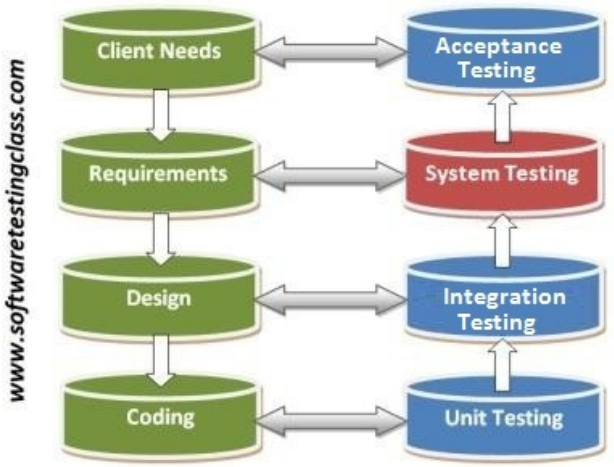
System Level Testing

System Level Testing validates the entire software architecture system and its component programs, both separately and as a whole. It typically takes place in the late stages of a project, after integration testing is complete and before user acceptance testing begins. The main purpose for system level testing is to test, validate and verify the system against both the Application Architecture and the Business Requirements.

Below is a diagram of the various levels of testing in most Software Development Life Cycles (SDLC). Notice the System Testing occurs between Integration Testing and Acceptance Testing.



There are several different kinds of tests that can be performed on a system level to test a program, including:

- *Graphical User Interface (GUI) Testing* – This can be done by simply running the programs in the system and going through the various actions to see how the GUI responds. In this process, the testers can make sure each action does what it is supposed to as defined in the System Architecture and Business requirements. The challenge with this type of testing occurs in large systems consisting of hundreds or thousands of GUI elements. The answer to this is to create automated test scripts that simulate the actions a live user would take.

- *Usability Testing* – This kind of testing is centered around the user experience (UIX). This is primarily performed by end users, often in an isolated environment, to catch errors and obtain input comparing system requirements with actual product usability and user expectations.

- *Load Testing* – This testing is typically done with servers or online services and webservers. It is designed to test how much the program or service can handle. It is a great method to perform stress testing to discover the threshold of simultaneous users before performance degrades or systems crash; and enables the company to ensure adequate hardware and online services are in place to handle actual users when the system officially launches.

- *Security Testing* – This testing is focused on overall system security. It helps identify security weakness and exploits prior to system launch. It can be completed through automated vulnerability scans as well as cybersecurity experts trained in ethical hacking that perform penetration testing.

A System Test Plan should include the following components:

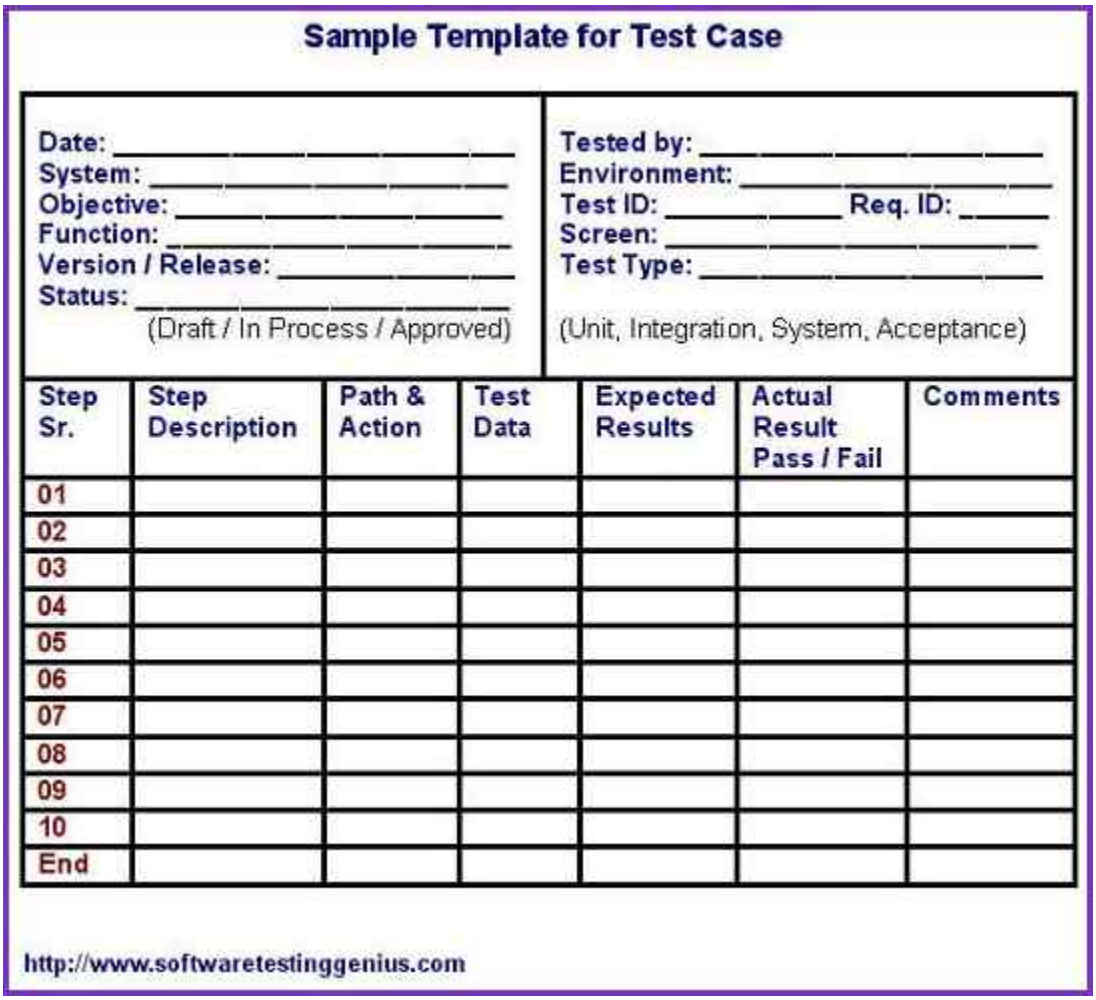
- Goals & Objective
- Scope
- Critical Focus Areas
- Test Deliverable
- Testing Strategy
- Testing Schedule
- Entry and Exit Criteria
- Suspension & Resumption Criteria
- Test Environment
- Roles and Responsibilities
- Glossary of Terms

(Source: http://www.softwaretestingclass.com/system-testing-what-why-how/)

Below is a sample Test Case in simple format. It is a great way to track system testing for smaller projects:

Test Case ID	Test Suite Name	How to Test?	Test Data	Expected Result	Actual Result	Pass/Fail

Below is another Test Case sample format. This format would be more appropriate for larger systems that require more detailed tracking of completed tests.



Sample Template for Test Case

Date: _____ System: _____ Objective: _____ Function: _____ Version / Release: _____ Status: _____ (Draft / In Process / Approved)	Tested by: _____ Environment: _____ Test ID: _____ Req. ID: _____ Screen: _____ Test Type: _____ (Unit, Integration, System, Acceptance)

Step Sr.	Step Description	Path & Action	Test Data	Expected Results	Actual Result Pass / Fail	Comments
01						
02						
03						
04						
05						
06						
07						
08						
09						
10						
End						

http://www.softwaretestinggenius.com

(Source: http://www.jobproposalideas.com/wp-content/uploads/2017/02/test-case-template-tpltc1-jpg.jpg)