# Introduction to Manual Testing and QA Processes

**What is SDLC?**
SDLC is referred to as the Software Development Life Cycle.
It is the process of developing software through business needs/requirements.

**What is SDLC used for?**
SDLC is useful in developing the products or customers which ensures the high quality and efficiency.

**Phases of Software Development Life Cycle?**
Phases of SDLC: The entire SDLC process divided into the following stages:
i) Phase 1: Planning and Requirement Analysis
ii) Phase 2: Defining Requirements and designing architecture
iii) Phase 3: Building or Developing the Product
iv) Phase 4: Testing the Product
v) Phase 5: Deployment and Maintenance

**What are the different types of SDLC methodologies?**
- Waterfall methodology.
- Agile methodology.
- Incremental Model.
- V-shaped model.
- Kanban.

## Agile:

i) In Agile is an incremental methodology where all the phases of SDLC occurs at the same time.
ii) In agile the cycle is broken down into smaller sprints each lasting 2 to 3 weeks.
iii) In agile preferred discussion and continuous improvement.
iv) Agile allows changes in project development requirement.

Sprints are core components of the agile methodology.
Sprint is the basic unit of development in Scrum which progresses via iterations called Sprint.
It lasts from two to three weeks.

## What is Scrum?

Scrum is a software development process framework for project management based on Agile Methodology.
Scrum uses self-organizing cross-functional teams to incrementally develop products in short Cycles.

**What are the roles in Scrum?**
The roles in Scrum are :
i) Product Owner
ii) Development Team
iii) Scrum Master
iv) End Users
14) What is a Sprint?

# Process on How QA Works

### Describe Scrum Development Process? Or What are the meetings you attend in your Agile methodology?

**Backlog grooming:** As an on-going activity, the product owner and the scrum team should be actively reviewing their backlog and ensuring the work is appropriately prioritized and contains clear information. This allows each piece of work to be more easily planned into a sprint as it can be more accurately estimated. As a tester I actively try to be involved in this process as it is my first opportunity to assess the requirements and the information provided. It also allows me a chance to gather information required for testing, which allows me to provide more reliable estimates.
The objective is to be in a position where for any piece of work presented in planning you know exactly what the work requires. You should then have a good idea of what you will test and therefore provide reliable estimates. If this is not the case, then the work cannot be effectively planned into the sprint.
**Sprint Planning meeting**, **Daily Scrum Meeting**, **Review Meeting**, **Sprint Retrospective**.
**A SPRINT PLANNING MEETING** is the starting point of Sprint. It is the meeting where the entire scrum team gathers, the SCRUM Master selects a user story based on the priority from the product backlog and the team brainstorms on it. Based on the discussion, the scrum team decides the complexity of the story and sizes it as per the Fibonacci series. The team identifies the tasks along with the efforts (in hours) which would be done to complete the implementation of the user story.

On each day of the sprint, all team members attend a **DAILY SCRUM MEETING** for 15 to 20 minutes to provide updates on the workday before, the task they will perform today and highlight any obstacles.

At the end of every sprint cycle, the SCRUM team meets again and demonstrates the implemented user stories to the product owner. The product owner may cross verify the stories as per its acceptance criteria. It’s again the responsibility of the Scrum master to preside over this **REVIEW MEETING.**

**SPRINT RETROSPECTIVE** is conducted at the end of each sprint with the entire team including Scrum Master and to discuss what went well, what did not go well and what steps should be taken to improve the process.

## What is a Defect/Bug?

Defect is the unusual behavior or problem in the software program.
It’s a condition when a product does not meet the requirement or end-user expectation.

## What are the few types of defects?

- Arithmetic,
- Logical,
- Syntax
- Functional defect, non-functional defect, Ui defects.

## What should you do after finding a defect?

**RECREATE THE DEFECT:** Once you find a defect, we must try to recreate (meaning that we should be able to reproduce it).
**ATTACH THE SCREEN SHOT** (SUPPORTING DOCUMENT): Once we confirm that it is a defect, and then it is a good idea to attach supporting documents when we log (write) a defect. For example,
screen shot, requirement document etc.
**LOG THE DEFECT**: Now, the next step is, we need to log it. Depending on the company what kind of tools they are using. In some cases, JIRA, Excel sheet is used log defects.

### Explain Bug Life Cycle?

When the bug is logged it gets **NEW** status
When the bug is accepted it gets **OPEN** status
When the bug is assigned to developer it gets **ASSIGNED** status
When the bug is being fixed it gets “**In Progress Status**”
When the bug is fixed it gets “**DEV COMPLETE**” or “**FIXED**” status
When the bug is tested and passed it gets “**VERIFIED**” status
If a decision is made to fix the bug in later release it gets “**DEFERRED**” status
And also **NOT A BUG**, **REOPEN** and **REJECTED**.

## 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 a 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 a 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 happen 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 are within a paragraph or in the report (Not on cover page, heading, title).

**What is STLC?**
STLC stands for Software Testing Life Cycle which performed by the testing team to ensure the quality of the software.

**Different stages of STLC?**
**Requirement Analysis** – it should be clear, consistent, and testable.
**Test Planning** – developing the test cases and managing them using tools.
**Designing/Development** – developing the test cases based on the requirement.
**Environment Setup** – when the integrated environment is ready to validate the product/test case.

**Execution** – we execute the case in real time and try finding bugs.
**Closure** – once testing is completed. Matrix reports are ready to publish/share with the team.

**Difference between SDLC vs STLC?**

## SDLC :

i) **Concept** – Business Analyst gathers requirements. Development team analyzes the requirements.
ii) **Requirement** – the development team starts analyzing from the architecture and the design perspective.
iii) **Design Stage** – The architecture of SDLC helps you develop a high-level and low-level design of the software based on the requirements.
iv) **Development Stag**e – Development team starts developing the software. Integrate with different systems. Once all integration is done, a ready to test software or product is provided.
v) **Testing Stage** – The actual testing is carried out in this phase. It includes unit testing, integration testing, system testing, defect retesting, regression testing, etc.
vi) **Implementation & Maintenance** - Once sign-off is received from various testing team, application is deployed in a prod environment for real end users.

**STLC**

i) **Concept** – no active involvement but can participate in meetings.
ii) **Requirement** – read requirement document try to understand as requirement: clear, consistency and testability.

iii) **Design Stage** – read design document, writing test cases based on design prototype of application.
iv) **Development Stage** – black box tester not active but white box tester does unit testing.
v) **Testing Stage** – all testing needs to be done in this stage with managing the Defect Life Cycle.
vi) **Implementation & Maintenance** – do regression test if any update version or defect fix.

**What is verification and validation?**

**Verification:**

i) Verification typically involves reviews and meetings to evaluate documents, plans, code, and specifications.
ii) This can be done with checklists, issues list, walkthroughs, and inspection meetings.
iii) It answers the question, Am I building a product, right?

**Validation:**

i) Validation typically involves actual testing and takes place after verifications are completed.
ii) This can be done by doing testing such as white box, Gray box & Black box testing.
iii) It answers the question, Am I building the right product?

**Difference between Black-Box and White Box Testing?**

**BLACK BOX TESTING**: Black Box Testing is a software testing method where testing is possible
without the internal knowledge of code.
i) Functionality Testing is a form of Black box testing.

**WHITE BOX TESTING**: White Box Testing is a software testing method where testing is done with
the internal knowledge of code.
i) Unit testing is a form of white box testing.

**What is Gray box testing?**
It's a combination of white and black box testing.
The aim is to search for defects and the tester knows partially internal structure.

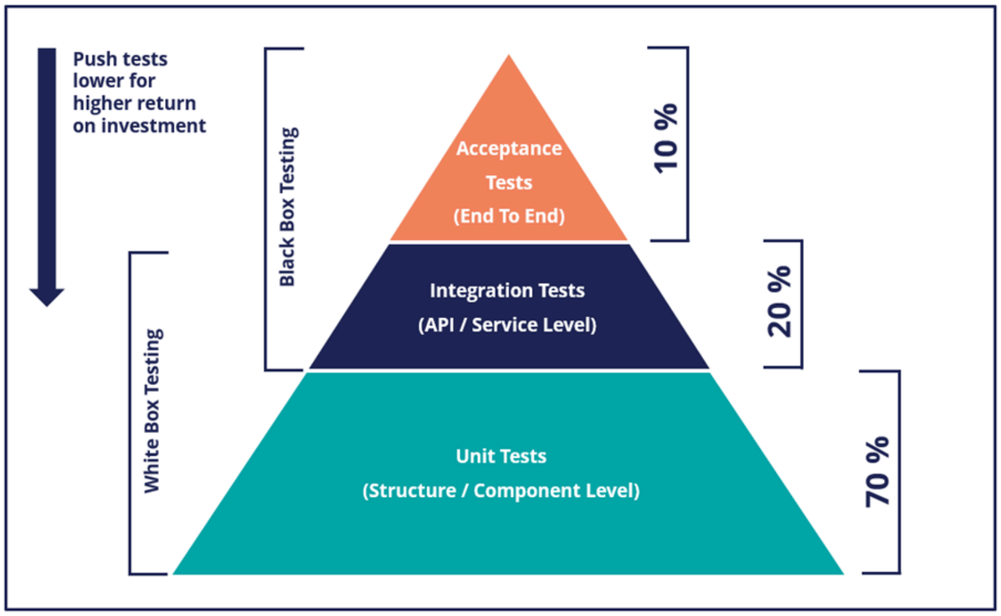
**Level Of TESTING?**

**UNIT TESTING**
**INTEGRATION TESTING**
**SYSTEM TESTING**
**UAT (USER ACCEPTANCE TESTING)**
**FUNCTIONAL TESTING**
**NON-FUNCTIONAL TESTING**
**DATABASE TESTING**
**PERFORMANCE TESTING**
**USABILITY TESTING**
**COMPATIBILITY TESTING**
**USER INTERFACE TESTING**
**END-TO-END TESTING**
**REGRESSION TESTING**

**How do we do regression testing? Regression Testing Manual or Automation?**
Regression tests are generally extremely tedious and time-consuming. We do regression testing after every deployment, so it would make life easy to automate test cases instead of running manually on each and every time.

Example : If we have thousands of test cases, it's better to create
automation test scripts for the test cases which we do on every build(i.e., regression testing). Automated regression test is the best practice and it is the choice of organizations to save a lot of time and to run nightly builds.

**What is a Test Pyramid?**
Testing Pyramid is a framework that can help both developers and QAs create high-quality software. It reduces the time required for developers to identify if a change they introduced breaks the code. It can also be helpful in building a more reliable test suite.
Testing Pyramid had 3 level of testing:
i) Unit Testing (using Junit or TestNG Assert,assertTrue, Assert.assertEqual etc...)
ii) Integration Testing (API testing manually with postman or rest assured with automation testing)
iii) End to end Testing (UI level testing using multiple scenarios cucumber feature file, StepDefinition, Pagefactory with Selenium and Java)


Push tests lower for higher return on investment
Black Box Testing
White Box Testing
Acceptance Tests (End To End)
Integration Tests (API / Service Level)
Unit Tests (Structure / Component Level)
10 %
20 %
70 %

**What is a Test Case?**
Test Case is a detailed document that describes the step by step process to execute a test.

- Detail steps to test specific function of an application

- Expected and actual result
- Pass or fail result
- Description for each step
- Information on person executing the test

**What do you include in a test case?**
Test cases should include Test case ID, Test case title, Description, Test steps, expected result and Actual result (once tested).

## Test Case Design Techniques in Manual Testing

### 1. Equivalence Partitioning (EP)

### 2. Boundary Value Analysis (BVA)

### 3. Error Guessing

## 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."

## Key Considerations for Test Data Preparation:

- **Data Quality:** Ensure that the data is accurate, consistent, and representative of real-world scenarios.
- **Data Security:** Protect sensitive data and comply with privacy regulations.
- **Data Volume:** Generate sufficient data to test the application's performance and scalability.
- **Data Variety:** Include a variety of data types and values to cover different test cases.
- **Data Reusability:** Design test data to be reusable across different test scenarios.

## Test data commonly include the following types

**No data:** Verify the system returns a proper response when the field is left empty

**Null Data:** Verify the system returns a proper response when the field value is passed as "NULL"

**Default data:** Verify the system's behavior on default data

**Valid test data:** Also known as positive testing. It is necessary to verify the system is working as expected with valid data.

**Invalid test data:** Proper response or error messages in the case for invalid data

**Boundary test data:** Prepare test cases for upper and lower boundary conditions.

**Equivalence Partition Data Set:** Test data qualifying the equivalence partitions.

## Test data creation techniques

Some of the ways to generate test data are below:

### 1. Manual Test Data Generation

- Adding data from the admin panel
- Creating data using POST method
- Using Front End of the application to create data e.g making a dummy order from the FE (website or app)
- Data mocking

Pros:

- a) This technique does not require any additional resources.
- b) The test team is not dependent on any other team and can create the data using their own skills and judgments.

Cons:

- a) The technique is time-taking especially if you need a large data set.
- b) The person entering the data must have domain knowledge.

### 2. Back-end data injection

This is done by inserting data into the database using DB queries. CURD operations can also be done using this technique.

Pros:

- a) A huge volume of data can be quickly available
- b) Quickly inject data into the system

Cons:

- a) It requires expert knowledge of the data flow in the DB
- b) Risk of database and application getting corrupted is high. Therefore, it's important to have a proper database backup
- c) Data insertion into complex databases systems may require additional help from database administrator

### 3. Importing Data

- Importing data from production to testing environment
- Importing subset of data from production to testing environment
- Importing data from one testing environment to another

Pros:

- a) Testers get to test data similar or closely similar data as in Production environment
- b) Can get a huge volume of data in less time as compared to manually entering the data

Cons:

- a) It might need additional technical expertise to abstract data to be dumped into test environments
- b) Can be time taking depending upon the volume of the data

- c) If more than one teams are using the same environment, there might be a conflict for the downtime
- d) Risk of getting the environment not working properly after the data import

## 4. Automated Test Data Generation

- This is done with the help of data generation tools which can accelerate the process for generating the high volume of data. One of the common tools that are used in this technique is selenium and web services APIs.

Pros:

- a) The main advantage of this approach is its better speed
- b) Once the process has been automated it can be performed without the need for human intervention
- c) If one requires thousands of data, this technique can be scheduled to run during nonworking hours to save time

Cons:

- a) It comes at a higher cost than manual test data generation , as in order to automate the data generation process, the person needs to have requisite skill and domain knowledge as well
- b) Automating the process itself can be a time taking process and requires regular updates.

## 5. Third-Party Tools

Third-party tools are a great way to create data. These tools can insert similar or real-time data into the system which makes diverse data available in high volume for the testers.

Pros:

- a) Easily available in the market
- b) It offers accuracy of the data
- c) These tools don't require you to have domain knowledge and expertise in the tool.

Cons:

- a) They are quite costly
- b) Their limited use to a specific type of system, application, or data.

# Test Plan

Test plan is one of the documents in test deliverables. Like other test deliverables, the test plan document is also shared with the stakeholders.

**Note : Always keep the test plan short and simple to understand, and keep the test plan up-to-date.**

## Who will Prepare a Test Plan Template?

Usually, Test Lead prepares Test Plan and Testers involved in the process of preparing test plan document

## Sections of Test Plan Template:

Following are the sections of test plan document :

1. **Test Plan Identifier**
2. **References**
3. **Introduction**
4. **Test Items**
5. **Features To Be Tested**
6. **Features Not To Be Tested**
7. **Approach**
8. **Pass/Fail Criteria**
9. **Suspension Criteria**
10. **Test Deliverables**
11. **Testing Tasks**
12. **Environmental Needs**
13. **Responsibilities**
14. **Staffing and Training Needs**
15. **Schedule**

16. **Risks and Contingencies**
17. **Approvals**

## Test Strategy

Test Strategy is a high level document (static document) and usually developed by project manager.This is also one of the important documents in test deliverables.

### Sections of test strategy document:

1. **Scope and overview**
2. **Test Approach**
3. **Testing tools**
4. **Industry standards to follow**
5. **Test deliverables**
6. **Testing metrics**
7. **Requirement Traceability Matrix**
8. **Risk and mitigation**
9. **Reporting tool**
10. **Test summary**

## What is a Test Scenario?

- Test Scenario gives the idea of what we have to test. Test Scenario is like a high-level test case.

- Test Scenario answers “What to be tested”.
- Test scenarios are high level actions.
- Test scenarios are written by Test Leads, Business Analysts, and Testers.
- Purpose of test scenario is to test end to end functionality of a software application.

-

- Assume that we need to test the functionality of a login page of a Gmail application. Test scenario for the Gmail login page functionality as follows:
- Test Scenario Example: Verify the login functionality.

## What is a Test Case?

Test cases are the set of positive and negative executable steps of a test scenario which has a set of pre-conditions, test data, expected result, post-conditions and actual results.

Test Case answers “How to be tested”.
Test cases are low level actions.
Test cases are written by Testers.
Purpose of test case is to validate the test scenario by executing a set of steps

Assume that we need to test the functionality of a login page of a Gmail application. Test cases for the above login page functionality as follows:
Test Case Examples:

- Test Case 1: Enter valid User Name and valid Password
- Test Case 2: Enter valid User Name and invalid Password
- Test Case 3: Enter invalid User Name and valid Password
- Test Case 4: Enter invalid User Name and invalid Password

**Note** : It's a best practice to write Test Scenarios and then move on to Test Cases. Even though it's a best practice, in today’s Agile era, most of the companies prefer Test Scenarios. Test cases are being replaced with test scenarios in the agile era to save time.

| Phase | Manual team | Automation team |
|---|---|---|
| Test Design | Test case designing | Automation scripting |
| Test Execution | Manual execution | Automation Run |
| Test Closure | Reporting, Documentation | Reporting, Documentation |