Test Cases

My Web Server

1. General precondition and assumptions

To perform the setup to run the server and perform the tests the assumption is the tester have basic knowledge in java, development of java and handling of terminals. Java Runtime and JDK-version 11 or higher installed.

A general precondition for all tests is that the project from the original source code has been compiled with sdk-version 11 to be able to run in Raspberry Pi OS. A jar file artefact named “MyWebSever.jar” has been built and the folder named “resources” with its files, located in the original source code at “MyWebServer-master/bin/se/lnu/http/”, should be moved to the same directory as where the built jar-file is located.

The test cases and input steps in the example are, if not anything else mentioned in the specific test case, done in Windows 10 with the terminal Git Bash Terminal and Terminal in Raspberry Pi OS. Firefox will be the used browser in Windows 10 and Chromium Web Browser in Raspberry Pi OS.

Some of the test will use Postman to make some of the HTTP requests. Postman can be downloaded here. If the tester have no experience in Postman from before, here is a guide how to setup requests. To be able to perform Postman request to server running on Raspberry Pi change the address from localhost:9090 to <your raspberry pi local ip>:9090 for all test cases.

When general precondition is applied in the test case the following must be true:

- The tester should have a terminal opened and the directory should be the same as where MyWebServer.jar is located with no server running (stop server if server have started in another test case before). All folders named in test cases should have no access restrictions unless mentioned otherwise in a specific test case.

2. Overview manual tests

Test	Requirement
TC 1.1 Succesfully start server	UC 1
TC 1.2 Start server with already taken port	UC 1
TC 1.3 Start server with to low port number	UC 1
TC 1.4 Start server with low port number	UC 1
TC 1.5 Start server with high port number	UC 1
TC 1.6 Start server with to high port number	UC 1
TC 1.7 Start server with folder with restricted acc	UC 1
TC 1.8 Start server with file as shared container	UC 1
TC 1.9 Start server with only valid port number	UC1
TC 1.10 Start server with just valid shared container	UC 1
TC 1.11 Start server without arguments	UC 1
TC 2.1 Successfully stop server	UC 2

TC 2.2 Try to stop server with wrong command	UC 2
TC 2.3 Stop the server with aborting in terminal	UC 2
TC 3.1 Request the home of the server	UC 3
TC 3.2 Request a existing file – HTTP 200	UC 3
TC 3.3 Request a non-existing file – HTTP 404	UC 3
TC 3.4 Request a file outside the shared container	UC 3
TC 3.5 Request a file with malformed headers – HTTP 400	UC 3
TC 3.6 Receive internal server response on request – HTTP 500	UC 3
TC 4.1 Simple deploying on a Raspberry Pi	REQ 4
TC 5.1 Compatibility with Raspberry Pi	REQ 5
TC 6.1 Web server should be able to handle 25 request per seconds	REQ 6
TC 7.1 Follow minimum requirements for HTTP 1.1	REQ 7
TC 8.1 Source code release under GPL-2.0	REQ 8
TC 9.1 Access log viewable from text editor	REQ 9
TC 10.1 Web server should be able to handle massive number of req	REQ 10

3. Manual test cases

Test Case **TC 1.1 Successfully start the server**
Use Case UC1 Start Server
Scenario When administrator wants to start the server
Precondition General precondition
Input • In terminal type “java -jar MyWebServer.jar 9000 $(pwd)/resources/inner/” and press enter. • In browser, in the address-field type “localhost:9000” and press enter.
Expected output • Terminal prints “HTTP Server object constructed”, “HTTP Server started”, “Accept” first. Thereafter that two ClientThread have started and later stopped. • A note in the access log “log.txt” that is created and located in the same directory as “MyWebServer.jar” is written about the successful connection. • Browser shows heading text “It works” and two images that shows the same text.
Actual output No log.txt can be found, otherwise as expected.
Pass / Fail **Fail**

Test Case

TC 1.2 Start the server with already taken port

Use Case

UC1 Start Server

Scenario

When administrator wants to start the server but chooses a port that is already taken.

Precondition

Perform TC 1.1

Input

- Open a new terminal and navigate to the directory where “MyWebServer.jar” is located.
- In the second terminal type “java -jar MyWebServer.jar 9000 $(pwd)/resources/inner/” and press enter.
- In browser, in the address-field type “localhost:9000” and press enter.

Expected output

- Terminal prints “HTTP Server object constructed” and “Port is taken” and then exits and returns to accept new inputs.
- Browser shows heading text “It works” and two images that shows the same text.

Actual output

As expected

Pass / Fail

Pass

Test Case

TC 1.3 Start the server with to low non existing port number

Use Case

UC1 Start Server

Scenario

When administrator wants to start the server with port number 0.

Precondition

General precondition

Input

- In terminal type “java -jar MyWebServer.jar 0 $(pwd)/resources/inner/” and press enter.
- In browser, in the address-field type “localhost:0” and press enter.

Expected output

- Terminal prints “Enter a valid port 1-65535 and an optional URL” and then exits and returns to accept new inputs.
- Browser displays a text that the address is restricted for other purposes than web browsing with heading “This address is restricted” (Firefox)

Actual output

As expected

Pass / Fail

Pass

Test Case

TC 1.4 Start the server with a low port number

Use Case

UC1 Start Server

Scenario

When administrator wants to start the server with port number 1.

Precondition

General precondition

Input

- In terminal type “java -jar MyWebServer.jar 1 $(pwd)/resources/inner/” and press enter.
- In browser, in the address-field type “localhost:1” and press enter.

Expected output

- Terminal prints “HTTP Server object constructed”, “HTTP Server started”, “Accept” first. Thereafter that two ClientThread have started and later stopped.
- A note in the access log “log.txt” is written about the successful connection.
- Browser shows heading text “It works” and two images that shows the same text.

Actual output

No log.txt can be found, and browser shows “This address is restricted”, otherwise as expected.

Pass / Fail

Fail

Test Case

TC 1.5 Start the server with a high port number

Use Case

UC1 Start Server

Scenario

When administrator wants to start the server with port number 65535.

Precondition

General precondition

Input

- In terminal type “java -jar MyWebServer.jar 65535 $(pwd)/resources/inner/” and press enter.
- In browser, in the address-field type “localhost:65535” and press enter.

Expected output

- Terminal prints “HTTP Server object constructed”, “HTTP Server started”, “Accept” first. Thereafter that two ClientThread have started and later stopped.
- A note in the access log “log.txt” is written about the successful connection.
- Browser shows heading text “It works” and two images that shows the same text.

Actual output

No log.txt can be found, otherwise as expected.

Pass / Fail

Fail

Test Case
TC 1.6 Start the server with to high non existing port number
Use Case
UC1 Start Server
Scenario
When administrator wants to start the server with port number 65536.
Precondition
General precondition
Input
• In terminal type “java -jar MyWebServer.jar 65536 $(pwd)/resources/inner/” and press enter. • In browser, in the address-field type “localhost:65536” and press enter.
Expected output
• Terminal prints “Enter a valid port 1-65535 and an optional URL” and then exits and returns to accept new inputs. • Browser will treat the URL as not a correct URL and instead perform a search in the chosen search-provider website (default is Google in Firefox)
Actual output
As expected
Pass / Fail
Pass

Test Case
TC 1.7 Start the server with folder with restricted access
Use Case
UC1 Start Server
Scenario
When administrator wants to start the server with port number 65536.
Precondition
General precondition
Input
• Restrict access to the “inner” directory by for example following the steps described here. • In terminal type “java -jar MyWebServer.jar 65536 $(pwd)/resources/inner/” and press enter. • In browser, in the address-field type “localhost:65536” and press enter.
Expected output
• Terminal prints an error with text “No access to folder inner” and then exits and returns to accept new inputs. • Browser can’t establish a connection and displays a header text “Unable to connect” (Firefox).
Actual output
Terminal connects to the server. Browser shows text 404 Not Found.
Pass / Fail
Fail

Test Case

TC 1.8 Start the server with a file as shared container

Use Case

UC1 Start Server

Scenario

When administrator wants to start the server but enter a file as shared container.

Precondition

General precondition

Input

- In terminal type “java -jar MyWebServer.jar 9000 $(pwd)/resources/inner/index.html” and press enter.
- In browser, in the address-field type “localhost:9000” and press enter.

Expected output

- Terminal prints a NotADirectoryException with text “url is not a directory” and then exits and returns to accept new inputs.
- Browser can’t establish a connection and displays a header text “Unable to connect” (Firefox).

Actual output

As expected

Pass / Fail

Pass

Test Case

TC 1.9 Start the server with just a valid port number

Use Case

UC1 Start Server

Scenario

When administrator wants to start the server with port number 9000 but enters no shared container.

Precondition

General precondition

Input

- In terminal type “java -jar MyWebServer.jar 9000” and press enter.
- In browser, in the address-field type “localhost:9000” and press enter.

Expected output

- Terminal prints a NotADirectoryException with text “Index 1 out of bounds for length 1” and then exits and returns to accept new inputs.
- Browser can’t establish a connection and displays a header text “Unable to connect” (Firefox).

Actual output

As expected

Pass / Fail

Pass

Test Case
TC 1.10 Start the server with just a valid shared container
Use Case
UC1 Start Server
Scenario
When administrator wants to start the server but forgets to add a valid port number.
Precondition
General precondition
Input
• In terminal type “java -jar MyWebServer.jar $(pwd)/resources/inner/” and press enter.
Expected output
• Terminal prints “Enter a valid port 1-65535 and an optional URL” and then exits and returns to accept new inputs.
Actual output
As expected
Pass / Fail
Pass

Test Case
TC 1.11 Start the server without arguments
Use Case
UC1 Start Server
Scenario
When administrator wants to start the server but forgets to add any arguments
Precondition
General precondition
Input
• In terminal type “java -jar MyWebServer.jar” and press enter.
Expected output
• Terminal prints a WrongNumberOfArgumentsException with text “This program should only have one or two arguments” and then exits and returns to accept new inputs.
Actual output
As expected
Pass / Fail
Pass

Test Case
TC 2.1 Successfully stop the server
Use Case
UC2 Stop Server
Scenario
When administrator wants to stop the server.
Precondition
Perform TC 1.1
Input
• In terminal type “stop” and press enter. • In browser, in the address-field type “localhost:9000” and press enter.
Expected output
• Terminal prints “HTTP Server Accept thread stopped”, “HTTP Server stopped” and then exits and returns to accept new inputs. • Browser can’t establish a connection and displays a header text “Unable to connect” (Firefox). • A note in the access log “log.txt” is written about the successful stoppage of the server.
Actual output
No log.txt can be found otherwise as expected
Pass / Fail
Fail

Test Case
TC 2.2 Try to stop the server with wrong command
Use Case
UC2 Stop Server
Scenario
When administrator wants to stop the server but writes wrong word.
Precondition
Perform TC 1.1
Input
• In terminal type “stopa” and press enter. • In browser, in the address-field type “localhost:9000” and press enter.
Expected output
• Terminal prints nothing new. • Browser shows heading text “It works” and two images that shows the same text.
Actual output
As expected
Pass / Fail
Pass

Test Case **TC 2.3 Stop the server with aborting in terminal**
Use Case UC2 Stop Server
Scenario When administrator wants to stop the server with aborting from terminal
Precondition Perform TC 1.1
Input • In terminal hold “CTRL” + “C” on keyboard at the same time. • In browser, in the address-field type “localhost:9000” and press enter.
Expected output • Terminal exits and return to wait on new terminal input. • A note in the access log “log.txt” is written about the stopped connection. • Browser can’t establish a connection and displays a header text “Unable to connect” (Firefox).
Actual output No log.txt can be found otherwise as expected
Pass / Fail **Fail**

Test Case

TC 3.1 Request the home of the server

Use Case

UC3 Request shared resource

Scenario

When user wants to access the home directory to display “index.html”

Precondition

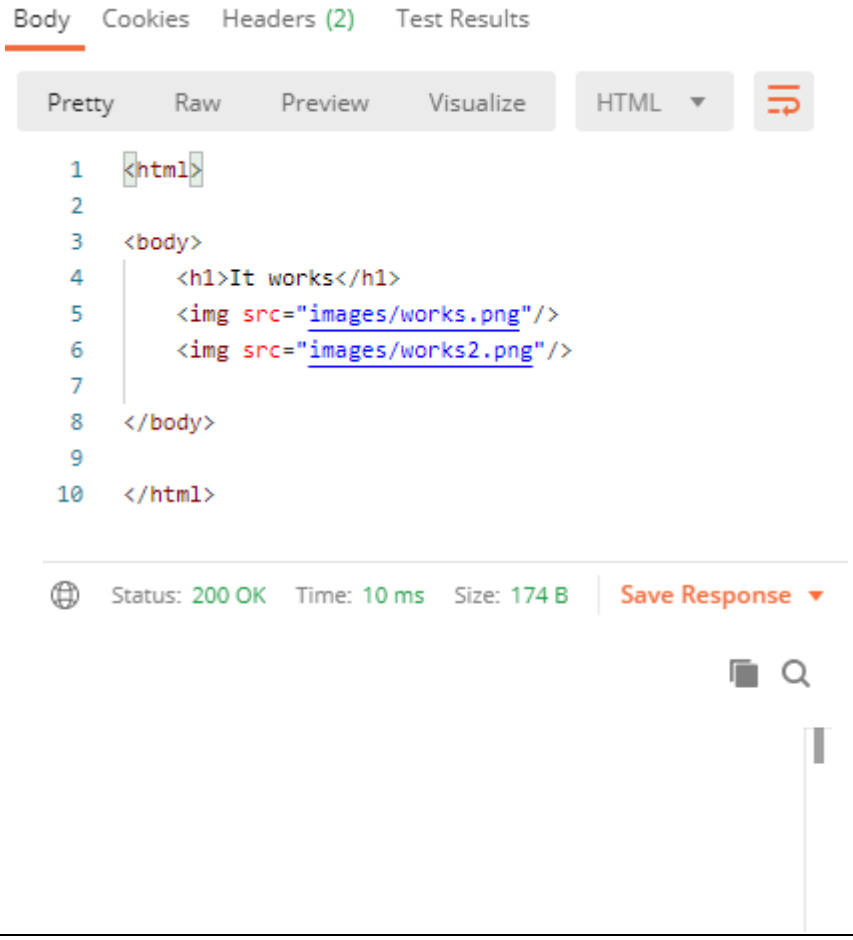
Perform TC 1.1
Open Postman and be ready to fill in a new request (follow linked tutorial in General preconditions and assumptions if needed)

Input

- In Postman, in the address-field type “localhost:9000/”.
- Click send.

Expected output

- Postman displays a body with the HTML-text with <h1>-tag with text “It works” and 2 img-tags.
- Postman displays the status code as 200 OK.
- A note in the access log “log.txt” is written about the successful request.



Actual output

No log.txt can be found otherwise as expected

Pass / Fail

Fail

Test Case

TC 3.2 Request a existing file – HTTP 200

Use Case

UC3 Request shared resource

Scenario

When user wants to access a file that exist in the shared container

Precondition

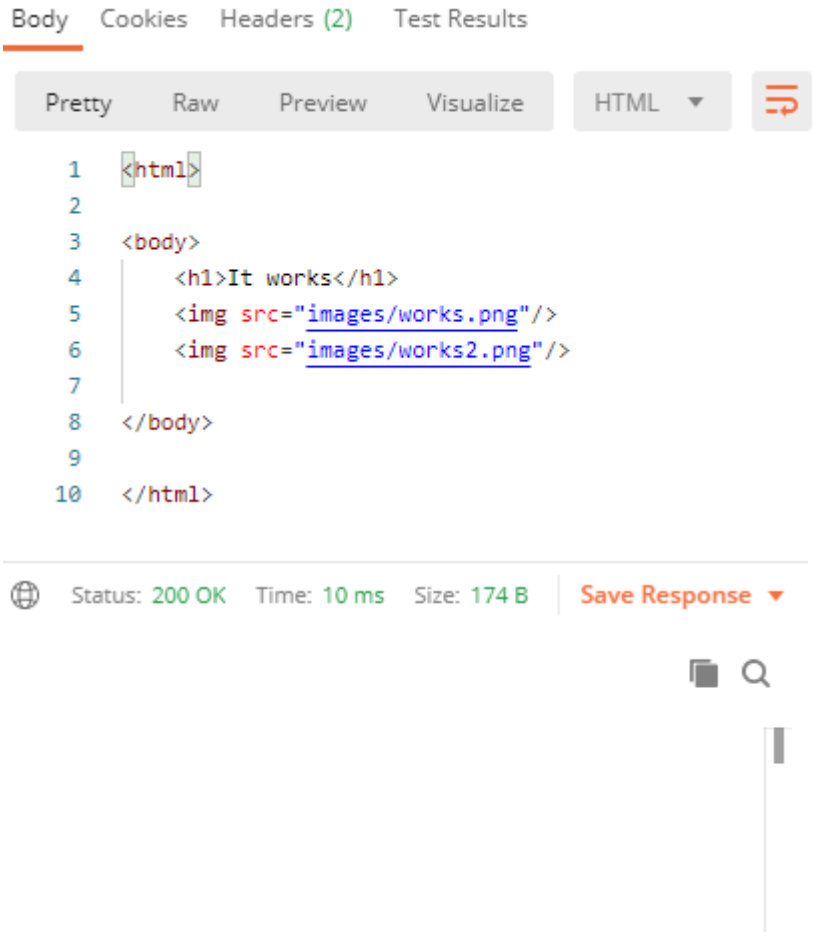
Perform TC 1.1
Open Postman and be ready to fill in a new request (follow linked tutorial in General preconditions and assumptions if needed)

Input

- In Postman, in the address-field type “localhost:9000/index.html”.
- Click send.

Expected output

- Postman displays a body with the HTML-text with <h1>-tag with text “It works” and 2 img-tags.
- Postman displays the status code as 200 OK.
- A note in the access log “log.txt” is written about the successful request.



Actual output

No log.txt can be found otherwise as expected

Pass / Fail

Fail

Test Case

TC 3.3 Request a non-existing file – HTTP 404

Use Case

UC3 Request shared resource

Scenario

When user wants to access a file that does not exist

Precondition

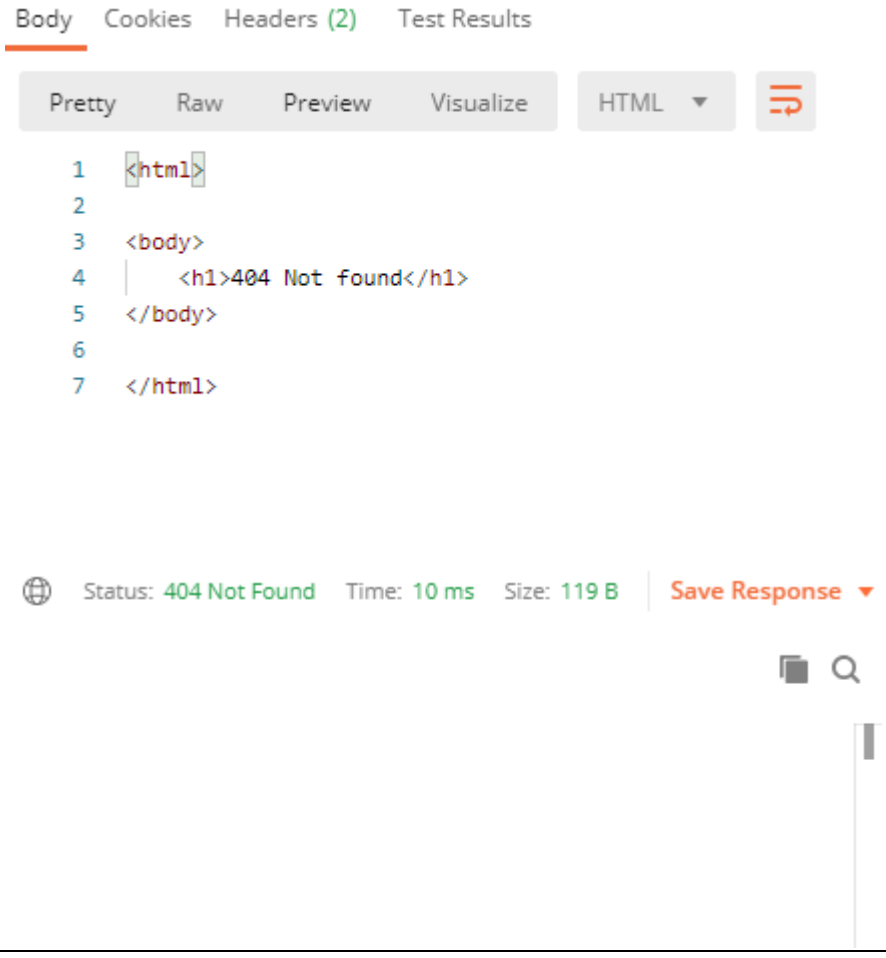
Perform TC 1.1
Open Postman and be ready to fill in a new request (follow linked tutorial in General preconditions and assumptions if needed)

Input

- In Postman, in the address-field type “localhost:9000/cool.html”.
- Click send.

Expected output

- Postman displays a body with the HTML-text with <h1>-tag with text “404 Not Found”.
- Postman displays the status code as 404 Not Found.
- A note in the access log “log.txt” is written about the unsuccessful request.



Actual output

No log.txt can be found otherwise as expected

Pass / Fail

Fail

Test Case

TC 3.4 Request a file outside the shared container – HTTP 403

Use Case

UC3 Request shared resource

Scenario

When user trying to access a file outside the shared container

Precondition

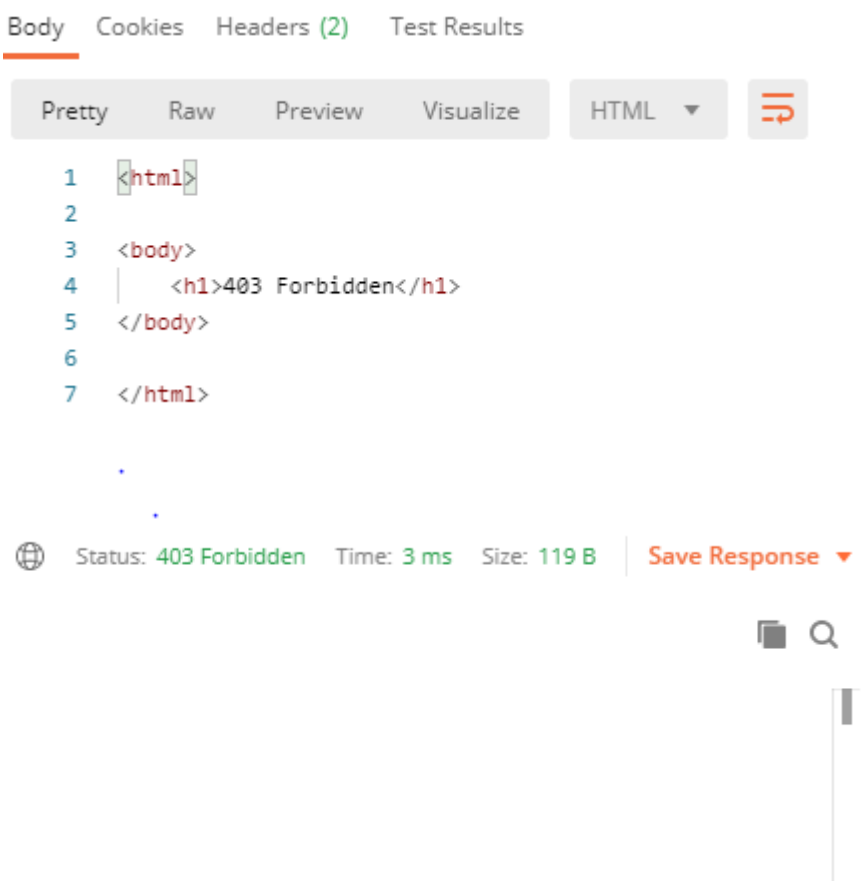
Perform TC 1.1
Open Postman and be ready to fill in a new request (follow linked tutorial in General preconditions and assumptions if needed)

Input

- In Postman, in the address-field type “localhost:9000/../secret.html”.
- Click send.

Expected output

- Postman displays a HTML-text with <h1>-tag with text “403 Forbidden”.
- Postman displays the status code as 403 Forbidden.
- A note in the access log “log.txt” is written about the unsuccessful request.



Actual output

No log.txt can be found otherwise as expected

Pass / Fail

Fail

Test Case

TC 3.5 Request a file with malformed headers – HTTP 400

Use Case

UC3 Request shared resource

Scenario

When user does a request with malformed headers

Precondition

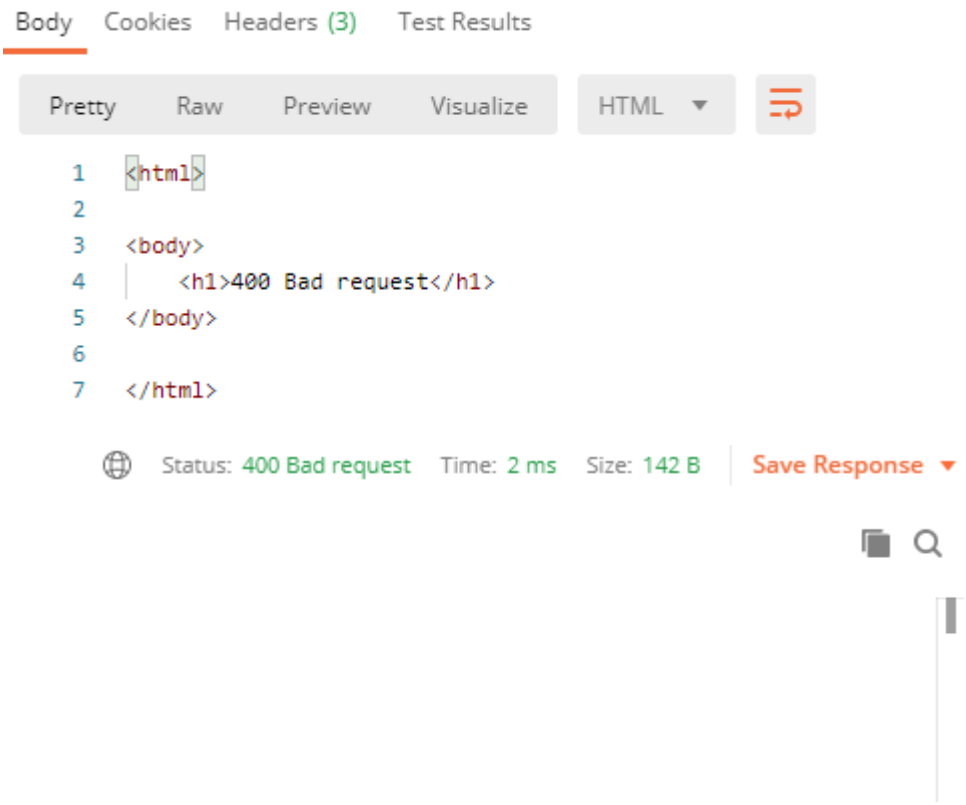
Perform TC 1.1
Open Postman and be ready to fill in a new request (follow linked tutorial in General preconditions and assumptions if needed)

Input

- In Postman, in the address-field type “localhost:9000”.
- In the headers tab in Postman, uncheck Host.
- Click send.

Expected output

- Postman displays a HTML-text with <h1>-tag with text “400 Bad request”.
- Postman displays the status code as 400 Bad Request.
- A note in the access log “log.txt” is written about the unsuccessful request.



Actual output

No log.txt can be found otherwise as expected

Pass / Fail

Fail

Test Case **TC 3.6 Receive internal server response on request – HTTP 500**
Use Case UC3 Request shared resource
Scenario When there is an internal server error upon a request
Precondition None
Input • Open up the project in the IDE • Do a basic code review, search for 500, to find when a 500 response is sent.
Expected output • 500 Internal error response are sent when there is no specific error response, as a last step.
Actual output No code for error status code response 500 could be found in code
Pass / Fail **Fail**

Test Case

TC 4.1 Simple deploying on a Raspberry Pi

Use Case / Requirement

REQ4

Description

Developer wants to deploy the web server on a Raspberry Pi

Precondition

Have a Raspberry Pi with Raspberry Pi OS installed.
Install FileZilla on Windows computer and setup a connection to the Raspberry Pi by following this guide (note is also of course is possible to transfer files from the terminal if that is preferred)

Input

- In FileZilla, open the connection to the Raspberry Pi in Site Manager under File.
- Locate the local folder were “MyWebServer.jar” is located. (left side of screen)
- Create a new folder in your Raspberry Pi located of your choosing (right side of screen)
- Double click on the newly created folder that is empty inside.
- Select “MyWebServer.jar” and “resources” folder. Right click and chose “Send”.
- Close FileZilla
- In Raspberry Pi OS, open terminal and locate yourself to the created folder with “MyWebServer.jar” and “resources” inside.
- In terminal in Raspberry Pi OS type “java -jar MyWebServer.jar 9000 $(pwd)/resources/inner/index.html” and press enter.

Expected output

- Terminal prints “HTTP Server object constructed”, “HTTP Server started” and “Accept”.
- A note in the access log “log.txt” that is created and located in the same directory as “MyWebServer.jar” is written about the successful connection.
- Browser shows heading text “It works” and two images that shows the same text.
- Be able to perform the deployment within 15 minutes.

Actual output

As expected, the tester have however already done this before so the time limit may potentially be hard for someone less experienced. And this test is potentially not a realistic alternative to deploy the server on a IOT-device…

Pass / Fail

Pass

Test Case **TC 5.1 Compatibility with Raspberry Pi**
Use Case / Requirement REQ5
Description Web server should be compatible and run in Raspberry Pi OS
Precondition Perform TC 4.1
Input • Perform all test cases starting with number 1, 2, 3 and report separately.
Expected output • Same output on each test as when performed on Windows 10.
Actual output There is a difference in output between Windows and Raspberry Pi OS at TC 1.4 where terminal instead responds port is already taken. Chromium Browser also use a different text when for example server can not be reached.
Pass / Fail **Fail**

Test Case **TC 6.1** Web server should be able to handle 25 request per seconds
Use Case / Requirement REQ6
Description Web server should be able to handle a high but not totally unreasonable amount of request without any error responses occurs. This test only needs to be done at the server installed in Raspberry Pi.
Precondition Perform TC 1.1 Install Jmeter and get a understanding of how load test works by following this tutorial Locate Load Test.jmx in the deliverables
Input • In Jmeter, press File -> Open and locate the Load Test.jmx-file. • Select HTTP Request to the left. • Update the field Server Name or IP from “localhost” to the ip of your Raspberry Pi. • Press the green play button.
Expected output • Summary report should display a throughput rate >25/sec • Summary report should display a Error% of 0.
Actual output As expected
Pass / Fail **Pass**

Test Case **TC 7.1 Follow minimum requirements for HTTP 1.1**
Use Case / Requirement REQ7
Description 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.” GET is tested at 3.1 but HEAD should also be tested.
Precondition Perform TC 1.1 Open Postman and be ready to fill in a new request (follow linked tutorial in General preconditions and assumptions if needed)
Input • In Postman, in the address-field type “localhost:9000”. • Change method to HEAD. • Click send.
Expected output • Get an empty body in Postman • Get headers response
Actual output Could not get response at all
Pass / Fail **Fail**

Test Case **TC 8.1 Source code released under GPL-2.0**
Use Case / Requirement REQ8
Description MyWebServer should be released under open source GPL-2.0 certificate.
Precondition None
Input • Locate the LISCENSE.txt in the root folder of source code. • Open file in Notepad
Expected output • Display a GPL-2.0 Certificate
Actual output The certificate is a MIT.
Pass / Fail **Fail**

Test Case
TC 9.1 Access log viewable from text editor
Use Case / Requirement
REQ9
Scenario
Administrator should be able to view the access log from a text editor
Precondition
Perform TC 1.1
Input
• Locate yourself to the same directory as “MyWebServer.jar” • Click on “log.txt”
Expected output
• An access log displaying at least one log message is displayed
Actual output
No access log could be found.
Pass / Fail
Fail

Test Case
TC 10.1 Web server should be able to handle massive number of requests
Use Case / Requirement
REQ10
Description
Web server should be able to handle massive request load with less than 20% error responses and without crashing. This test only need to be done at the server installed in Raspberry Pi.
Precondition
Perform TC 4.1 Install Jmeter on Windows computer and get a understanding of how load test works by following this tutorial Locate Stress Test.jmx in the deliverables
Input
• In Jmeter, press File -> Open and locate the Stress Test.jmx-file. • Select HTTP Request to the left. • Update the field Server Name or IP from “localhost” to the ip of your Raspberry Pi. • Press the green play button.
Expected output
• Summary report should display a Error% of < 20%. • Server should not crash
Actual output
Error% of the request is however 26,35% and is extremely slow at the end.
Pass / Fail
Fail

4. Automated tests

The already existing automated tests in the original source code will be regression tested. It is 57 tests and are following:

Automated test id	Test file	Test name
AUT1	AcceptThreadTest	testStopmeNotRunning
AUT2	AcceptThreadTest	testrun
AUT3	AcceptThreadTest	testsockedfailed
AUT4	ClientSocketTest	testWriteResponseBody
AUT5	ClientSocketTest	testWriteResponseHeader
AUT6	ClientSocketTest	testGetRequest
AUT7	ClientThreadTest	testMalformed
AUT8	ClientThreadTest	testRun
AUT9	ClientThreadTest	testMultipleConnections
AUT10	HTTPReaderTest	testbroken
AUT11	HTTPReaderTest	testbroken2
AUT12	HTTPReaderTest	testReadBody
AUT13	HTTPReaderTest	testReadkOk
AUT14	HTTPRequestParserTest	testMalformedRequest
AUT15	HTTPRequestParserTest	testMalformedRequestEmpty
AUT16	HTTPRequestParserTest	testMalformedRequest2
AUT17	HTTPRequestParserTest	testParseRequest
AUT18	HTTPRequestParserTest	testMalformedRequestNoHost
AUT19	HTTPRequestTest	testGetURL
AUT20	HTTPServerConsoleTest	testrunConsoleNoPort
AUT21	HTTPServerConsoleTest	testrunConsolePort80
AUT22	HTTPServerConsoleTest	runOnTakenPort
AUT23	HeaderTest	testFromString
AUT24	HeaderTest	testFromString2
AUT25	HeaderTest	testFromString3
AUT26	HeaderTest	testFromString4
AUT27	PortTest	testPort0
AUT28	PortTest	testPortTooLarge
AUT29	PortTest	testPortOk
AUT30	ResponseFactoryTest	getBad
AUT31	ResponseFactoryTest	testUnknownMethod
AUT32	ResponseFactoryTest	testGetResponseGETRoot
AUT33	ResponseFactoryTest	testGetResponseUnexistingFile
AUT34	ResponseFactoryTest	testIllegalFile
AUT35	ServerFactoryTest	testCreate
AUT36	SharedFolderTest	testGetNonExistantFile
AUT37	SharedFolderTest	testGetIllegalFile
AUT38	SharedFolderTest	testGetRootURL
AUT39	HTTPServerTest	testStart
AUT40	HTTPServerTest	testStop
AUT41	HTTPServerTest	testHTTPServer
AUT42	HTTPServerTest	testStopWhenNotStarted
AUT43	SocketClientTest	testGetFromOnlineServer
AUT44	StressTest	stressTest

AUT45	ContentTypeTest	testGetFromFileEnding
AUT46	ErrorResponses	test400
AUT47	HTMLFileResponseTest	testWriteResponse
AUT48	ConsoleViewTest	testNoArguments
AUT49	ConsoleViewTest	testOkDirectory
AUT50	ConsoleViewTest	testOkDirectory1
AUT51	ConsoleViewTest	testDoNotStop
AUT52	ConsoleViewTest	testShowhelp
AUT53	ConsoleViewTest	testDoStop
AUT54	ConsoleViewTest	testOkPort
AUT55	ConsoleViewTest	testCrapArgument1
AUT56	ConsoleViewTest	testCrapArgument2
AUT57	ConsoleViewTest	testPortTaken