TEST CASES

TABLE OF CONTENTS

REQUIREMENTS TO BE COVERED THIS ITERATION BY TESTS

REQUIREMENTS FOR USE CASE 1:

Requirement 1.1: The server should be able to be started with different port, by the needs of the administrator.

Requirement 1.2: The system is loading properly the resources provided.

Requirement 1.3: The system is keeping a log on startup action.

Requirement 1.4: The system is refusing a startup in case of missing elements.

REQUIREMENTS FOR USE CASE 2:

Requirement 2.1: The server should stop on demand of an administrator.

Requirement 2.2: The server should keep log on stopping of the server.

REQUIREMENTS FOR USE CASE 3:

Requirement 3.1: The server should share the resources with the browser.

Requirement 3.2: The server should throw an appropriate error dependent on the situation.

Requirement 3.3: The server should keep a log on all access to and from the browser as well as administrator changes.

Requirement 3.4: The server should present that it has internal error when encountering an error prone process.

Requirement 3.5: If an invalid request is sent to the server it should respond that the request cannot be handled.

Requirement 3.6: If the shared resource is outside of its container the system should present that the resource is forbidden.

REQUIREMENTS OUTSIDE OF THE USE CASES DEFINED BY THE STAKEHOLDERS

Requirement 4.1: The system should have basic web vulnerabilities protection.

Requirement 4.2: The system should be easily modifiable by future IOT developers.

Requirement 4.3: The system should be adaptable to different devices associated with IOT.

Requirement 4.4: The system should be responsive under high load.

MANUAL TEST CASES

TEST CASES 1.1

Scenario: Starting the server successfully.

Precondition: The server is not deployed yet.

Input:

In the command prompt start the file with java and provide argument to the program which in this case is port to the server as well as the directory of the resource folder.

Output:

The command prompt should return appropriate message on server started and it is running.

The system is deployed.

TEST CASES 1.2

Scenario: Starting the server with port that is already in use.

Precondition: The server is not running on the specified port.

Input:

In the command prompt start the file with java and provide argument to the program which in this case is port to the server but provide port of the server that is already running, as well as the directory of the resource folder.

Output:

The server which is already running keeps running.

The one that is being started is showing a message regarding the port being used.

TEST CASES 1.3

Scenario: Starting the server with directory that the server does not have.

Precondition: The server is not started.

Input: The server is being started with an appropriate argument for the port but for the path to the resource folder it is picked a directory which is not valid.

Output: The server is showing a message that the directory is not found. The server is not continuing the startup process.

TEST CASES 2.1

Scenario: The server is stopped from the terminal after a normal startup.

Precondition: The server must be running normally.

Input:

On server running the command prompt should receive a command “stop”.

Output:

The terminal is showing appropriate message that the server has been properly stopped.

The server is not running anymore.

TEST CASES 3.1

Scenario: On the server that is being run, the files in the resource folder should be provided to the browser client.

Precondition: The server should be up and running properly.

Input:

While the server is running on the host machine, from a browser the host’s device external IP should be written and after that should be written behind a column the host’s port to which the server is running.

Output:

On the client side the content of the folder that is considered “resource directory” is being show.

On the host side the prompt is displaying thread and the file names of what is being supplied to the client side.

TEST CASES 4.1

Scenario: The log is updated on startup and stop action as well on sharing resources.

Precondition: The server should be up and running properly.

Input:

The server is being started, then from the browser’s side there are multiple requests for sharing resources then the host is stopping the server.

Output:

The log is not being properly updated on the aforementioned actions as well on all cases in the tests above.

RESULTS

The test case associated with the log being maintained by the server, keeping information on startup and stop of the server as well as client access to features of the server is not handled. The log availability in the server source code is not being implemented thus leading to this test not passing the general requirements. The other tests are reasonably passing and answering to the needed requirements defined by the stakeholders.

SECURITY TEST CASES

The test cases should be numbered 4.2

APPROACH

The server was tested for basic vulnerabilities exploiting a FTP(File Transfer Protocol). The test was conducted with the CURL and WPUT and the main objective was to upload a picture to the resource folder thus modifying the content of the web page being shown.
The test was conducted from an external operator while the host is maintaining the server open with appropriate permissions. The initial test was conducted with a simple picture file upload to the resources folder.

RESULTS

On launch CURL established a connection with the server but instead of uploading the picture it started overflowing the server since the system was not able to handle the request that CURL did, thus leading to an unintentional Denial of Services(DoS). The website was timing out when another browser tried to open it and did not show anything from the resources provided. The host reported overflow and timeout from his side as well.
WIth different arguments of the same program it gives an error 405: Unsupported command, which is showing that the server has basic security against running unsupported commands.
After the CURL tests the same one was conducted with WPUT but gave results with error message 400 : Bad request, probably due to the nature of the different requests that the two programs are using.
In general the application is prone to vulnerabilities but has basic defense against unnatural requests from the side of the client.

CONFIGURATION TEST CASES

The test cases should be numbered 4.3

PURPOSE

Due to the nature and purpose of the request by the stakeholders the server ought to be runnable by different devices associated with the nature of IOT. The configuration tests fall into place to answer the need of the stakeholders to test different devices running “MyWebServer”.

APPROACH

For the tests to be held some microchips were being chosen filtered by usability in the IOT branch. One to be considered was the ubuntu core which might be described as lightweight ubuntu specifically created for microcontrollers. Another was the Raspbian OS which is coming with another popular in the IOT branch microcomputer: Raspberry Pi. The tests conducted are the one described above as “Manual Test Cases”.

TEST CASES

On ubuntu core the server was started and stopped by executing the appropriate commands from the terminal however not all of the functionality could have been tested. Due to the lack of proper GUI (Graphical User Interface) in this lightweight OS the test cases defined by the use case #3 was conducted but did not give meaningful result. The tests are conducted with images attached into the resource folder and the same cannot be shown in an OS without GUI.

The Raspbian OS could load the picture but experienced high load due to the low powered and low performance parts.

Due to the nature of the operating systems the tests needing a GUI were not performed properly and did not give proper results. Furthermore, the IOT microcontrollers are hardly considered powerful and the data transferred would probably not be with graphical nature which would reason a different approach for next iteration of testing and development.

SUPPLEMENTARY TEST CASES

The test cases should be numbered 4.4

UNIT TEST CASES

The test cases provided with the source code are testing the direction of properly handled exceptions or if different HTTP elements are passed properly to other objects on call. Many but not all entries are being covered. One of the cases is when the system is being deployed with multiple arguments.

In general, despite few missing cases and few outdated cases and some tests being redundant inside the test methods, the overall usability is high.

INTEGRATION TESTS CASES

The integration test cases are verifying multiple things and are in a package. They verify actions relevant to starting and stopping the server and furthermore things such as availability of the server from external clients defined by the use cases as the actor “browser”. The other useful aspect is the stress testing that shows the server handling multiple simultaneous connections. The test creates multiple threads which tries to simultaneously connect to the server.

RESULTS

The code is well covered with automated tests. Most of them are in passing state. Almost 100 percent is being covered with missing automated tests on main method and one other block. One of the tests which was not good or explicitly useful was part of the integration package in particular the test verifying the accessibility by external client. Otherwise the automated tests proved to be useful for most of the requirements defined by the stakeholders.

The following page contains the traceability matrix for requirements and test cases.

REQUIREMENT TRACEABILITY MATRIX

Req. ID	*1.1*	*1.2*	*1.3*	*1.4*	*2.1*	*2.2*	*3.1*	*3.3*	*3.4*	*3.5*	*3.6*	*4.1*	*4.2*	*4.3*	*4.4*
Case 1.1	X	X	**X**												
Case 1.2	X		**X**												
Case 1.3			**X**	**X**											
Case 2.1					**X**										
Case 3.1						**X**	**X**								
Case 4.1							**X**								
Case 4.2													**X**		**X**
Case 4.3														**X**	
Case 4.4								**X**	**X**	**X**	**X**	**X**		**X**	