**Experiment No.:** 08

**Title:** Preparing Software Test Document (STD)

**Batch: A3 Roll No.: 16010421119 Experiment No.: 08**

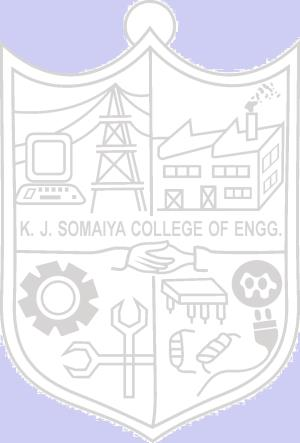
**Aim:** To prepare Software Test Document (STD)

**Resources needed:** Internet Explorer, Latex Editor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Theory** This system test document specifies the tests for the entire software system, defines the test schedule, and records the test results. This document does not cover unit testing (the testing of individual sub-systems or components of the system).

The system may consist of multiple items that are to be tested separately. The system may be tested in one or more increments of functionality; the system test document should cover each version of the system separately. When different versions of the system are tested, make sure to clearly identify the version of the software and the relevant test and test result. Also, extend the unique identifier scheme to include the version of the software system under test (SUT) – for example, use TC-vvvv-nnnn to identify test case nnnn for software system version vvvv.



Software Test Document (STD) Template:

1 INTRODUCTION

1.1 System Overview

Briefly detail the software system and items to be tested. Identify the version(s) of the software to be tested.

1.2 Test Approach

Describe the overall approach to testing. For each major group of features or feature combinations, specify the approach that will ensure that these feature groups are adequately tested. Specify the major activities, techniques, and tools that are used to test the designated groups of features. The approach should be described in sufficient detail to permit identification of the major testing tasks and estimation of the time required to do each one. Identify significant constraints on testing, such as deadlines.

2 TEST PLAN

Describe the scope, approach, resources, and schedule of the testing activities. Identify the items being tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task in the case of a group project.

2.1 Features to be tested

Identify all software features and combinations of software features to be tested. Identify the test case(s) associated with each feature and

each combination of features. Identify the version of the software to be tested.

When multiple versions of the software are tested in a planned, incremental

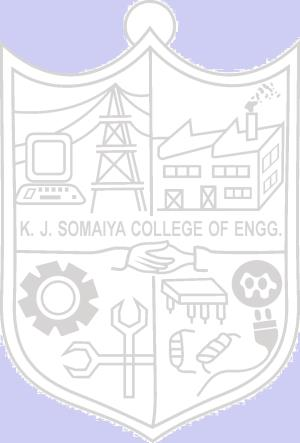
manner, then use section numbers 2.1.n to identify the features to be tested for each version.

2.2 Features not to be tested

Identify all features and significant combinations of features that will not be tested and the reasons for not doing so.

2.3 Testing Tools and Environment

Specify test staffing needs. For an individual project, specify the time to be spent on testing. For a group project, specify the number of testers and the time needed.



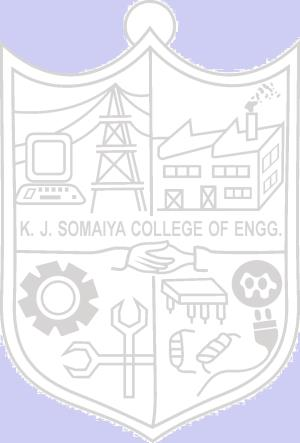
Specify the requirements of the test environment: space, equipment, hardware, software, special test tools. Identify the source for all needs that are not currently available.

3 TEST CASES

A test case specification refines the test approach and identifies the features to be covered by the case. It also identifies the procedures required to

accomplish the testing and specifies the feature pass/fail criteria. It documents the actual values used for input along with the anticipated outputs.

If an automated test tool is to be used:



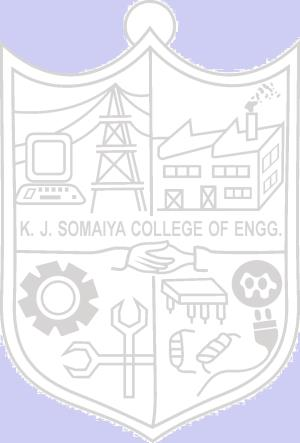
|  |  |  |
| --- | --- | --- |
|  | 1. | document each test case here as a specification for the test tool; |
| 2. | document the procedure that must be followed to use the test tool. |
| 3.n |  | Case-n (use a unique ID of the form TC-nnnn for this heading) |
| 3.n.1 |  | Purpose |
|  |  | Identify the version of the software and the test items, and describe the features and combinations of features that are the object of this test case. For each feature, or feature combination, a reference to its associated requirements in the software requirement specification (SRS) should be included. |

3.n.2 Inputs

Specify each input required to execute the test case. Some of the inputs will be specified by value (with tolerances where appropriate), while others, such as files or URLs, will be specified by name. Specify all required relationships between inputs (e.g., ordering of the inputs).

3.n.3 Expected Outputs & Pass/Fail criteria

Specify all of the expected outputs and features (e.g., response time) required of the test items. Provide the exact value (with tolerances where appropriate) for each required output or feature. Specify the criteria to be used to determine whether each test item has passed or failed testing. If an automated test tool is used, identify how the results of that tool are to be analysed.



3.n.4 Test Procedure

Detail the test procedure(s) needed to execute this test case. Describe any special constraints, such as: special set up, operator intervention, output determination procedures, and special wrap up.

|  |
| --- |
| 4.Test Logs  A test log is used by the test team to record what occurred during test execution. A.n  Log for test-n (use a unique ID of the form TL-nnnn for this heading)  5.Test Results  For each execution, record the date/time and observed results (e.g., error messages generated). Also record the location of any output (e.g., window on the screen). Record the successful or unsuccessful execution of the test.  Incident Report (add a unique ID of the form TIR-nnnn to this heading) If the test failed, or passed with some unusual event, fill in this incident  report with the details. Summarize the incident, identifying the test items involved, and the anomaly in the results. Indicate what impact this  incident will have on the project. |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Procedure:** 1 Prepare Test plan

2 Give descriptions of minimum five test cases.

3 Prepare Test case Table for test cases.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Results: STD document in given format**

\documentclass{article}

\usepackage{graphicx}

\title{Software Test Document (STD)}

\author{}

\date{}

\begin{document}

\maketitle

\section{Introduction}

\subsection{System Overview}

Briefly detail the software system and items to be tested. Identify the version(s) of the software to be tested.

\subsection{Test Approach}

Describe the overall approach to testing. Specify major activities, techniques, and tools used. Identify significant constraints on testing.

\section{Test Plan}

Describe the scope, approach, resources, and schedule of testing activities. Identify items being tested, features to be tested, testing tasks, and responsible personnel.

\subsection{Features to be Tested}

Identify all software features and combinations of features to be tested. Identify associated test cases and software versions.

\subsection{Features not to be Tested}

Identify features and combinations not to be tested and reasons for exclusion.

\subsection{Testing Tools and Environment}

Specify test staffing needs and requirements of the test environment.

\section{Test Cases}

\subsection{Case-n (use a unique ID of the form TC-nnnn for this heading)}

\subsubsection{Purpose}

Identify the version of the software and test items. Describe features and reference associated requirements.

\subsubsection{Inputs}

Specify inputs required for test case execution.

\subsubsection{Expected Outputs \& Pass/Fail Criteria}

Specify expected outputs and pass/fail criteria for test items.

\subsubsection{Test Procedure}

Detail procedures needed to execute the test case.

\section{Test Logs}

\subsection{Log for test-n (use a unique ID of the form TL-nnnn for this heading)}

\section{Test Results}

Record date/time and observed results for each execution. Indicate successful or unsuccessful execution.

\section{Incident Report (add a unique ID of the form TIR-nnnn to this heading)}

Summarize incidents, identifying test items involved and anomalies in results.

\end{document}

**Questions:**

1. Differentiate between verification and validation.

* Verification:
  + Definition: Verification ensures that the software is developed correctly according to its specification. It involves evaluating the intermediate work products of a software development process to check if they meet the specified requirements.
  + Focus: Verification focuses on the process of building the software correctly, ensuring that each step of the development process adheres to the predefined standards and requirements.
  + Activities: Verification activities include reviews, inspections, walkthroughs, and testing at various stages of the development process. These activities aim to identify defects early in the development lifecycle.
  + Objective: The objective of verification is to answer the question, "Are we building the product right?"
* Validation:
  + Definition: Validation ensures that the software meets the customer's needs and expectations. It involves evaluating the final product to ensure that it satisfies the intended use and purpose in its specific environment.
  + Focus: Validation focuses on evaluating the end product to ensure that it fulfills the user's requirements and expectations, providing value in its intended context.
  + Activities: Validation activities typically involve testing the software against the customer's requirements and conducting user acceptance testing (UAT) to ensure that it meets user needs.
  + Objective: The objective of validation is to answer the question, "Are we building the right product?"

2. List down all OO software testing strategies.

* Class Testing:
  + Focuses on testing individual classes or objects.
  + Verifies the behavior of each class's methods and attributes.
  + Includes testing of constructors, methods, state changes, and interactions between objects.
* Integration Testing:
  + Tests the interactions and collaborations between classes and objects.
  + Ensures that classes work together correctly when integrated into larger components or systems.
  + May involve testing individual components, subsystems, or layers of the application.
* Inheritance Testing:
  + Verifies the correct implementation of inheritance hierarchies.
  + Tests the behavior of derived classes to ensure they inherit and extend the functionality of their parent classes properly.
  + Checks for proper method overriding, polymorphism, and adherence to the Liskov Substitution Principle (LSP).
* Polymorphism Testing:
  + Focuses on testing methods that exhibit polymorphic behavior.
  + Ensures that methods invoked on objects of different types produce the expected results.
  + Tests for dynamic dispatch and method overriding correctness.
* State-Based Testing:
  + Tests the behavior of objects based on their internal states.
  + Covers different states an object can be in and transitions between states.
  + Includes state transition testing and state coverage analysis.
* Collaboration Testing:
  + Verifies the interactions and communication between objects.
  + Tests how objects collaborate to achieve specific functionalities or use cases.
  + Includes testing message passing, method calls, and object collaborations.
* Mocking and Stubbing:
  + Involves creating mock objects or stubs to simulate the behavior of dependencies.
  + Facilitates isolated testing of individual classes or components.
  + Allows testing interactions with external systems or components without relying on their actual implementations.
* Behavior-Driven Development (BDD):
  + Focuses on defining test cases and behavior specifications in natural language.
  + Tests are written in a format that is easily understandable by stakeholders.
  + Promotes collaboration between developers, testers, and business analysts to define test scenarios and acceptance criteria.
* Test-Driven Development (TDD):
  + Involves writing tests before writing the actual code.
  + Tests serve as specifications for the desired behavior of the code.
  + Helps ensure code correctness and maintainability by continuously running tests during development.
* Mutation Testing:
  + Involves introducing small changes (mutations) to the code to create faulty versions (mutants).
  + Tests are then run against these mutants to determine the effectiveness of the test suite in detecting the faults.
  + Helps assess the quality of the test suite by measuring its ability to detect changes in the code.

**Outcomes: CO4 Demonstrate test case design**

**Conclusion:** We learnt about Software test document and implemented it and made an STD document for our project.

**Grade: AA / AB / BB / BC / CC / CD /DD Signature of faculty in-charge with date**

**References:**

**Books / Websites:**

1. Roger S. Pressman, Software Engineering: A practitioners Approach, 7th Edition,

McGraw Hill, 2010.

2. Ian Somerville, Software Engineering, 9th edition, Addison Wesley, 2011.

3. http://vlabs.iitkgp.ernet.in/se/