ADHD Task Manager MASTER TEST PLAN

Version Information

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Remarks** | **Author** |
| 0.1 | 05/04/2023 | Initial | Matthew Neil |
| 1.0 | 09/06/2023 | Finl | Corie Rhodes |

Executive summary

|  |
| --- |
| **Project objective**  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**  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 e.g., spelling errors. |
| **Test objectives**   * 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

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Document name** | **Version** | **Date** | **Author** |
| LCOMProjectVision.docx | 1.0 | 29/03/23 | Deepak Chand, Sam McConchie, Corie Rhodes, Matthew Neil |
| LCOMInitialRequirements.docx | 1.0 | 29/03/23 | Deepak Chand, Sam McConchie, Corie Rhodes, Matthew Neil |

## Test basis

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Document name** | **Version** | **Date** | **Author** |
| **LCOMProposedArchitecture.docx** | 1.0 | 29/03/23 | Deepak Chand, Sam McConchie, Corie Rhodes, Matthew Neil |
| **LCOMRiskList.xlsx** | 1.0 | 04/04/23 | Deepak Chand |

# Test strategy

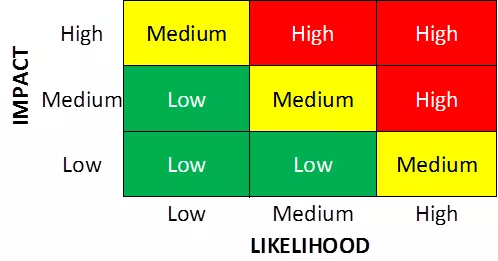
The time available for testing is limited; not everything can be tested with equal thoroughness. This means that choices must be made regarding the depth of testing. Also, it is strived to divide test capacity as effective and efficient as possible over the total test project. This principle is the basis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

The test strategy is based on risks: a system must 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:

|  |  |  |  |
| --- | --- | --- | --- |
| **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:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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, Editing, 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. |

## The Test Levels

**Unit Testing:** The Primary Goal of Unit Testing is to ensure that the individual functions, function as they should.

**Integration Testing:** The Primary Goal of Integration Testing is to ensure that functions can be integrated and work together to achieve a larger goal.

**System Testing:** The Primary goal of System Testing is to test a fully functioning application on a system to monitor the systems capabilities while ensuring the application is up to standard.

**Acceptance Testing:** The Primary goal of Acceptance Testing is to ensure the requirements outlines in the requirements document have been met.

### Entrance and Exit Criteria

|  |  |  |
| --- | --- | --- |
| Testing Type | Entry Criteria | Exit Criteria |
| Unit Testing | * The function’s code has been completed. * Requirements defined. * Test cases ready * Test environment ready. | * Successful execution of unit test cases. * All Bugs and Issues resolved. * Code is complete |
| Integration Testing | * Completion of Unit Testing for all required functions. * Bugs and Issues from Unit testing fixed and closed. * Integration Plan Ready * Test Case Ready * Test Environment Ready | * Successful Execution of Integration Test Cases. * Bugs and Issues have been resolved |
| System Testing | * Completion of Integration Testing * Bugs and Issues found in Integration Testing fixed and closed. * Test Cases Ready * Test Environment Ready | * Successful Execution of System Test Cases. * All Business and Function Requirements are achieved. * Bugs and Issues have been resolved. * Systems Compatibility with supported software and hardware is sound. |
| Acceptance Testing | * Completion of System Testing. * Bugs and Issues found in System Testing fixed and closed. * Business and Functional Requirements are achieved. * Test Cases Ready. * Test Environment Ready. | * Successful Execution of User Acceptance Test Cases. * Business Requirements have been met. * No defects were left out. |

### Test Environment

Android Studio comes with a complete suite of testing functionality, including Kotlin Virtual Machines. We should be able to complete all our tests using these tools with some being tested on a real android device.

-System Testing will be done with Java Virtual Machine for an android emulator as well as real android systems.

- User Acceptance tests will have a similar environment to the System Testing environment. It will be done with real Android Systems.

### 4.1.3 Test Objectives

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Test Goals** | **Risk Verification** | **Schedule** |
| Adding Task | Ensure a Task is saved to the local memory and can be viewed. | User Input Variables are recorded and stored in the AddTask Class | During implementation of Add Task Use Case |
| Rewards: Completing Task | Ensure User and Task Records are updated, and the Rewards Function awards the correct rewards and stores them properly | User is received the correct rewards for the tasks completed. | During implementation of the Complete Task Use Case. |
| Usability | To ensure it is easy to do everything on this application. | Ensure there is not a lot of steps to achieve anything in this app | During Usability Test Case |
| Reminders: | To ensure reminders for tasks work on the android’s notification bar | The notification bar receives an alert about an upcoming due task | During Reminder Test Case |