	☑	☑						☑						
TC1.1.14	3	☑	☑						☑						
TC1.2.1	3	☑		☑					☑						
TC1.2.2	3	☑		☑					☑						
TC1.2.3	3	☑		☑					☑						
TC1.2.4	3	☑		☑					☑						
TC1.3.1	4	☑			☑		☑		☑						
TC1.3.2	4	☑			☑		☑		☑						
TC1.3.3	4	☑			☑		☑		☑						
TC1.3.4	4	☑			☑		☑		☑						
TC1.4.1	2	☑				☑									
TC1.6.1	3	☑						☑	☑						
TC1.8.1	3	☑							☑	☑					

As mentioned in testPlan.pdf, the focus of this iteration was on testing legacy requirements provided in MWS documents. That is why requirement 1, which consists of requirements 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, and 1.8 in MWS tested mainly in this iteration. Since all test cases run

in Windows 10, requirement 1.7 is checked in almost every test case. As mentioned in testPlan.pdf, requirements 2, 3, 4, 5, and 6 will be tested in the next iteration.

4.Explaining about TC1.1.6 and TC1.1.14

These two test cases are similar but testing two various aspects of the system. TC1.1.6 shows the server should not start with one argument, as it needs two arguments. Later, the test lead explored the source code and faced an error message, “This program should only have one or two arguments” for that reason, the tester in TC1.1.14 revealed that the error message needs to be changed. the server could only be started with exactly two arguments.

5.Requirement 1.5 The web server must follow minimum requirements for HTTP 1.1

After exploring source codes, the tester found out that the only method that can be used when using MWS is the GET method. In order to have minimum requirements for HTTP1.1, the other methods might be required (for instance, POST or PUT based on customers’ requirements) if it is supposed to be used by IoT developers as clients who need to send their requests through HTTP1.1 to the server.

6.Automation Tests Results

After running the automation tests available in legacy source code following results obtained

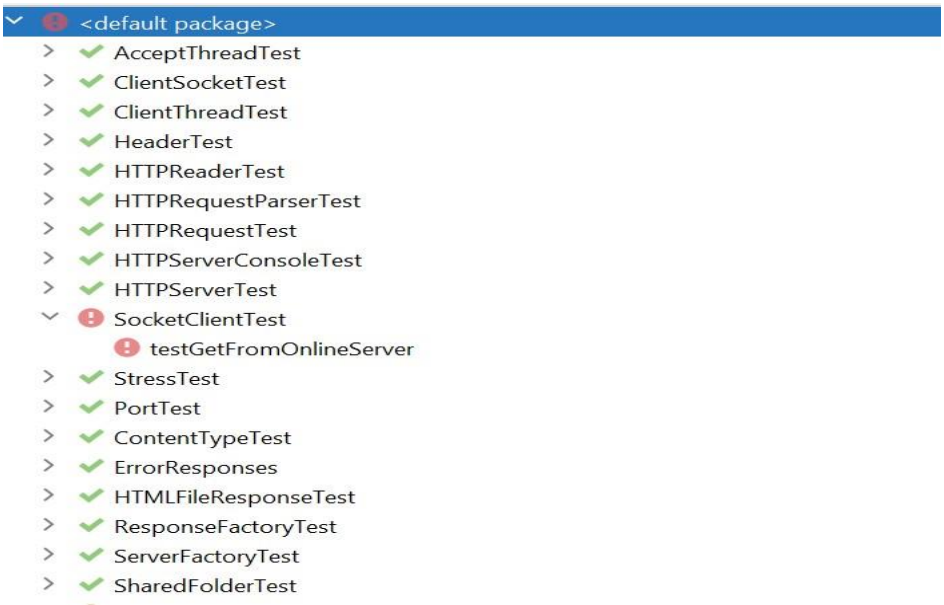

Figure 1 automated test result 1

Figure 2 automated test results 2

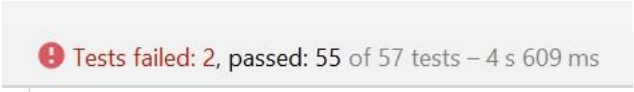

Figure 3 number of failed and passed tests

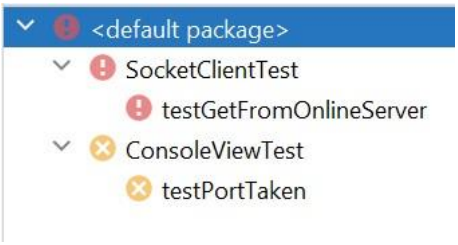

Figure 4 failed tests.

se	100% (31/31)	98% (85/86)	98% (384/391)

Figure 5 total test coverage

Element	Class, %	Method, %	Line, %
client	100% (2/2)	100% (3/3)	100% (22/22)
exceptions	100% (5/5)	100% (5/5)	100% (11/11)
response	100% (7/7)	100% (15/15)	100% (72/72)
view	100% (1/1)	100% (14/14)	100% (46/46)
AcceptThread	100% (1/1)	100% (4/4)	100% (20/20)
ClientFactory	100% (1/1)	100% (2/2)	100% (7/7)
ClientSocket	100% (1/1)	100% (6/6)	100% (24/24)
ClientThread	100% (1/1)	100% (4/4)	93% (27/29)
Header	100% (2/2)	100% (7/7)	100% (27/27)
HTTPReader	100% (1/1)	100% (3/3)	100% (21/21)
HTTPRequest	100% (2/2)	100% (7/7)	100% (19/19)
HTTPRequestParser	100% (1/1)	100% (1/1)	100% (11/11)
HTTPServer	100% (1/1)	100% (3/3)	100% (20/20)
HTTPServerConsole	100% (1/1)	66% (2/3)	77% (17/22)
Port	100% (1/1)	100% (2/2)	100% (6/6)
ResponseFactory	100% (1/1)	100% (3/3)	100% (15/15)
ServerFactory	100% (1/1)	100% (1/1)	100% (2/2)
SharedFolder	100% (1/1)	100% (3/3)	100% (17/17)

Figure 6 test coverage of classes and packages in detail

Figure 7 method which failed under test in ConsoleViewTest

Figure 8 method failed under test in SocketClientTest

As it can be seen in Figures 1 to 4, other than two tests, all provided automated tests passed successfully. The rate of passing tests is 96% which is beyond SDC management expectation. Below in figures 5 and 6, the test coverage could be seen, which shows, 100% of classes tested and 98% of method and lines covered, which is an acceptable rate for the SDC company. In figure 7, the reason that ConsoleViewTest failed is demonstrated. in the assertEquals method, the expected result was “Port is taken[]” while the actual result was “Port is taken[ ]” which can be fixed later.

After checking the source codes and see the comments provided by the original developer(s)/tester(s) of MWS, the reason why SocketClientTest was failed revealed. To pass, the test lead needs to have access to the internet, the client needs to send the request to an online server.