**Test cases** that can be **automated:**

Note: The **point of automating test cases** is to **expand** the **test coverage** and **helps testers can test more in less time**.

* Tests used repeatedly.
* Tests involved a lot of data entry / more data combinations.
* Tests clearly pass or fail.
* Tests use consistent UI and regular controls.
* Tests are only to do what they a meant do – not checking anything else.

**Test cases** that cannot be **automated:**

* Exploratory testing.
* UI/UX testing.
* Test cases that are not testing a clear function.
* Test where elements are of dynamic in nature.

**Which Test Cases should not be automated?**

* Extreme Test Cases which may not be executed more than once. If the Tests are Passed.
* UI Test Cases for which UI is not stable.
* Basic validation testing which can be executed quickly by manual testing [Eg. Blank username and passwords, providing special characters]
* Look and feel Test cases which are visual part of the application.
* Application which are very stable and don’t have much changes in future

Explain about how one tester can decide that provided **product** are **actually ready to move in the live environment?**

* This is one of the critical decisions, so it never been taken by the single person, higher management is periodically involved in that. Management test mainly ensure by validating below for ensuring product delivery are bugless:
* Validating bug reports provided by the tester. How bug got resolved and retesting done by the tester or not.
* Validating all the test cases written by the tester for that specific functionality, documentation, and confirmation taken from the tester on the same.
* Run automate test cases for ensuring new functionalities does not break any existing functionality.
* Sometimes validating test coverage report, which ensures all the developing component has been covered by test cases written.

If someone needs one specific format of bug reports from a tester, then what will be the best way or approach can take by the tester for providing the same?

One bug report normally contains below:

* Bug Summary
* Reproduce steps
* Expected behaviour and current behaviour of one specific bug.

Explain in detail about different kinds of testing called **Alpha** and **Beta**?

* Alpha testing done by the tester identified bugs before moving the product to live environment or to the end user.
* The beta bug is normally identified by the end user who is the actual users of the product or application.

What is **Risk-Based Testing**?

* Risk-Based testing is defined as the **functionalities of a product** are **tested based** on the **priority of the deliverables**.
* Risk-Based testing includes testing of **crucial features** of a **product** which will have a **business impact** and the **probability** of the **failure** of those **features is very high**.
* The **priority for all functionalities** of a **product** is **set based** on the **business requirement** then the high priority functionalities will be tested first then medium and then low priority functionalities.
* Risk-Bases testing will be performed when there is **no sufficient time** **to test all** the **functionalities** of a product.

Describe what happens **behind the scenes** between the time when you enter an URL in the address bar and hit enter and the time when the page finishes loading on your screen.

When we type URL into the browser,

* Browser checks cache, if the requested object is in cache and is fresh goes to step 9.
* Browser ask OS or browser for server’s IP address (stored in cache).
* OS makes a Domain Name Server (DNS) lookup and replies the IP address to browser.
* Browser opens a TCP connection to server.
* Browser sends the HTTP request through TCP connection.
* Browser receives HTTP response and may close the TCP connection, or reuse it for another request.
* Browser checks if the response is a redirect or a conditional response (3XX result status codes, authorization request (401), error (4XX and 5XX), etc these are handled differently from normal responses(2XX).
* If cacheable, response is stored in cache.
* Browser decodes response.
* Browser determines what to do with response (e.g. is it a HTML page, is it an image, is it a sound clip?).
* The browser then uses HTML parser to re-create document structure which is later presented to the user on screen.

**Epic**: Large stories with distinct start and end multiple sprints contains stories, bugs and tasks.

**Stories**: Also called as User Stories to ensure focus on users.

**Bugs**: are defects or issues that should be prioritized against new items.

**What Is Software Testing?**

Software testing is the process of checking that your product performs as expected. It’s a critical step in software engineering.

You can test software to verify many things. This includes whether:

* Functionality works as expected.
* Requirements are fulfilled.
* The software runs without producing bugs.

**Why Is Software Testing Important?**

Software testing is important for producing quality software — without defects.

* Testing Software improves performance.
* It increases customer satisfaction. It helps you avoid failures — which minimizes the risk of customers finding errors in the software.
* It improves the long-term success for your software.
* Testing does take considerable time and resources. But testing your application early on can help you save on costs in the long-term. After all, defects cost less when fixed earlier in the development process.
* Every development team needs to have a software testing process. But testing methods may vary. The needs for Agile testing teams are different from teams with serious compliance requirements.

That’s why it’s important to test the right things first.

**Software Testing Basics**: Software testing has two purposes. **Verification** and **Validation**.

* **Verification** confirms that each **function works properly**.
* **Validation** confirms that the **software fulfils requirements**.

Both verification and validation are important. And every type of software test you run should either verify or validate the software.

**7 Software Testing Principles**:

There are seven software testing principles used by testers everywhere.

* Testing reveals bugs - Testing shows presence of bugs.
* You can’t test everything - Exhaustive testing is not possible,
* Testing early reduces the cost of bugs - Early testing
* Most reported defects come from the same functionality - Defect clustering.
* Review and update tests regularly to avoid the pesticide paradox - Pesticide paradox.
* Different types of software require different types of testing- Testing is context dependent.
* Error-free doesn’t mean ready-to-ship - Absence of error is fallacy.

**What will happen if you don’t have time for testing?**

We do ad-hoc testing, testing the application randomly without any requirements, planning and documentation, is also called Monkey testing.

**The Software Testing Lifecycle (STLC):**

The software testing lifecycle (STLC) is the sequence of testing activities throughout development.

**How the typical software testing lifecycle works.**

**1. Requirements Review:**

Every test should come from a requirement. And every requirement should have a test.

That’s why it’s important that development teams do thorough requirements reviews. Stakeholders — including developers, testers, and managers — should be on the same page about what you’re going to build.

After all, it’s much easier to know what to test when you know what’s being built — and why.

**2. Test Planning:**

Preparing a test plan will help you align testing efforts with requirements.

Your test plan should include:

What you’re going to test.

How you’ll test it.

Who will do the testing.

Of course, test plans may vary in depth based on your team’s needs.

**3. Test Cases:**

Once you have a test plan, you’ll need to create test cases.

A test case should detail how you’re going to test a particular requirement. It should also include a step-by-step process for running the test. Your test case may also indicate what type of test you’ll be running (e.g., manual vs. automated testing).

Test cases should also be prioritized, so you test the most important things first. And individual test cases may be part of a test suite.

**4. Test Environment Setup**

Your test environment determines the software and hardware conditions that will be used for your test. Depending on your team, this task may be done by developers or it may be done by testers. You can even set up a test environment as you create test cases.

**5. Test Runs:**

When your test environment is ready, you can start running your tests.

Tests will either pass or fail. If a test fails, bugs found in the test should be documented (and connected to the requirement that was tested). You should run the test again — as changes are made to your software — to verify that the bug has been fixed.

Creating a test matrix — or a requirement traceability matrix — will help you keep track of the status of testing.

**6. Test Reporting**

You may need to report on test results to your team — or to other stakeholders within your company.

Here are some common testing metrics to report on:

* Test velocity
* Test status
* Test results
* Overall test summary
* **Severity in defect** defines the **impact** that a given defect on the application.
* **Priority in defect** defines the order in which it should be resolved.

**Tell me how you decide priority and severity?**

* It depends upon the defect which is found and at what stage it was found.
* If the defect has **impact on application crashing** or **unable to test further**. It will be reported as critical and priority will be high.
* For eg:- if a application crashes when a remote link is clicked, if the clicking a remote is very rare by the user but impact of the application crashing is severe, then the severity is high but priority is low.

**Severity types**

**Critical** [if there is no acceptable alternative method to achieve required results, termination]

**Major** [ if there is alternative available to achieve required results, termination]

**Moderate** [ if the defect does not result in termination but s/w produces incorrect results]

**Minor** [ if defect does not result in termination and desired results achieved easily]

**Cosmetic** [ look and feel of application]

**Priority types**

**High** [ need to fix immediately]

**Medium** [ not to fix immediately to but if time permits]

**Low** [ can be fixed in next phase/sprint]

**What is the cost of the defect?**

The cost of the defect is measured by the impact of the defects and when the defect was found. If the defect is found in requirement gathering it is cheap to fix it. The correction of specifications can be done, and it can be resumed. If the defect is found in design, then it can be corrected and resumed. If defect is found in User acceptance, then the cost of fixing the bug will expensive.

**What is Smoke test and Sanity test?**

Smoke test is to confirm whether the build is stable or not. It confirms to test further or not.

Sanity test is similar to smoke test but is performed on production environment before the end users start using the product.

**What is Regression testing and Retesting?**

Regression testing is carried out to ensure that the existing functionality is working fine and there are no side effects of any changes or enhancement done to the application.

Retesting is carried out to ensure that the particular defect has been fixed and its functionality working as expected.

**Quality Assurance:** is focusing on preventing defects [ Verification, Product Right, start of development process, Low level activity]

**Quality Control:** is focusing on identifying the defects [ Validation, Right Product, end of the development process, High level activity].

**What is automation Testing?**

Is a process of automating the existing manual process to test application using an automation tools to find the defects, in this process executing the test scripts and generating results are performed automatically by automation tools, the most popular automation tools are HP QTP, Selenium WebDriver, etc.

Benefits of automation testing

* Saves time, money and faster execution
* Code reusability
* Easy reporting
* Enable parallel execution -combination of different o/s and browsers
* More reliable
* Mostly used for Regression, Smoke and Sanity testing
* Minimal manual intervention
* Maximum coverage

**What is Test Plan?**

Test plan is detailed document which consist of what to test, when to test, how to test.

Test plan ID – unique number assigned.

Test plan description – use of the test plan/purpose if the test plan.

Test items – details of functionalities/features [how many to features].

Features to be tested.

Features not to be tested.

Entry Criteria.

Exit Criteria.

Suspension Criteria.

Roles and Responsibilities.

Staff and training needs

Test Approach

Risk and Assumptions

**What is Test strategy?** [ prepared by Project Manager]

Is a High level document which defines the **approach for the software testing**, It is basically derived from **BRS**. It is static document which sets the standards for testing so not updated often

.

**What are Test Cases?**

A test case is a set of actions executed to verify a particular feature/functionality of software application.

Once requirements are finalized/freezed, we will create Test cases.

**What type of test you have automated?**

Our main **focus** is to **automate test cases** to do **Regression** testing**, Smoke** testing **and Sanity** testing.

**What is the process of creating a test script?**

* **Understanding the requirement** document thoroughly. [ in absence of document, we could understand any point of reference we have in previous version of application or wireframe]
* after understanding the requirement we make a list of **what areas** **to be tested** in the s/w. The **outcome of this step** is **list of test scenarios**.
* once we have test scenarios, we concentrate on **how to test**. This phase involves writing detailed steps about a particular feature, test data and expected result.

Once these three steps are done, we are ready for testing.

**What is the importance of the Test Data and how do you prepare it?**

Test Data is data that is used to execute the tests on software application. The test data needs to be precise and exhaustive to uncover the defects.

We prepare Test Data from SRS and usually provided by the Developer.

**Testing Techniques [ Risk based testing]**

1. Boundary Value Analysis (BVA)

min-1 min+1 middle max-1 max+1 [ Test data]

1. Equivalence Class Portioning [ ECP]

Valid Invalid [Test data]

1. Error guessing [random]

**What will you do if you find the defect?**

Once the defect is found we log it as a new defect with all details.

*Details provided are:*

Tile.

Description.

Expected Result.

Actual Result.

Project details.

Environment details.

Steps to reproduce.

Type of defect – functional, Integration, database.

Priority of the defect.

Severity of the defect.

Attachments/screenshots.

Test Data

**Test Closure:**

Test Closure is a document that gives a summary of all the tests conducted during the software development life cycle, it also gives a detailed analysis of the bugs removed and errors found . In other words, Test Closure is a memo that is prepared prior to formally completing the testing process. This memo contains a report of test cases executed, type and number of defects found, the density of defects, etc.

**What is a dynamic testing technique? [ It is done during validation process. Software tested by executing on computer].**

* **Unit Testing**: A Unit testing is the smallest testable part of an application. It is done by Developers.
* **Integration Testing:** Testing the Integrations/interfaces between a component interaction to different parts of the system such as o/s, file system and h/w. done by Developers.
* **System Testing:** Testing the behavior of the whole system (or) application, is tested as defined by the scope of the development project. Referred as final test to verify that system is developed meets the specifications and the purpose.

System Testing involves both functional and non- functional requirement of testing.

* **User Acceptance Testing:** Requirements are met, this testing is done by user or customer.

