Test Report

My Web Server

1. Test traceability matrix

Requirement Identifiers	Reqs Tested	UC1	UC2	UC3	REQ4	REQ5	REQ6	REQ7	REQ8	REQ9	REQ10
Test Cases	30	12	4	7	1	1	2	1	1	1	
1.1	1	x									
1.2	1	x									
1.3	1	x									
1.4	1	x									
1.5	1	x									
1.6	1	x									
1.7	1	x									
1.8	1	x									
1.9	1	x									
1.10	1	x									
1.11	1	x									
2.1	1		x								
2.2	1		x								
2.3	1		x								
3.1	1			x							
3.2	1			x							
3.3	1			x							
3.4	1			x							
3.5	1			x							
3.6	1			x							
4.1	1				x						
5.1	1					x					
6.1	1						x				
7.1	1							x			
8.1	1								x		
9.1	1									x	
10.1	1										x
Exploratory		x	x	x							

2. Test result

2.1 Overview manual tests

Test	Requirement	Pass / Fail
TC 1.1 Succesfully start server	UC 1	Fail
TC 1.2 Start server with already taken port	UC 1	Pass
TC 1.3 Start server with to low port number	UC 1	Pass
TC 1.4 Start server with low port number	UC 1	Fail
TC 1.5 Start server with high port number	UC 1	Fail
TC 1.6 Start server with to high port number	UC 1	Pass
TC 1.7 Start server with folder with restricted acc	UC 1	Fail
TC 1.8 Start server with file as shared container	UC 1	Pass
TC 1.9 Start server with only valid port number	UC1	Pass
TC 1.10 Start server with just valid shared container	UC 1	Pass
TC 1.11 Start server without arguments	UC 1	Pass
TC 2.1 Successfully stop server	UC 2	Fail
TC 2.2 Try to stop server with wrong command	UC 2	Pass
TC 2.3 Stop the server with aborting in terminal	UC 2	Fail
TC 3.1 Request the home of the server	UC 3	Fail
TC 3.2 Request a existing file – HTTP 200	UC 3	Fail
TC 3.3 Request a non-existing file – HTTP 404	UC 3	Fail
TC 3.4 Request a file outside the shared container	UC 3	Fail
TC 3.5 Request a file with malformed headers – HTTP 400	UC 3	Fail
TC 3.6 Receive internal server response on request – HTTP 500	UC 3	Fail
TC 4.1 Simple deploying on a Raspberry Pi	REQ 4	Pass
TC 5.1 Compatibility with Raspberry Pi	REQ 5	Fail
TC 6.1 Web server should be able to handle 25 request per seconds	REQ 6	Pass
TC 7.1 Follow minimum requirements for HTTP 1.1	REQ 7	Fail
TC 8.1 Source code release under GPL-2.0	REQ 8	Fail
TC 9.1 Access log viewable from text editor	REQ 9	Fail
TC 10.1 Web server should be able to handle massive number of req	REQ 10	Fail

2.2 Overview automated tests

Automated test id	Test file	Test name	Pass / Fail
AUT1	AcceptThreadTest	testStopmeNotRunning	Pass
AUT2	AcceptThreadTest	testrun	Pass
AUT3	AcceptThreadTest	testsockedfailed	Pass
AUT4	ClientSocketTest	testWriteResponseBody	Pass
AUT5	ClientSocketTest	testWriteResponseHeader	Pass
AUT6	ClientSocketTest	testGetRequest	Pass
AUT7	ClientThreadTest	testMalformed	Pass
AUT8	ClientThreadTest	testRun	Pass
AUT9	ClientThreadTest	testMultipleConnections	Pass
AUT10	HTTPReaderTest	testbroken	Pass
AUT11	HTTPReaderTest	testbroken2	Pass
AUT12	HTTPReaderTest	testReadBody	Pass

AUT13	HTTPReaderTest	testReadkOk	Pass
AUT14	HTTPRequestParserTest	testMalformedRequest	Pass
AUT15	HTTPRequestParserTest	testMalformedRequestEmpty	Pass
AUT16	HTTPRequestParserTest	testMalformedRequest2	Pass
AUT17	HTTPRequestParserTest	testParseRequest	Pass
AUT18	HTTPRequestParserTest	testMalformedRequestNoHost	Pass
AUT19	HTTPRequestTest	testGetURL	Pass
AUT20	HTTPServerConsoleTest	testrunConsoleNoPort	Pass
AUT21	HTTPServerConsoleTest	testrunConsolePort80	Pass
AUT22	HTTPServerConsoleTest	runOnTakenPort	Pass
AUT23	HeaderTest	testFromString	Pass
AUT24	HeaderTest	testFromString2	Pass
AUT25	HeaderTest	testFromString3	Pass
AUT26	HeaderTest	testFromString4	Pass
AUT27	PortTest	testPort0	Pass
AUT28	PortTest	testPortTooLarge	Pass
AUT29	PortTest	testPortOk	Pass
AUT30	ResponseFactoryTest	getBad	Pass
AUT31	ResponseFactoryTest	testUnknownMethod	Pass
AUT32	ResponseFactoryTest	testGetResponseGETRoot	Pass
AUT33	ResponseFactoryTest	testGetResponseUnexistingFile	Pass
AUT34	ResponseFactoryTest	testIllegalFile	Pass
AUT35	ServerFactoryTest	testCreate	Pass
AUT36	SharedFolderTest	testGetNonExistantFile	Pass
AUT37	SharedFolderTest	testGetIllegalFile	Pass
AUT38	SharedFolderTest	testGetRootURL	Pass
AUT39	HTTPServerTest	testStart	Pass
AUT40	HTTPServerTest	testStop	Pass
AUT41	HTTPServerTest	testHTTPServer	Pass
AUT42	HTTPServerTest	testStopWhenNotStarted	Pass
AUT43	SocketClientTest	testGetFromOnlineServer	Fail
AUT44	StressTest	stressTest	Pass
AUT45	ContentTypeTest	testGetFromFileEnding	Pass
AUT46	ErrorResponses	test400	Pass
AUT47	HTMLFileResponseTest	testWriteResponse	Pass
AUT48	ConsoleViewTest	testNoArguments	Pass
AUT49	ConsoleViewTest	testOkDirectory	Pass
AUT50	ConsoleViewTest	testOkDirectory1	Pass
AUT51	ConsoleViewTest	testDoNotStop	Pass
AUT52	ConsoleViewTest	testShowhelp	Pass
AUT53	ConsoleViewTest	testDoStop	Pass
AUT54	ConsoleViewTest	testOkPort	Pass
AUT55	ConsoleViewTest	testCrapArgument1	Pass
AUT56	ConsoleViewTest	testCrapArgument2	Pass
AUT57	ConsoleViewTest	testPortTaken	Fail

3. Code Coverage Automated Tests

Class	Class, %	Method, %	Line, %
AcceptThread	100% (1/ 1)	100% (4/ 4)	100% (20/ 20)
AcceptThreadTest	100% (1/ 1)	100% (6/ 6)	100% (22/ 22)
ClientFactory	100% (1/ 1)	100% (2/ 2)	100% (7/ 7)
ClientSocket	100% (1/ 1)	100% (6/ 6)	100% (24/ 24)
ClientSocketTest	100% (1/ 1)	100% (6/ 6)	100% (24/ 24)
ClientThread	100% (1/ 1)	100% (4/ 4)	93,1% (27/ 29)
ClientThreadTest	100% (1/ 1)	100% (6/ 6)	100% (27/ 27)
HTTPReader	100% (1/ 1)	100% (3/ 3)	100% (21/ 21)
HTTPReaderTest	100% (1/ 1)	100% (6/ 6)	90,9% (20/ 22)
HTTPRequest	100% (2/ 2)	100% (7/ 7)	100% (19/ 19)
HTTPRequestParser	100% (1/ 1)	50% (1/ 2)	91,7% (11/ 12)
HTTPRequestParserTest	100% (1/ 1)	100% (6/ 6)	85,2% (23/ 27)
HTTPRequestTest	100% (1/ 1)	100% (4/ 4)	80% (4/ 5)
HTTPServer	100% (1/ 1)	100% (3/ 3)	100% (20/ 20)
HTTPServerConsole	100% (1/ 1)	66,7% (2/ 3)	77,3% (17/ 22)
HTTPServerConsoleTest	100% (1/ 1)	100% (5/ 5)	100% (31/ 31)
Header	100% (2/ 2)	100% (7/ 7)	100% (27/ 27)
HeaderTest	100% (1/ 1)	100% (5/ 5)	84,6% (11/ 13)
Port	100% (1/ 1)	100% (2/ 2)	100% (6/ 6)
PortTest	100% (1/ 1)	100% (4/ 4)	71,4% (5/ 7)
ResponseFactory	100% (1/ 1)	100% (3/ 3)	100% (15/ 15)
ResponseFactoryTest	100% (1/ 1)	100% (8/ 8)	100% (32/ 32)
ServerFactory	100% (1/ 1)	100% (2/ 2)	100% (2/ 2)
ServerFactoryTest	100% (1/ 1)	100% (2/ 2)	100% (6/ 6)
SharedFolder	100% (1/ 1)	100% (3/ 3)	100% (17/ 17)
SharedFolderTest	100% (1/ 1)	100% (6/ 6)	85,7% (12/ 14)

Class	Class, %	Method, %	Line, %
ContentType	100% (1/ 1)	100% (5/ 5)	100% (18/ 18)
ContentTypeTest	100% (1/ 1)	100% (4/ 4)	100% (9/ 9)
ErrorResponses	100% (1/ 1)	100% (2/ 2)	100% (21/ 21)
HTMLFileResponseTest	100% (1/ 1)	100% (4/ 4)	100% (14/ 14)
HTTP200OKFileResponse	100% (1/ 1)	100% (2/ 2)	100% (14/ 14)
HTTP400BadRequest	100% (1/ 1)	100% (2/ 2)	100% (7/ 7)
HTTP403Forbidden	100% (1/ 1)	100% (2/ 2)	100% (8/ 8)
HTTP404FileNotFoundResponse	100% (1/ 1)	100% (2/ 2)	100% (8/ 8)
HTTP405MethodNotSupportedResponse	100% (1/ 1)	100% (2/ 2)	100% (9/ 9)
HTTPResponse	100% (1/ 1)	100% (2/ 2)	100% (8/ 8)

Class	Class, %	Method, %	Line, %
InvalidPortException	100% (1/ 1)	100% (1/ 1)	100% (2/ 2)
MalformedRequestException	100% (1/ 1)	100% (1/ 1)	100% (2/ 2)
NotADirectoryException	100% (1/ 1)	100% (2/ 2)	100% (4/ 4)
NotStartedException	100% (1/ 1)	100% (1/ 1)	100% (1/ 1)
WrongNumberOfArgumentsException	100% (1/ 1)	100% (1/ 1)	100% (2/ 2)

Class	Class, %	Method, %	Line, %
ConsoleView	100% (1/ 1)	100% (14/ 14)	100% (46/ 46)
ConsoleViewTest	100% (1/ 1)	100% (13/ 13)	90,5% (67/ 74)

Class	Class, %	Method, %	Line, %
HTTPGetProtocoll	100% (1/ 1)	100% (2/ 2)	100% (8/ 8)
SocketClient	100% (1/ 1)	100% (2/ 2)	100% (14/ 14)

Class	Class, %	Method, %	Line, %
HTTPServerTest	100% (1/ 1)	100% (8/ 8)	92,1% (35/ 38)
SocketClientTest	100% (1/ 1)	100% (2/ 2)	21,4% (3/ 14)
StressTest	100% (4/ 4)	100% (12/ 12)	87,5% (77/ 88)

4. Conclusion

The manual testing resulted in a lot of failed tests. However, the vast majority of these are very simple and many of this test fails because there is not a access log named “log.txt” to be found.

Other failures are for example response status code 500 for internal errors are not implemented. It also seems to be not following the minimum requirements for HTTP 1.1 since the HEAD method is not giving a proper response like stated in the Hypertext Transfer Protocol Standard for HTTP 1.1 states at 4.1 that “All general-purpose servers MUST support the methods GET and HEAD.”

The source code is also not released under GPL 2.0 and instead have a MIT-certificate but as of my understanding this should not affect SDC ability to redistribute the server. The performance seems to be ok under realistic number of requests even though the server have some problem responding good under stress test when running on a Raspberry Pi. The performance testing is however a bit tricky to get a realistic scenario for a IOT-device, but my feeling is the server performs ok for an IOT-device.

The user experience for the administrator probably also needs to be improved since exceptions are thrown into the console.

Regarding the failed automated test, they do not seem to be indicating something troublesome. The “testPortTaken” just a extra space in the text. The “testGetFromOnlineServer” have static IP-address that is not working.

There are a lot of things that needs to develop further, so if SDC want something that is ready for the market today this is not the product.