Test Approach Document

Task Process Monitor

Authors

S.No.	Name	Email
1	Vivek Vellaiyappan Surulimuthu	Vivek.Surulimuthu1@marist.edu
2	Neha Ersavadla	Neha.Ersavadla1@marist.edu
3	Deeksha Sudini	Deeksha.Sudini1@marist.edu

Revision History

Revision	Dates & Time	Revised By	Current Plan	Comments
1	10/26 & 6pm	Vivek	Listed out the type of testing to be used and what we will be doing in those each testing types	
2	10/29 &2pm	Deeksha & Neha	Revised the test document by adding the test cases	More improvement needed
3	11/1 & 5pm	Vivek & Deeksha	Updated all the required testing	

			approach that is needed for the project	

Table of Contents

1. Objective

To perform various testing process involved in STLC to make the software product bug-free and reliable

1.1 Approach

- Following STLC process
- Following Spring boot best testing practices

The functional testing will be done by unit testing the independent modules making sure the test case is passed and all corresponding requirements are satisfied. After completion of the unit tests and making sure all successfully pass, integration will be done and validation of the integrated code will take place; this will be done by considering the whole software as one unit with several inputs and outputs

2. Purpose

This test plan document describes the objectives, scope, approach and focus of the testing of the Task Process Monitor. This document begins with an overview of the Task Process Monitor. It then lists the features to be tested as part of the testing cycle. The document also lists the application features that will not be tested. It further delves into the details of the software and environmental needs for carrying out an efficient and smooth testing. It then discusses the testing schedule and the resource responsible for testing each of the features. It then briefly describes the pass/fail criteria and risks associated with the testing. The document ends with a detailed description of the test cases that will be executed as part of the testing

2.2 Target Audience

This test plan document has been written for the following audience

- Project Manager
- Test Manager

- Business Analyst
- Development Team
- Test Team
- Onsite coordinators

3. Introduction

The software will be tested and determined to be ready for delivery and of accepted quality based on the results of the functional testing and non-functional testing. In brief the tests have to satisfy all of the customer's and internal requirements by successfully passing all test cases.

The functional testing verifies conformance to the customer requirements and it verifies as well that not unexpected behavior outside the requirements is present. The non-functional testing addresses the parameters such as performance and load etc.

3.1 Relationship to other documents:

The requirements specification presents the requirements from the customer, as well as internal requirements. The requirements are numbered to provide for an easy way to trace the test cases to each requirement. As well, the design document provide more information about how the implementation of the functionality is done which serves to better understand how to perform the white box testing

3.2 Features to be tested/not to be tested:

Testing will be done to the Task Process Monitor in order to satisfy all requirements. Verification of this will assure that the design is working as expected by the developers, and will verify as well that the requirements have been satisfied.

3.3. Pass/Fail criteria:

A test case will be determined to have been successfully executed and passed if and only if all conditions stated are satisfied. All inputs have to be exercised, environment has to be setup, and outputs must be read and verified as having the expected values. Else, the test case is declared failed.

4. Testing Planned to Execute

4.1. Functional testing types include: (will be updated regularly after each project sprints)

- Unit testing
    - Configured using spring-boot-starter-test config file in pom.xml
    - Coded on every push to master branch: Junit will be done
    - Will complete this if possible: Mockito
- Integration testing
    - Testing between Spring boot UI and Sigar Data
    - Database and Sigar pull program
- System testing
    - OS environment testing
- Sanity testing
    - Information on the type of process being pulled and shown
- Smoke testing
    - Verifying on overall basic working functionality
- Interface testing
    - Using dependency injection of Spring
- Regression testing
    - Tested after each release of our project sprint
- Beta/Acceptance testing
    - Just making sure this project conforms with the requirements available in the design document
- Automation Testing
    - Selenium with Python will be used to make the testing of the website

4.2 Non-functional testing types include:

- Performance Testing
    - How it performs for load and stress testing?`
- Load testing
    - Running many processes at the same time
- Stress testing
    - Refreshing the page again and again and verifying
- Compatibility testing
    - Backward compatibility testing
- Install testing
    - Doing in different work machines
- Reliability testing
    - What if I close the application
    - What if i close db connection

    - Accuracy of data
- Usability testing
    - Visual representation & interpretation
- Compliance testing
    - Non-offending, License verification

5.Testing requirements(Hardware & Software)

5.1 Test Cases (to be updated over time)

5.1.1 Configure Maven Setup

Purpose: To see whether maven has successfully configured with all the dependencies and perform a build.

Audience:
The audience for this test case is the customer and project team.

Test Case Specification Identifier:TC-01

Input: 1. Install maven on (Linux/Windows) System and check the version of maven using mvn --version

Expected Output: Maven should configure properly and display the version of maven installed in the command line.

5.1.2 Configure SQLite using SQLite JDBC driver

Purpose: To see whether SQLite has successfully configured with all the dependencies.

Audience:
The audience for this test case is the customer and project team.

Test Case Specification Identifier:TC-02

Input: 1. Install SQLite on (Linux/Windows) System and check the SQLite installation has been done successfully by running the command c:\cd sqlite

Expected Output: SQLite should configure properly and display the version of SQLite installed.

5.1.3 Fix Jar file dependencies

Purpose: To see whether all the useful jar files has been added to the project.

Audience:
The audience for this test case is the customer and project team.

Test Case Specification Identifier:TC-03

Input: Add all the required jar files to the project and try to build the project.

Expected Output: Build should be successful.

5.1.4 Pull cpuinfo data using Sigar.

Purpose: System information and cpu information should be pulled properly using sigar.

Audience:
The audience for this test case is the customer and project team.

Test Case Specification Identifier: TC-04

Input: cpu information should be pulled properly.

Expected Output: Should be able to pull all the System level CPU information. Verify whether this is appropriate by running the built-in task manager of the system.

5.1.5 Run the project file and navigate to the URL

Purpose: On click of bash script, execute the .war file and then on ready, navigate to the localhost

Audience:

The audience for this test case is the customer and project team.

Test Case Specification Identifier: TC-05

Input: Execute the .war file and then on ready, navigate to the localhost.

Expected Output: The localhost should open successfully with the data pulled using sigar.

5.1.6 Collection of Metrics

Purpose: To see whether all the real time data is being pulled to the database using Sigar

Audience:
The audience for this test case is the customer and project team.

Test Case Specification Identifier:TC-06

Input: Data pulled by Sigar API

Expected Output: CRUD operations should be performed and metrics should be collected based on the real time data.

5.1.7 Logging Actions

Purpose: To verify the log output

Audience:
The audience for this test case is the site monitoring team.

Test Case Specification Identifier:TC-07

Input: running the program

Expected Output: get the log of how exactly the program is able to run. Verify the logs

5.1.8 Reproduce Webapp on windows/linux

Purpose: To see whether we can reproduce the webapp in windows and linux

Audience:
The audience for this test case is the customer and project team.

Test Case Specification Identifier:TC-08

Input: Setup the environment such as maven,sqlite configuration in windows and linux

Expected Output: The project should be able to reproduce in both windows and linux.

6. Spring-boot-starter-test

- It is the dependency provider to configure traditionally used testing dependency libraries

6.1 Testing

Spring Boot provides a number of utilities and annotations to help when testing application. Test support is provided by two modules: spring-boot-test contains core items, and spring-boot-test-autoconfigure supports auto-configuration for tests.

Most developers use the spring-boot-starter-test "Starter", which imports both Spring Boot test modules as well as JUnit, AssertJ, Hamcrest, and a number of other useful libraries.

6.2 Test Scope Dependencies

The spring-boot-starter-test "Starter" (in the test scope) contains the following provided libraries:

- JUnit: The de-facto standard for unit testing Java applications.
- Spring Test & Spring Boot Test: Utilities and integration test support for Spring Boot applications.
- AssertJ: A fluent assertion library.
- Hamcrest: A library of matcher objects (also known as constraints or predicates).
- Mockito: A Java mocking framework.

- JSONassert: An assertion library for JSON.
- JsonPath: XPath for JSON.

These common libraries are useful when writing tests. If these libraries do not meet our needs, we can add additional test dependencies of our own.

Planned Tests using starter-test config

We will be using JUnit, Spring Test, Spring Boot Test, AssertJ, JsonPath from the aforementioned starter-test dependency configuration file.

7. Testing Risk and Contingency Plan

Human Resource Risk

The following are the possible human resource risk that the project can run into:

1. Test Manager goes on an unexpected long leave or quits the project.
2. The testers are not able to cope up with the project pressure or are not capable enough to execute their responsibilities efficiently.

These risks can be resolved using the following two approaches.

1. Hire extra resources at the beginning of the project as back-up resources. This has an adverse effect of increasing the project budget as the back-up resource might never be used if the risk never occurs.
2. Hire contractors and train them once the risk happens.

Unrealistic Schedule

There is a possibility that during the course of the project, the project team realizes that the initial estimation of the project complexity was inaccurate and hence the project can run into schedule risks. Also, there can be a budget overrun due to unforeseen circumstances.

These risks can be resolved by using the following approach:

1. Hire contractor to speed up testing.
2. Reduce the test scope and push the low priority functionalities to a later release of the project.

Unavailability of S/W or H/W resources

There might be a possibility that the required software license is not available or the test bed is not available on time for the performance testing.

This risk can be resolved by using the following approach

1. Use an alternate software for which the license is available

2. Use the hardware of a lower capacity to perform the initial performance test until the actual test bed is available.

8. References

8.1 Websites:

- https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-testing.html - Testing feature in spring boot
- https://www.softwaretestinghelp.com/types-of-software-testing/
- https://www.softwaretestinghelp.com/types-of-software-testing/ - types of testing

8.2 Books

- https://www.oreilly.com/library/view/software-testing-principles/9788177581218/xhtml/chapter001.xhtml