Chapter 1 Fundamentals of Software Testing

1.1 Introduction:

- **Software -** A Software is collection of programs and programs are set of instruction.
- **Testing -** Testing is a process in which we can check the software in specific manner.
- **Software Testing –** Software testing is a process in which we can check the system is working is properly or not as per user requirement.
- **Quality Assurance –** Quality Refers to the features or Characteristic of product which define it's ability to satisfy user requirement / specification.
- QA is the process of determining whether product meets customer / users Expectations or Requirements.

We have learned about the test activities performed at a high level to meet the defined testing objectives; let us now find out how these activities are mapped to the different phases of a test lifecycle.
As you can see below, these phases are:

1. Test Planning and Control
2. Test Analysis and Design
3. Test Implementation and Execution
4. Evaluating Exit Criteria and Reporting
5. Test Closure

It is important to note that while these phases are sequential, they are also iterative in nature. For example, during test execution, there may be a need to go back to test design to introduce more test cases or test data before the test execution process is resumed.
Alternatively, during exit criteria evaluation, it can be decided to execute some more tests before the application is considered fit for release. Hence, all phases interact and might transition from one to the other based on the needs of the project.

1.2 Basics of Software Testing:

What is Testing?

- Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.
-

Who does Testing?

- In most cases, the following professionals are involved in testing a system within their respective capacities −
- Software Tester
- Software Developer
- Project Leader/Manager
- End User
- 1. Finding defects which may get created by the programmer while developing the software.
- 2. Gaining confidence in and providing information about the level of quality.
- 3. To prevent defects.
- 4. To make sure that the end result meets the business and user requirements.
- 5. To gain the confidence of the customers by providing them a quality product.

- 6. **Identification of Bugs, and Errors:-** Once the developer finishes coding, the tester starts testing. During testing, QA validates each module under various conditions.
- 7. **Quality Product: -** The main aim of testing is to maintain the quality of the product. Also, testing has its own cycle and in each phase, all focus revolves around quality only.
- **8.Enhances Growth:-** A quality delivery increases the potential of a business. And we all know quality comes through testing only.

Importance of Software Testing

- 1.It Saves Money.
- 2.Maintain Security.
- 3.Product Quality.
- 4.Customer Satisfaction

1.3 Approaches to Testing:

Test Approach:- A test approach is the test strategy implementation of a project, defines how testing would be carried out. Test approach has **two techniques:**

- **1. Proactive -** An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the **build** is created.
  Proactive Testing enables you to do more effective testing in less time, while also providing the value that overcomes traditional user, manager, and developer resistance to testing. By continually refocusing on the highest risks, and applying special techniques that spot many ordinarily-overlooked risks, Proactive Testing makes sure the most important testing is done in available time. Moreover, by catching more defects earlier when they are easier to fix, and actually preventing many showstoppers and other errors, Proactive Testing also can cut developers time, effort, and aggravation. This interactive workshop shows powerful proven Proactive structured test planning and design techniques that produce value, not busywork, and enable productive use of automated test tools. To enhance learning, participants practice each key technique in a series of exercises with various aspects of a real case fact situation.

  2. Reactive - An approach in which the testing is not started until **after design and coding are completed.**

In Reactive approach, tests are designed after software development.
Before selecting a 'test approach', we should consider the following factors:

- Analyse the various risks associated with failure of the project - to the people,the environment, and the company.
- Analyse the expertise and efficiency of the people in the proposed tools and techniques.
- The nature of product and the business.

1.4. Testing During Development Life Cycle

- **What is SDLC?**

SDLC stands for **Software Development Life Cycle** and is also referred to as the Application Development life-cycle.

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.
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.

Here, are prime reasons why SDLC is important for developing a software system-
1. It offers a basis for **project planning, scheduling, and estimating.**
2. Provides a framework for a standard set of **activities** and **deliverables**.
3. It is a mechanism for **project tracking and control.**
4. Increases visibility of project planning to all involved stakeholders of the **development process**.
5. Increased and enhance **development speed.**
6. Improved **client relations**.
7. Helps you to **decrease** project risk..

- The entire SDLC process divided into the following SDLC steps:

Phase 1: Requirement collection and analysis
Phase 2: Feasibility study
Phase 3: Design
Phase 4: Coding
Phase 5: Testing
Phase 6: Installation/Deployment
Phase 7: Maintenance

Phase 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 recognition 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.
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. It includes everything which should be designed and developed during the **project life cycle**.

There are mainly five types of feasibilities checks:

1. Economic: Can we complete the project within the **budget or not?**

2.Legal: Can we handle this project as **cyber law** and other regulatory framework/compliances.

3. Operation feasibility: Can we create operations which is expected by **the client?**

4. Technical: Need to check whether the current **computer system** can support the software

5. Schedule: Decide that the project can be completed within **the given schedule or not.**

Phase 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.
-

Phase 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.

Phase 7: Maintenance:-

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

1. Bug fixing – bugs are reported because of some scenarios which are not tested at all

2. Upgrade – Upgrading the application to the newer versions of the Software

3. Enhancement – Adding some new features into the existing software

1.6 .Features of Testing

1.Test Management and Planning -
A good testing plan will have a plan and management system in place. Developers and tech teams need some way to systematically test parts and pieces of software. When you fail to test systematically, you increase the chances of missing out on key areas or losing time tracking the same sections of these tests over and over again.

2.Automated Testing -
The future of work and productivity depends heavily on automation. Using automation tools in testing can help eliminate hours wasted on repetitive tasks when testing and lessen a development team's workload significantly.
3.Quality Assurance Reporting and Insights -
Part of the QA testing tools list for automated software testing should include reporting.
4.Mobile Testing Capabilities -
More and more people are switching to mobile solutions to run several of their tech-based activities. Choosing right QA software testing tool that tests all devices of all sizes can help save time and resources.
5.Design and Experience Testing -
When testing an e-commerce website, design and user experience play a big factor in determining whether a site is ready for launch or not. This testing should include both functional and visual aspects of a website. In the past, website developers and designers had to test and check website pages one by one.
6.Solution Integrations -
A QA testing software tool should have native integration with CI/CD tools. Having integrations will help in ensuring test automation remains integral to the development process.

1.7.Misconceptions About Testing

1. **Testers are Breaking Product** - Misconceptions about *software testing*. "Testers break the software". The most common misconceptions about software for testing are testers **are breaking product and software**.
2. Testers Involve Only in **Post-Development**
3. Automation Testers **Don'**t Bother About Manual Testing
4. Test Leaders Don't **Test.**
5. Testing is All About **Writing Test Cases**
6. Testers are **Gatekeepers**
7. Anyone can Test
8 . Software Testing Slows Down
9. Software testers only worry about test cases.
10. Only **seasoned experts** can be involved in testing
11. No special skills are required for testing.

12. Software testing is nothing but software development
13. Testing Can Find All The Bugs
14. Testing Can Be Automated

1.8.Principles of Software Testing

1) Exhaustive testing is not possible

Yes! Exhaustive testing is not possible , because it test **all product at a time**. And also it check number of valid and invalid inputs. For ex. Modules(Pages)

2) Defect Clustering

Defect Clustering which states that a **small number of modules** contain most of the defects **detected.** By experience, you can identify such **risky modules**. But this approach has its own problems if the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.
For Ex-

Module	A	B	C	
Module Size	**20%**	**70%**	**10%**	□**Start first**
Defect Density	**10%**	**10%**	**80%**	□ **Defect density**

3) Pesticide Paradox

It is used to check the test cases as per requirement. To perform regression testing we use this principle.

4) Testing shows a presence of defects

The goal of software testing is to make the software fail. Software testing reduces the presence of defects. It is also one of the most important principle , It is used to show the defect at the time of execution Testing can reduce the number of defects but not remove all defects.

5) Absence of Error – Failure

It is used to build the defect free software as per user requirement. But if defect is present at that time we can say it is failure of tester.

6) Early Testing

Early Testing should start as **early as possible** in the Software Development Life Cycle. So that any defects in the requirements or design phase are **captured in early stages**. It is much cheaper **to fix a Defect** in the early stages of testing. **7) Testing is context dependent**

Testing is context dependent which basically means that the way you test an **e-commerce** site will be different from the way you test a **Banking application**. All the developed software’s are not identical. You might use a **different approach, methodologies, techniques, and types of testing depending** upon the application type.

1.9. Test Policy

- Test policy is a document described at the organization level and gives the organizational insight for the test activities.
- It is determined by the senior management of the organization and defines the test principles that the organization has adopt.
- The test policy is very high-level document and at the top of the test documentation structure.
- Organizations may prefer to publish their test policy in a sentence, as well as a separate document.

Also they may use this policy in both development and maintenance projects

1.10. Defect Classification

Defects are classified from the QA team perspective as **Priority** and from the development perspective as **Severity** (complexity of code to fix it). These are two major classifications that play an important role in the timeframe and the amount of work that goes in to fix defects.

What is Priority?

Priority is defined as the order in which the defects should be resolved.
Priority Listing:
A Priority can be categorized in the following ways −

- **Low** − This defect can be fixed after the critical ones are fixed.
- **Medium** − The defect should be resolved in the subsequent builds.
- **High** − The defect must be resolved immediately because the defect affects the application to a considerable extent and the relevant modules cannot be used until it is fixed.
- **Urgent** − The defect must be resolved immediately because the defect affects the application or the product severely and the product cannot be used until it has been fixed.
- **What is Severity?**

- Severity is defined as the impishness of defect on the application and complexity of code to fix it from development perspective. It is related to the development aspect of the product. Severity can be decided based on how bad/crucial is the defect for the system.

- **Example** − For flight operating website, defect in generating the ticket number against reservation is high severity and also high priority.

- Severity can be categorized in the following ways −

- **Critical /Severity 1** − Defect impacts most crucial functionality of Application and the QA team cannot continue with the validation of application under test without fixing it. For example, App/Product crash frequently.

- **Major / Severity 2** − Defect impacts a functional module; the QA team cannot test that particular module but continue with the validation of other modules. For example, flight reservation is not working.
- **Medium / Severity 3** − Defect has issue with single screen or related to a single function, but the system is still functioning. The defect here does not block any functionality. For example, Ticket# is a representation which does not follow proper alpha numeric characters like the first five characters and the last five as numeric.
- **Low / Severity 4** − It does not impact the functionality. It may be a cosmetic defect, UI inconsistency for a field or a suggestion to improve the end user experience from the UI side. For example, the background colour of the Submit button does not match with that of the Save button.

1.11 Defect:

Defect are defined as the deviation of the actual and expected result of system or software application.

Defects can also be defined as any deviation or irregularity from the specifications mentioned in the product functional specification document. Defects are caused by the developer in development phase of software. When a developer or programmer during the development phase makes some mistake then that turns into bugs that are called defects.
It is basically caused by the developers' mistakes.

- **Types of Defects:**
  Following are some of the basic types of defects in the software development:
- **Arithmetic Defects:**
  It include the defects made by the developer in some arithmetic expression or mistake in finding solution of such arithmetic expression.
- Code congestion may also lead to the arithmetic defects as programmer is unable to properly watch the written code.

- **Logical Defects:**
  Logical defects are mistakes done regarding the implementation of the code. When the programmer doesn't understand the problem clearly or thinks in a wrong way then such types of defects happen.
- **Syntax Defects:**
  Syntax defects means mistake in the writing style of the code. It also focuses on the small mistake made by developer while writing the code.

- **Multithreading Defects:**
  Multithreading means running or executing the multiple tasks at the same time.
- **Interface Defects:**
  Interface defects means the defects in the interaction of the software and the users. The system may suffer different kinds of the interface testing in the forms of the complicated interface, unclear interface or the platform based interface.
- **Performance Defects:**
  Performance defects are the defects when the system or the software application is unable to meet the desired and the expected results.

1.12. Error:

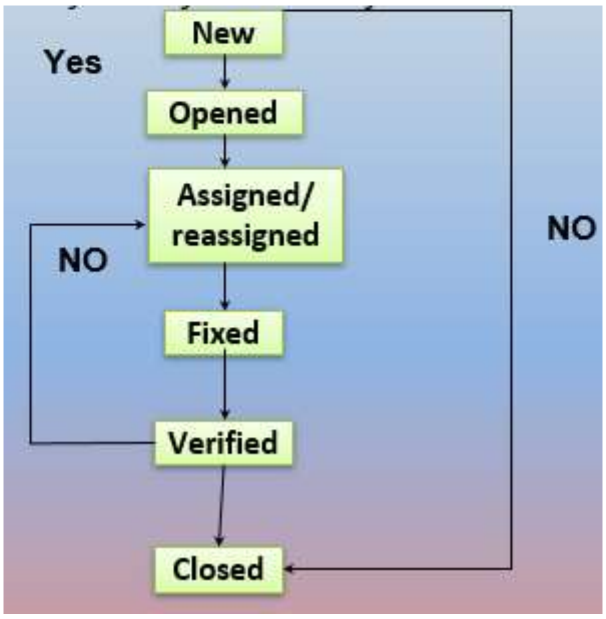
An error is a mistake, misconception, or misunderstanding on the part of a software developer. In the category of the developer, we include software engineers, programmers, analysts, and testers. For example, a developer may misunderstand a de-sign notation, or a programmer might type a variable name incorrectly – leads to an Error. It is the one that is generated because of the wrong login, loop or syntax. The error normally arises in software; it leads to a change in the functionality of the program.

Let us see the comparision of mistake, error and defect :

Mistake	**Error**	**Defect**
An issue identified while reviewing own documents or peer review may be termed "mistake".	An issue identified internally or in unit testing may be termed 'error'.	An issue identified while in black box testing or by customer is termed "defect"
Very low cost of finding mistakes and can be fixed immediately	Slightly more cost of finding an error and needs some time for fixing	Most costly and needs longer time for fixing defects.
Most of the time, problems and resolutions are not documented properly.	Sometimes, problems and resolutions are documented, but may not be used for process improvements	Problems and resolutions are officially documented and used for process improvements.

1.14. Defect Life Cycle:

When a software system undergoes verification/validation cycles, there are possibilities that defects can be found in the product and corresponding processes or work artifacts. The defects may be termed as issues, problems, mistakes, or bugs depending upon organization's way of naming it. Defects so found must be tracked in defect tracking system automatically to the maximum extent possible. Initially, it may not be very clear to the testers whether it is a defect or not.



New:
When a new defect is logged and posted for the first time. It is assigned a status as NEW.
Open:
Here, the developer starts the process of analyzing the defect and works on fixing it, if required.
Assigned/reassigned:
In this stage, a newly created defect is assigned to the development team to work on the defect. This is assigned by the project lead or the manager of the testing team to a developer.
Fixed:
When the developer finishes the task of fixing a defect by making the required changes then he can mark the status of the defect as "Fixed".
Verified:
If the tester does not find any issue in the defect after being assigned to the developer for retesting and he feels that if the defect has been fixed accurately then the status of the defect gets assigned to 'Verified'.
Closed:

When the defect does not exist any longer, then the tester changes the status of the defect to “Closed”.

1.15. Defect Management Process(Fixing and root cause of defect):

Defect management process must include the appraisal of a defect finding process, software development process and the actions initiated to close the defects and strengthen the product/processes associated with development, so that defects are not repeated again and again.
It typically includes correction, corrective actions and preventive actions.

- **Defect Naming**: Defect naming conventions are defined in a test plan. Defects must be named according to the problem's nature, effect on user, etc.
- **Defect Resolution:** Defect resolution includes defect fixing, retesting, resolving, and taking actions on the processes and methods which have produced the defect.
- **Deliverable Baselining**: Once the defect is fixed, retested and found to be closed, the product is created again. If this newly created product satisfies the acceptance criteria, it is baselined.
- **Process Improvement:** Defect is an instance of a class problem. All problems are due to failures in the processes involved in creating software. Defects give an opportunity to identify the problems with processes used and update them to suite the organizational requirements.
- **Management Reporting:** Management reporting is required to maintain information flow-to and from management about the defects and status of the software being developed, and also the improvement proposals for processes.
- It is required to get a 'buy-in' of the senior management for updating the processes if required.
- Senior management must be aware of the status of projects, and their needs, so that they can offer their support in continuous (continual) improvement.
- Test reports must cover SWOT (strength, weakness, opportunities for improvement, and Threats) analysis so that they may take a decision about rework and releases.
- The customer may or may not be a part of management report.

1.16.Developing Test Stratergy:

Test planning includes developing a strategy about how the test team will perform testing. Some key components of testing strategy are as follows.

- Test factors required in particular phase of development
- Test phase corresponding to development phase

Process of developing test strategy goes through the following stages.

Select and Rank Test Factors for the Given Application

A software may have some specific requirements from user's point of view. Test factors must be analysed and prioritised or ranked.

Identify System Development Phases and Related Test Factors

The critical success factors may have varying importance as per development life cycle phases. One needs to consider the importance of these factors as per the life cycle phase that one is going through. The test approach will change accordingly.

Identify Associated Risks with Each Selected Test Factor In Case if it is Not Achieved

Trade-offs may lead to few risks of development and testing the software. Customer must be involved in doing trade-offs of test factors and the possible risks of not selecting proper test factor.

The risks with probability and impact need to be used to arrive at the decision of trade-off.

Identify Phase in Which Risks of Not Meeting a Test Factor Need to Be Addressed

The risks may be tackled in different ways during development life cycle phases.

As the phase is over, one needs to assess the actual impact of the risks and effectiveness of devised countermeasures for the same.

1.17.Developing Test Methodologies:

Developing test tactics is the job of project-level test manager/test lead. Different Projects may need different tactics as per type of product/customer. Designing and defining of test methodology may take the following route.

- **Acquire and study test strategy as defined earlier:** Test strategy is developed by a test team familiar with business risks associated with software usage. Testing must address the critical success factors for the project and the risk involved in not achieving it.
- **Determine the type of development project being executed:** Development projects may be categorized differently by different organisation
- **Determine the type of software system being made:** Type of Software system defines how data processing will be performed by the software.
- **Identify tactical risks related to development:** Risks may be introduced in a software due to its nature, type of customer, type of developing organisation, development methodologies used and skills of teams. Risk may differ from project to project.
- **Determine when testing must occur during life cycle.**
- **Steps to develop customized test strategy.**