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	A. Telenius	2021-11-26
2.0	O. Löfgren, E. Parö	New chapter usability testing, changed chapter acceptance testing	A. Telenius	2021-12-01
2.1	A. Telenius	Added Quality part in 2.2.3	O. Löfgren	2021-12-01
2.2	S. Gharedaghi	Grammatical Review	A. Telenius	2021-12-06
2.3	A. Telenius	Added explination for post-project plans in 2.2.1 and 2.3	O. Löfgren	2021-12-06

1 Introduction

In this document, an explanation of 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 in 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 are used and when a test is considered complete.

We base our testing procedure on the V-model. This model is also called the 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 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 means that it is performed iteratively. Therefore, the testing is also done iteratively. 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 system's usability and satisfyingly visualize data. Therefore the testing will focus on measuring if the developed application achieves the requested levels. The testing will also focus on the functionality of the system. This means that most of the testing in this project is related to System testing which tests the integration between different modules, and acceptance testing that verifies the requirements. These tests and how they are going to be performed in the project are going to be explained later in this document. 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 module testing and unit testing. This is due to the focus on usability and displaying data. This will result in a low amount of components needed to be integrated during the project together with the basic level of functionality and output of the implemented functions. Unit and module level testing can be time-consuming, and we have a limited amount of time in the project is another reason why we will not conduct testing on these levels.

The customer does not want us to focus on how to store data and database management, and therefore tests on the application back-end will not be conducted. Because storing data is not crucial 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 is 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 merge requests are created from a team's branch to the develop branch, the tester in that team is responsible for testing that merge request.

From iteration 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 responsibility for a merge request by assigning themselves to test that merge request in GitLab.

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 to perform 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 an analyst and software quality manager which will help verification with requirements in higher-level testing. The testing is done iterative which means that all testing is done in each sprint. This will help us to continuously improve both our application and our testing through each iteration.

Main workflow for testing new merge requests:
The The developer posts a merge request in GitLab; in the merge request, the developer must specify which feature has been added and what they believe is essential to test. The automated tests in the pipeline are performed. All merge requests need two people to accept them, a company member and a tester. A company member performs a peer review of the code according to the setup rules. 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 on desktop and in iPad mode. If a fault is found, the tester will post a comment on the merge request which specifies in which mode(iPad or desktop) the bug is found and precisely specify how to replicate the problem so that the developer can easily find the problem by themselves. The developer who posted the merge is responsible for fixing the comment and replies to the comment, saying that the problem is fixed. Then the tester will perform the testing procedure described above again. If no problem is found, the tester can approve the merge request.

The testing of merge requests will occur as soon as possible when a merge request has been posted in git. A merge request must be tested in 24 hours. If the testing team does not have enough time to test, 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 System testing

The leading testing will be performed on the system level in this project. We think this is the most critical level of testing for this project since it is focused on the front-end and the user's experience. In this chapter, the different types of system-level tests that are going to be performed are presented.

Pipeline
When a merge request is posted, 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 is a smoke test and is used to ensure that no functions are unused in the application. To check if the test is accepted, enter the specific job created for the merge request. Check at the bottom of the pipeline terminal and see if all test was successful. If all tests 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.
Features and automated tests that are not implemented before the end of the project will be noted and handed over to customer after project completion.
Functional testing
To test and verify our requirements, the test team will produce a list explaining how the re-

quirement is tested. Several linked requirements will be tested together in a test case. 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 a different order because the user can click on things in a different order. All requirements will have an explanation text of how they are going to be verified and which test cases they are part of. This information will be found in the document: *Testing_TestCases*

Non-functional requirements
Each non-functional requirement will be tested separately, and how it is tested will be documented in the excel file *Testing_TestCases*. The testing procedure we will mainly use is code review which will be performed by checking a requirement and then checking if it is implemented by looking in the code. Another way we will verify requirements, if possible, is to simply check on our application if it is fulfilled. 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 in the code. All non-functional requirements will be tested separately and marked if they are passed or not in the test case file.

Regression testing
Regression testing is going to be done to make sure that new functionality does not affect the old. When can we achieve this by always running all of our selenium test cases for each merge request? In Selenium, we set up test cases that test a couple of requirements, and these are performed at each merge request. Selenium is used to help reduce the workload for the tester. In an event-driven environment, the user can cause events in different orders which means that there are a lot of different test cases. Therefore, the test team will set up test cases in Selenium IDE that can be done for every merge request.

The Selenium tests will be updated and added iterative throughout the project. If new features are implemented, a selenium will be developed to test that feature. If a small change is made to the system then the selenium test can be updated if necessary. At the end of the project, there will be a Selenium test for each part of the system, for example, testing the main menu, notifications, and graphs. When running the selenium test, paying close attention to see that the system is working correctly is vital. If a problem is found, it is reported to the developer of the code by commenting on the merge request in the same way that is explained under "the workflow for testing new merge requests". New Selenium tests will continuously be added when a new module is added to the application.

2.2.2 Usability testing

Usability testing is also a big part of the project since the customer has expressed that the product's usability is essential. Since we are working on 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 via teams and conducted with one tester and one customer. The meeting will be recorded so that the tester can go through the recording afterward and write down the customer's response in a usability report. The customer will be provided with different tasks, and they are instructed to think aloud according to the CTA protocol. The tasks are created to test the whole application and will be revised before each usability test after the end of each iteration. This is done because new features will be developed over time and should be reflected in the usability tests. A template will be created with the tasks and some initial questions about the customer. A SUS-test will also be performed at the end of the usability test to get a quantitative result and measure the usability of the application. 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 ranging 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 regard to 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 base their score on how the application has changed. The tasks can be found in the output file in teams. Because we are constantly developing new features, the tasks are going to be changed between the usability tests. This is a source of error that needs to be taken into consideration when reviewing the SUS score. This is because each usability test from the user will be different with regards to the task they perform.

After each iteration, the test will be conducted with the customer to gather their response. Because the test is done after each iteration, the developer can get feedback on the not-yet-complete application from 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 essential to conduct these usability tests with different levels of technical experience, age, and profession to get a complete picture of the customer's actual needs. Unfortunately, the customer can not provide people with different professions due to specific professions that do not have time to spare. Therefore, some profession-specific feedback can be lost. Nonetheless, the doctors, nurses, and assistant nurses work closely together and give some insight into what other professions 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 usability testing: 3/12

Each test with the customer will result in a test report. The test report shall contain whether the customer could complete each task and their feedback. The test report will follow a standardized template that the test team will create. The different test reports for each test occasion will be summarized to a single test report for that iteration. This report will be distributed to the UX team. This test report will be the foundation of changes in UX design for the subsequent sprint. The UX team will create prototypes so that the developers can easily understand our ambitions. This will help us continuously improve our application from the response from the customer.

UX/design-verification
Because UX and presentation of data is an essential part of this project, we have a specific procedure for this verification. Two members from the UX team will work together when reviewing the UX and design of the application. 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 will be reviewed, both on a web browser setting and in an iPad setting. The review will compare the developed feature 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 will clarify the prototype on Figma to assist the developer. This will be performed for every iteration to ensure that passed requirements remain correct and that failed reviews eventually become passed.

Traceability
To track and trace which components have been accepted throughout the process, the UX team will use a worksheet found in the output folder on teams, where every requirement is present. The sheet consists of the requirement, the latest review date, whom 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.

To make sure that every requirement is going to be tested and verified, the test will go through the file with all the test cases. Because every requirement is mentioned, it is easy to see if and how it is tested. It will also include a link to the relevant issue in git where information of who created, tested, and peer-reviewed the code.

Quality
To ensure quality throughout the testing procedure, no tester is going to test their own code. This is to minimize bias when evaluating the quality of the code. To identify the maximum number of bugs, the tester must complete each test case before a merge request is approved. This strengthens the regression testing of the testing procedure as each application function 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 and suggestions of change on specific parts of the application from the user is also noted for each usability test. This ensures that already implemented parts of the application are becoming more accepted by the user.

2.2.3 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, and this test is going to be 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 on the 8th of December to see if any changes made during the last day result in the customer accepting the product.

Entry Criteria
When we conduct the acceptance testing, several criteria must be fulfilled. If we not manage to fulfill these criteria before the date of acceptance testing, we will have to postpone the test. If this event occur ,we will perform an additional usability test with the customer instead and perform the acceptance test on the 8th of December instead. A list with the entry criteria is presented below:

1. All features should be completed
2. No significant bugs should be found on the application. If minor bugs are found, a list should be available for the customer with these bugs.
3. All system testing should be finished.

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.

The test team will create a template for the acceptance test with real-life scenarios that the customer will perform on the application. The real-life scenarios will be created based on the user stories and connected to several requirements. The customer will also give the chance to roam the application freely. If the customer can complete the task and accept the design, the task is defined as accepted. If all tasks are accepted, then the application is defined as accepted. 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 computers/tablets from their workplace. A report from 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 tester does not approve the merge request. The merge request must then be edited to solve the bug or 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. Suppose a bug is found during the testing procedure but does not relate to the specific merge request. It will follow the same procedure described above minus reporting to the test team because the test team found the bug.

Bugs found but not resolved within a week of code stop will be reported in a post project document delivered to the customer. Each of the bugs will have a description, suspected underlying issue and steps to replicate the bug.

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.

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 simultaneously, 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. In addition, it also has to pass the UX review. 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 in a test report 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 quickly understands how our requirements shall be implemented. If needed, the test report will also be distributed to the analyst team if some requirements need 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.
At the end 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 are planning to do these tests four times with the customer during the project. This is to test the new features that has been developed during the iteration that has been so that the customer can give feedback on what they like and do not 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 initial 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 prototype and the application were evaluated on a detailed 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 going to be performed for each merge request; any bugs found will be commented in code and as a comment in the merge request text fields. The creator of the merge request is then notified by the automated mail sent to their GitLab accounts of the discovered bug. If

The final system testing will be performed and will result in an excel file with all the requirements and how they are tested. The file will have information on whether the requirement is completed or not. The document name is *Testing_TestCases* and can be found in the output folder.
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 the 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 will be no code to test. The testers will spend their time in preparation for the upcoming iterations. This preparation consisted of deciding upon which testing tools were to be used during the project lifespan and how the process of testing will be implemented in the project. These processes aimed to answer 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 is going to be written during the pre-study phase. A workshop will be held to understand which tools we should use during this project.

3.1.2 Iteration 1

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 find bugs better and improve our process. We are going to continue to test the new code. In this stage, the development has probably increased in speed, and therefore, more time needs to be set aside for actual testing. In this stage, the pipeline is going to be created to have the automated test. At the end of the iteration, the tester will perform usability testing with the customer to understand how usable our application is. To be able to conduct these tests, a plan for usability tests needs to be created. Each tester will perform a test on their own and write a test report of how well the test went, and then all test report is summarized into a single report.

3.1.4 Iteration 3

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

3.1.5 Iteration 4

In the last iteration, the test team should ensure that the final acceptance testing is performed. In this iteration, the test team should check that every created requirement is verified. We set up final acceptance testing with the customer for the final product and perform 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 does not 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 concrete the test plan with 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 exact 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 possesses. The requirement is then marked as done in the analysts' requirements document.

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 the 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 is 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
- Karma 2.0.0

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