Test Strategy

Background

- The small Software Development Company (SDC) has found a possible market niche for giving out a simple to deploy web-server. SDC aims to redistribute this server on a wide range of Internet Of Things (IOT) to present information from sensors etc. SDC wants an easy to deploy java-web-server that can be deployed on many different devices and therefore that can attract attention of a wide range of IOT developers. IOT-developers want minimal configuration as well as easy integration and adaptation of the web-server. End-customers want easy access and absolute security. [1]

Scope

- What to test: The Software Development Company's abandonware they've found.
- Why to test it: To find out if it is suitable for the SDC's needs, and if it is good enough for their end-users.
- How to test: By a range of testing techniques which will be described further down in this document.
- Who are involved: A single individual tester, with feedback from peers.

Objectives/Goals

- Test if the software fulfills the requirements by performing:
  - **Black box testing** which helps find out how the server works without knowing the implementation details. It is also used in the sense of discovering issues of the application with the use of a web browser. An example of this is to use the knowledge of relative urls to try to access hidden resources (e.g., the file "secret.html").
  - **White box (glass box) testing**, which means looking at the source code to figure out how to emulate certain responses. For example, by finding out how to intentionally cause a "400 Bad Request" error by looking at the classes at the location: "se.lnu.http.response".
  - **Stress/load testing** to see how the software handles concurrent requests in different quantities.
  - **API testing** to evaluate the HTTP responses when sending HTTP requests, such as deliberately causing a "400 Bad Request" with the knowledge of white/glass box testing as described above.
  - **Explorative testing**, meaning that while evaluating the software and looking at the source code, new knowledge is acquired, and that knowledge will be used to write additional tests, which in its turn again gives us even better understanding of the system to derive further use cases.
  - **Write manual test cases** based on the knowledge gained from the above steps.
  - **Conduct the manual test cases and report all the results** in a test matrix in a separate document (testReport) which also includes a summary of the tests.

Who are the stakeholders?

- Software Development Company (SDC) which are the ones interested in the software to be evaluted.
- End-customers (users of the software when it is in production).

What is the motivation of testing?

- The SDC wants to confirm that the open source software, "My Web Server" fulfills the requirements in the requirements specification document. This is to ensure that the software is of high quality before deploying it to be accessible by other actors; such as customers and/or associates of the SDC. As the software is regarded as being "abandonware" (developers have since long ago abandoned the development of this product), the SDC wants to know if it is worth investing in this application, or if they should look elsewhere.

What is to be tested?

- The provided use cases.
- Use cases derived from the requirements.
- Additional use cases based on explorative testing.
- Automatic testing with JUnit.
- API testing that let us find out if there are any issues with reaching the server, and if there are any concerns that need to be addressed, as in the sense of security for example.
- General tests, such as testing port numbers, how the server handles invalid requests etc.

What will NOT be tested?

Some areas may need specific fields of expertise, or there are some constraints that somehow otherwise affect what is not to be included in the testing activity. These are the following:

- Security.
- Optimization.
- Other operative systems than Windows 7, as they are not available for consideration.
- Other hardware than a desktop personal computer.
- Multi-browser compatibility testing. All tests will be performed on Mozilla FireFox.

What are the available resources?

- Staff: One individual tester conducting the tests and performing evaluations of the application's state versus how it is intended to work according to the specifications provided in the bundle of content (incl. High-level details, Low-level details, source code, code-coverage etc).
- Time resource: 20 hours.
- Operative System: Windows 7 64-bit.
- Integrated Development Environment (IDE):
  Eclipse (version number: 4.17.0) for Java Developers.
- Web-browser: Mozilla FireFox, version number: 83.0 (64-bit).
- Tools and other software:
  - JUnit (Testing framework)
  - JMeter (Load testing, analytics)
  - Insomnia REST (API testing)

How is a test evaluated as being successful?
- When the test case covers a specific use case and all goes exactly as specified.
- The tests must be consistent: the result must be the same for occasion A and B respectively, given that both occasions have values that belong to the same equivalance partition (could be positive numbers or strings of equal length for example).

What is the optimal code coverage percentage to be deemed as high quality software?
- At least 95% code coverage as there will be end-customers relying on the software's functionality in real-time, due to it being deployed on a live web server.

What are the test activities and how are they met from a quality assurance perspective?
- The test activities includes both manual and automatic tests to increase the total coverage, as some tests can not be automatically tested. Some examples of tests that automatic tests can not verify could be user experience, accessibility, and CSS styling (applies to both personal computers and mobile devices).

What will be the test level?
- This will depend solely on the requirements (use cases), and could therefore be a mix of test levels. It could be any combination of unit, integration and system level tests.
  See the document testCases.pdf for an overview of the tests to be conducted.

What roles exists, and what are their responsibilities?
- The tests to be conducted, will be by an individual tester that executes and logs the result of all tests and present them in the final document: testReport.pdf.

What are the environmental requirements for the tests?
- The requirements specification lists Linus, Mac and Windows* as prerequisites.
  In this case, Windows 7 64-bit is being used as the target operative system to conduct the testing, along with Eclipse IDE, version number: 4.17.0. The web server will be hosted locally.

* XP, Vista, 7, 8, Server 2008.

What are the risks, and how are they mitigated?

Issue	Resolution
Unable to check for compatibility on other operative systems.	Where applicable, simulate other operative system environments or exclude the operative system(s), but specify this in the test plan.
Unable to test for compatibility on other hardware.	Where applicable, simulate other hardware environments or exclude the devices altogether, but specify this in the test plan.
Tools does not work as intended.	Find a substitute tool that performs the same job in an equivalent manner.
There are bugs preventing the software to work properly.	Apply the latest patches to the software being used, and/or try other software to resolve this issue.
Some tests cases can not be evaluated, due to the nature of the preconditions, which may not apply to the current testing environment.	State these issues in the test plan and try to substitute those where applicable.

Notes

- There is an inconsistency: The requirements document says to use port 9000 to access the server's resources, but the images that are served at the home page (localhost:9000) displays that the port number :8080 is used. Although, it does not affect much, it may still be of interest to know about.

References

[1]: https://coursepress.lnu.se/kurser/mjukvarutestning/03-examinations/03-assignment-2/01-task-1