Testing Plan

TDDC88 Programutvecklingsmetodik

Company 3 - Testing Group

October 18, 2024

1 Document Change History

Note: This change history table was generated by Autoleaf AI under the supervision of the Technical Writer. Only the most significant changes are highlighted, check the readme.md, found in gitlab, for more detailed information.

Ver.	**Date**	**Modified Areas**	**Changed By**	**Description of Changes**
2.1	2024-10-18	Workflow, Traceability, System Testing, Risk Assessment	Testing Group	Clarifies testing processes, expands on traceability, details the system testing approach, and incorporates risk assessment related to limited time and automated testing knowledge.
2.0	2024-10-10	Bug Handling, Time Plan, External Libraries	Testing Group, Martin	Introduces a detailed bug handling process, refines the time plan with iteration details, and adds a section for testing external libraries.
1.0	2024-10-04	Overall Structure	Testing Group	Establishes the initial structure and content of the Testing Plan, including sections for different testing types, a time plan, and risk assessment.

Contents

1 Document Change History **2**

2 Introduction **5**

2.1 In Scope 5

2.2 Out of Scope 5

3 Test Process **5**

3.1 Workflow 5

3.2 Bug Handling Process 6

3.2.1 Selected bug tracking tool 6

3.2.2 Process for reporting, prioritizing, and assigning bugs 6

3.2.3 Workflow for bug resolution and verification 6

3.2.4 Metrics for tracking bug trends and resolution efficiency 6

3.2.5 Bug tracking for continuous improvement 6

3.3 Traceability 6

3.3.1 Requirement-to-Test Traceability 7

3.3.2 Test Traceability Matrix 7

4 Unit Testing **7**

5 System Testing **7**

6 User Experience Test **8**

7 Testing External Libraries **8**

8 Acceptance Test **8**

8.1 Acceptance Test Process 8

8.1.1 Execution Plan for Acceptance Test 8

9 Time Plan **9**

9.1 Iteration 1 9

9.2 Iteration 2 9

9.3 Iteration 3 10

9.4 Iteration 4 10

9.5 Special Considerations 10

9.6 Summary of Milestones 10

10 Risk Assessment **10**

2 Introduction

2.1 In Scope

These are the main components the following Testplan has been centered around

- Qualitative usability
- The basic functions of each unit

2.2 Out of Scope

The following components will not be a centre piece of the Testplan

- The security aspect
- Quality of the code

3 Test Process

3.1 Workflow

In large projects, a well structured workflow is essential for performing an organizing tasks in an efficient way. This section outlines the workflow for testing and the workflow for executing and evaluating test cases. The testing will be divided into stages to track the process in the workflow.

Before initiating the workflow,it is important to define and make sure the rolls for testing are clear. This ensures tasks and responsibilities are divided and it is clear who will do what to prevent overlap in work.

Then the first stage of testing is understanding the project requirements. This step is crucial for developing the test cases and must be done in collaboration with analysts. Once the requirements are defined and have been properly understood, the process will continue to the next stage, developing a testing strategy. This involves determining what types of tests will be performed, scheduling the execution and layout of the testing process.

Next, the testing moves to creating test cases for the selected testing methods. This involves writing a thorough script for the test and describing how the test will be executed. Right before executing the test, the testing environment has to be set up. This will vary depending on what type of test it is but this involves setting up necessary tools and resources.

Once the environment is ready for testing the tests are executed. This process is done in different ways depending on the type of test. For instance, acceptance tests will be performed together with customers, user tests involving external participants and automated tests will run without monitoring.

After tests are complete, the final stage is reviewing results and addressing bugs and issues. This includes determining if requirements must be retested or if they can be labeled as achieved.

3.2 Bug Handling Process

3.2.1 Selected bug tracking tool

Bugs will be tracked using the Microsoft Planner tool. This will be a way to track unexpected problems that come up in the project and have a way of managing them efficiently. This tool will be used to label the incidents in how important they are and in what order they will be handled. This is a way of tracking that will be efficient for development since Microsoft tools are used for more parts in the project.

3.2.2 Process for reporting, prioritizing, and assigning bugs

The way that bugs will be reported is by creating an incident in Microsoft planner. After this is created the bug will be labeled by the creator on how important it is and how it should be prioritized. With Microsoft planner the team can create different boards for different types of bugs or label as severity. After this, for efficient communication, the scrum meetings will be used to discuss these incidents and who will be responsible for the resolution of the bug. This way every developer will have knowledge of the bug and a resolution can be discussed with the entire group. If bugs are not solved properly or other developers are not aware of the issue then that can cause more issues and unnecessary work.

3.2.3 Workflow for bug resolution and verification

The bug resolution will be solved in different ways depending on the issue, the developer, together with the team, will come up with a solution and the code will be fixed. After debugging, the member will merge the fixed code with the old code. The incident in Microsoft planner will be closed. The other developers can after that continue with their work in that code without being affected by the incident.

3.2.4 Metrics for tracking bug trends and resolution efficiency

Tracking bug trends will be done manually from the incidents created in Microsoft planner when or if there is something that needs tracking. The aim for the project is that the debugging is done thoroughly so that there are no trends in bugs to track. If needed, the tracking of trends will be done in excel. The Microsoft planner board that need analysis and tracking of trends will be exported into excel. There the developers can see trends in metrics and analyze the data.

3.2.5 Bug tracking for continuous improvement

Bug tracking is important for continuous improvements as this allows the team of developers to see and understand the issues that has been raised and how they have been resolved. This means that if this issue comes up again there will be a quick resolution, but perhaps the issue will never come up again as the team knows what to avoid. The bug tracking enforces this when the group of developers discuss this during the scrum-meetings and have this communications on bugs and debugging.

3.3 Traceability

The plan is to implement traceability in two key ways to ensure that all tests are directly linked to the corresponding requirements and user stories, providing a clear understanding of which features have been verified.

3.3.1 Requirement-to-Test Traceability

Each requirement defined in the project will be linked to a specific test or set of tests. This ensures that every requirement is accounted for and tested through a structured process. For example, when a test is executed, it will be easy to trace back to which requirement the test is verifying. Additionally, each requirement will be associated with a relevant user story, ensuring that the user perspective is always kept in focus during the testing process.

3.3.2 Test Traceability Matrix

To effectively manage and track the testing process, we will use a Test Traceability Matrix. This matrix will provide a comprehensive overview of all tests, showing the following details:

- Which tests have been performed.
- When the tests were executed.
- Whether each test passed or failed.
- Which requirement each test covers.

By using this matrix, we will have a clear view of testing progress and be able to easily identify any missing or untested requirements. Additionally, it helps ensure that any failed tests are properly addressed, preventing gaps in coverage. This approach provides both clarity and accountability throughout the testing phase, ensuring continuous improvement and alignment with the project's goals.

For automated tests, the *Test Traceability Matrix* will not be updated every time they are run. Instead, it will only be updated when there is a change in the outcome. For example, if an automated test has been consistently successful but suddenly fails, the *Test Traceability Matrix* will be updated at that point. This ensures that the matrix reflects significant changes in the testing results, avoiding unnecessary updates while still maintaining an accurate record of test performance.

4 Unit Testing

During the sprints, the code—divided into 4 sections—will need to be tested to ensure continuous integration, primarily focusing on each section separately, with system tests being conducted less frequently. The approach to executing these unit tests is still under discussion until actual tests are performed, but the current plan is to use a Git pipeline for unit testing. Given that the code is segmented, each section requires its own set of unit tests. By utilizing an xUnit-based framework (such as JUnit, NUnit, or pytest), we can maintain a consistent and structured method for testing each code segment. This will automate the testing process early on, freeing up time later and providing a continuous and clear understanding of how the code development is progressing. Additionally, since there are more testers than developers in the company, this automated framework will streamline the workflow and help manage the volume of tests efficiently.

5 System Testing

System testing tests the entire system when the four components are put together. This involves testing the different parts of the project to make sure it all works together. Together with unit testing we can make sure all components work and that the entire integrated system works . The system tests will be performed using functional testing and non-functional testing. The system tests will be executed as a way to control requirements to see that the entire integrated system aligns with these. System testing will be used to ensure there is a deliverable system at the end of each sprint, therefor these tests will be spread out during the project.

6 User Experience Test

7 Testing External Libraries

Throughout the development process, it is essential to ensure that all external libraries integrated into the application are properly tested. These libraries, such as those handling JSON parsing (e.g., "jansson"), must be validated to guarantee they perform their intended functions correctly within the system.

Testing of external libraries will be conducted continuously as new libraries are introduced. This approach ensures that any potential issues are identified and resolved early. Close collaboration with the developers will be maintained to verify that each external component integrates seamlessly with the application and meets the required standards for functionality and reliability.

8 Acceptance Test

An acceptance test is conducted to determine whether a system or product meets Axis' specifications and is ready for delivery. These tests will be held after each prototype stage, following unit, integration, and system testing.

8.1 Acceptance Test Process

We plan to implement structured development cycles, where an acceptance test will be conducted after each prototype stage. These tests will focus on evaluating the features that have been completed during the cycle. The primary objective of the acceptance test is to gather detailed feedback from Axis, ensuring that the product is progressing in line with their specifications and expectations. By continuously validating the development with Axis, we aim to ensure that any necessary adjustments are made early, facilitating smoother progress towards the final product.

8.1.1 Execution Plan for Acceptance Test

In the later stages of each development cycle, the testing team within the company will conduct internal tests. The primary objective is to evaluate the product from a customer perspective, assessing which user stories have been completed and identifying functionalities that are still under development. Based on these findings, the testing team will create an acceptance test designed to present Axis with a user-centric view of the application's current capabilities. Axis will then provide feedback on how our solutions align with the established requirements, offering insights that we will relay to the developers to guide further improvements.

Option 1: Live, face to face meeting
This option could be defined as the booking of a certain amount of meetings, where the customer will have the chance to get hands-on experience with the application and go through a variety of pre-set tasks, with the chance of questioning and feedback. The pros of this option is the level of understanding a customer actually gets from hands-on experience, along with the change of immediate feedback and

discussion, which could be very valuable. The cons of this option is of course the amount of time this option actually takes, having to book and go through with a physical meeting.

Option 2: Digital meeting
Option 2 is having digital meetings where a tester in the company will be live with the system to show the user. The user test will be performed similarly to if the test was live, but this will be more of a visual walkthrough of the system and the features. Axis will here have the possibility to leave their feedback and comments on what they like and what they do not like. However, if this option is chosen then there will possibly be a decrease in performance as the computer will have to work harder. Showing the system live or in video will not create the same issue. The tester at axis will not have the ability to test and play around with the system, therefore only seeing the parts of the system that the tester from Company 3 shows them. This can be an efficient way in testing as there will be no time spent for the customer to have to learn the system and the time will be well spent.

Option 3: Over email, videos, pictures and text
This option could be described as the sending of pictures and videos of testing of the application, with the aim to show the results of functionality and usability of the application at that certain time. The pros of this option is the resource management aspect, since qualitative requirements and testing should ensure that each iteration will produce a qualitative product. The cons of this option is of course that the customer will have little hands-on experience with the application, which could present hidden defects a video or picture could not.

Axis decision
Axis has decided to proceed with a combination of face-to-face and digital meetings for the acceptance test. They believe that these options will provide a more comprehensive evaluation of the product, allowing for hands-on experience and detailed feedback. However, they have opted not to use email, videos, or pictures, as these methods may not provide sufficient insight into the system's usability and functionality.

9 Time Plan

The testing process will follow the project's structured development cycles, with each cycle concluding with an acceptance test. There will be four iterations, each designed to refine the system and ensure that it meets the required specifications from Axis.

9.1 Iteration 1

- Duration: Week 40 - Week 41 - Focus: Creating the base of the Quality Assurance plan, starting creating tests to cover the requirements. Rough draft of the testing plan, focusing on the acceptance test and unit testing since this will be what is testable at this stage. Gaining technical knowledge in how automated tests can be done. - Acceptance Test: Conducted during Week 42.

9.2 Iteration 2

Duration: Week 42 - Week 45
Focus: finishing up most parts of the test plan. Implementing automated tests and starting with system testing.
Acceptance Test: Conducted during Week 46.

9.3 Iteration 3

Duration: Week 46 - Week 47
Focus: Starting with user experience testing. Gaining input from users to ensure that the product meets customer demands,
Acceptance Test: Conducted during Week 48.

9.4 Iteration 4

Duration: Week 48 - Week 49
Focus: Final testing ensuring as few bugs are
Acceptance Test: Conducted during Week 50.

9.5 Special Considerations

Post-Exam Push: Due to high workload during the first study period (Study Period 1), we anticipate dedicating additional resources to finalizing the test plan and completing remaining tests directly after the exam season. This is planned for Week 2 (Study Period 2), where the team will have more available time to focus on testing tasks.

9.6 Summary of Milestones

- **Week 42**: First acceptance test
- **Week 46**: Second acceptance test
- **Week 48**: Third acceptance test
- **Week 50**: Final acceptance test and test plan completion

10 Risk Assessment

It is important to identify and mitigate risks that may affect the success of the testing process. Below are some potential risks that have been identified:

Limited Time in Study Period 1
Due to a heavy academic workload during Study Period 1, the team has limited time to dedicate to this project. This constraint could lead to delays in the completion of tests and documentation. To mitigate this, we have planned a **Post-Exam Push** in Week 2 of Study Period 2, where additional resources will be allocated to finalize any remaining tasks, including testing and bug fixing.

Technical Knowledge for Automated Testing
Another risk is the need for the team to acquire sufficient technical knowledge to implement automated testing in the Git pipeline. As automated testing is an integral part of our development process, any delays in learning or implementation could affect the consistency and reliability of our testing framework. To mitigate this, we will prioritize learning this system early in the project and seek support from developers to ensure we are prepared to use the Git pipeline effectively for automated testing.

[1]

References

[1] Pang-Ning Tan. *Introduction to data mining*. Pearson Education Limited, 2014.