Software Testing

# Fundamentals of Testing

## Introduction to Software Testing

Software testing is a crucial step in the software development lifecycle (SDLC). It ensures that software functions as intended, meets quality standards, and is free from critical bugs.

## Software Testing

Process of finding or identifying defects in a software is called as software testing. In other words, it is verifying the functionality (behaviour) of the application(software) against requirements specification. It can also be termed as the execution of the code with the intention of finding defects.

Software testing is done

* To ensure customer’s business not affected because of bugs.
* To ensure that the developed software is a quality product.
* To ensure customer needs are met.

## Error

An error refers to a human mistake made during the development or design of software. It occurs when developers or designers misinterpret requirements or make logical mistakes while coding or designing.

## Bug

It refers to a flaw or error in the software code that causes it to produce incorrect or unintended results. It is identified by the test engineers while testing the application.

## Defect

A defect is a flaw in the software that occurs due to an error in the code, design, or requirements. It is identified end user when they are using the product, and the software does not perform as expected.

## Failure

A failure is an observable incorrect behaviour of the software when it is executed. It is the manifestation of one or more defects during the software’s runtime. Failures can occur in testing or in production.

## Causes of Software Defects

Software defects arise due to various reasons during the development, testing, and deployment phases.

* Software defects can arise from various causes, including **human errors**, such as coding mistakes, misunderstanding requirements, or skill gaps.
* **Inadequate requirement gathering**, involving incomplete, ambiguous, or frequently changing requirements.
* **Communication gaps** between teams or due to outdated documentation.
* **Design issues** like flawed architecture or improper design patterns.
* **Development process problems** such as lack of standards, insufficient testing, or unrealistic timelines.
* **Environmental factors**, like compatibility issues or resource limitations.
* **Tool or technology limitations**, including bugs in third-party tools or platform constraints.
* **Deficiencies in testing**, such as poor test design or insufficient coverage.
* **Deployment and maintenance issues**, like configuration errors or patch defects.

## Cost of Software Defects

The cost of software defects increases significantly the later they are identified in the development lifecycle. Defects caught in the requirements or design stages are cheaper to fix compared to those discovered in development, testing, or production, where costs can escalate due to debugging, rework, downtime, or reputational damage. Early detection, robust testing, and proactive strategies can minimize these costs and ensure higher software quality.

## What does Software Testing reveal?

Software testing reveals defects, verifies functionality, and ensures the software meets user and business requirements. It evaluates performance, security, usability, compatibility, reliability, and compliance. By uncovering issues early, testing ensures quality and reduces risks before deployment.

## Importance of Software Testing:

1. **Identifies and Fixes Defects**: Ensures software operates as intended by detecting and resolving issues early.
2. **Enhances Quality**: Improves functionality, performance, and user experience.
3. **Ensures Security**: Protects against vulnerabilities and potential data breaches.
4. **Builds User Confidence**: Delivers reliable and user-friendly software, boosting customer satisfaction.
5. **Saves Costs**: Reduces expenses by catching issues early in the development lifecycle.
6. **Ensures Compliance**: Meets industry standards and legal regulations.
7. **Prevents Failures**: Reduces the risk of costly downtimes or critical system errors in production.

**Testing** is the process of evaluating software to identify defects, ensure functionality, and verify that it meets user and business requirements. It plays a crucial role in maintaining quality by enhancing reliability, performance, and security. **Quality** ensures that the software delivers value, meets standards, and provides a seamless user experience, making testing essential for delivering high-quality products.

## Seven Testing Principles

**Testing Shows the Presence of Defects**

Testing identifies defects but cannot prove software is defect-free. It reduces the number of bugs but can never ensure the absence of all defects, even with extensive testing.

**Exhaustive Testing is Not Possible**

Testing every possible input and condition is impractical due to cost and effort constraints. Only select test cases are tested, assuming the software will handle others correctly.

**Early Testing**

Detecting defects early in the software lifecycle is cost-effective. Starting testing during the requirement analysis phase improves software quality and reduces expenses.

**Defect Clustering**

Most defects (80%) are often found in a small number (20%) of modules, following the Pareto Principle. Focusing testing on these areas can improve efficiency.

**Pesticide Paradox**

Reusing the same test cases repeatedly may fail to uncover new bugs. Test cases should be regularly reviewed and updated to remain effective.

**Testing is Context-Dependent**

Testing approaches vary based on the software being developed. For example, testing an e-commerce site differs significantly from testing a mobile application.

**Absence of Errors Fallacy**

Even if software is nearly bug-free, it is unusable if it doesn’t meet user requirements. Software must be both functional and aligned with customer needs.

## Attributes of a Good Tester

1. **Attention to Detail**: Notices even the smallest issues and inconsistencies in the software.
2. **Analytical Thinking**: Breaks down complex systems to understand how they work and identifies potential problem areas.
3. **Curiosity**: Eager to explore and test software beyond standard scenarios to uncover hidden defects.
4. **Strong Communication Skills**: Clearly documents and explains defects, test cases, and results to team members.
5. **Problem-Solving Ability**: Thinks critically to troubleshoot issues and find effective solutions.
6. **Technical Knowledge**: Understands software development concepts, tools, and testing methodologies.
7. **Team Collaboration**: Works effectively with developers, product managers, and other stakeholders.
8. **Adaptability**: Adjusts quickly to new tools, technologies, and changing requirements.
9. **Customer Focus**: Ensures software meets user needs and expectations.
10. **Persistence**: Stays determined to find defects, even in challenging scenarios.

## Psychology of Testing

1. **Tester and Developer Perspectives Differ**: Developers aim to create functioning software, while testers focus on finding flaws and ensuring quality.
2. **Testing is Destructive by Nature**: The goal is to intentionally break the software to identify defects and weaknesses.
3. **Critical Thinking is Essential**: Testers must question assumptions, requirements, and functionality to uncover hidden issues.
4. **Defects are Normal**: Finding bugs is expected and helps improve the software; it is not a personal attack on developers.
5. **Communication is Key**: Testers should report defects professionally and collaboratively to maintain a positive team dynamic.
6. **User-Centric Approach**: Testers think from the end-user’s perspective to ensure the software meets user needs and expectations.

## Code of Ethics for Testers

1. **Integrity and Honesty**: Always report findings truthfully, whether they are positive or negative, and avoid exaggerating issues.
2. **Confidentiality**: Protect sensitive information about the software, users, and company; never disclose details without authorization.
3. **Respect for Individuals**: Treat all team members, including developers, with respect, fostering a positive and collaborative environment.
4. **Accountability**: Take responsibility for your work, including test results, and be open to feedback.
5. **Fairness**: Testers should evaluate software objectively and ensure it meets both business requirements and user needs without bias.
6. **Continuous Improvement**: Stay updated with new testing techniques, tools, and industry practices to enhance your skills and contribute to team success.

## Limitations of Software Testing:

1. **Cannot Prove Software is Bug-Free:** Testing can identify defects but cannot guarantee that all bugs are found or that the software is entirely error-free.
2. **Limited Coverage:** Exhaustive testing of all possible scenarios is impractical due to time, cost, and resource constraints.
3. **Time Constraints:** Limited time for testing may result in incomplete test coverage, missing some defects.
4. **Dependence on Test Cases:** Testing is based on predefined test cases, and if they are not comprehensive or updated, new defects may be missed.
5. **Changing Requirements:** Evolving requirements during development can make it difficult to keep tests relevant and up to date.
6. **Human Error:** Testers can overlook defects or make mistakes during testing or reporting, which can impact the quality of testing.

## Component Testing

Testing every component thoroughly (rigorously) against requirement specifications is known as component testing.

**How to perform component testing?**

Start testing the application with the valid data. If the application works for valid data, only then start testing for invalid data. If the application is not working for 1 of the invalid values, then can continue testing for all the other invalid values and then submit the defect reports for invalid values.

Don’t do **over testing** on the application as the time and effort gets wasted. Also, **under testing** the application will not allow the test engineers to find proper bugs. Test the application with sufficient amount of test cases which will be **optimal**.

Do not assume or propose requirement. If there are any queries, talk to the person who knows the requirements very well and clarify the queries.

**Example:**

Consider the following example of a Signup / Registration form. Let’s see how to perform component testing on one of the components based on the requirements given below.

***Signup Form***

***Username*** 

***Password***  

***Confirm Password*** 

***Email***  

***Contact***  

***Date of Birth*** 



Requirements:

Username

* It can be from 5 to 15 characters.
* Cannot contain special characters or numbers

Password:

* It can contain 8 characters and must include one number, one upper case and one special character.

Confirm Password:

* It must be same as the password mentioned.

Email:

* It should be at least 15 characters in length

Contact:

* It should be only numbers
* Cannot exceed 10 digits
* Cannot start with 0

Date of Birth:

* Should be in (DD/MM/YYYY) format

Based on the above given requirements, if a specific component called “Username” needs to be tested, below are the component testing test steps that needs to be followed.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step #** | **Step Details** | **Test Data** | **Expected Results** |
|
| **1** |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Integration Testing

Testing the data flow between two different modules (features) is called as Integration Testing.

**How to perform Integration Testing?**

* Understand the application thoroughly. That is understand how every feature works. Also understand how every feature is related or linked to each other.
* Identify all possible scenarios.
* Prioritize all the scenarios for execution.
* Test all the scenarios.
* If you find defects, communicate defect report to developers.