Software Test Plan

for

NASA Maestro Format Test Tool

**Version 1.0**

**Prepared by:**

**NASA TEAM 1**

**University of Maryland Global Campus**

**25 April 2020**

Table of Contents

[1. Introduction 5](#_Toc38816088)

[1.1. Purpose 5](#_Toc38816089)

[1.2. Scope 5](#_Toc38816090)

[1.2.1. In-Scope 5](#_Toc38816091)

[1.2.2. Out-of-Scope 5](#_Toc38816092)

[1.3. Test Case Development Approach 5](#_Toc38816093)

[1.4. Test Approach 6](#_Toc38816094)

[1.5. Document Overview 6](#_Toc38816095)

[1.6. Definitions and Acronyms 6](#_Toc38816096)

[2. Requirements 7](#_Toc38816097)

[2.1. Hardware 7](#_Toc38816098)

[2.2. Software 7](#_Toc38816099)

[3. Software Test Approach 8](#_Toc38816100)

[3.1. Unit Testing 8](#_Toc38816101)

[3.2. Smoke Testing 8](#_Toc38816102)

[3.3. System and Integration Testing (SIT) 8](#_Toc38816103)

[3.3.1. Approach: 8](#_Toc38816104)

[3.3.1.1. Entry Criteria: 8](#_Toc38816105)

[3.3.1.2. Exit Criteria: 8](#_Toc38816106)

[3.4. Regression Testing 9](#_Toc38816107)

[3.4.1. Approach: 9](#_Toc38816108)

[3.4.1.1. Entry Criteria: 9](#_Toc38816109)

[3.4.1.2. Exit Criteria: 9](#_Toc38816110)

[3.5. User Acceptance Testing (UAT) 9](#_Toc38816111)

[4. Test Design 10](#_Toc38816112)

[4.1. Test Execution Results 10](#_Toc38816113)

[4.2. Defect Management 10](#_Toc38816114)

[4.2.1. Defect Identification 10](#_Toc38816115)

[4.2.2. Defect Severity 10](#_Toc38816116)

[4.2.3. Defect Assignment 10](#_Toc38816117)

[5. Assumptions and Constraints 11](#_Toc38816118)

[1. 5.1. Assumptions 11](#_Toc38816119)

[2. 5.2. Constraints 11](#_Toc38816120)

[2. 11](#_Toc38816121)

[6. Functional Test Cases of node.js 11](#_Toc38816122)

[1. 11](#_Toc38816123)

[7. Functional Test Cases of UIs 11](#_Toc38816124)

[7.1. Test Case 1: Create a New Project 12](#_Toc38816125)

[7.2. Test Case 2: Upload a DOCX File 13](#_Toc38816126)

[7.3. Test Case 3: Convert DOCX file to PDF file 15](#_Toc38816127)

[7.4. Test Case 4: Convert PDF document to PNG file 15](#_Toc38816128)

[7.5. Test Case 5: Compare Two Images 16](#_Toc38816129)

**Revision History**

|  |  |  |
| --- | --- | --- |
| Date | Reason for Changes | Version |
| 04/25/2020 | Initial Document | 1.0 |

# Introduction

## Purpose

The purpose of Software Test Plan (STP) is to prescribe the scope, approach, and resources associated with testing activities that will be administered through the software development life cycle of Maestro Format Test Tool (FTT) system that can be utilized for NASA’s Extra-vehicular Activity/Intra-vehicular Activity (EVA/IVA). This STP document will pinpoint all the times and features that need to be tested to ensure a high-quality application that meets the expectations.

## Scope

The STP document will cover the functional testing of the NASA Maestro FTT application. The in-scope and out-of-scope sections below specify what type of tests will be administered for the NASA Maestro project.

### In-Scope

The types of tests required for this project will ensure that the NASA Maestro project known as Format Test Tool Maestro (FTT) is working properly and meets the business expectation. Testing is very important to ensure high-quality software that runs without any issues. This STP document showcases the type of tests required for this project.

* Smoke Testing.
* System and Integration Testing (SIT).
* Regression Testing.

The scope of functional testing is derived based on the NASA Maestro Format Test Tool system request document. Below functionalities are identified as in scope for this test plan:

* The user launches the application
* The user selects a document
* The user selects an application to launch the document
* Then is controlled by the user to display the required documents

### Out-of-Scope

* Performance testing is out of this scope for the quality assurance team (QA)
* UAT testing is also out of the scope and will be performed by the project sponsors and the client team.

## Test Case Development Approach

Test cases will originate from the project owner’s requirement specified for the Maestro project. All test cases will be reviewed by the team before execution. The priority of these test cases will be based on their importance and severity.

## Test Approach

The following are the type of tests to be conducted within test execution:

* Unit Testing.
* Smoke Testing.
* System and Integration Testing (SIT).
* Regression Testing.

## Document Overview

This document describes the NASA Maestro Format Test Tool (FTT) system test plan and test cases that ensure high-quality software meets the business expectations specified by the product owner. The intended audience of this document includes the project stakeholders to verify the project functional toy and intended and the developer who will create and verify the system functionality.

## Definitions and Acronyms

|  |  |
| --- | --- |
| Abbreviation | Definition |
| FTT | Format Test Tool |
| SDLC | System development life cycle |
| SIT | System and Integration Testing |
| STP | Software Test Plan |
| UAT | User Acceptance Testing |
| UI | User Interface |
|  |  |

# Requirements

## Hardware

* Windows, Linux, and Mac OS

## Software

* Web browser – Safari, Chrome, and Firefox.
* Visual Studio Code

# Software Test Approach

The software test approach for the STP is developed to document the scope, testing progression, defect management, and resource requirements for the NASA Maestro FTT development project. This document also describes all the testing involved in this project and deliverable to be produced during the system and regression testing of this project.

The test approach is critical to ensure that all major scenarios of the Maestro FTT project are covered to ensure a high-quality product that meets the business expectations.

## Unit Testing

The analyst /development team will utilize unit tests to ensure the software is working correctly. Also, these unit tests can detect any issues when integrated into the main product. These unit tests will be written by developers as it pertains to code itself and internal program design.

## Smoke Testing

A quick test that will be performed when new functionality is introduced to ensure that the major functionality of the software is still working as expected. Blockers or any major bugs should be cleared by this type of testing.

## System and Integration Testing (SIT)

This process pertains to testing the integrated system to ensure that it is still within the scope and the proposed requirements. This document will cover the functional testing of the project.

### Approach:

The system and integration tests are both combined for this project. The QA team will ensure that testing planning, test case execution, and reporting activities are covered during this phase.

#### Entry Criteria:

* All identified smoke tests for this project should pass
* All updates to project documentation are also reflected in this document
* All test cases are linked to business requirement document
* System tests cases are reviewed by the project team
* Project code is united tests and all failed tests are shared with the QA team.

#### Exit Criteria:

* The completion of system and integration testing – a pass rate of 97%.
* All blockers and major defects are fixed and resolved.
* Test report for test cases testing completion.

## Regression Testing

This type of test will be conducted by the quality assurance team to ensure existing functionality was not impacted by code fixes and new enhancements.

### Approach:

This ensures that new functionality or a current fix to functionality will not impact the rest of the code and that previous functionality will not be changed.

#### Entry Criteria:

* Regression test cases for affected areas of the project should be ready.
* The system and integration testing outlet criteria are completed.

#### Exit Criteria:

* Regression testing will be completed.

## User Acceptance Testing (UAT)

The tests that are out of scope for this project is performance and UAT testing. These tests will be performed by the project manager and one of the team members.

# Test Design

## Test Execution Results

The test execution results showcase failed or passed test cases. All these results should and will be recorded and shared in a test report document.

## Defect Management

### Defect Identification

All new defects found during test execution shall be logged in as a defect

### Defect Severity

The defect severities are as follows:

* ***Severity -1: Blocker*** – This type of defect prevents the application from working. As a result, it must be dealt with immediately.
* ***Severity-2: High*** – This will impair the functionality of the software and must be addressed in higher priority.
* ***Severity-3: Medium*** – This type of defect that may impact certain aspects of the application that are considered a medium priority.
* ***Severity-4: Low*** – These defects are trivial and can be fixed later.

### Defect Assignment

Defects assigned to the project team lead to be reviewed and assessed. Then the project team leaders will reach out to developers to pinpoint resolution and find causes.

|  |  |
| --- | --- |
| Name | Responsibility |
| Rick Stuart | QA |

# Assumptions and Constraints

## 5.1. Assumptions

This project scope is limited to the activities required to develop the Maestro Format Test Tool (FTT) interface for the NASA Maestro project. Based on the existing Maestro application, we assume the following:

* The GitLab repository (Repo) is set up to detect any code changes and trigger a build
* When committing and pushing an update of the source file into the Maestro project Repo, it will generate a build as an output
* All NASA team members will be involved through the SDLC of the project
* Dependency identification to ensure the facilitation of Maestro project development pull and push files from the Repo, as well as creating, editing, and updating YAML files.

## 5.2. Constraints

The following are a few limitations which may delay project success:

* Constructing an existing Maestro application would require knowledge regarding the pre-existing technologies used to speed up the development process for the actual tool.
* The application would require a server to run and that will be of a cost to the team.
* Pre-existing or unknown bugs which might delay the development process from the beginning.
* Maestro project features might not be fully developed during the previous phase of this project.
* Daily communication of the team, and real-time cooperation within a time of team members that are in different time zones can be very challenging.

# 

# Functional Test Cases of node.js

For this testing, Stakeholders have provided documents that will be used for formatting. Testers will use this document to verify that all documents will be processed and formatted correctly.

1. Create a new Project
2. Upload a DOCX file
3. Convert DOCX file to PDF file
4. Convert PDF document to PNG file
5. Compare Images

## 

# Functional Test Cases of UIs

For this testing, Stakeholders have provided documents that will be used for formatting. Testers will use this document to verify that all documents will be processed and formatted correctly.

1. Create a new Project
2. Upload a DOCX file
3. Convert DOCX file to PDF file
4. Convert PDF document to PNG file
5. Compare Images

## Test Case 1: Create a New Project

5.1.1 create a new project with a project name

|  |  |
| --- | --- |
| **Description:** | The user will create a new project to run and save. |
| **Requirements:** | The user must enter a project name to save. |
| **Prerequisites:** | The application is open and running. |
| **Steps:** | Click the “Create a New Project” link.  Enter a project name in the textbox that has not been used.  Click the “OK” button. |
| **Expected output:** | The system displays a new folder, with the entered project name, that has been saved. |
| **Assumptions:** | The user’s project name has not been used. |

5.1.2 No project name is entered

|  |  |
| --- | --- |
| **Description:** | The user will fail in creating a new project to run and save. |
| **Requirements:** | The user leaves an empty project name. |
| **Prerequisites:** | The application is open and running. |
| **Steps:** | Click the “Create a New Project” link.  Click the “OK” button. |
| **Expected output:** | The system will pop up a new window showing no project entering.  The system will not display a new folder and project name. |
| **Assumptions:** | The user’s project name has not been used. |

5.1.3 Enter the duplicated project name

|  |  |
| --- | --- |
| **Description:** | The user will fail in creating a new project to run and save. |
| **Requirements:** | The user enters a duplicated project name that has been used. |
| **Prerequisites:** | The application is open and running. |
| **Steps:** | Click the “Create a New Project” link.  Enter a project name in the textbox that has been used.  Click the “OK” button. |
| **Expected output:** | The system will pop up a new window showing the project name has been used before.  The system will not display a new folder and project name. |
| **Assumptions:** | The user’s project name has not been used. |

## Test Case 2: Upload a DOCX File

5.2.1 upload a correct DOCX file

|  |  |
| --- | --- |
| **Description:** | The user selects a DOCX file to format in the system project. |
| **Requirements:** | The file must be a DOCX file in the user’s environment. |
| **Prerequisites:** | There has been a project created. |
| **Steps:** | Click “Upload a DOCX file” link.  Use the dropdown list to select a project.  Use “Choose File” button and select a DOCX file  Click “Upload the selected file” button |
| **Expected output:** | The system displays the selected DOCX file, in the created project folder, and the file is saved in Maestro. |
| **Assumptions:** | The project has been created. |

5.2.2 Recognize wrong format file

|  |  |
| --- | --- |
| **Description:** | The user selects a none DOCX file namely PDF file. |
| **Requirements:** | The file is not a DOCX file in the user’s environment. |
| **Prerequisites:** | There has been a project created. |
| **Steps:** | Click “Upload a DOCX file” link.  Use the dropdown list to select a project.  Use “Choose File” button and select a wrong DOCX file  Click “Upload the selected file” button |
| **Expected output:** | The system will not display the selected DOCX file, in the created project folder.  The system will pop up a window alerting that the uploaded file is wrong. |
| **Assumptions:** | The project has been created. |

5.2.3 No DOCX file is selected

|  |  |
| --- | --- |
| **Description:** | The user doesn’t select a file. |
| **Requirements:** | None |
| **Prerequisites:** | There has been a project created. |
| **Steps:** | Click “Upload a DOCX file” link.  Use the dropdown list to select a project.  Click “Upload the selected file” button |
| **Expected output:** | The system will not display the selected DOCX file, in the created project folder.  The system will pop up a window alerting that none DOCX file is selected. |
| **Assumptions:** | The project has been created. |

## Test Case 3: Convert DOCX file to PDF file

|  |  |
| --- | --- |
| **Description:** | The user will choose to convert a DOCX file to a PDF file. |
| **Requirements:** | The file chosen is a DOCX file. |
| **Prerequisites:** | DOCX file has been saved to the project. |
| **Steps:** | 1. Click the “Convert DOCX to PDF” link. 2. Use the drop-down list to select a project. 3. Use the drop-down list to select DOCX file 4. Click “Convert DOCX” |
| **Expected output:** | The system displays the selected DOCX file as a saved PDF file, in the created project folder, and the file is saved in Maestro. |
| **Assumptions:** | The project has been created. |

## Test Case 4: Convert PDF document to PNG file

|  |  |
| --- | --- |
| **Description:** | The user will choose to convert the PDF file to PNG file. |
| **Requirements:** | The file chosen is a PDF file. |
| **Prerequisites:** | A PDF file has been saved to the project. |
| **Steps:** | 1. Click the “Convert PDF to PNG” link. 2. Use the drop-down list to select a project. 3. Use the drop-down list to select PDF file 4. Click “Convert PDF” |
| **Expected output:** | The system displays the selected PDF file as a saved PNG file, in the created project folder, and the file is saved in Maestro. |
| **Assumptions:** | The project has been created. |

## Test Case 5: Compare Two Images

|  |  |
| --- | --- |
| **Description:** | The user will choose to compare images. |
| **Requirements:** | PNG files have been chosen. |
| **Prerequisites:** | PNG files have been saved to the project. |
| **Steps:** | 1. Click the “Compare two images” link. 2. Use the drop-down list to select a project. 3. Use the drop-down list to select the first PNG file 4. Use the drop-down list to select the second PNG file 5. Use a scale to select Tolerance threshold level 6. Click “Compare Images” button |
| **Expected output:** | The system displays a new file with both PNG files with a new name, which has the first image over the second image, saved as a PNG file. |
| **Assumptions:** | A project has been created.  A second PNG file has been saved in the project. |

# 