

Table of Contents

[Mock Interview Question 3](#_Toc137748714)

[1. Introduction to Software Testing 3](#_Toc137748715)

[1.1. What is Software Testing? 3](#_Toc137748716)

[1.2. Why Software Testing? 3](#_Toc137748717)

[1.3. Benefits of Software Testing? 4](#_Toc137748718)

[1.4. What is Quality? 4](#_Toc137748719)

[1.5. What is defect? 4](#_Toc137748720)

[1.6. Project vs Product 4](#_Toc137748721)

[2. Software Development Life Cycle (SDLC) 4](#_Toc137748722)

[2.1 What is Software Development Life Cycle 4](#_Toc137748723)

[2.2 Why Software Development Life Cycle? 4](#_Toc137748724)

[2.3 Phases of Software Development Life Cycle 4](#_Toc137748725)

[2.3.1 Initial 5](#_Toc137748726)

[2.3.2 Analysis 5](#_Toc137748727)

[2.3.3 Design 5](#_Toc137748728)

[2.3.4 Coding 5](#_Toc137748729)

[2.3.5 Testing 5](#_Toc137748730)

[2.3.6 Delivery & Maintenance 6](#_Toc137748731)

[3. Software Development Life Cycle (SDLC) Models 6](#_Toc137748732)

[3.1. Water Fall Model 6](#_Toc137748733)

[3.2 V Model 7](#_Toc137748734)

[3.3 Prototype Model 8](#_Toc137748735)

[3.4 Agile Model 9](#_Toc137748736)

[4. Software Testing Methodologies 9](#_Toc137748737)

[4.1 Block Box Testing 10](#_Toc137748738)

[4.1.1 What is Block box testing? 10](#_Toc137748739)

[4.1.2 How will perform block box testing? 10](#_Toc137748740)

[4.1.3 How to perform block box testing? 10](#_Toc137748741)

[4.1.4 Block box testing techniques 10](#_Toc137748742)

[4.2 White box testing 10](#_Toc137748743)

[4.2.1 What is white box testing? 10](#_Toc137748744)

[4.2.2 Who will perform White box testing? 10](#_Toc137748745)

[4.2.3 How to perform White box testing? 10](#_Toc137748746)

[4.3 Grey Box testing 11](#_Toc137748747)

[4.3.1 What is grey box testing? 11](#_Toc137748748)

[4.3.2 Who will perform Grey box testing 11](#_Toc137748749)

[5. Levels of Testing 11](#_Toc137748750)

[5.1 Unit Testing 11](#_Toc137748751)

[5.2 Module/Component Testing 12](#_Toc137748752)

[5.3 Integration Testing 12](#_Toc137748753)

[5.4 System Testing 13](#_Toc137748754)

[5.5 User Acceptance Testing (UAT) 14](#_Toc137748755)

[6. Verification & Validation 16](#_Toc137748756)

[6.1 Verification 16](#_Toc137748757)

[6.1.1 What is Verification: 16](#_Toc137748758)

[**6.1.2. Who will perform:** 16](#_Toc137748759)

[6.1.3 Verification Techniques: 16](#_Toc137748760)

[6.1.4 Inspection: 16](#_Toc137748761)

[6.1.5 Walkthrough 16](#_Toc137748762)

[6.2 Validation 17](#_Toc137748763)

[6.1.1 What is Validation: 17](#_Toc137748764)

[6.1.2 Who will perform: 17](#_Toc137748765)

[7. Functional & Non Functional Testing 17](#_Toc137748766)

[7.1 Smoke Testing: 17](#_Toc137748767)

[7.2 Sanity Testing: 17](#_Toc137748768)

[7.3 Re-testing: 17](#_Toc137748769)

[7.5 Exploratory Testing: 18](#_Toc137748770)

[7.6 Monkey Testing: 18](#_Toc137748771)

[7.7 End to End Testing: 18](#_Toc137748772)

[7.8 User Interface Testing: 18](#_Toc137748773)

[7.9 Usability Testing: “ 18](#_Toc137748774)

[7.10 Stress Testing: 18](#_Toc137748775)

[7.11 Load Testing: 18](#_Toc137748776)

[7.12 Performance Testing: 19](#_Toc137748777)

[7.13 Localization Testing: 19](#_Toc137748778)

[7.14 Globalization Testing: 19](#_Toc137748779)

[7.15 Security Testing: 19](#_Toc137748780)

[7.16 Compatibility Testing: 20](#_Toc137748781)

[7.17 Installation Testing: 20](#_Toc137748782)

[7.18 Recovery Testing: 20](#_Toc137748783)

[8. Windows & Web Application 20](#_Toc137748784)

[8.1 What is Windows application: 20](#_Toc137748785)

[8.2 What is Web application: 20](#_Toc137748786)

[8.3 Difference between Windows & Web: 20](#_Toc137748787)

[9. Software Testing Life Cycle (STLC) 21](#_Toc137748788)

[9.1 Requirement Analysis: 21](#_Toc137748789)

[9.2 Test Planning: 22](#_Toc137748790)

[9.3 Test Case Development: 23](#_Toc137748791)

[9.3.1 What is test case: 23](#_Toc137748792)

[9.3.2 Test Case Techniques: 23](#_Toc137748793)

[9.3.3 Test Data Preparation What is Test Data: 24](#_Toc137748794)

[9.3.4 Types of test cases 24](#_Toc137748795)

[9.3.5 Test Case Review: 24](#_Toc137748796)

[9.4 Test Environment Setup 25](#_Toc137748797)

[9.5 Test Execution 25](#_Toc137748798)

[9.6 Result Analysis & Reporting: 25](#_Toc137748799)

[9.7 Delivery & Maintenance 28](#_Toc137748800)

# Mock Interview Question

1. What is S/W Testing?
2. What is Defect?
3. What is verification and validation?
4. What is SDLC Steps?
5. S/W development model?
6. Types of Reviews
7. Level of software testing?
8. Method of Software testing?
9. What is Graybox testing?
10. When to start QA(Testing Eng job)?
11. Difference between error and Defect/Bug and Failure?
12. What is Project / Product?
13. Ways of Software testing?
14. What is Manual testing / automation testing?

# Introduction to Software Testing

## What is Software?

A software is collection of computer program that helps us to perform a task.

## What is Software Testing?

“Software testing is a process of executing the application with the intent of finding the defects by comparing the output behavior of the application with expected behavior (requirement).” In other words, it’s comparing the actual behavior of an application with expected behavior.

The objective of testing is to release quality product to client.

## Why Software Testing?

Humans make mistakes all the time!!

“Software testing is really required to point out the defects and errors that were made during the development phases”.

We humans can’t identify our mistakes in a work done by us. We should get someone else to check our work because another person may identify the mistakes done by us. In the same way software developers may not identify the mismatches in a program or application implemented by them which can be identify by the another department called Software Test Engineer.

## Benefits of Software Testing?

“Software testing helps in finalizing the software application against business requirements.”

Software testing makes sure that the testing is being done properly and hence the system is ready for the customers to use.

Below are few benefits of software testing.

- Finding the defects before delivery

- Gaines the confidence about quality

- To Prevent defects

- Ensure the requirements are delivered to client.

## What is Quality?

“Software quality is nothing but delivering a bug free application and delivered on time with all requirements.”

ISO 8402-1986 standard defines quality as “the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs.”

**Quality Software Is reasonably**

* Bug-free
* Delivered on time.
* Within budget.
* Meets requirements and / or expectation
* Maintainable

## What is defect?

“A defect is a deviation or mismatch from the requirements”.

When actual result deviates from the expected result while testing a software application or product then it results into a defect. Hence, any deviation from the specification mentioned in the functional specification document is a defect. In different organizations it’s called differently like bug, issue, incidents or problem.

## Project vs Product

“If software application is developed for specific customer requirement then it is called **Project**.”

“If software application is developed for multiple customers’ requirement then it is called **Product**.”

# Software Development Life Cycle (SDLC)

## 2.1 What is Software Development Life Cycle

“SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software”

## 2.2 Why Software Development Life Cycle?

“SDLC ensure success in process of software development.”

## 2.3 Phases of Software Development Life Cycle

• Initial

• Analysis

• Design

• Coding

• Testing

• Delivery & Maintenance

### 2.3.1 Initial

“Business requirements are gathered in this phase. “

This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements like;

- Who is going to use the system?

- How will they use the system?

- What data should be input into the system?

- What data should be output by the system?

**Roles Involved:** Business Analyst (BA), System Architects

**Outcome:** System Requirement Specification (SRS)

### 2.3.2 Analysis

“After requirement gathering these requirements are analyzed for their validity and the possibility of developing the requirements in the system.”

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by both development team and testing team.

**Roles Involved:** Dev & QA team, Architects, Project Managers

**Outcome:** Final SRS approved by customer, Technology selection for both Dev & QA

### 2.3.3 Design

“During this part of the design phase, the consultants/architects break down the system into pieces that can be programmed.”

System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

**Roles Involved:** Architects & Team

**Outcome:** Technical Design Document (TDD)

### 2.3.4 Coding

“The actual development starts and the product is built in coding phase. “

The work is divided in modules/units and actual coding is started in this coding phase and it is the main focus for developer. Coding is one of the longest phase of SDLC.

**Roles involved:** Developers and Architects

**Outcome:** Programs or Application or Module

### 2.3.5 Testing

“In Testing phase testers execute the test cases against the application, report the defects and retested the fixed defects. “

During this phase unit testing, integration testing, system testing, acceptance testing are done.

**Roles Involved:** Testers, Developers

**Outcome:** Defects, Test Summary Report, Test Plan, Test Case document

### 2.3.6 Delivery & Maintenance

“Delivery: After successful testing the product is delivered / deployed to the customer.

During the Delivery phase, customer will perform user acceptance testing (UAT) in a real time environment.

Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

**Roles Involved:** Testers, Developers, Customer, Business team, Architects, Project Manager, and Delivery Manager

**Outcome:** Quality Product, Enhancements &Production Issues (Maintenance

# Software Development Life Cycle (SDLC) Models

“There are many development models life that have cycle been developed in order to achieve different required objectives.”

The selection of model has very high impact on the testing that is carried out. It will define the what, where and when of our planned testing, influence regression testing and largely determines which test techniques to use.

* Water Fall Model
* V Model
* Prototype Model
* Agile Model

## Water Fall Model

“Ina waterfall model, each phase must be completed fully before the next phase.

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. This type of model is basically used for the project which is small and there are no uncertain requirements.



**Advantages of waterfall model:**

- This model is simple and easy to understand and use.

- It is easy to manage due to the rigidity of the model

- In this model phases are processed and completed one at a time.

**Disadvantages of waterfall model:**

- Once an application is in the testing stage, it is very difficult to go back and change

- No working software is produced until late during the life cycle.

- High amounts of risk - Not a good model for complex and object-oriented projects.

- Poor model for long and ongoing projects.

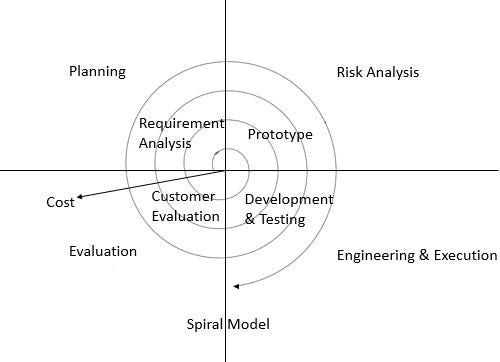
**When to use the waterfall model:**

- This model is used only when the requirements are very well known, clear and fixed.

- The project is short.

## Spiral Model

* Spiral Model is iterative model.
* Spiral Model overcome drawback of waterfall model.
* We follow spiral model whenever there is dependency on the modules.
* In every cycle new software will be released to customer.
* Software will be released in multiple versions. So it is also called version control model.



**Advantages of Spiral-model:**

* Testing is done in every cycle, before going to the next cycle.
* Customer will get to use the software for every module.
* Requirements changes are allowed after every cycle before going to the next cycle.

**Disadvantages of Spiral-model:**

* Requirements changes are not allowed in between the cycle.
* Every cycle of spiral model looks like waterfall model.
* There is no testing in requirement & design phase.

## V Model

“The – V model is a SDLC model where execution of processes happens in a sequential manner in V-shape. “

V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development.

**Advantages of V-model:**

- Simple and easy to use.

- Testing activities like planning, test designing happens well before coding. This saves a lot of time.

- Proactive defect tracking

- That is defects are found at early stage.

**Disadvantages of V-model:**

- Software is developed during the implementation phase, so no early prototypes of the software are produced.

- If any changes happen in midway, then the test documents along with requirement documents has to be updated. When to use the V-model:

- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.

- The V-Shaped model should be chosen when sample technical resources are available with needed technical expertise.

**When to use the V-model:**

- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.

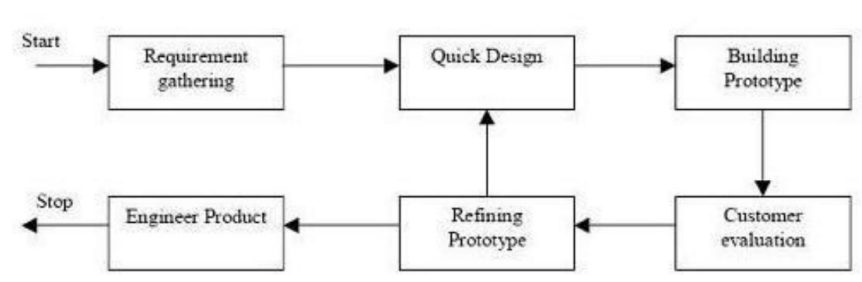
- The V-Shaped model should be chosen when sample technical resources are available with needed technical expertise.



## Prototype Model

“Blue print of the software”

* Initial requirements from the customer 🡪Prototype🡪Customer🡪Design, coding, testing …..



**Advantages of Prototype model:**

- Users are actively involved in the development

- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.

- Errors can be detected much earlier. - Quicker user feedback is available leading to better solutions. - Missing functionality can be identified easily - Confusing or difficult functions can be identified

**Disadvantages of Prototype model:**

- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.

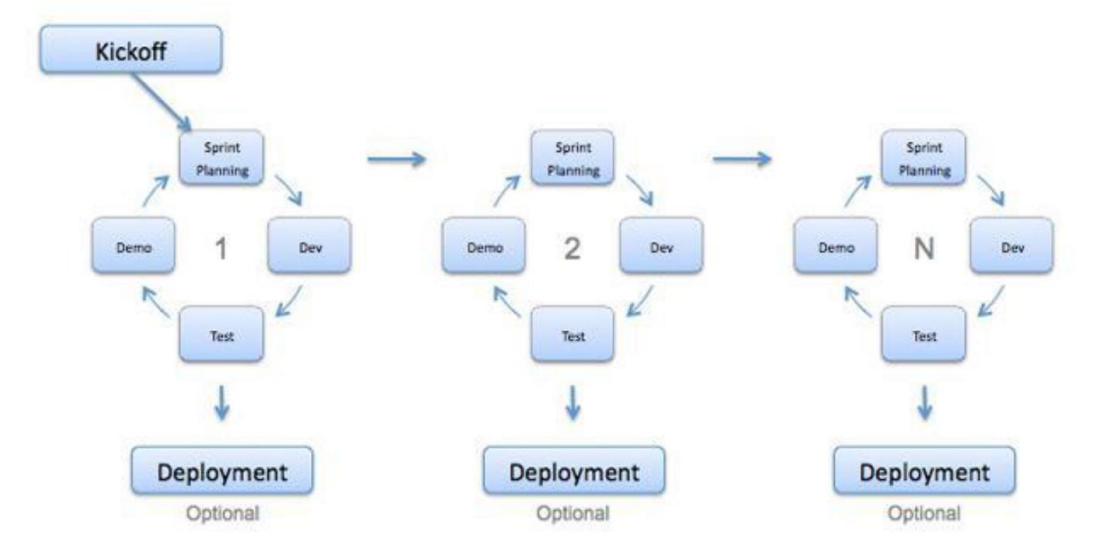
- Incomplete application may cause application not to be used as the full system was designed

**When to use Prototype model:**

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.

## Agile Model

“Software is developed in incremental, rapid small incremental cycles releases. This with result search release in building on previous functionality.”



## Static v/s Dynamic Testing

* **Static testing** is an approach to test the project documents in the form of Reviews, Walkthroughs and Inspection.
* **Dynamic testing** is an approach to test the actual software by giving inputs and observing results.

# Software Testing Methodologies

## 4.1 Block Box Testing

### 4.1.1 What is Block box testing?

“The technique of testing without having technical knowledge of an application i

Specification-based testing technique is also known as ‘black-box’ or input/output driven testing techniques because they view the software as a black-box with inputs and outputs.

### 4.1.2 How will perform block box testing?

“Testing team will perform the block box testing”

The testers have no knowledge of how the system or component is structured inside the box. In black-box testing the tester is concentrating on what the software does, not how it does it.

### 4.1.3 How to perform block box testing?

Block box testing covers both functional and non-functional testing. Functional testing is concerned with what the system does its features or functions. Non-functional testing is concerned with examining how well the system does. Non-functional testing like performance, usability, portability, maintainability, etc

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Well suited and efficient for large code segments. | Limited Coverage since only a selected number of test scenarios are actually performed. |
| Code Access not required. | Inefficient testing, due to the fact that the tester only has limited knowledge about an application. |
| Clearly separates user's perspective from the developer's perspective through visibly defined roles. | Blind Coverage, since the tester cannot target specific code segments or error prone areas. |
| Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems. | The test cases are difficult to design. |

### 4.1.4 Block box testing techniques

Below is the black box testing techniques:

- Equivalence partitioning

- Boundary value analysis

- Error Guessing

## 4.2 White box testing

### 4.2.1 What is white box testing?

“Structure-based testing technique is also known as ‘white-box’ or ‘glass-box’ testing technique because here testers require knowledge of how the software is implemented, how it works “

### 4.2.2 Who will perform White box testing?

“Developers use –based structure techniques in component testing and component integration testing, especially where there is good tool support for code coverage.”

### 4.2.3 How to perform White box testing?

White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also called g lass testing or Open box testing.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively. | Due to the fact that a skilled tester is needed to perform white box testing, the costs are increased. |
| It helps in optimizing the code | Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems as many paths |
| Extra lines of code can be removed which can bring in hidden defects | will go untested |
| Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing | It is difficult to maintain white box testing as the use of specialized tools like code analyzers and debugging tools are required. |

## 4.3 Grey Box testing

### 4.3.1 What is grey box testing?

“Grey Box testing is a technique application with to limited test knowledge the of the internal working of an application.”

### 4.3.2 Who will perform Grey box testing

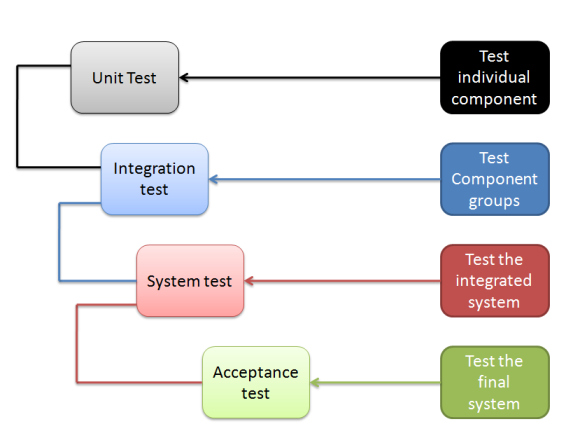
Unlike black testing, box where the tester only tests the application's user interface, in grey box testing, the tester has access to design documents and the database. Having this knowledge, the tester is able to better prepare test data and test scenarios when making the test plan.”

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Offers combined benefits of black box and white box testing wherever possible. | Since the access to source code is not available, the ability to go over the code and test coverage is limited. |
| Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications. | The tests can be redundant if the software designer has already run a test case. |
| Based on the limited information available, a grey box tester can design excellent test scenarios especially around communication protocols and data type handling. | Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program paths will go untested. |
| The test is done from the point of view of the user and not the designer. |  |

# Levels of Testing

“In software development life cycle models there are defined phases like require coding or implementation, testing and deployment. Each phase goes through the testing.”

Hence there are various levels of testing. The various levels of testing are:

s

## 5.1 Unit Testing

**What is Unit testing:** “A unit is the smallest testable part of an application procedures, ion like function interfaces.”

Unit testing is a method by which individual units of source code are tested to determine if they are fit for use.

* A unit is a single component or module of a software.
* Unit testing conduct on a single program or single module.
* Unit testing is white box testing technique.
* Unit testing is conducted by the developers.
* Unit testing technique
  + Basic path testing
  + Control structure testing
    - Conditional coverage
    - Loops coverage
  + Mutation Testing

**Who will perform Unit testing:** “Unit tests are basically written and executed to make by software sure that code meets its design and requirements and behaves as expected.”

## 5.2 Module/Component Testing

**What is Module/Component Testing:** “Component testing is a method where testing of each c application is done separately.”

Suppose, in an application there are 5 components. Testing of each 5 components separately and efficiently is called as component testing.

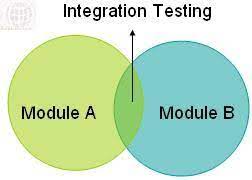
**Who will perform Component Testing:** “Component testing is done by the tester.”

## 5.3 Integration Testing

**What is Integration Testing:** “Integrations testing done when two modules are integrated, in order to test the behavior and functionality of both the modules after integration.”

As displayed in the image below when two different modules ‘Module A’ and ‘Module B’ integration testing is done.

* In Integration testing, individual software modules are integrated logically and tested as group.
* Integration testing focuses on checking data communication among these modules.
* Integrated Testing is white box testing technique.
* Integrated testing is conducted by the developers.



**Who will perform Integration Testing:** “Integration testing is done by a specific integration tester or test team.

**Types of integration testing:**

* + Incremental Integration Testing
  + Non-Functional Integration Testing

What is Incremental Integration Testing?

* Integration Testing is done after unit testing.
* It is the process of verifying interfaces and interactions between modules.
* As the developers integrate modules one by one this type of testing is called as Incremental Integration Testing.
* In Incremental Integration Testing, the developers integrate the modules one by one using stubs or drivers to uncover the defects.

**Stubs:** Is called by the Module under test.

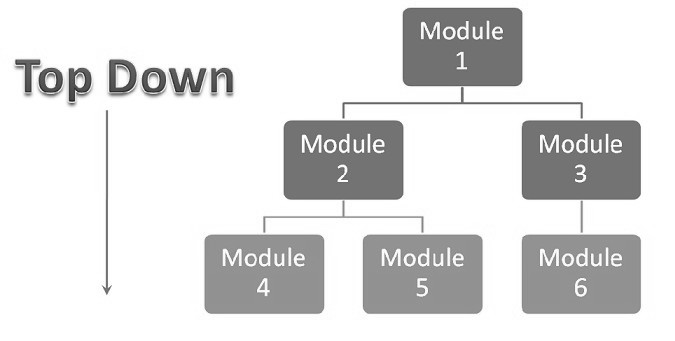
**Driver**: Calls the Module to be tested.

**Methodologies in Incremental Integration Testing**

* Top Down Integration
* Bottom Up Integration
* Functional Incremental

**Top-down integration testing:** Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs.

Below is the diagram of ‘Top down App roach’:



**Advantages of Top-Down approach:**

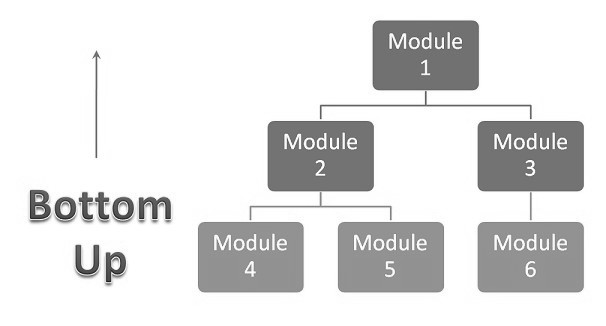
* The tested product is very consistent because the integration testing is basically performed in an environment that almost similar to that of reality
* Stubs can be written with lesser time because when compared to the drivers then Stubs are simpler to author.

**Disadvantages of Top-Down approach:**

* Basic functionality is tested at the end of cycle

**Bottom-up integration testing:** Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers.

Below is the image of ‘Bottom up approach’:



**Advantage of Bottom-Up approach:**

* In this approach development and testing can be done together so that the product or application will be efficient and as per the customer specifications.

**Disadvantages of Bottom-Up approach:**

* We can catch the Key interface defects at the end of cycle
* It is required to create the test drivers for modules at all levels except the top control

**Non-Incremental Integration Testing:** Adding all the module in a single shot and test the data flow between modules.

Drawback of Non-Incremental Integration testing

* We might miss data flow between some of the modules.
* If you find any defect, we can’t understand the root cause of defect.

## 5.4 System Testing

**What is system testing:**

* Testing over all functionality of the application with respective client requirements.
* It is a black box testing technique.
* The testing is conducted by testing team.
* Before conducting system testing we should know the requirements.
* System Testing focuses on below aspects.
* User Interface Testing(GUI)
* Functional Testing
* Non-Functional Testing
* Usability Testing

What is GUI?

* **Graphical User-Interface testing** or **GUI testing** is the process of testing the System’s GUI.
* GUI testing involves checking the screens with the control like menus, buttons, icons and all type of bars, tool bar, menu bar, dialog boxes and windows etc.

Following aspects:

* Look & feel
* Easy to use
* Navigation & Shortcuts keys
* GUI objects:
* Window, Dialog box, push button, Radio button, radio group, Tool bar, Edit box, Text Box, check box, List box, drop down box, Combo box, Tab, Tree view, progress bar, table, scroll bar etc.

GUI Testing Checklist

* Testing the size, position, width, height of the elements.
* Testing of the error message that are getting displayed.
* Testing the different sections of the screen.
* Testing of the font whether it is readable or not.
* Testing of the screen in different resolutions with the help of zooming in and zooming out.
* Testing the alignment of the texts and other elements like icons, buttons, etc. are in proper place or not
* Testing the colors of the fonts.
* Testing whether the image has clarity or not.
* Testing the alignment of the images.
* Testing of the spelling.
* Testing user must not get frustrated while using the system interface.
* Testing whether the interface is attractive or not.
* Testing of the scrollbars according to the size of the page if any.
* Testing of the size of the images.
* Testing of the heading whether it is properly aligned or not.
* Testing of the color of the hyperlink
* Testing UI Elements like button, textbox, text area, checkbox, radio buttons, dropdowns, link etc.

**Usability Testing**

* During the testing validates application provided context sensitive help or not to the user.
* Check hoe easily the end users are able to understand and operate the application is called usability testing.

**Functional Testing**

* Functionality is nothing but behavior of application.
* Functional testing talks about how your feature should work.
* Object Properties Testing
* Input Domain Coverage (BVA, ECP)
* Database Testing/Backend Coverage
* Error Handling
* Calculation/Manipulation Testing
* Links Existence & Link Execution
* Cookies & Sessions

**Object Properties Testing**

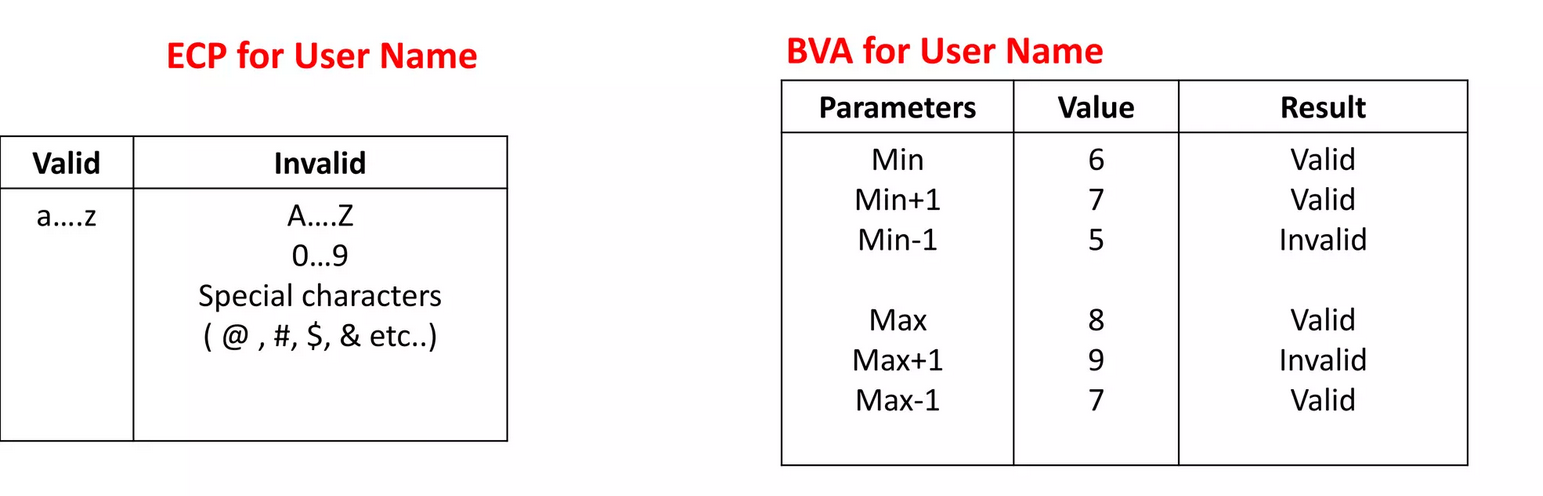
* Every object has certain properties.
* Ex: Enable, Disable, Focus, Text etc.
* During Functional testing test Engineers validate properties of object in run time.

**Input Domain Testing**

* During input domain testing test engineers validate data provided to the application w.r.t value and length.
* There are two techniques in input domain technique.
* Equalance Class Partition (ECP)
* Boundary Value Analysis (BVA)

**ECA & BVA**

* Requirements: -
* User name field allows only lower case with min 6 and max 8 letters.



**Database Testing**

* During database testing test engineers validate the data w.r.t. database.
* Validate DML operations (Insert, Update, Delete & Select).
* SQL language: DDL, DML, DCL etc.
* DDL – Data definition language – CREATE, alter, drop.
* DML – Data manipulation language – Insert, update, select, delete.
* DCL – Commit, roll back etc.

**Error Handling Testing**

* Validate error message thrown by the application when we provide invalid data.
* The error message should be clear and easy to understand to the user.

**Calculation Testing/Manipulation Testing**

* Validate mathematical calculations.

**Links Existence & Links Execution**

* Link existence – Links places in the appropriated location or not.
* Links execution – Link is navigating to appropriate page or not.
* Types of links:
* Internal links
* External links
* Broken links

**Cookies & Sessions**

* Cookie – Temporary internet files which are created at client side when we open the web sites. These files contain User data.
* Session – Sessions are time slots which are allocated to the user at the serve side.

**Non-Functional Testing**

* Performance Testing
* Load Testing
* Stress Testing
* Volume Testing
* Security Testing
* Recovery Testing
* Compatibility Testing
* Configuration Testing
* Installation Testing
* Sanitation Testing

**Performance Testing:** Speed of the application.

* **Load:** Gradually increasing the load on the application then check the speed of the application.
* **Stress:** Suddenly increase/decrease the load on the application and check speed of the application.
* **Volume:** Check how much data is able to handle by the application.

**Security Testing:** How secure our application.

* **Authentication:** Users are valid or not.
* **Authorization/Access Control:** Permission of the valid user.
* **Encryption/Decryption:**

**Recovery Testing**

* Testing Recovery provided by the system. Whether the system recovering abnormal to normal condition or not.

**Compatibility Testing**

* Testing Compatibility of the system w.r.t. OS, H/W & Browsers.
* Operating System Compatibility
* Hardware Compatibility (Configuration Testing)
* Browser Compatibility
* Forward & Backward Compatibility

**Installation Testing**

* Testing Installation pf the application on Customer expected platforms and check installation steps, navigation steps, navigation, how much space is occupied in memory.
* Check Un-Installation.

**Sanitation Testing/Garbage Testing**

* Check whether application is providing any extra/additional featured beyond the customer requirements.

|  |  |
| --- | --- |
| Functional Testing | Non-Functional Testing |
| Validates functionality of software | Verify the performance, security, reliability of the software. |
| Functionality describes what software does | Non-Functionality describes how software works |
| Concentrates on user requirement | Concentrates on user expectation |
| Functional testing takes place before Non-Functional testing | Non-Functional testing performed after Functional testing |

## 5.5 User Acceptance Testing (UAT)

**What is Acceptance testing or User Acceptance Testing (UAT)?**

* After the system test has corrected all or most defects, the system will be delivered to the user or customer for **Acceptance Testing** or **User Acceptance Testing (UAT)**.
* Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well.
* The goal of acceptance [testing](https://tryqa.com/what-is-a-software-testing/) is to establish confidence in the system.
* Acceptance testing is most often focused on a validation type testing.
* Acceptance testing may occur at more than just a single level, for example:
* A **Commercial Off the shelf (COTS)**software product may be acceptance tested when it is installed or integrated.
* **Acceptance testing of the**[usability of the component](https://tryqa.com/what-is-usability-testing-in-software-and-its-benifits-to-end-user/)may be done during component testing.
* **Acceptance testing of a new functional enhancement**may come before [system testing](https://tryqa.com/what-is-system-testing/).

**The types of acceptance testing are:**

* The User Acceptance test: focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user. The user acceptance test is performed by the users and application managers.
* The Operational Acceptance test: also known as Production acceptance test validates whether the system meets the requirements for operation. In most of the organization the operational acceptance test is performed by the system administration before the system is released. The operational acceptance test may include testing of backup/restore, disaster recovery, maintenance tasks and periodic check of security vulnerabilities.
* Contract Acceptance testing: It is performed against the contract’s acceptance criteria for producing custom developed software. Acceptance should be formally defined when the contract is agreed.
* Compliance acceptance testing: It is also known as regulation acceptance testing is performed against the regulations which must be adhered to, such as governmental, legal or safety regulations.

**Who will perform UAT:** “Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well.”

**What is Alpha testing?**

Alpha testing is one of the most common [software testing](https://tryqa.com/what-is-a-software-testing/) [strategy](https://tryqa.com/what-are-the-test-approaches-or-strategies-in-software-testing/) used in software development. Its specially used by product development organizations.

* This test takes place at the developer’s site. Developers observe the users and note problems.
* Alpha testing is testing of an application when development is about to complete. Minor design changes can still be made as a result of alpha testing.
* Alpha testing is typically performed by a group that is independent of the design team, but still within the company, e.g. in-house software test engineers, or software QA engineers.
* Alpha testing is final testing before the software is released to the general public. It has two phases:
  + In the first phase of alpha testing, the software is tested by in-house developers. They use either debugger software, or hardware-assisted debuggers. The goal is to catch bugs quickly.
    - In the second phase of alpha testing, the software is handed over to the software QA staff, for additional testing in an environment that is similar to the intended use.
* Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers’ site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

**What is Beta testing?**

* **Beta Testing** is also known as field testing. It takes place at **customer’s site**. It sends the system/software to users who install it and use it under real-world working conditions.
* A beta test is the second phase of [software testing](https://tryqa.com/what-is-a-software-testing/) in which a sampling of the intended audience tries the product out. (Beta is the second letter of the Greek alphabet.) Originally, the term [alpha testing](https://tryqa.com/what-is-alpha-testing/) meant the first phase of testing in a software development process. The first phase includes [unit testing](https://tryqa.com/what-is-unit-testing/), [component testing](https://tryqa.com/what-is-component-testing/), and [system testing](https://tryqa.com/what-is-system-testing/). Beta testing can be considered “pre-release testing.
* The goal of beta testing is to place your application in the hands of real users outside of your own engineering team to discover any flaws or issues from the user’s perspective that you would not want to have in your final, released version of the application. Example: Microsoft and many other organizations release beta versions of their products to be tested by users.

**Advantages of beta testing**

* You have the opportunity to get your application into the hands of users prior to releasing it to the general public.
* Users can install, test your application, and send feedback to you during this beta testing period.
* Your beta testers can discover issues with your application that you may have not noticed, such as confusing application flow, and even crashes.
* Using the feedback, you get from these users, you can fix problems before it is released to the general public.
* The more issues you fix that solve real user problems, the higher the quality of your application when you release it to the general public.
* Having a higher-quality application when you release to the general public will increase customer satisfaction.
* These users, who are early adopters of your application, will generate excitement about your application.

# Verification & Validation

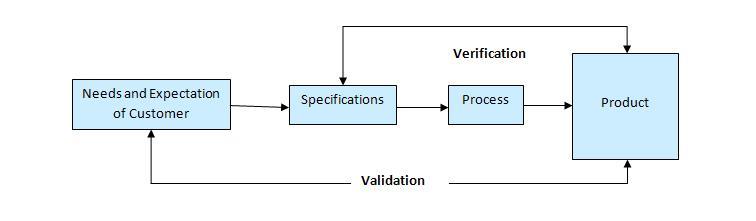
## 6.1 Verification

### 6.1.1 What is Verification:

“The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements.

* Verification checks whether we are building the right product.
* Focus on Documentation.

Verification is done at the starting of the development process. It includes reviews and meetings, walkthroughs, inspection, etc. to evaluate documents, plans, code, requirements and specifications.



### **6.1.2. Who will perform:**

Peers (Sr Team Members, Architects, Analysts)

### 6.1.3 Verification Techniques:

Below are the validation techniques

- Reviews

- Inspections

- Walk through

**Reviews:**

“A review is a systematic examination of document by one or more people with the main aim of finding and removing errors early in the software development life cycle.”

There are two types of Reviews held in verification. They are Formal Review and Informal Review.

* 1. **Formal Review:** “Formal reviews follow a formal process. It is well structured and regulated. It contains: Planning, Kick-off, Preparation, Review meeting, Rework.”
  2. **Informal Review:** “Informal reviews are applied many times during the early stages of the life cycle of the document. A two-person team can conduct an informal review.” The most important thing to keep in mind about the informal reviews is that they are not documented
* Requirements Review
* Design Reviews
* Code Reviews
* Test plan reviews
* Test cases reviews etc.

### 6.1.4 Inspection:

“Inspection is the most formal form of reviews, a strategy adopted during static testing phase.”

- It is the most formal review type

- It is led by the trained moderators

- During inspection the documents are prepared and checked thoroughly by the reviewers before the meeting.

- In which at least 3-8people will sit in the meeting 1–reader 2–writer 3-moderator plus concerned.

- Inspection will have a proper schedule which will be intimated via email to the concerned developer/tester.

### 6.1.5 Walkthrough

“A walkthrough is conducted by ‘document he author under of the review’ who takes the participants document and his or her thought processes, to achieve a common understanding and to gather feedback.”

* It is an informal review.
* Author reads the documents or code and discuss with peers.
* It’s not pre-planned and can be done whenever required.
* Also walkthrough does not have minutes of the meet.

## 6.2 Validation

### 6.1.1 What is Validation:

“The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements.”

### 6.1.2 Who will perform:

Testing Team, Dev Team, Client or BA team

|  |  |  |
| --- | --- | --- |
| Criteria | Verification | Validation |
| objective | To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements. | To ensure that the product actually meets the user’s needs, and that the specifications were objective specifications. In other words, to ensure that demonstrate that the product fulfills its intended u se when placed in its intended environment. |
| Question | Are we building the product right? | Are we building the right product? |
| Evaluation | Items Plans, Requirement Specs, Design Specs, Code, Test Cases | The actual product/software. |
| Activities | Reviews, Walkthroughs, Inspections | Testing |

## QA vs QC/QE

- QA is process related.

- QC is the actual testing of the software.

- QA focuses on building in quality.

- QC focuses on testing for quality.

- QA is preventing defects.

- QC is detecting defects.

- QA is process oriented.

- QC is product oriented.

- QA for entire life cycle.

- QC for testing part in SDLC.

# Testing Terminology

## Regression Testing:

* “Regression testing is performed to verify if the build has NOT broken any other parts of the application by the recent code changes for defect fixing or for enhancement.
* Testing conducts on modified build to make sure there will not be impact on existing functionality because of changes like adding/deleting/modifying features.
* **Unit Regression Testing**: Testing only the changes/modification done by the developer.
* **Regional Regression Testing**: Testing the modified along with the impacted modules.

Impact analysis meeting conducts to identify impacted module with QA & Dev.

* **Full Regression**: Testing the main feature & remaining part of the application.

**Ex**: Dev has done changes in many modules, instead of identifying impacted modules, we perform one round of full regression.

## Re-testing:

* Whenever the developer fixed a bug, tester will test the bug fix is called Re-testing.
* Tester close the bug if it worked otherwise re-open and send to developer.
* To ensure that the defects which were found and posted in the earlier build were fixed or not in the current build.
* Example
* Build 1.0 was released. Test team found some defect (Defect Id 1.0.1, 1.0.2) and posted
* Build 1.1. was released, now testing the defects 1.0.1 and 1.0.2 in this build is retesting.

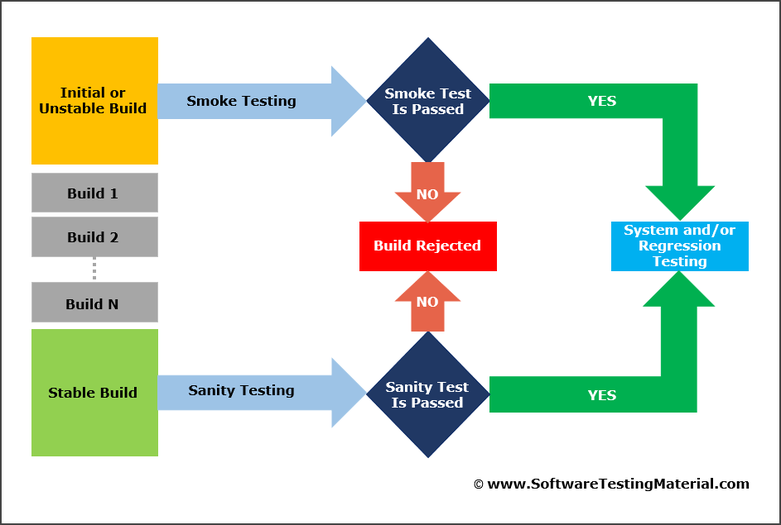
**Example: Re-Testing Vs Regression Testing**

* An Application Under Test has three modules namely Admin, Purchase and Finance.
* Finance module depends on purchase module.
* If a tester found a bug on purchase module and posted. Once the bug is fixed, the tester needs to do Resting to verify whether the bug related to the purchase is fixed or not and also tester need to do regression testing to test the Finance module which depends on the Purchase module.

## Smoke Vs Sanity Testing

* Smoke and Sanity Testing come into the picture after build release.

|  |  |
| --- | --- |
| Smoke Testing | Sanity Testing |
| Smoke Test is done to make sure the build we received from development team is testable/stable or not. | Sanity Test is done during the release phase to check for main functionalities of the application without going deeper. |
| Smoke Testing is performed by both developers and testers. | Sanity Testing is performed by Testers alone. |
| Smoke Testing, build may be either stable or unstable. | Sanity Testing, build is relatively stable. |
| It is done on initial builds. | It is done on stable builds. |
| It is a part of basic testing. | It is a part of regression testing. |
| Usually it is done every time there is a new build release. | It is planned when there is no enough time to do in-depth testing. |



## Exploratory Testing:

“Testing of software without any documents (test cases or test planning) and Identify the functionality of application by exploring the application.

* We have to explore the application, understand completely and test it.
* Understand the application, identify all possible scenario, document it then use it for testing.
* We do exploratory testing when the Application ready but there is no requirement.
* Test Engineer will do exploratory testing when there is no requirement.

**Drawbacks**:

* You might misunderstand any feature as a bug (or) any bus as a feature since you do not have requirements.
* Time consuming
* If there is any bug in application, you will never know about it.

## Adhoc Testing

* Testing application randomly without any test cases or any business requirements document.
* Adhoc testing is an informal testing type with an aim to break the system.
* Tester should have knowledge of application even though he doesn’t have requirements/test cases.
* This testing is usually an unplanned activity.



## Monkey Testing:

* Testing application randomly without any test cases or any business requirement document.
* Adhoc is an informal testing type with an aim to break the system.
* Tester do not have knowledge of application.
* Suitable for gaming application.

## End to End Testing:

“End-to-end testing is a methodology used to test whether the flow of an application is performing as designed from start to finish.”

The purpose of carrying out end-to-end tests is to identify system dependencies and to ensure that the right information is passed between various system components and systems.

**Difference between System testing and End to End testing:**

There isn't really a huge difference between the two and in some establishments the terms could be used interchangeably. Everywhere is different.

**System testing:** You're testing the whole system i.e. all of its components to ensure that each is functioning as intended. This is more from a functional side to check against requirements.

**End to end testing:** This is more about the actual flow through a system in a more realistic end user scenario. Can a user navigate the application as expected and does it work. You're testing the workflow.

**Non Functional Testing:**

## User Interface Testing:

“Graphical User Interface (GUI) testing is checking the application design of an application”.

Ex: Required/Optional, Fields Align, Lengths, Progress Bars, Scroll Bars, Alignments, etc

## Usability Testing: “

In usability testing basically the testers tests the ease with which the user interfaces can be used. It tests that whether the application is user-friendly or not. “

Usability Testing tests the following features of the software.

– How easy it is to use the software.

– How easy it is to learn the software.

– How convenient is the software to end user.

## Stress Testing:

“It is a form of testing that is used to determine the stability of a given system, Stress testing involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results.

“Stress testing is a generic term used to describe the process of putting a system through stress.

## Load Testing:

“Load testing is performed to determine a system’s behavior under both normal and at peak conditions. “

A load test is usually conducted to understand the behavior of the application under a specific expected load.

E.g. If the number of users are increased then how much CPU, memory will be consumed, what is the network and bandwidth response time.

## Performance Testing:

“Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload. “

It can serve different purposes like it can demonstrate that the system meets performance criteria.

## Localization Testing:

“Localization translates the product UI and occasionally changes some initial settings to make it suitable for another region.” Localization testing checks the quality of a product's localization for a particular target culture/locale.

The test effort during localization testing focuses on:

- Areas affected by localization, such as UI and content

- Culture/locale-specific, language-specific, and region-specific areas

## Globalization Testing:

“Globalization Testing is testing process to check whether software can perform properly in any locale or culture & functioning properly with all types of international inputs and steps to effectively make your product truly global.”

This type of testing validates whether the application is capable for using all over the world and to check whether the input accepts all the language texts.

Ex: Let’s see another example of a Zip code field in Sign up form:

- For globalized, it should allow to enter alphanumeric inputs

- For localized (country like INDIA), it should allow only numbers in input field

## Security Testing:

“Security testing is basically to check that whether the application or the product is secured or not.

“Can anyone came tomorrow and hack the system or login the application without any authorization. It is a process to determine that an information system protects data and maintains functionality as intended.

Security testing is related to the security of data and the functionality of the application. You should be aware of the following concepts while performing security testing:

1. Confidentiality - The application should only provide the data to the relevant party e.g. one customer's transactional data should not be visible to another customer; the irrelevant personal details of the customer should not be visible to the administrator and so on.

2. Integrity - The data stored and displayed by the application should be correct e.g. after a withdrawal, the customer's account should be debited by the correct amount.

3. Authentication - It should be possible to attribute the data transmitted in the application to either the application or the customer. In other words, no one other than the customer or the bank should be able to create or modify any data.

4. Authorization - The application or a user should only be able to perform the tasks which they are respectively authorized to perform e.g. a customer should not be able to withdraw more than the balance in their account without having an overdraft facility, the application should not be able to levy charges on a customer account without prior customer approval.

5. Availability - The data and functionality should be available to the users throughout the working period e.g. if the bank's operating times are from 8 a.m. to 8 p.m. on all working days, it should be possible for a customer to access their account and make the necessary transactions on their account.

6. Non-repudiation - At a later date, it should not be possible for a party to deny that a particular transaction or data change took place e.g. if a customer withdraws an amount from their account, this should trigger the relevant actions (posting to their transaction records, debiting their account and sending them a notification etc.)

## Compatibility Testing:

” Compatibility Testing ensure compatibility of the application built with various other objects such as other web browsers, hardware platforms, operating systems etc.”

This type of testing helps find out how well a system performs in a particular environment that includes hardware, network, operating system and other software etc.

Ex: Browser Compatibility Testing, OS Compatibility Testing

## Installation Testing:

“Installation testing is performed to ensure that all necessary components are installed properly and working as per the requirements of the software, post installation. Installation process may include partial, full or upgrade install. “

## Recovery Testing:

“Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or failure “

Ex: For example, when an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application’s ability to continue receiving data from the point at which the network connection got disappeared. Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.

# Windows & Web Application

## 8.1 What is Windows application:

“A program that is written to run under Microsoft's Windows operating system. “

## 8.2 What is Web application:

“Web application is an application that is accessed via Web browser over a network such as the Internet or an intranet. “

## 8.3 Difference between Windows & Web:

**Windows Application:**

1. Windows applications typically run under all 32-bit versions of Windows, but earlier applications might also run under the 16-bit versions (Windows 3.x) as well.
2. Runs on personal computers and work stations
3. c. Window based app. need to be installed on your machine to access.
4. Windows applications (desktop) need to be installed on each client's PC.
5. Windows application runs faster than Web application.
6. Windows application has many inbuilt classes in .Net compared to Web application.

**Web application**:

a. It is a computer software application that is coded in a browser-supported language (such as HTML, ASP, PHP, Perl, Python etc.) and reliant on a common web browser to render the application executable.

b. Web applications are very much useful when they are hosted.Web app. can be access from any ware in the world through the internet.

c. Web application is tested mainly for browser compatibility and operating system compatibility, error handling, static pages, back-end testing and load testing.

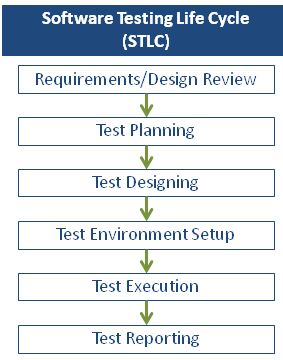
d. Web applications are programs that used to run inside some web server (e.g., IIS) to fulfill the user requests over the http.

e. Common Web applications include Webmail, online retail sales, online auctions, wikis, discussion boards, Weblogs

# 9. Software Testing Life Cycle (STLC)

“STLC consists of series of activities carried out methodologically to help certify your software product. These activities are part of the Software Testing Life Cycle.”

The different stages in Software Test Life Cycle:



## 9.1 Requirement Analysis:

“During this phase, test team studies the requirements from a testing point of view to identify the testable requirements.”

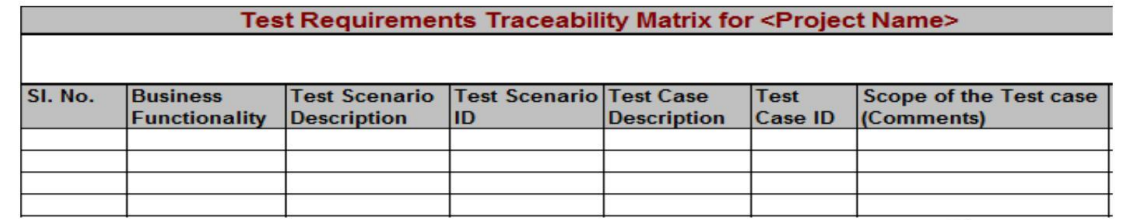
The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, and System Architects etc.) to understand the requirements in detail. Requirements could be either Functional (defining what the software must do) or Non Functional (defining system performance /security availability).

**Deliverables:** Requirement Traceability Matrix (RTM), Clarification Document

**RTM:** “Traceability matrix links a business requirement to its corresponding functional requirement.”

If a Test Case fails, traceability helps determine the corresponding functionality easily; it also helps ensure that all requirements are tested

**Sample RTM:**



**Clarification Document:** “Contains all clarifications which will arise during the requirement analysis phase”.

**Sample:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Date** | **Module/Page Name** | **Query Description** | **Comments** |
|  |  |  |  |  |
|  |  |  |  |  |

## 9.2 Test Planning:

**What is Test plan:** “Test planning is the first step of the testing process. In this phase we identify the activities and resources which would help to meet the testing objectives.”

**What Test Plan contains: (IEEE 829 STANDARD TEST PLAN TEMPLATE)**

Test plan identifier

Test deliverables

Introduction

Test tasks

Test items

Environmental needs

Features to be tested Responsibilities

Features not to be tested

Staffing and training needs

Approach Schedule

Item pass/fail criteria

Risks and contingencies

**Who will prepare the test plan:** Test Lead or Test Manager.

**What is Test Strategy**: “A Test Strategy document is a high level document and normally developed by project manager. This document defines “Software Testing Approach” to achieve testing objectives. The Test Strategy is normally derived from the Business Requirement Specification document.”

**Test plan Vs Test Strategy**: Generally, it doesn’t matter which comes first. Test planning document is a combination of strategy plugged with overall project plan. According to IEEE Standard 829-2008, strategy plan is a sub item of test plan.

Every organization has their own standards and processes to maintain these documents. Some organizations include strategy details in test plan itself. Some organizations list strategy as a subsection in testing plan but details is separated out in different test strategy document.

Ex: Test plan gives the information of who is going to test at what time. For example: Module 1 is going to be tested by “X tester”. If tester Y replaces X for some reason, the test plan has to be updated.

On the contrary, test strategy is going to have details like – “Individual modules are to be tested by test team members. “In this case, it does not matter who is testing it- so it’s generic and the change in the team member does not have to be updated, keeping it static.

**Deliverables:** Test Plan with estimation

## 9.3 Test Case Development:

“During this phase the test cases will be prepared”.

### 9.3.1 What is test case:

“A test case is a set of conditions under which a tester will determine whether a system under test satisfies requirements or works correctly.”

**Writing Good Test Cases:**

* As far as possible, write test cases in such a way that you test only one thing at a time. Do not overlap or complicate test cases. Attempt to make your test cases ‘atomic’.
* Ensure that all positive scenarios and negative scenarios are covered.
* Language:
* Write in simple and easy to understand language.
* Use active voice: Do this, do that.
* Use exact and consistent names (of forms, fields, etc.).
* Characteristics of a good test case:
* Accurate: Exacts the purpose.
* Economical: No unnecessary steps or words.
* Traceable: Capable of being traced to requirements.
* Repeatable: Can be used to perform the test over and over.
* Reusable: Can be reused if necessary

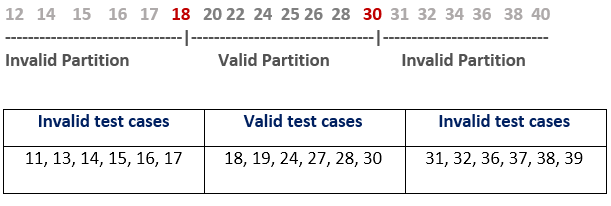
### 9.3.2 Test Case Techniques:

* Equivalence Portioning (EP)
* Boundary Value Analysis (BVP)
* Error Guessing

**Equivalence Portioning**: “EP divides the input data of a software unit into partitions of equivalent data from which test cases can be derived.”

Test cases are designed to cover each partition at least once. This technique tries to define test cases that uncover classes of errors, thereby reducing the total number of test cases that must be developed. An advantage of this approach is reduction in the time required for testing a software due to lesser number of test cases.

**Boundary Value Analysis:** “Tests are designed to include representatives of boundary values “



**Error Guessing:** “The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.”

The success of error guessing is very much dependent on the skill of the tester, as good testers know where the defects are most likely to be. This is why an error guessing approach, used after more formal techniques have been applied to some extent, can be very effective.

### 9.3.3 Test Data Preparation What is Test Data:

“In order to test a software application you need to enter some data for testing most of the features. Any such specifically identified data which is used in tests is known as test data.”

**Test data preparation:** In the above example we can generate the inputs for Valid and Invalid partitions. Ex: Valid: 1, 2, 5, 10, 11, 12 Invalid: -2, -1, 0, 13, 15, 99, 150, 999

### 9.3.4 Types of test cases

**Functional Test Cases:** “The test cases based on functional requirement specifications”

**Positive Test Cases:** “Test Cases with valid input and also verifying that the outputs are correct.”

**Negative Test Cases:** “This testing involves exercising application functionality using a combination of invalid inputs, some unexpected operating conditions and by some other “out-of-bounds” scenarios.”

**Non Functional Test Cases:** “The test cases based on functional requirement specifications like performance, Load, Stress, Security, etc.”

### 9.3.5 Test Case Review:

Reviewing is a form of testing too – the verification part of the V&V, also called static testing.

**Why Review:** For exactly the same reason we test the software

* To uncover errors
* To check for completeness
* To make sure the standards and guidelines are followed

**Review Checklist:**

* Do test cases cover all requirements?
* Has each test case been assigned a test case identifier?
* Does each test case specify?

- Actions - Test condition

- Expected result

* Have the expected results been recorded in detail?
* Is any method for validating expected results specified?
* Do test cases for field validations, record validations and database updates include the following?

- Valid conditions

- Invalid conditions

- Boundary or unusual conditions

* Do the test cases for reports include the test data along with the expected output?
* Have the inter test case dependencies been described?
* Have Pass/Fail criteria been specified?
* Have all requested environments been specified?
* Has the method for logging on to the test environment been specified?
* Are pre-conditions for the test specified?
* Is the number of Test cases met customer standards?

**Self-Review:** “Review our own work by us”

**Peer Review**: “Review our own work by colleague”

**Lead Review:** “Review our work by our Test lead or Manager”

**Client Review:** “Review our work by client or business team after lead review.”

**Deliverables:** Test Case Document, Test Data

## 9.4 Test Environment Setup

“Test environment decides the software and hardware conditions under which a work product is tested. “Test environment set-up is one of the critical aspects of testing process and can be done in parallel with Test Case Development Stage. Test team may not be involved in this activity if the customer or dev team provides the test environment in which case the test team is required to do a readiness check (smoke testing) of the given environment.

**Deliverables:** Test Environment

**How many environments do we have:** “A Typical project can have following environments”?

- Dev

- QA

- Pre-Production

- Production

## 9.5 Test Execution

“In this phase testing team start executing test cases based on prepared test planning & prepared test cases in the prior step in testing environment”.

**Deliverables:** Test cases updated with result

## 9.6 Result Analysis & Reporting:

“After test execution phase, if the test case is passed then same can be marked as Passed. If any test case is failed then corresponding defect can be reported to developer team via bug tracking system & bug can be linked for corresponding test case for further analysis.”

**Severity:** It is the extent to which the defect can affect the software. In other words, it defines the impact that a given defect has on the system.

If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.

* **Critical**: The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
* **Major**: The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
* **Medium**: The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.
* **Minor**: The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
* **Cosmetic**: The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

**Priority:** “Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait?”

This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned, then the developer has to fix it at the earliest. The priority status is set based on the customer requirements.

For example: If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

* Low: The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect has been fixed.
* Medium: The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
* High: The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.

**Few very important scenarios related to the severity and priority which are asked during the interview:**

* High Priority & High Severity: An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Eg. A site maintaining the student details, on saving record if it, doesn’t allow to save the record then this is high priority and high severity bug.)
* High Priority & Low Severity: The spelling mistakes that happens on the cover page or heading or title of an application.
* High Severity & Low Priority: An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.
* Low Priority and Low Severity: Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

**Famous Defect Tracking Tools:**

- Bugzilla

- JIRA

- ALM (QC)

**A typical Defect Tracker will have:**

- Defect id

- Date

- Created By

- Assigned TO

- Bug description

- Steps to Reproduce

- Expected Result

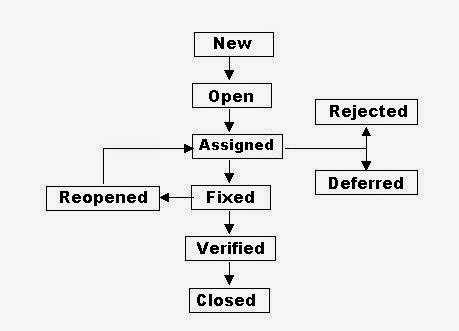
- Comments

- Screen shots

**Defect Life Cycle:** “Defect life cycle is a cycle which a defect goes through during its lifetime. “

It starts when defect is found and ends when a defect is closed, after ensuring it’s not reproduced. Defect life cycle is related to the bug found during testing.

The Life cycle of the bug can be shown diagrammatically as follows:



**New:** When a defect is logged and posted for the first time. Its state is given as new.

**Assigned:** After the tester has posted the bug, the lead of the tester approves that the bug is genuine and he assigns the bug to corresponding developer and the developer team. Its state given as assigned.

**Open:** At this state the developer has started analyzing and working on the defect fix

**Fixed**: When developer makes necessary code changes and verifies the changes then he/she can make bug status as ‘Fixed’ and the bug is passed to testing team.

**Pending** **retest**: After fixing the defect the developer has given that particular code for retesting to the tester. Here the testing is pending on the testers end. Hence its status is pending retest.

**Retest**: At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.

**Verified**: The tester tests the bug again after it got fixed by the developer. If the bug is not present in the software, he approves that the bug is fixed and changes the status to “verified”.

**Reopen**: If the bug still exists even after the bug is fixed by the developer, the tester changes the status to “reopened”. The bug goes through the life cycle once again.

**Closed**: Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, he changes the status of the bug to “closed”. This state means that the bug is fixed, tested and approved.

**Duplicate**: If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to “duplicate “.

**Rejected**: If the developer feels that the bug is not genuine, he rejects the bug. Then the state of the bug is changed to “rejected”.

**Deferred**: The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are priority of the bug may be low, lack of time for the release or the bug may not have major effect on the software.

**Not** **a** **bug**: The state given as “Not a bug” if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and field of the application like change of color of some text then it is not a bug but just some change in the looks of the application.

## 9.7 Delivery & Maintenance

Evaluate cycle completion criteria based on Time, Test overage, Cost, Software, Critical Business Objectives, Quality.

Typical Exit criteria may include the following:

* All Test plans have been run.
* A certain level of requirements coverage has been achieved.
* No high priority or severe bugs are left outstanding.
* All high-risk areas have been fully tested, with only minor residual risks left outstanding.
* Cost – when the budget has been spent.
* The schedule has been achieved.

**Maintenance**: Once a system is deployed it is in service for years and decades. During this time the system and its operational environment is often corrected, changed or extended. Testing that is provided during this phase is called maintenance testing.

Usually maintenance testing is consisting of two parts:

First one is, testing the changes that has been made because of the correction in the system or if the system is extended or because of some additional features added to it.

Second one is regression tests to prove that the rest of the system has not been affected by the maintenance work.

**Metrics**: A Metric is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute.

Metrics can be defined as “STANDARDS OF MEASUREMENT”.

Software Metrics are used to measure the quality of the project. Simply, Metric is a unit used for describing an attribute. Metric is a scale for measurement.

Suppose, in general, “Kilogram” is a metric for measuring the attribute “Weight”. Similarly, in software, “How many issues are found in thousand lines of code?”, here No. of issues is one measurement & No. of lines of code is another measurement. Metric is defined from these two measurements.

**Test metrics example:**

* How many defects are existed within the module?
* How many test cases are executed per person?
* What is the Test coverage %?

**Why Test Metrics?**

Generation of Software Test Metrics is the most important responsibility of the Software Test Lead/Manager.

“We cannot improve what we cannot measure.”

“We cannot control what we cannot measure”

* Take decision for next phase of activities
* Evidence of the claim or prediction
* Understand the type of improvement required
* Take decision on process or technology change

**Effectiveness**: Doing the right thing. It deals with meeting the desirable attributes that are expected by the customer.

**Efficiency**: Doing the thing right. It concerns the resources used for the service to be rendered

1. **Test Plan coverage on Functionality:**

Formula: (No of requirements covered / total number of requirements) \* 100

1. **Test Case defect density: Formula**:

(Defective Test Scripts /Total Test Scripts) \* 100

**Example**: Total test script developed 1360, total test script executed 1280, total test script passed 1065, total test script failed 215.

So, test case defect density is :215 X 100 / 1280 = 16.8%

This 16.8% value can also be called as test case efficiency %, which is depends upon total number of test cases which uncovered defects.

1. **Defect Slippage Ratio:** Number of defects slipped (reported from production) v/s number of defects reported during execution.

**Formula**: Number of Defects Slipped / (Number of Defects Raised - Number of Defects Withdrawn)

**Example:** Customer filed defects are 21, total defect found while testing is 267, total number of invalid defects are 17.

So, Slippage Ratio is [21/ (267-17)] X 100 = 8.4%

1. **Requirement Volatility**: Number of requirements agreed v/s number of requirements changed.

**Formula**: Number of Requirements Added + Deleted + Modified) \*100 / Number of Original Requirements

**Example**: VSS 1.3 release had total 67 requirements initially, later they added another 7 new requirements and removed 3 from initial requirements and modified 11 requirements.

So, requirement Volatility is (7 + 3 + 11) \* 100/67 = 31.34%

Means almost 1/3 of the requirement changed after initial identification

1. **Review Efficiency**: The Review Efficiency is a metric that offers insight on the review quality and testing some organization also use this term as “Static Testing” efficiency and they are aiming to get min of 30% defects in static testing.

**Formula**=100\*Total number of defects found by reviews/Total number of project defects

**Example**: A project found total 269 defects in different reviews, which were fixed and test team got 476 defects which were reported and valid So, Review efficiency is [269/(269+476)] X 100 = 36.1% f. Defect Removal Effectiveness(DRE):

DRE= (Defects removed during development phase x100%) / Defects latent in the product Defects latent in the product = Defects removed during development Phase+ defects found later by user

------------------------------------------------------------ End ----------------------------------------------------------------

**Advantages of Agile model:**

- Customer satisfaction by continuous delivery of software.

- Customers, developers and testers constantly interact with each other.

- Working software is delivered frequently

- Continuous attention to technical excellence and good design.

- Even late changes in requirements are welcomed

**Disadvantages of Agile model:**

- It is difficult to assess the effort required at the beginning of the software development life cycle.

- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.

- Only senior programmers are capable of taking the kind of decisions required during the development process. When to use Agile model:

- New changes can be implemented at very little cost

- To implement a new feature the developers need to lose only the work of a few days.

- Unlike the waterfall model in agile model very limited planning is required to get started with the project.

**Software Testing Principles**

1. Early Testing -
2. Defect Clustering - It is not possible to test all possible combination of data and scenarios.
3. Pesticide Paradox - If I run the same test case in application version 4. Will we get the bug ? NO.
4. Testing is Context Dependent
5. Absence of Errors Fallacy - The testing is done but we didn’t get the error.
6. Shows Presence of Error - Testing shows the presence of Bugs. The test cases that we have to design are very important.
7. Exhaustive Testing is Impossible

**Basic Terms**

Test Plan

Items to be Tested

Level of Testing

Sequence of Testing

Test strategy to be applied to test each item

Describe the test environment

**Test Plan Document**

* Scope
* Approach
* Resources
* Schedule
* Features to be tested
* Task Planning
* Test Environment
* Acceptance & system test planning
* Integration & unit testing

Test Case

* Input System
* Output System
* Pres-conditions
* Expected Results
* That is called Test Case
* And after adding a few input values we get to know about the system that the output that we are expecting, is really coming with the system or it is not coming.

**Test Data**

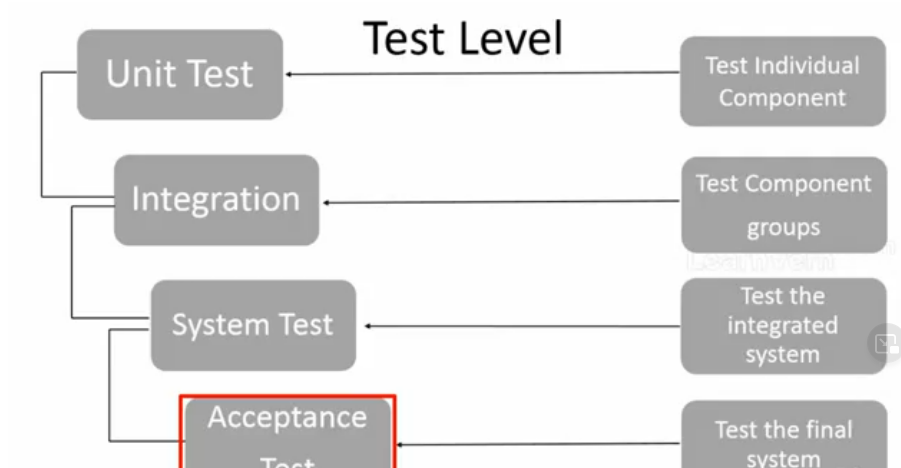
* Test data is the data which is used to execute the test.
* The test data should be precise and exhaustive by which all the defect are uncovered.

**Test Script**

* Test script are the set of instruction that we will follow when we will perform the test scripts on the system.
* Test scripts are commonly called test case.

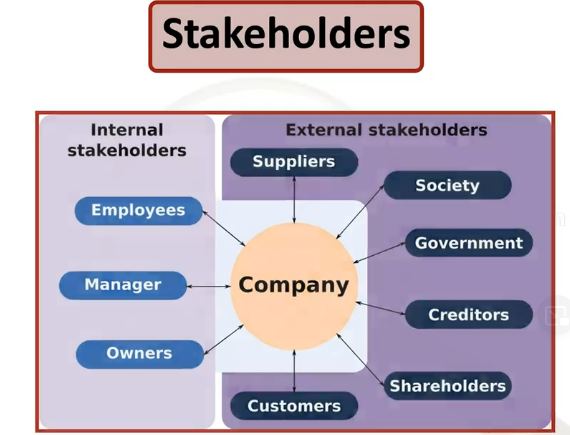
**Test Log**

* Which test cases have passed and which ones have failed.



**Test Summary**

* Inputs
* Result
* Actual Result
* Expected Results



What is Bug?

* A bug is result of programming error because of which of functionality does not give us the expected output than it will called as bug.

What is an Error?

* Expected output and actual output difference called as an error.

What is Failure?

* IF a software does not work as its capacity than we will called failure.

What is Defect?

* When actual output differs from the customer expectations, then we consider it as a Defect.

What is Mistake?

* If there is any mistake by user, then we will call as mistake.

What is HLR?

* It stands for High Level Requirement.
* HLR was used to do the Tracing.
* HLR should be made.

How to make HLR of a Website?

* These three column are require for HLR Document.
* Functionality ID
* Functionality Name
* Description

|  |  |  |
| --- | --- | --- |
| **Function-ality ID** | **Functionality Name** | **Description** |
| 1 | check Website URL | while entering URL into address bar it should be open the learn vern Website |
| Home Page | | |
| 100 | Check All Courses | while hover the cursor on All Courses its open sub details of the course |
| 101 | check Technology under All Courses | while hover the cursor on Technology under the All Courses to open the sub Technology details |
| 102 | check Business under All Courses | while hover the cursor on Business under the All Courses to open the sub Business details |
| 103 | check Designing Tools under All Courses | while hover the cursor on Designing Tools under the All Courses to open the sub Designing Tools details |
| 104 | check Mechanical/Civil under All Courses | while hover the cursor on Mechanical/Civil under the All Courses to open the sub Mechanical/Civil details |
| 105 | check Live Project under All Courses | while hover the cursor on Live Project under the All Courses to open the sub Live Project details |
| 106 | check Free Webinar under All Courses | while hover the cursor on Free Webinar under the All Courses to open the sub Free Webinar details |
|  |  |  |
| 200 | check Login icon Menu | While clicking on that icon it should be open Login Page |
| 201 | check login via Gmail | while clicking on that button its must require login first on gmail |
| 202 | check login via Facebook | while clicking on that button its must require login first on facebook |
| 203 | check login via email | while clicking on that button its must require login first on email |
| 204 | check create account button | while clicking on that create account should be open |

**Software Testing Life Cycle**

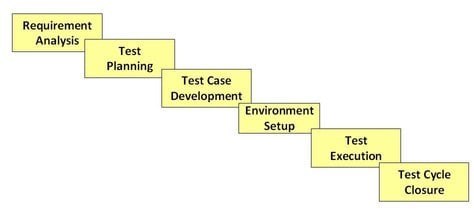
What is STLC?

* Software Testing Life Cycle (STLC) is a sequence of specific activities conducted during the testing process to ensure software quality goals are met.
* STLC involves both verification and validation activities.
* It consists of a series of activities carried out methodologically to help certify your software product. STLC stands for Software Testing Life Cycle.

**STLS Phases**

There are following six major phases in every Software Testing Life Cycle Model(STLS Model):

1. Requirement Analysis
2. Test Planning
3. Test Case development
4. Test Environment Setup
5. Test Execution
6. Test Cycle closure



Each of these stages has a definite Entry and Exit criteria, Activities & Deliverables associated with it.

What is Entry and Exit Criteria in STLC?

**Entry Criteria**: Entry Criteria gives the prerequisite items that must be completed before testing can begin.

**Exit Criteria**: Exit Criteria defines the items that must be completed before testing can be concluded

You have Entry and Exit Criteria for all levels in the Software Testing Life Cycle (STLC)

In an Ideal world, you will not enter the next stage until the exit criteria for the previous stage is met. But practically this is not always possible. So for this tutorial, we will focus on activities and deliverables for the different stages in STLC life cycle. Let’s look into them in detail.

**1. Requirement Phase: -**

Requirement Phase Testing also known as Requirement Analysis in which test team studies the requirements from a testing point of view to identify testable requirements and the QA team may interact with various stakeholders to understand requirements in detail. Requirements could be either functional or non-functional. Automation feasibility for the testing project is also done in this stage.

**Activities in Requirement Phase Testing**

* Identify types of tests to be performed.
* Gather details about testing priorities and focus.
* Prepare Requirement Traceability Matrix (RTM).
* Identify test environment details where testing is supposed to be carried out.
* Automation feasibility analysis (if required).

**Deliverables of Requirement Phase Testing**

* RTM
* Automation feasibility report. (if applicable)

**2. Test Planning: -**

Test Planning in STLC is a phase in which a Senior QA manager determines the test plan strategy along with efforts and cost estimates for the project. Moreover, the resources, test environment, test limitations and the testing schedule are also determined. The Test Plan gets prepared and finalized in the same phase.

**Test Planning Activities**

* Preparation of test plan/strategy document for various types of testing
* Test tool selection
* Test effort estimation
* Resource planning and determining roles and responsibilities.
* Training requirement

**Deliverables of Test Planning**

* Test plan /strategy document.
* Effort estimation document.

**3. Test Case Development: -**

The Test Case Development Phase involves the creation, verification and rework of test cases & test scripts after the test plan is ready. Initially, the Test data is identified then created and reviewed and then reworked based on the preconditions. Then the QA team starts the development process of test cases for individual units.

**Test Case Development Activities**

* Create test cases, automation scripts (if applicable)
* Review and baseline test cases and scripts
* Create test data (If Test Environment is available)

**Deliverables of Test Case Development**

* Test cases/scripts
* Test data

**4. Test Environment Setup: -**

Test Environment Setup decides the software and hardware conditions under which a work product is tested. It is one of the critical aspects of the testing process and can be done in parallel with the Test Case Development Phase. Test team may not be involved in this activity if the development team provides the test environment. The test team is required to do a readiness check (smoke testing) of the given environment.

**Test Environment Setup Activities**

* Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
* Setup test Environment and test data
* Perform smoke test on the build

**Deliverables of Test Environment Setup**

* Environment ready with test data set up
* Smoke Test Results.

1. **Test Execution**

Test Execution Phase is carried out by the testers in which testing of the software build is done based on test plans and test cases prepared. The process consists of test script execution, test script maintenance and bug reporting. If bugs are reported then it is reverted back to development team for correction and retesting will be performed.

**Test Execution Activities**

* Execute tests as per plan
* Document test results, and log defects for failed case
* Map defects to test cases in RTM
* Retest the Defect fixes
* Track the defects to closure

**Deliverables of Test Execution**

* Completed RTM with the execution status
* Test cases updated with results
* Defect reports

1. **Test Cycle closure**

Test Cycle Closure phase is completion of test execution which involves several activities like test completion reporting, collection of test completion matrices and test results. Testing team members meet, discuss and analyze testing artifacts to identify strategies that have to be implemented in future, taking lessons from current test cycle. The idea is to remove process bottlenecks for future test cycles.

**Test Cycle Closure Activities**

* Evaluate cycle completion criteria based on Time, Test coverage, Cost,Software, Critical Business Objectives, Quality
* Prepare test metrics based on the above parameters.
* Document the learning out of the project
* Prepare Test closure report
* Qualitative and quantitative reporting of quality of the work product to the customer.
* Test result analysis to find out the defect distribution by type and severity.

**Deliverables of Test Cycle Closure**

* Test Closure report
* Test metrics

**Development Models**

**What is SDLS?**

**SDLC** is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase. SDLC stands for **Software Development Life Cycle** and is also referred to as the Application Development life-cycle.

**Why SDLC?**

Here, are prime reasons why SDLC is important for developing a software system.

* It offers a basis for project planning, scheduling, and estimating
* Provides a framework for a standard set of activities and deliverables
* It is a mechanism for project tracking and control
* Increases visibility of project planning to all involved stakeholders of the development process Increased and enhance development speed
* Improved client relations
* Helps you to decrease project risk and project management plan overhead

**SDLC Phases**

The entire SDLC process divided into the following SDLC steps:

**SDLC Phases**



1: Requirement collection and analysis

2: Defining Requirement

3: Design

4: Coding

5: Testing

6: Installation/Deployment

7: Maintenance

**1: Requirement collection and analysis**

The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and reorganization of the risks involved is also done at this stage.

This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.

Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

**Phase 2: Feasibility study**

Once the requirement analysis phase is completed the next sdlc step is to define and document software needs. This process conducted with the help of ‘Software Requirement Specification’ document also known as ‘SRS’ document. It includes everything which should be designed and developed during the project life cycle.

**There are mainly five types of feasibilities checks:**

* Economic: Can we complete the project within the budget or not?
* Legal: Can we handle this project as cyber law and other regulatory framework/compliances.
* Operation feasibility: Can we create operations which is expected by the client?
* Technical: Need to check whether the current computer system can support the software
* Schedule: Decide that the project can be completed within the given schedule or not.

**3: Design**

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

This design phase serves as input for the next phase of the model.

There are two kinds of design documents developed in this phase:

**High-Level Design (HLD)**

* Brief description and name of each module
* An outline about the functionality of every module
* Interface relationship and dependencies between modules
* Database tables identified along with their key elements
* Complete architecture diagrams along with technology details

**Low-Level Design (LLD)**

* Functional logic of the modules
* Database tables, which include type and size
* Complete detail of the interface
* Addresses all types of dependency issues
* Listing of error messages
* Complete input and outputs for every module

**4: Coding**

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

**5: Testing**

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

**6: Installation/Deployment**

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

**7: Maintenance**

Once the system is deployed, and customers start using the developed system, following 3 activities occur

* Bug fixing – bugs are reported because of some scenarios which are not tested at all
* Upgrade – Upgrading the application to the newer versions of the Software
* Enhancement – Adding some new features into the existing software

**What is Waterfall model?**

The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**.  It is very simple to understand and use.  In a waterfall model, each phase must be completed fully before the next phase can begin. This type of [software development model](http://tryqa.com/what-are-the-software-development-models/) is basically used for the project which is small and there are no uncertain requirements.

**Advantages of waterfall model**

* This model is simple and easy to understand and use.
* It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
* In this model phases are processed and completed one at a time. Phases do not overlap.
* Waterfall model works well for smaller projects where requirements are clearly defined and very well understood.

**Disadvantages of waterfall model**

* Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects
* Not suitable for the projects where requirements are at a moderate to high risk of changing.

**When to use the waterfall model**

* This model is used only when the requirements are very well known, clear and fixed.
* Product definition is stable.
* Technology is understood.
* There are no ambiguous requirements
* Ample resources with required expertise are available freely
* The project is short.

In Waterfall model, very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demonstrated to the end users.

Once the product is developed and if any failure occurs then the cost of fixing such issues are very high, because we need to update everything from document till the logic.

In today’s world, Waterfall model has been replaced by other models like iterative, agile etc.

**Forms of Testing**

**Which are forms of Testing?**

There are two forms of testing

* Static Testing
* Dynamic Testing

**What is Static Testing?**

* Static testing was started in the early days of the life cycle.
* Static testing is a review.
* At the documentation time if there is an Error or if there is Unclear Requirement there, then we clear that.
* There are some test cases made in such a way that we cannot understand them or we are getting misguided.

**Static Analysis**

* The code that the developers are writing are analyzed by some tools, so that going ahead it doesn’t raise any defect.

**When & Where the Static testing should be used?**

* We have to use static testing as it will give you the defect at the start.
* Static Testing is 100 times more effective than dynamic testing.
* Like there is a 10-line code. If in the 10-line code, if we are doing code analysis with the Static Testing.
* We will get to know if the 10-line code is really required or it can get over in 7-line or for that requirement, writing this big code is not necessary? All these things you can cover in Static Testing.
* Most of the bugs are found in Static Testing.
* There are static tool that are also there. Some companies use static tools which benefits them.
* If you skip the tools of dynamic testing but it will be better if you don’t skip static tools.

**Which are the techniques that are used for Static testing?**

* Informal
* Walkthrough
* Peer Review
* Inspection

**What is Informal Review?**

* We have to find the errors in the document but there is not particular process, like 1st you will have the meeting and after that you will follow the process, there is no such process.
* We will see the documents and give our comments that this particular requirement cannot go.
* We can make the requirements better in this way and many more.

**What is a Walkthrough?**

* Walkthrough is an evaluation process which is an informal meeting, which does not require us to prepare anything.
* Walkthrough is important for a High-Level Document which is called requirement specification which is important for any architectural document.

**What are the goals of Walkthrough?**

* All the documentations that are there we will present them to the stakeholders and we will take information from that whether anything is missing in the documentation or somewhere we have to correct it.

**What is Code Review?**

**What is Inspection?**

**What is Dynamic Testing?**

* Dynamic Testing is done after the executing the software.

**Types of Testing**

* Unit Testing
* Integration Testing
* System Testing

**Unit Testing**

* In Unit Testing we will test the small units.
* The other name of units is module.
* Developer will test the small functions like addition.

**Integration Testing**

**System Testing**

* Entire System is Tested.
* Applications is Developed According to requirements Specifications Documentations(SRD).

**When & Where to use Dynamic Testing?**

* Dynamic Testing is About finding & fixing the defects.
* Dynamic Testing involves test cases for execution.
* Dynamic Testing covers the executables files of the code.
* It is done during validation process.
* It is tested by executing it on computer.

**Levels of Testing**

**What is Unit Testing?**

**What is Incremental Integration Testing?**

* Integration Testing is done after unit testing.
* It is the process of verifying interfaces and interactions between modules.
* As the developers integrate modules one by one this type of testing is called as Incremental Integration Testing.
* In Incremental Integration Testing, the developers integrate the modules one by one using stubs or drivers to uncover the defects.

**Methodologies in Incremental Integration Testing**

* Top Down Integration
* Bottom Up Integration
* Functional Incremental

**Top Down Integration**

* The testing beings by testing the top level module and adds it in lower levels.

• Unavailable components or nodules are substituted by stubs.

* Stubs are dummy functions.

**What is System Testing?**

**Testing Methods**

**Types of Testing**

**Types of Testing - Non Functional Testing**

**Defect Management**

**How to Write test Case**

**Project - Testing**

**Interview Questions**

**Test Case Development**

**Test Documentation in Software Testing**

**What is Test Documentation?**

Test documentation is documentation of artifacts created before or during the testing of software. It helps the testing team to estimate testing effort needed, test coverage, resource tracking, execution progress, etc. It is a complete suite of documents that allows you to describe and document test planning, test design, test execution, test results that are drawn from the testing activity.

**Example of Test Documentation**

Here, are important Types of Test Documentation:

|  |  |
| --- | --- |
| Types of Testing Documents | Description |
| Test policy | It is a high-level document which describes principles, methods and all the important testing goals of the organization. |
| Test strategy | A high-level document which identifies the Test Levels (types) to be executed for the project. |
| Test plan | A test plan is a complete planning document which contains the scope, approach, resources, schedule, etc. of testing activities. |
| Requirements Traceability Matrix | This is a document which connects the requirements to the test cases. |
| Test Scenario | Test scenario is an item or event of a software system which could be verified by one or more Test cases. |
| Test case | It is a group of input values, execution preconditions, expected execution post conditions and results. It is developed for a Test Scenario. |
| Test Data | Test Data is a data which exists before a test is executed. It used to execute the test case. |
| Defect Report | Defect report is a documented report of any flaw in a Software System which fails to perform its expected function. |
| Test summary report | Test summary report is a high-level document which summarizes testing activities conducted as well as the test result. |

**Best practice to Achieve Test Documentation**

* QA team needs to be involved in the initial phase of the project so that Test Documentation is created in parallel
* Don’t just create and leave the document, but update whenever required
* Use version control to manage and track your documents
* Try to document what is needed for you to understand your work and what you will need to produce to your stakeholders
* You should use a standard template for documentation like excel sheet or doc file
* Store all your project related documents at a single location. It should be accessible to every team member for reference as well as to update when needed
* Not providing enough detail is also a common mistake while creating a test document

**Advantages of Test Documentation**

* The main reason behind creating test documentation is to either reduce or remove any uncertainties about the testing activities. Helps you to remove ambiguity which often arises when it comes to the allocation of tasks
* Documentation not only offers a systematic approach to software testing, but it also acts as training material to fresher’s in the software testing process
* It is also a good marketing & sales strategy to showcase Test Documentation to exhibit a mature testing process
* Test documentation helps you to offer a quality product to the client within specific time limits
* In Software Engineering, Test Documentation also helps to configure or set-up the program through the configuration document and operator manuals
* Test documentation helps you to improve transparency with the client

**Disadvantages of Test Documentation**

* The cost of the documentation may surpass its value as it is very time-consuming
* Many times, it is written by people who can’t write well or who don’t know the material
* Keeping track of changes requested by the client and updating corresponding documents is tiring.
* Poor documentation directly reflects the quality of the product as a misunderstanding between the client and the organization can occur

**Summary**

* Test documentation is documentation of artifacts created before or during the testing of software.
* The degree of test formality depends on 1) the type of application under test 2) standards followed by your organization 3) the maturity of the development process.
* Important types of Test Documents are Test policy, Test strategy, Test plan, Test case etc.
* QA team needs to be involved in the initial phase of the project so that Test Documentation is created in parallel
* The main reason behind creating test documentation is to either reduce or remove any uncertainties about the testing activities.
* The cost of the documentation may surpass its value as it is very time-consuming

**What is Test Scenario? Template with Examples**

**What is A Test Scenario?**

A Test Scenario is defined as any functionality that can be tested. It is also called Test Condition or Test Possibility. As a tester, you should put yourself in the end user’s shoes and figure out the real-world scenarios and use cases of the Application Under Test.

**Scenario Testing**

Scenario Testing in software testing is a method in which actual scenarios are used for testing the software application instead of test cases. The purpose of scenario testing is to test end to end scenarios for a specific complex problem of the software. Scenarios help in an easier way to test and evaluate end to end complicated problems.

**Why create Test Scenarios?**

Test Scenarios are created for the following reasons,

* Creating Test Scenarios ensures complete Test Coverage
* Test Scenarios can be approved by various stakeholders like Business Analyst, Developers, Customers to ensure the Application Under Test is thoroughly tested. It ensures that the software is working for the most common use cases.
* They serve as a quick tool to determine the testing work effort and accordingly create a proposal for the client or organize the workforce.
* They help determine the most important end-to-end transactions or the real use of the software applications.
* For studying the end-to-end functioning of the program, Test Scenario is critical.

**When not create Test Scenario?**

Test Scenarios may not be created when

* The Application Under Test is complicated, unstable and there is a time crunch in the project.
* Projects that follow Agile Methodology like Scrum, Kanban may not create Test Scenarios.
* Test Scenario may not be created for a new bug fix or Regression Testing. In such cases, Test Scenarios must be already heavily documented in the previous test cycles. This is especially true for Maintenance projects.

**How to Write Test Scenarios**

As a tester, you can follow these five steps to create Test Scenarios-

* **Step 1**: Read the Requirement Documents like BRS, SRS, FRS, of the System Under Test (SUT).  You could also refer uses cases, books, manuals, etc. of the application to be tested.
* **Step 2**: For each requirement, figure out possible users actions and objectives. Determine the technical aspects of the requirement. Ascertain possible scenarios of system abuse and evaluate users with hacker’s mindset.
* **Step 3:** After reading the Requirements Document and doing your due Analysis, list out different test scenarios that verify each feature of the software.
* **Step 4:** Once you have listed all possible Test Scenarios, a[Traceability Matrix](https://www.guru99.com/traceability-matrix.html)is created to verify that each & every requirement has a corresponding Test Scenario
* **Step 5:**The scenarios created are reviewed by your supervisor. Later, they are also reviewed by other Stakeholders in the project.

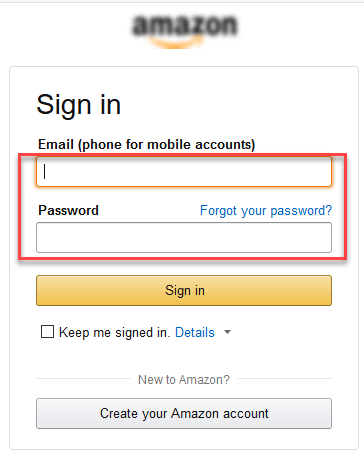
**Tips to Create Test Scenarios**

* Each Test Scenario should be tied to a minimum of one Requirement or User Story as per the Project Methodology.
* Before creating a Test Scenario that verifies multiple Requirements at once, ensure you have a Test Scenario that checks that requirement in isolation.
* Avoid creating overly complicated Test Scenarios spanning multiple Requirements.
* The number of scenarios may be large, and it is expensive to run them all. Based on customer priorities only run selected Test Scenarios

**Example 1: Test Scenario for eCommerce Application**

For an eCommerce Application, a few test scenarios would be

**Test Scenario 1:**Check the Login Functionality



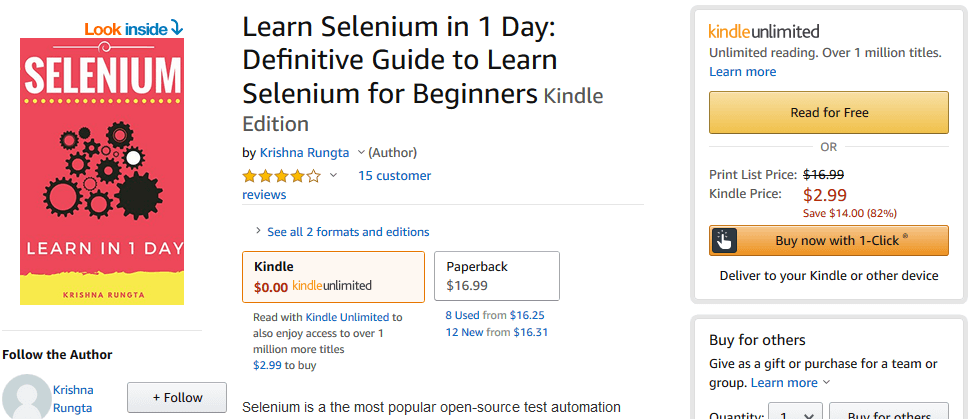
In order to help you understand the difference Test Scenario and Test Cases, specific test cases for this Test Scenario would be

1. Check system behavior when valid email id and password is entered.
2. Check system behavior when *invalid* email id and *valid* password is entered.
3. Check system behavior when *valid* email id and *invalid*password is entered.
4. Check system behavior when *invalid* email id and *invalid*password is entered.
5. Check system behavior when email id and password are left blank and Sign in entered.
6. Check Forgot your password is working as expected
7. Check system behavior when valid/invalid phone number and password is entered.
8. Check system behavior when “Keep me signed” is checked

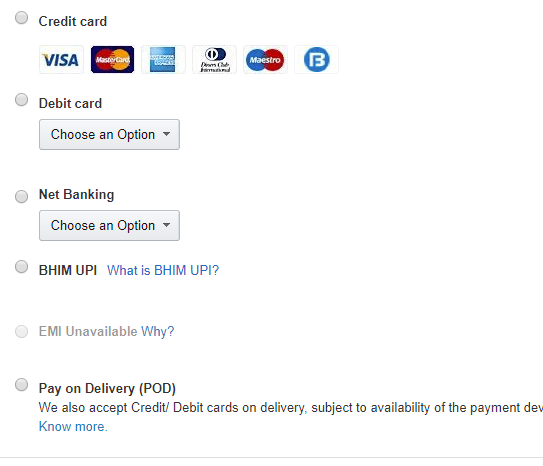
**Test Scenario 2:**Check the Search Functionality

test scenario

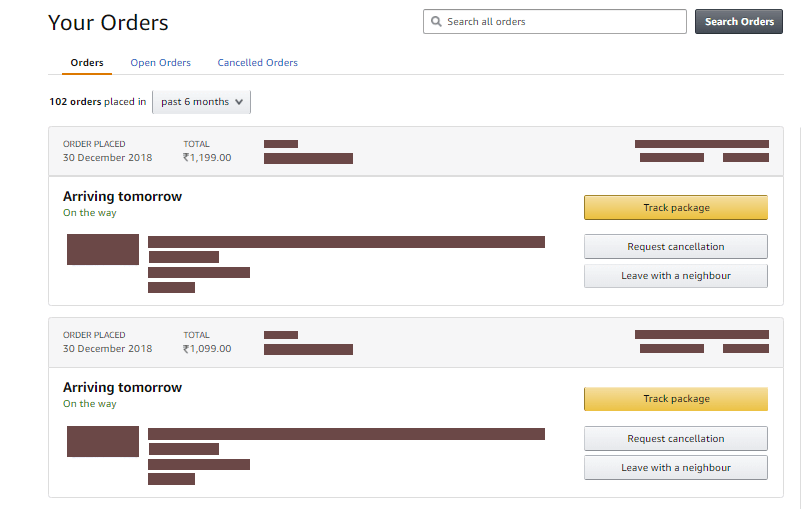
**Test Scenario 3:**Check the Product Description Page



**Test Scenario 4:**Check the Payments Functionality



**Test Scenario 5:**Check the Order History



Apart from these 5 scenarios here is the list of all other scenarios

* Check Home Page behavior for returning customers
* Check Category/Product Pages
* Check Customer Service/Contact Pages
* Check Daily Deals pages

**Example 2:  Test Scenarios for a Banking Site**

**Test Scenario 1**: Check the Login and Authentication Functionality

**Test Scenario 2**: Check Money Transfer can be done

**Test Scenario 3**: Check Account Statement can be viewed

**Test Scenario 4**: Check Fixed Deposit/Recurring Deposit can be created

Test Scenario Template

|  |  |  |  |
| --- | --- | --- | --- |
| Test Scenario # | Requirement ID | Test Scenario Description | Test Cases |
| 1 | S1.1 | Check the Login Functionality | 1. Check system behavior when valid email id and password is entered.  2. Check system behavior when invalid email id and valid password is entered.  3. Check system behavior when valid email id and invalid password is entered.  4. Check system behavior when invalid email id and invalid password is entered.  5. Check system behavior when email id and password are left blank and Sign in entered.  6. Check Forgot your password is working as expected  7. Check system behavior when valid/invalid phone number and password is entered.  8. Check system behavior when "Keep me signed" is checked |
| 2 | S1.2 | Check the Search Functionality |  |
| 3 | S1.3 | Check the Product Description Page |  |
| 4 | S1.4 | Check the Payments Functionality |  |
| 5 | S1.5 | Check the Order History |  |
| 6 | S1.6 | Check Home Page behavior for returning customers |  |
| 7 | S1.7 | Check Category/Product Pages |  |

**How to Write Test Cases: Sample Template with Examples?**

**What is a Test Case?**

A Test Case is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, post condition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

[**Test Scenario Vs Test Case**](https://www.guru99.com/test-case-vs-test-scenario.html)

Test scenarios are rather vague and cover a wide range of possibilities. Testing is all about being very specific.

For a [Test Scenario](https://www.guru99.com/test-scenario.html): Check Login Functionality there many possible test cases are:

* Test Case 1: Check results on entering valid User Id & Password
* Test Case 2: Check results on entering Invalid User ID & Password
* Test Case 3: Check response when a User ID is Empty & Login Button is pressed, and many more