Test Plan

---

1. Revision history

Date	Version	Author	Description
2020-11-26	1.0	Test designer	Planning documentation for test plan.
2020-11-27	1.1	Test designer	Analyzing the product and defining objectives and tasks.
2020-11-30	1.2	Test designer	Deciding and defining test strategy.
2020-12-1	1.3	Test manager	Listing features to test and not to test.
2020-12-3	1.4	Test manager	Defining risks and how to manage them.
2020-12-7	1.5	Test designer	Defining exit criteria
2020-12-9	1.6	Implementer	Implementing listed strategies.

2. Introduction

The product to be tested is a simple to deploy web server that supports multiple platforms.Developers want minimal configuration as well as easy integration. End-customers want easy access as well as absolute security.
Down below is a supplementary specification for the web server.

- The web server should be responsive during high load.
- The web server must follow minimum requirements for HTTP 1.1.
- The web server must work on Linux, Mac and windows.
- The source code should be released under GPL-2.0.
- The access log should be viewable from a text editor.

3. Objectives and tasks

3.1 Objectives

The roles and corresponding responsibilities are listed in 7.1. In order for completion of the test plan all tasks below have to be completed. Else test plan is deemed insufficient.

3.2 Tasks

Below is a list of tasks related for the testing of the web server:

Planning the testing process

- Identification of the requirements to test
- Assessment of possible risk that could occur as well as the level of criticality
- Creation of the test strategy
- Creation of the schedule to work against
- Creation of the test plan

Designing the tests

- Creation of the test suites
- Identification and description of test cases
- Identification and structorialization of test scripts
- Reviewing the coverage of created tests

Implementing the tests

- Setting up the environment for tests to occur in
- Programming the test scripts

Executing the tests

- Execution of the test scripts
- Evaluation of said test scripts
- Verifying the results
- Investigation of unexpected occurrences as well as logging the defects

Evaluation of the test

- Evaluation regarding the coverage of the test cases
- Evaluation regarding the coverage of the code
- Determining if the test completion meets the success criteria
- Creation of the test report

4. Scope

General:

Things in the general scope to test are listed below. In the next section we describe how we are going to test corresponding areas.

- Testing specific operations, functions and classes. This is described and explained in 5.1.
- Testing the system in general along with how to integrate it. This is described and explained in 5.2.
- Testing the system during high load. This is described and explained in 5.3

To accompany this we write manual test cases to get a broader coverage that some tools and frameworks cant test.

Tactics:

Things that are out of scope are things that most likely will be tested in future iterations. These things are listed in section 8. The time taken to test corresponding areas are covered in 10.1. By keeping track of time as well as estimating it we can more successfully follow the timeline and schedule.

5. Testing strategy

5.1 Unit testing

Definition:

To test the application on a lower level we will use unit testing. To have the unit testing deemed successful they will have to pass at a 95% complete rate. The test will be run through testpad, which is a very good tool to keep an overview and traceability over run tests, successful tests as well as failing tests.

Participants:

Participating in the unit testing is the test manager as well as the designer. The test designer will have to participate to get a broader overview of the coverage of the current testing and how to cover possible areas not yet tested.

Methodology:

The test designer will write all the tests using JUnit. JUnit is a very good tool to test applications on a lower level and ensures that specific operations, methods and classes work the way they are intended. The testing activity will be conducted using testpad to keep as mentioned overview over the tests and how to further proceed.

5.2 System and integration testing

Definition:

A defined requirement is that the system operates and is functional as expected. To meet this requirement we must test the system along with how it is integrated in different environments.

Participants:

Participating in this testing process is the system tester, test designer and implementer. They will need to participate to get an understanding of how the system is integrated in different environments and how to test this.

Methodology:

This will be tested with manual test cases on various operating systems as well as different browsers. By using manual test cases in this process we can get a full verification that the system performs as it should on both the operating systems as well as all the common browsers.

5.3 Performance and stress testing

Definition: A defined requirement is that the system must be responsive during high load. This is tested during the process performance and stress testing. To test this we are going to use Apache Jmeter.

Participants:

Participating in this process is the system tester along with the test designer. They will need to take part to verify that the system meets such corresponding requirements.

Methodology:

When testing this subject we will take use of Apache Jmeter as well as some JUnit tests. Jmeter is a stress testing platform where we can get an understanding of how the system behaves during high load and stress. By using Apache Jmeter we can verify that this defined requirement is met, as well as spending more time on different topics.

5.4 Regression testing

Definition:

Regression testing is performed when a new build is deployed for testing which contains defect fixes and new enhancements if any.
Regression Testing is being done on the entire application and not just the new functionality and Defect fixes.
This testing ensures that existing functionality works fine after defect fix and new enhancements are added to the existing application.

Participants:

Participating in this process is the test designer as well as the implementer.

Methodology:

Each time a new build has been made, new functionality has been added; all manual test cases are to be retested to check for possible errors, bugs or defects.

6. Roles and responsibilities

This table shows people that are critical to the project.

Human resources

Role	Name	Specific responsibilities
Test manager	Classified	Provides oversight Responsibilities: ● Push testing in correct direction ● Provide appropriate resources to staff ● Report to management
Test designer	Classified	Identification, implementation and prioritization of test cases Responsibilities: ● Construct the test plan ● Construct the test suites ● Evaluation of the testing effectiveness
System tester	Classified	Execution of all tests Responsibilities: ● Test execution ● Log results of test ● Error-recovery ● Documentation of possible defects
Designer	Classified	Identification and definition of tests on attributes, operations and associations of the corresponding test classes. Responsibilities: ● Construct test classes based on previous identifications ● Construct test packages based on previous identifications
Implementer	Classified	Implements the test classes, test packages as well as the

		unit tests Responsibilities: ● Creates the things listed above

7. Features to be tested

Down below is a list of features that needs to be tested:

Feature	How to test it
● The web server should be responsive during high load.	This will be tested using Jmeter and JUnit.
● The web server must follow requirements for HTTP 1.1.	This is tested using manual test cases.
● The web server must work on linux, mac and windows.	This is tested using manual test cases.
● The source code should be released under GPL-2.0.	This is not tested in the current iteration.
● The access log should be viewable from a text editor.	This is tested using manual test cases.
● Start the server.	This is tested using manual test cases.
● Stop the server.	This is tested using manual test cases.
● Request shared resource.	This is tested using manual test cases.

8. Features to not be tested

Down below are the features and strategies not tested as well as corresponding reasons:

- Security

Security will not be tested as it is a very complex matter. The team employed currently does not have the shared knowledge regarding security to test it enough. Maybe in a future iteration of the project, a full testing of security will be completed. If say the team has gained new knowledge regarding security.

- Beta testing

Beta testing is performed by a large amount of end users. Data and experience is gathered as the end users take use of the product. The end user validates the products functionality, reliability, usability etc.
Beta testing won't be performed this iteration because it is now considered to be in alpha testing stages.

- User acceptance testing

When doing this we need to compare the system to its initial requirements. This process's purpose is to confirm that the system is ready for operational use. As the name of the testing strategy, it describes very well the thing intended with it. To have the acceptance of future users before moving to a production environment. This testing strategy is performed with the end users so it will most likely be done with beta testing in future iterations.

- Exploratory testing

This iteration we will not take use of exploratory testing as it relies on discovery of possible bugs and side effects. In future iterations exploratory testing will most likely take place because it can find and test defects not previously found by current tests.

- Exploratory Testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.

  Under scripted testing, you design test cases first and later proceed with test execution. On the contrary, exploratory testing is a simultaneous process of test design and test execution all done at the same time.
  Scripted Test Execution is usually a non-thinking activity where testers execute the test steps and compare the actual results with expected results. Such test execution activity can be automated and does not require many cognitive skills. Exploratory testing when using this testing strategy the focus was "thinking" activity. It has not been used to a great extent since we are not experienced testers and because we have had directions on what to test from knowledge of requirements from the stakeholders.

9. Exit criteria

In order to stop testing the application and deem it working properly without side effects etc. the amount of unit tests to pass must be at least 95%. All manual test cases need to pass at a rate of 100%. This provides confidence in the software that it's working properly. The stakeholders listed in the test strategy will have a big interest in how the testing has unfolded and are not keen on a badly tested product with a lacking test coverage as well as a product not passing less than agreed.

When the exit criteria is met the testing can seize and a test report can be generated. Testing will always be an important matter as well as a tool to further develop the web server.

10. Schedule and estimation

10.1 Schedule

Week	Task/s	Time spent
47	Creating test plan	5 hours
48	Definition of requirements etc.	5 hours
49	● Setting up test environment ● Defining tests	10 hours
50	Running tests, reporting, creating test deliverables etc.	10 hours

10.2 Estimation

Task	Members	Estimated effort

Create test plan	Test designer, Test manager	5 hours
Definition of requirements to be met etc.	Test designer	5 hours
Create test cases	Test designer	5 hours
Set up test environment	Implementer	5 hours
Performing tests	Implementer	5 hours
Test report	Test manager, Test designer	2,5 hours
Test delivery	Test manager, Test designer	2,5 hours
Total:		30 hours

11. Risk Analysis

Risks can always happen there for a planning for risks that can be devastating increases the chances of handling them better if they were about to occur. Therefore a list of risks will be made and a table for probability which also checks the effects of the risk.

List of risks

- Human - Illness, death, injury, or other loss of a key individual.
- Deadline -The project is not done in time.
- Technical - Advances in technology, or from technical failure.
- Requirement Change - The client wants another software to be made instead.

Risk	**Probability**	**Effects**
Human	Low	Serious

Deadline	Low	Serious
Technical	Low	Tolerable
Requirement Change	Low	Tolerable

Prevent Risks

Human, keep all employees in a healthy environment where everyone can enjoy their work.

Deadline, Push the project forward with regular meetings that are held by the Test manager.

Technical, keep all software used in the project updated, store all progress on DevOps platforms like GitLab.

Requirement Change, schedule regular meetings with stakeholder and ask for any updates for new information about any upcoming changes to the project.

Contingency plan

Human, if there would be a loss of a key individual, a new developer with the same skills in the same field, would need to be hired ASAP.

Deadline, hire another leader role that has responsibility for the project's Test Driven Development(TDD) progress for completion and more overtime work for the developers with extra pay.

Technical, use other versions of the software not the latest version, might want to find other software that are similar to the previous used software.

Requirement Change, get extra time for the deadline to be able to make the changes complete in time.