ADHD Taks Manager MASTER TEST PLAN

Version Information

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Remarks** | **Author** |
| 0.1 | <<Date of Change>> | <<Describe what changes were made and why>> | <<Author>> |
| 0.1 | 5/4/2023 | Initial | Matthew Neil |

Executive summary

|  |
| --- |
| **Project objective**  <<Summarize the goal the project is intended to achieve.>>  The ADHD Task Manager Application is designed to keep people who suffer from ADHD on track of their tasks. The application’s goal help manage the user’s time and organize their thoughts to help them keep focused. One of the ways this will be done is by implementing a rewards system. |
| **Test approach**  <<Briefly outline the test levels to be used>>  We will use all four testing levels for this project to ensure the application runs smoothly and as intended:   1. Unit Testing – This will test the individual functions of the application. 2. Integration Testing – This will test a scenario where multiple functions are used together. 3. System Testing – This will test the application runs smoothly and as it should on a mobile device. 4. Acceptance Testing – This test will ensure a polished product. Looking to find bugs in user data rather than the application itself eg, spelling errors. |
| **Test objectives**   * <<Itemize the major risks addressed by testing and the outcomes to be achieved>> * Ensure the Task List itself works. Adding, Removing and Completing tasks is the core functions of the application. Failure to test this functionality will ruin the user’s experience on the application and render it useless. The task functions of the application must work reliably and seamlessly. This is the most important part of the application. * Ensuring user data is stored for re-use. The application must remember what tasks have been saved and when to send reminders. Once again, the application is rendered useless unless it can save the data. * Application must send achievement rewards for completing tasks. A large attraction of the application is the rewards for completing tasks. The risks addressed in this objective is if the rewards system does not work, then the application will be just another to-do list application. |

Table of Contents

[1 Introduction 4](#_Toc37168483)

[1.1 Project and project objective 4](#_Toc37168484)

[1.2 Objective of the master test plan 4](#_Toc37168485)

[2 Documentation 5](#_Toc37168486)

[2.1 Basis for the master test plan 5](#_Toc37168487)

[2.2 Test basis 5](#_Toc37168488)

[3 Test strategy 6](#_Toc37168489)

[3.1 Risk analyses 6](#_Toc37168490)

[3.1.1 Product Risk Analysis 6](#_Toc37168491)

[3.1.2 Technical Risk Analysis 7](#_Toc37168492)

[3.2 Test strategy 8](#_Toc37168493)

[4 Test Levels 9](#_Toc37168494)

[4.1 The <name test level> 9](#_Toc37168495)

[4.1.1 Entrance and Exit Criteria 9](#_Toc37168496)

[4.1.2 Test Environment 9](#_Toc37168497)

[4.1.3 Test Objectives 9](#_Toc37168498)

# Introduction

## Project and project objective

<<Outline the objective of the overall project>>

The ADHD Task Manager Application is designed to keep people who suffer from ADHD on track of their tasks. The application’s goal help manage the user’s time and organize their thoughts to help them keep focused. One of the ways this will be done is by implementing a rewards system.

## Objective of the master test plan

The objective of the Master Test Plan (MTP) is to inform all who are involved in the test process about the approach, the activities, including the mutual relations and dependencies, and the (end) products to be delivered.

The master test plan describes the test approach, the activities and (end) products.

# Documentation

This chapter describes the documentation used in relation with the master test plan. The described documentation concerns a first inventory and will be elaborated, actualized and detailed at a later stage, during the separate test levels.

## Basis for the master test plan

<< Consider the Project Vision, requirements analyses, specific project or test planning, an implementation plan or other documents of importance. >>

The following documents are used as basis for this master test plan.

|  |  |  |  |
| --- | --- | --- | --- |
| **Document name** | **Version** | **Date** | **Author** |
| **LCOMProject Vision.docx** | 1.0 | 29.3.23 | Deepak Chand, Sam McConchie, Corie Rhodes, Matthew Neil |
| LCOMInitialRequirements.docx | 1.0 | 29.3.23 | Deepak Chand, Sam McConchie, Corie Rhodes, Matthew Neil |

## Test basis

The test basis contains the documentation that serves as basis for the tests that have to be executed. The overview below describes the documentation that is the starting point for testing.

<< Consider requirements analysis documents, technical designs, data models, system architecture, user manuals, ‘old’ testware and AO-procedures >>.

|  |  |  |  |
| --- | --- | --- | --- |
| **Document name** | **Version** | **Date** | **Author** |
| **LCOMProposedArchitecture.docx** | 1.0 | 29.3.23 | Deepak Chand, Sam McConchie, Corie Rhodes, Matthew Neil |
| **LCOMRiskList.xlsx** | 1.0 | 4.4.23 | Deepak Chand |

<< If it’s already definite that the test basis is (partly) missing or is of poor quality, also mention here the measures taken in this area, for example interviews to get the necessary information on the table. It is also possible to mention the document type if the document is not yet available at the time of writing this document. >>

# Test strategy

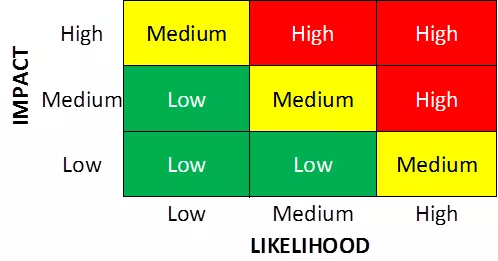
The time available for testing is limited; not everything can be tested with equal thoroughness. This means that choices have to be made regarding the depth of testing. Also, it is strived to divide test capacity as effective and efficient as possible over the total test project. This principle is the basis of the test strategy.

The test strategy is based on risks: a system has to function in practice to an extent that no unacceptable risks for the organization arise from it. If the delivery of a system brings along many risks, thorough testing needs to be put in place; the opposite of the spectrum is also true: 'no risk, no test'.

The first step in determining the test strategy is the execution of a product risk analyses. This is elaborated in §3.1.

The test strategy is subsequently based on the results of the risk analyses. The test strategy lays down what, how and when (in which test level) is being tested and is focused in finding the most important defects as early as possible for the lowest costs. This can be summarized as testing with an optimal use of the available capacity and time. The test strategy is described in §3.3.

## Risk analyses



**Low** is given the Value of **C.**

**Medium** is given the value of **B.  
High** is given the value of **A.**

### Product Risk Analysis

The product risks are determined in cooperation with the client and the other parties involved. Product risks are those risks associated with the final product failing to meet functional requirements and required system quality characteristics (NFRs) This product risk analyses (PRA) is comprised of two steps:

<<Make an inventory of the risks that are of interest and classify the risks.>>

|  |  |  |  |
| --- | --- | --- | --- |
| **Product Risk** | **Characteristic** | **Description** | **Risk Class** |
| 1 | Usability | Too difficult to use | B |
| 2 | Reliability | Unable to save tasks. User Input failure. | B |
| 3 | Performance | Application crashing regularly, Functions not working properly | A |
| 4 | Supportability | Unable to open supporting documents, or reminder information not working | C |

The extent of the risk (the risk class) is dependent on the chance of failure (how big the chance is that it goes wrong?) and it depends on the damage for the organization if it actually occurs.

### Technical Risk Analysis

Technical risks are determined in cooperation with the analyst/designers and programmers involved. Technical risks are development risks associated with failing to create a system that behaves according to specifications derived from requirements. (I.E. those aspects of development that pose particular challenges.) This technical risk analyses (TRA) is comprised of two steps:

<<Make an inventory of the risks that are of interest and classify the risks.>>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical risk** | | **Risk Area** | **Description** | **Risk Class** |
| 1 | Task Managing | | Problems with Adding, Editing, Deleting Tasks | A |
| 2 | Saving Task Data | | Task Data not recorded or saved. | A |
| 3 | Rewards System | | Rewards not giving proper rewards or unable to view achievements | B |

## Test strategy

For each risk from the product and technical risk analysis the risk class determines the thoroughness of the test. Risk class A is the highest risk class and C the lowest. The test strategy is subsequently focused on covering the risks with the highest risk class as early as possible in the test project.

<<Note: the content of the table below is only an example! Risk class A has to have in at least one test level a high thoroughness of the dynamic test (), risk class B has to have in at least one test level a medium thoroughness of the dynamic test () and risk class C has to have in minimal one test level a limited thoroughness of the dynamic test ()>>

<< **Attention**: There are some test levels mentioned in this table, but this is only done as an example. It can be possible that in your project there are more/less and/or other than the in this table mentioned test levels >>

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Risk | Description | Risk Cat | Test Level | | | | | |
| SR | Unit | Int | FAT | UAT | ST |
| Usability | Too difficult to use | B | \*\* |  |  | \*\* | \*\*\* |  |
| Reliability | Unable to save tasks. User Input failure. | B | \*\* |  | \*\* | \*\* |  | \*\* |
| Performance | Application crashing regularly, Functions not working properly | A | \* | \*\* | \*\*\* | \*\* | \*\* | \*\* |
| Supportability | Unable to open supporting documents, or reminder information not working | C | \* | \* |  |  | \*\* | \* |
| Task Managing | Problems with Adding, Editting, Deleting Tasks | A |  | \*\* | \*\*\* | \*\* | \*\* |  |
| Saving Task Data | Task Data not recorded or saved. | A |  | \*\* | \*\*\* | \*\* | \*\* |  |
| Rewards System | Rewards not giving proper rewards or unable to view achievements | B | \* | \* | \*\* | \*\*\* | \*\* |  |

Legend for the table above:

|  |  |
| --- | --- |
| RC | Risk class (from product and technical risk analysis, where A=high risk, B=average risk, C=low risk) |
| SR | Static Review of the various intermediary products (requirements, functional design, technical design). Checking and examining artefacts without executing the software |
| Unit | Unit test and Unit integration test |
| Integration | Integration tests (low level (L), high level(H)) |
| FAT | Functional acceptance test (alpha stage UAT) |
| UAT | User acceptance test (Beta stage UAT) |
| ST | System test (functional scenario testing (F), system quality scenario testing (S)) |
|  | Limited thoroughness of the test |
|  | Medium thoroughness of the test |
|  | High thoroughness of the test |
| <blank> | If a cell is blank, it means that the relevant test or evaluation level does not have to be concerned with the characteristic |

# Test Levels

For this MTP the following test levels are acknowledged:

|  |  |
| --- | --- |
| **Test level** | **Goal** |
| Unit testing: | The aim is to test each part of the software by separating it. It checks that component are fulfilling functionalities or not |
| Integration testing: | In this testing phase, different software modules are combined and tested as a group to make sure that integrated system is ready for system testing. Integrating testing checks the data flow from one module to other modules. |
| System testing: | System testing is performed on a complete, integrated system. It allows checking system's compliance as per the requirements. It tests the overall interaction of components. It involves load, performance, reliability and security testing. |
| Acceptance testing: | Acceptance testing is a test conducted to find if the requirements of a specification or contract are met as per its delivery. |

<<Now iterate through each test level describing the goals, entry and exit criteria, and test environment for each test level. Then Include a table detailing what aspects of what risks will be addressed for each test level, what outcome will be tested for, and when this aspect of testing will be scheduled within the overall project plan>>

## The <name test level>

The primary goal of <specify test level> is to <specify goal of this test level>.

### Entrance and Exit Criteria

<<Entry criteria refer to the desirable conditions in order to start test execution; exit criteria are the desirable conditions that need to be met in order to proceed with the implementation.>>

### Test Environment

<<Describe the test environment that will be used at this level of testing and the procedures used to carry out the testing.>>

### Test Objectives

<<escribe how the risks identified to be dealt with at this test level are being verified and/or tested in this test level. Include an indication of when this will occur within the overall project schedule>>

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Test Goals** | **Risk Verification** | **Schedule** |
| Borrowing: swipe card integration | Ensure card reader interacts correctly with software system | Functionality a specified in defined interface intended, | During implementation of Borrow Book Use Case |
| Returning items: Loans and patrons records not updated | Ensure loan and patron records are updated as appropriate during book return | Appropriate methods in correct DAOs are getting called by ReturnBook control class | During implementation of Return Book use case. |