SUPPLEMENTARY SCHEDULE MANAGEMENT SYSTEM SOFTWARE TEST SPECIFICATION (STS) DOCUMENT

Khartoum University 4th Year Students

Faculty of Engineering, University of Khartoum, Khartoum, Sudan

Date: 31 March 2023

TEST DOCUMENT HISTORY:

Document revisions:

Version	Primary Author(s)	Description of Version	Date
Initial (1.0)	Khartoum University 4th Year Students Group: - - Mugtba Mirghani - Mohamed Ahmed - Mohammed Ashraf - Monzer Omer - Ahmed Alsiddig	Initial draft.	31/03/2023

Document review & approval:

Reviewers	Version Approved	Signature	Date

1. Introduction:

Software testing is a process of verifying and validating that software application or program meets the business and technical requirements that guided its design and development and works as expected, and also identifies important errors or flaws categorized as per the severity level in the application.

Software testing is also used to test the software for other software quality factors like reliability, usability, integrity, security, capability, efficiency, portability, maintainability, compatibility etc.

1.1 Objectives of software testing:

- To discuss the distinctions between validation testing and defect testing.
- To describe the principles of system and component testing.
- To describe strategies for generating system test cases.
- To understand the essential characteristics of tool used for test automation.

1.2 The Basics of Software Testing:

There are two basic classes of software testing, black box testing and white box testing and there is a basic difference between the two classes clarified by the definitions below:

1. ***Black box testing*** (also called functional testing) is testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions.

2. ***White box testing*** (also called structural testing and glass box testing) is testing that takes into account the internal mechanism of a system or component.

Black box testing is often used for validation (are we building the right software?) and white box testing is often used for verification (are we building the software right?). This document focuses on black box testing [1].

1.3 Type of black box testing:

There are many types of black box testing but the following are the prominent ones:

i. ***Functional testing:*** this type of black box testing is related to functional requirements of system, it is done by software tester.

ii. ***Non-functional testing:*** this type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.

iii. ***Regression testing:*** regression testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

In black-box testing, a software testing technique whereby the internal workings of the item being tested are not known by the tester does not ever examine the programming code and does not need any further knowledge of the program other than its specification.

1.4 The advantages of black-box testing include:

1. The test is unbiased because the designer and the tester are independent of each other.
2. Tester does not need knowledge of any specific programming languages.
3. The test is done from the point of view of the user, not the designer.
4. The test case can be designed as soon as the specification are complete.
5. The test can be redundant if the software designer has already run a test case.

In this project we used black box testing to test the environment for software testing because software tester describe what can be expected from a component, without revealing how the component achieves its effects.

2. Test Plan:

The test plan include:

- Test ID
- Brief Introduction
- Test item (modules)
- Features to be tested
- Features not to be tested
- Test approach
- Entry/Exit criteria
- Test environment
- Roles and responsibilities
- Schedule

1. Test ID:

The Test Plan Identifier is just a type of unique number or reference-id to identify this test plan and the software that it is related to.

2. Introduction:

Software and software-based systems testing is a technical discipline of systems engineering. The purpose of software and software-based systems testing is to help the development organization build quality into the software and system during the life cycle processes and to validate that the quality was achieved.

The test processes determine whether the products of a given life cycle activity conform to the requirements of that activity, and whether the product satisfies its intended use and user needs. This determination can include inspection, demonstration, analysis, and testing of software and software based system products. Test activities are performed in parallel with software and system development, not just at the conclusion of the development effort.

3. Test item(modules):

These are the software products (c ode, user, manuals ... etc.) you intend to test within the scope of this test plan. This list of items will be populated from the software products identified in the master project plan as well as other sources of documentation and information.

You should include version numbers and configuration requirements where needed. Bear in mind that what you are testing is what you intend to deliver to the customer whether internal or external.

Test Item Name	Test Item Version No

4. Features to be tested:

This is a high-level view of what is to be tested from the user's viewpoint of what the system does and should refrain from being a technical testing breakdown of the system since that is covered in section below. It is also useful to list the features to be tested with respect to the names of the parent components, etc., as they are known by the configuration management system. A bulleted list format can serve well here, or use the table format given below.

5. Features not to be Tested

What is not to be tested can be sometimes just as important as stating what is to be tested. It removes any ambiguity in order that other project stakeholders are clear on what to expect from the test phases.

Make this list the same format as above. Additionally, however, you should state clear reasons why the feature is not being tested. There could be any number of reasons and all should be given alongside the mitigating factors.

6. Test approach:

The test approach is the overall test strategy that underpins the whole test plan. A test approach asks, "how are you going to test the software?" If this Test Plan is part of a larger parent project and there are other Test Plans for other parts of the overall system then the test approach should dovetail with the other test approaches.

7. Entry/Exit criteria :

In order that you can manage software stability and quality grade through successive test phases it is useful to plan for test phase entry and exit criteria. The review of such criteria should be the basis of formal management milestone reviews. Each test phase is likely to have its particular set of criteria. The following points below may help you formulate your own list of criteria.

Unit Test Phase Entry Criteria

- *All of unit tests have been peer-reviewed*
- *Software to be unit tested has been checked into configuration management system*
- *All planned functionality and bug fixes have been implemented*
- *Source code for software to be unit tested has been peer-reviewed*

Unit Test Phase Exit Criteria:

- *All of unit tests are executed*
- *All of unit tests pass*
- *Unit Test Report has been approved*
- *Unit tested software has been checked into configuration management system*

8. Test environment :

The test environment encompasses the software being tested, related platform software, third-party software, communications software, etc. Ensure that you have the resources required to install, set up and configure your test environment. The test environment also extends to hardware, test data, technical publications, consumables (like printer paper, ink, etc.).

Take a view of your test environment and plan for anything that is required in order to assure the smooth execution of your testing.

9. Roles and responsibilities :

It has always found that it useful to cover how stakeholders will communicate throughout all testing activities. It can be a good idea to include contact details of the key stakeholders in this section, for example the phone and email details of the author, test manager, etc. If you plan to hold morning stand-ups, then state that here. If it is planned to run review weekly meetings, or remote-conferences, then plan all aspects of your communication. Think about why, where, when, who and how. Some example tables are given below to help you structure this section.

Name	Role	Contact Details

Communication Aspect	Purpose

10. Schedule:

The testing schedule is a subset of the overall project schedule. Include a hyperlink to the file containing the project schedule rather than repeat any details here. Since project schedules can be volatile and subject to continual revision it makes sense to do this and avoid unnecessary reworking of this Test Plan.

The inherent risks associated with schedule slippage mean that this is an area that invariably finds its way into the testing Risk Register in paragraphs. If this project is part of a larger project, then you may wish to acknowledge any other relevant test/project schedules. Consider how this project and its schedule interface with these other projects and their schedules. For instance, are there any tasks that have to be handed over? [2].

2.1 **Test Case\s:**

Test Case#:1.0	**Test Case Name:**
System: Supplementary management system	**Subsystem:**
Designed by:	**Design Date:**
Executed by	**Executed date:**
Short Description: The system shall develop an algorithm for determining the best schedule.	

Pre-conditions

- Availability of data in a particular format.
- The system's user received training.

Step	Action	Expected System Response	Actual	Pass/Fail	Comment

Post-condition:

- The formulated supplementary table will be downloaded to the user's pc as an excel file.

Test Case#:1.0	**Test Case Name:**
System: Supplementary management system	**Subsystem:**
Designed by:	**Design Date:**
Executed by	**Executed date:**

Short Description: The system shall provide a method for efficiently distributing the monitoring staff.

Pre-conditions:

- The availability of monitoring personnel's schedules.

Step	Action	Expected System Response	Actual	Pass/Fail	Comment

Post-condition:

- The formulated monitoring schedule will be downloaded to the user's pc as an excel file.

Test Case#:1.0	**Test Case Name**
System: Supplementary management system	**Subsystem:**
Designed by:	**Design Date:**
Executed by	**Executed date:**

Short Description: Determines the number of exam halls available in the specific schedule dates based on the number of students taking the exams.

Pre-conditions

• Availability of data related to exam halls.

Step	Action	Expected System Response	Actual	Pass/Fail	Comment

Post-condition:

- A list of the available exam halls will be shown to the user.