Test Plan

Company 1 - TDDC88

Contents

Version	Author	Updates	Reviewed by	Date
0.1	E. Sköld	Initial version	A. Telenius	2021-09-13
0.2	E. Parö	Test Levels and more	D. Ma	2021-09-24
0.3	A. Telenius	Fill in last headers	D. Ma	2021-09-24
1.0	A. Telenius	Rewrite every section in chapters 2.3-4.2	E. Parö	2021-11-07
1.1	O. Löfgren	Rewrite chapters 1 - 1.3	E. Parö	2021-11-09
1.2	E. Parö	Rewrite chapter 2.1	A. Telenius	2021-11-09
1.3	E. Parö	Added one paragraph in chapter 1	E. Sköld	2021-11-16
1.4	E. Schadewitz, A. Nilsson	Updated 2.2.3 with additional information about system testing	E. Parö	2021-11-17
1.5	O. Löfgren	Updated chapter 2.2.3	A. Telenius	2021-11-19
1.6	E. Parö	Updated 2.1, 2.2.1 and 2.2.3 from review by supervisor	O.Löfgren	2021-11-19
1.7	A. Telenius	Changed 2.2.2 and 1.1.2 pipeline and out of scope	O.Löfgren	2021-11-20
1.8	O. Löfgren	Added 3.1, changed 1, 1.1.1 and 2.2.2	A. Telenius	2021-11-22
1.9	E. Parö	Added information in chapter 3.1	-	2021-11-26
2.0	O. Löfgren, E. Parö	New chapter usability testing, changed chapter acceptance testing	-	2021-12-01
2.1	A. Telenius	Added Quality part in 2.2.3	-	2021-12-01

1 Introduction

In this document an explanation about our test strategies, process, workflow and methodologies used in the project is given. The test plan will describe the structure of our testing and how they are incorporated in the process of developing the product. Every type of test that we will perform is going to be further explained into detail. The reason why we will perform certain tests at which stage in the project will also be explained. This is going to be used for all the testers to work in the same way and to have a standard within the company, this test plan will be read by new testers for learning how to work with the testing. Therefore there will be further explanation of how to handle bugs found during testing, what tools that are used and when a test is considered complete.

We base our testing procedure on the V-model. This is a model that has a testing phase parallel to the development phases. This model is also called Verification and Validation model, it is an extension of the waterfall model as it is also a sequential model. There is a testing phase parallel to each development stage. This means that it includes testing for the requirements, system design, module testing and the implementation. The advantage of using this model is that it is simple and easy to follow and when the requirements are well defined, it also works well for smaller projects like this.

The development will be performed agile which mean that it is performed iterative. Therefore, the testing is also done iterative. This will enhance the testing because retrospectives of the sprints will be done after each iteration that will highlight the pros and cons of the testing. This will also strengthen the customer relation because acceptance testing with the customer will be held after each sprint. The customer can validate our requirements, if we are going in the right direction or if they have changed their needs.

1.1 Scope

The scope of the test plan is defined below. This includes what will be tested (In Scope) and the parts that will not be tested (Out of Scope).

1.1.1 In Scope

Testing is to be conducted in system and acceptance level testing. The customer wants us to focus on the usability of the system and how to visualise data in a satisfying way. Therefore the testing will focus on that and measure if the developed application achieves requested levels. The testing will also focus on the functionality of the system. The testing in this project will focus on System testing which tests the integration between different modules and also acceptance testing that verifies the requirements. These tests and how they are going to be performed in the project is going to be explained in the test plan. The main focus of the testing is to ensure that the requirements are met in the finished product.

1.1.2 Out of Scope

Due to the limited implementation levels of the project the test plan will diverge from the V-model by not conducting tests at the final level, maintenance. It also diverges from a full V-model by not conducting integration testing and unit testing. This was due to the low amount of components needed to be integrated during the project together with the basic level of functionality and output of the implemented functions.

The customer does not want us to focus on how to store data and database management and therefore tests on the application backend will not be conducted. Because the storing of data is not important for this project, security will neither be tested during this project. The testing of the database could otherwise have been done by doing Structural Database Testing which

validates the elements that are inside the database that are stored and cannot be accessed or changed by the end users.

1.2 Roles and Responsibilities

Testers
In the first part of the project (iteration 1 and 2) every tester is part of a cross-functional team (CFT) and is responsible for performing tests for their CFT individually related to the system testing. When a merge requests are created from a teams branch to the develop branch the tester that is in that team is responsible for the testing of that merge request.

From interaction 3 and forward there will be a CFT that is more focused on the testing part, this means that the testers will work more together and are not responsible for specific merge requests. Instead the testers will take on responsibility for a merge request by assigning themselves for testing that merge request in Gitlab.

When a bug is found the tester is responsible for reporting it to the developer. This is done through not accepting the merge request and writing comments explaining what the bug is and the suspected underlying issue. If there is a bigger problem that will take longer than two hours to fix a new git issue will be made for that specific problem.

Test leader
The test leader is responsible for the acceptance testing and will coordinate this with the help of all the testers. The test leader will also be the contact person for non CFT-related inquiries for testing procedures. The test leader will also be responsible for setting up meetings with the customer for performing the acceptance testing.

2 Test Methodology

2.1 Overview

The testing is mainly performed by the test team and the testing cross functional team. In the testing cross functional team, we have a analyst and software quality manager which will help validation and verification in higher level testing. The testing is done iterative which mean that all testing is done in each sprint. This will help us to continuously improve both our application and our testing through each iteration.

The work flow for testing new merge requests:
The developer posts a merge request in GitLab, in the merge request the developer must specify which feature that have been added and what they believe is important to test. The automated tests in the pipeline is performed. All merge requests needs two people to accept it, a company member and a tester. A company member performs a peer review according to the set up rules to check that the code is good enough. If the code follows company standards, the company member can accept the merge request. A tester assigns themselves to the merge request and removes the other testers if they are already assigned to the merge request by default. The predefined Selenium IDE tests are performed in the browser. Then the tester will perform additional exploratory testing on the specific feature that the merge request is related to. This is done both in desktop and in iPad mode. If a fault is found then the tester will post a comment on the merge request which specifies in which mode(iPad or desktop) the bug is found and exactly specify what the problem is so that the developer easily can find and replicate the problem by themselves. The developer who posted the merge is responsible for fixing the comment before they can post a new merge request and the same procedure will be performed again. If no problem is found then the tester can approve the merge request.

The testing of merge request will occur as soon as possible when a merge request have been posted in git. A merge request must be tested in 24 hours. If the testing team not have enough time to test then they will ask for help from the line manager (Daniel Ma).

2.2 Test Levels

In this chapter the different test levels are presented and on which level we will test the application and why we don't test certain levels. The different testing levels are unit testing, module testing, system testing and acceptance testing. This chapter will further explain the specifics of the different tests and why they are done.

2.2.1 Unit testing & Module testing

In this project unit testing and module testing will not be performed. This will not be performed because the developed units are simple and few. Therefore, unit testing which take a lot of time because each unit needs to be tested without integration with the rest of the system. So, because we have limited time and the application only contains simple units we have chosen to not perform unit testing.

Module testing is not performed because it is very time consuming because testing the integration between units we have to make drivers or stubs depending on approach. With the same reasoning as above we believe that our developed units and the integration between them are simple and therefore we do not perform testing on this level. A major reason we don't perform these test are that this project have limited time so we have to priorities which test we want to do. Therefore, we have chosen to perform test on system level because we believe that those test will result in the most efficient testing.

2.2.2 System testing

In this project the part of testing where the most time and resources are spent is the System testing. We think this is the most important level of testing for this project since it is focused on the front end and the users experience. In this chapter the different types of system tests that are going to be performed in is presented.

Pipeline
When a merge request is purposed the gitlab pipeline is automatically activated. In the pipeline, a karma test is performed. This test will check if all the developed components are part of the application. This level of smoke testing is used to ensure that no functions are unused in the application. To check if the test is accepted, enter the specific job that was created for the merge request. Check at the bottom of the pipeline terminal and see if all test was successful. If all test were successful then the automated test in the pipeline can be seen as successful. The smoke test is performed to see that the build is stable and not causing any anomalies.

Functional testing
To test and verify our requirements, the test team will produce test cases that test several requirements at the same time. This is done to make sure that the application meets the requirements that are made by the analyst team. New test cases are added at the end of each sprint to test the added requirements that have been developed during the sprint. To help build test cases, Selenium IDE is used. When creating Selenium IDE it is important to cause events in different order because the user can click on things in different order. The test cases that are written comes from each of the requirements. The test team goes through the list of requirements and makes a test case for each requirement. Some test cases will verify more than one requirement. This is done to make sure that every requirement is met so that nothing is missed to be implemented or is implemented in the wrong way. If a requirement is not implemented or fails the test it is reported to the team leader of the developers.

Non-functional requirements
To test non-functional requirements a code review will be performed for the most part. The code review will be performed by checking a requirement and then checking if it is implemented. For example, NUC-016 - Server calls shall be through open API:s for example based on REST/HTTPS, we will check if the server calls are through either REST or HTTPS. All non functional requirements will be tested separately and marked if they are done or not based on the code review.

Regression testing
Regression testing is done to make sure that new functionality doesn't affect the old. When can achieve this by always running all of our test cases for each merge request. In Selenium we set up test cases which tests a couple of requirements and these are preformed at each merge request. Additional test cases are also performed during the merge request phase. Selenium is used to help reduce the workload for the tester. In a event-driven environment the user can cause events in different orders which mean that there are a lot of different test cases. Therefore, the test team sets up test cases in Selenium IDE that can be done for every merge request.

The Selenium tests are the same each time to ensure that the new functionality that has been added is working correctly and does not affect the code in some way. There is Selenium tests that are added for each part of the system, for example testing the main menu, notifications and graphs. These tests will like said before be run every time even though if the new code that is being tested is not part of that module. If a Selenium test fail the tester will look at what is causing the problem by testing that part of the system manually. The problem is then reported to the developer of the code by making a comment in the merge request in the same way that is explained under "the work flow for testing new merge requests". New Selenium tests will continuously be added when there is a new module that is added to the application.

2.2.3 Usability testing

Usability testing is a also a big part of the project since the customer has expressed that the usability of the product is important. Since we are working iterative these tests are going to be performed at the end of each iteration when a new version of the product is released.

The usability test will be performed with the concurrent think aloud method. To get a quantitative result and measure the usability of the application a SUS-test will also be part of the usability test. The test will be performed by giving the user tasks that shall be performed in the system. They will then answer a questionnaire with 10 questions and 5 answers to each question that range from 1 (strongly disagree) to 5 (strongly agree). The max score is 100. This test can be used to see how we are improving with regards to the usability by comparing the score in the different releases. The test is going to be performed by different users each time so that they do not get familiar with the application and the score increases in that way. The concurrent think aloud questions and SUS-test will be added as a appendix.

The test will be conducted with the customer after each iteration to gather their response. Because the test is done after each iteration this enables the developer to get feedback of the not-yet-complete application by the customer. This will hopefully make the application tailored for the customer. The developed application will be used by people with different professions and technical experience. Therefore, it is important to conduct these usability tests with different level of technical experience, age and profession to get a complete picture of the true needs of the customer. Unfortunately the customer can not provide people with different profession due to specific profession does not have the time to spare. Therefore, some perspectives can be lost nonetheless the doctors, nurses and assistant nurses works close together and can give some insight on what other profession need.

Dates for test with customer:
After iteration 1
Meeting with customer, IT-staff and nurses participated: 11/10 - 21
After iteration 2
Three different tests are going to be performed with different people:
Usability test 1 with nurse: 11/11 - 21
Usability test 2 with nurse: 12/11 - 21
Usability test 3: 15/11 - 21
After iteration 3
Purpose date for usability test: 22/11 - 21
After iteration 4, Final product
Purposed date for acceptance testing: 3/12

After iteration 3 and 4 there is also going to be multiple tests with different people like in iteration 2 but we have not confirmed dates or with who yet. The tests from each iteration is then going to be compared to each other to see how we have improved or if there is a specific area that has got a lower score than before to know which areas has to be improved to get a better score next time. After iteration 1 the application was working but there were to few features to perform worth-while CTA and SUS testing. Therefore, after iteration 1, the application was shown to the customer along with the prototype from the tollgate meeting. The meetings purpose was to give the customer an opportunity elaborate on their specific needs for different parts of the website, both functional and design related. From iteration 2 and forward the customer usability testing will all follow the same structure. The customer will be given a certain amount of tasks that they will perform on the application while following CTA-protocol. Afterwards, they will do a SUS-test. The tasks that the customer will be given can differ between the iterations because more features will be developed over time and thus, we want to include these feature in the test.

Each test with the customer will result in a test report. The different test reports for each test occasion will be summarized to a single test report that will be distributed to the whole

company. This test report will be the foundation of changes in requirements and UX-design for the subsequent sprint. The test report after iteration 4 will show if the usability requirements are accepted correctly from a customer point of view.

UX-verification
Two members from the UX-team work together when reviewing user interface requirements. By looking through the closed issues on GitLab, requirements to be tested can be found since every issue is linked to a requirement by the requirement code (e.g RC-002-001). The component is then reviewed, both on a web browser setting and in an iPad setting. The developed feature is compared to the prototype and if they match, the requirement is passed and the issue is marked by a "passed review" label on Gitlab. If any faults are discovered, the review is failed and marked by a "failed review" label on GitLab. Additionally, a comment describing why the review failed is published and the issue is reopened to allow the assigned developer to fix the issues. The UX-designers will follow the prototype when reviewing the application. If needed, the UX-team enhance the prototype on Figma to assist the developer. This is done every iteration to make sure that passed requirements remain correct and that failed reviews eventually becomes passed as well.

Traceability
To keep track and trace which components have been accepted throughout the process the UX-team will be using a work sheet, found in output folder on teams, where every requirement is present. The sheet consists of the requirement, a date of the latest review, who it is reviewed by and if the requirement has passed the review. It will also include a link to the relevant issue in git with all the relevant info of who developed, tested and peer-reviewed the code.

Quality
To ensure quality throughout the testing procedure no tester is to have written part of the code they are testing. This is to minimize bias when evaluating the quality of the code. To identify the maximum number of bugs each test case must be completed by the tester before a merge request is approved. This strengthens the regression testing of the testing procedure as each function of the application must function as intended when new functionality is added.

The SUS-test conducted at the end of usability testing helps verify that the application is improving in quality between iterations. If the average SUS-score from usability testing is lower than that of the previous iteration things are not improving and processes and requirements need reevaluating before work is started for the coming iteration. The number of comments or suggestions of change on specific parts of the application from a user for each usability test is also noted to ensure that already implemented parts of the application are becoming more accepted by the user.

2.2.4 Acceptance testing

To validate the requirements the test team will perform acceptance testing. These test are performed with the customer to validate that they accept the product. The acceptance testing is the final phase of testing, this test is performed by the intended users of the system. Black box testing will be used and the test will be performed manually. This test will be done on the 3rd of December and will have the outcome of either a pass or a fail. A fail means that the customer does not approve the final product. A second acceptance test will then be performed the 8th of December.

Entry Criteria
When we conduct the acceptance testing, several criteria must be fulfilled. A list of them is presented below:

1. All features should be completed

2. No major bugs should be found on the application. If small bugs are found, a list should be available for the customer with these bugs.

3. All system testing should be done.

Exit Criteria
These criterias need to be fulfilled to consider that the application can be launched.

1. All tasks in the acceptance testing should be done and passed

2. The customer shall formally accept our application during the acceptance testing meeting.

Test scenarios will be created and each of the scenarios will be connected to one or more requirements. From these scenarios we will create more specific test cases that cover all of the scenarios. The users will perform the test cases. The test will be done in the same environment that the application is going to be used in. This means that the application will be deployed and the users can access it on their own computers/tablets. A rapport for the acceptance test with the task and the outcome of the test will be created and added to the testing documents.

3-Dec- Final acceptance testing
8-Dec- Ev. Final acceptance testing

2.3 Bug Triage

Bugs found during the testing procedure must be solved or have the underlying code generating the bug removed. This is due to all testing (except acceptance testing) taking place when a merge request is created. If a bug is found, the merge request is not approved by the tester. The merge request must then be edited to solve the bug or it is not to be merged at all. Bugs found outside of the testing procedure (e.g by non-testers) shall be reported to the testing team which will investigate the bug, create a Gitlab issue explaining what the bug is, steps to reproduce it, suspected underlying issue and deadline for when it must be solved. This is usually set to the next iteration deadline as known bugs shall not be present during acceptance testing with the customer. The Gitlab issue is assigned to the author of the code.

2.4 Suspension Criteria and Resumption Requirements

Due to testers only working on testing and the small, limited number of functions in the application the suspension criteria is only a complete system failure rendering the application non-functional. Testers only working on testing means that continued runs after a program is not working correctly is reasonable in our time schedule. This is supported by the fact of a small, limited number of functions making each test short. This means that the only resumption criteria is when a non-functional program is made functional and thereby resuming testing.

2.4.1 Exam period, 19/10-21 → 30/10-21

During the exam period neither developers nor testers will have time to perform testing or programming because they will study for their exams. Because all company participants have this period at the same time this will not create a bottleneck, only a temporary stop in the continuous delivery.

2.5 Test Completeness

A test is considered complete when it passes the Gitlab pipeline, the selenium IDE-tests and exploratory testing for each merge. Not passing one of these tests means the merge request will not be approved.

2.6 Continuous Integration

Because we used continuous integration in this project we are going to have a working application at the end of each iteration. The working application will be used to conduct usability testing with the customer as previously described. The feedback we receive at the usability test will be summarized to a test rapport that will be distributed to the UX-team. The UX-team will create new and update our old prototypes based on the test rapport so that the developer easily understands how our requirements shall be implemented. If needed the test rapport will also be distributed to the analyst team if some requirements needs to be revised based on customer feedback. The process with testing and then forwarding information to the UX-team that will update our prototypes will occur for every iteration which will incrementally enhance our application with feedback from the customer.

3 Test deliverables

This chapter regards deliverables created for customer, CEO, company and/or testers.
In the beginning of each iteration we intend to have a meeting with the customer where they will do a System Usability Scale (SUS) test with the Concurrent Think Aloud (CTA) protocol. Every iteration is roughly two weeks so we planned to do these test four times with the customer during the project. This is to test the new features that have been developed during the iteration that has been so that the customer can give feedback to what they like and don't like. The feedback from the customer will then be delivered to the developers and UX-designers so that they can continue to improve the features and modules. This feedback is summarized by the testers, but the original meetings are recorded if possible and uploaded to enable the possibility to go back and analyze a specific meeting. All results from the meetings will be saved to the files section of the company "Testing" Teams channel.

The first customer meeting did not feature a SUS-test with a CTA protocol as the application was deemed too bare-bones to result in any meaningful feedback from a customer. Instead, a customer meeting was conducted where design ideas from the Figma protype and the application were evaluated on a detail level and written down. This interview was done with nurses and IT-staff from the customer. The notes from the meeting were then passed to the UX-team for evaluation.

System testing is done at each merge request and bugs found will be commented in code and as a comment in the merge request text fields. The creator of the merge request are then notified by the automated mail sent to their Gitlab accounts of the discovered bug.

The test plan is created to standardize and inform of the testing procedure. It is written by the testers. It is published both in the Gitlab repository as well as the Teams output folder for CEO and company to be informed about how testing is conducted. When a testing procedure is changed or created, it is written down into the test plan by testers so no procedure goes undocumented.

3.1 Milestones

3.1.1 Pre-study

In the pre-study phase of the project there was no code to test. At that time the testers spent their time in preparation for the upcoming iterations. This preparation consisted of deciding upon which testing tools that were to be used during the project lifespan and how the process of testing was going to be implemented in the project. These processes aimed to answer the question of how different tests were going to be performed, what defines a passed test and how bugs are to be reported. The first part of the test plan was written during the pre-study phase. A workshop is going to be held to understand which tools that we should use during this project. Further self-studies is going to be needed to get a good understand of software testing.

3.1.2 Iteration 1

During the first iteration several process is started and therefore not much code is going to be developed. Therefore the main focus is to set up a plan for how the testing should be performed. The plan should be specified in the test plan. At the end of the iteration a meeting with the customer is going to be held to get further opinions from the customer to refine our requirements. In this iteration the usability testing is specified how it should be performed.

3.1.3 Iteration 2

The goals for the second iteration are to refine the test plan to better find bugs and improve our process. We are going to continue to test new code. In this stage the development has increased in speed and therefore more time needs to be set aside for actual testing. In this stage the

pipeline is created to have automated test. At the end of the iteration, the tester will perform usability testing with the customer to understand how usable our application is. Each tester will perform a test on their own and write a test rapport of how well the test went and then all test rapports is summarized to a single rapport. The summarized rapport is distributed to the rest of the company.

3.1.4 Iteration 3

In this iteration we should further refine our testing procedures and continue testing all new developed features. We should also further enhance our documents and usability testing. If the pipeline is not done at this stage it is important to finalize it now.

3.1.5 Iteration 4

In the last iteration the test team should make sure that the final acceptance testing is performed. In this iteration the test team should check that every requirement that's created are verified. Setting up final acceptance testing with customer for the final product and performing the test. Further continue to test merge requests. In addition to the final system and acceptance testing we will schedule one more occasion for system and acceptance testing if the customer doesn't approve the product during the first acceptance test. There will be approximately one week between these occasions so that the developers have time to add or change features and we can test and do a final system testing before the acceptance testing.

To further concretize the test plan and the most important dates at the end of this project we have set up milestones dates for these events:
22-Nov - Freeze branch to be tested & review task for acceptance testing
23-Nov - Finish test plan v2.0 for reviewing
24-Nov - Totally finished test plan v2.0
25-Nov - Summarize test reports (from SUS-tests)
30-Nov - Test plan v3 ready for review
1-Dec - Totally finished test plan v3.0 & preparation for Final system testing
2-Dec - Final system testing
3-Dec - Final acceptance testing
7-Dec - Ev. Final system testing
8-Dec - Ev. Final acceptance testing

4 Rescource and Enviroments

This chapter regards testing tools and versions used by each tester to ensure the same prerequisites for each test done by the testers.

4.1 Testing Tools

The requirement tracking tool is verbal communication between testers and analysts where either a test is shown proving that a requirement is reflected in the application or by evaluating the application through existing functionality or properties the application possess. The requirement is then marked as done in the analysts document of requirements.

One of the two bug tracking tools is the merge request comment fields where information about what the bug is, steps to reproduce it and suspected underlying issue is relayed to the authors of the code and creator of merge request. The other bug tracking tool is git issues containing the same information as the first bug tracking tool and is used when bugs are found outside the testing procedure.

The automation tools are firstly the automated test done in Angular and implemented in the Gitlab pipeline which is conducted for each merge request. These tests are primarily used to ensure that the application is still having its basic functionality and is not the primary focus of the testing procedure. This is done due to the heavy focus on front-end testing as the customer application is heavily focused on front-end development. The second automated tool is a list of Selenium IDE tests and are run by the testers after they pull the merge request to their local machine.

4.2 Test Environment

- Angular 12.2.12
- Node 16.13.0
- Selenium 3.17.0

5 Terms/Acronyms

A list of terms or acronyms used in the test plan.

- CFT - Cross functional team
- SUS - System usability scale
- CTA - Concurrent think aloud
- UX-team - CFT2 from iteration 3