ASSIGNMENT – I
SOFTWARE TESTING METHODS AND TOOLS

NAME	**:**	DEBAYAN PAL
REGISTER NO	**:**	23370016
DEPARTMENT	**:**	COMPUTER SCIENCE
COURSE NAME	**:**	SOFTWARE QUALITY ASSURANCE
COURSE CODE	**:**	CSEL455
SEMESTER	**:**	3
DATE	**:**	22/10/2024

SOFTWARE TESTING METHODS AND TOOLS

(Functional Testing)

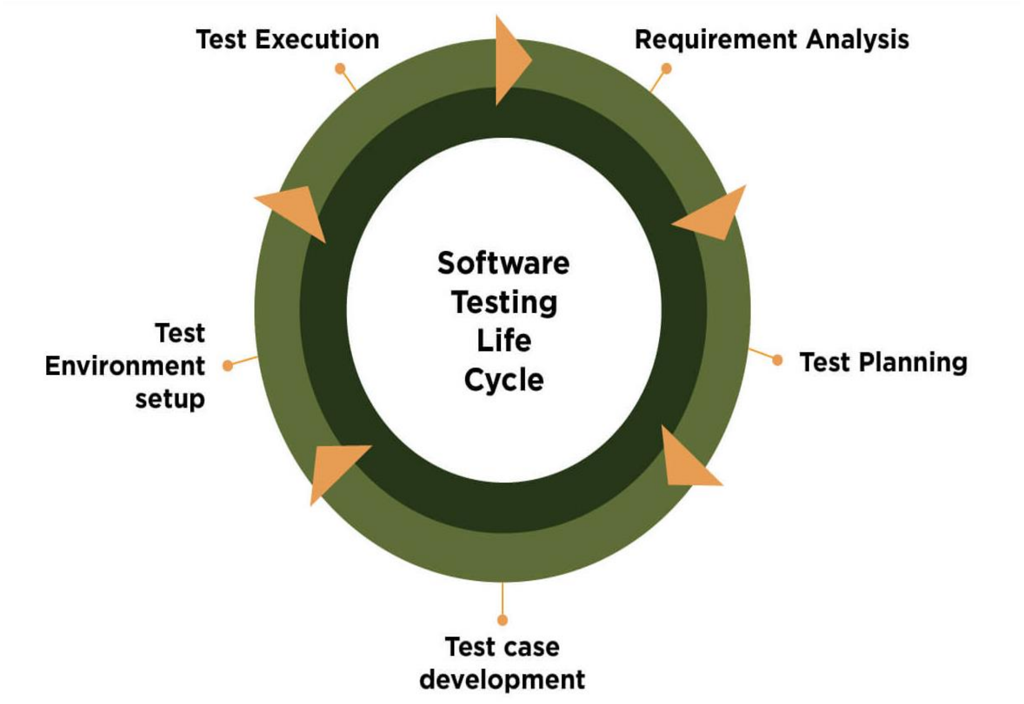
(Performance Testing)

INTRODUCTION

Software testing is the process of assessing the functionality of a software program. The process checks for errors and gaps and whether the outcome of the application matches desired expectations before the software is installed and goes live.

The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

Software testing life cycle:



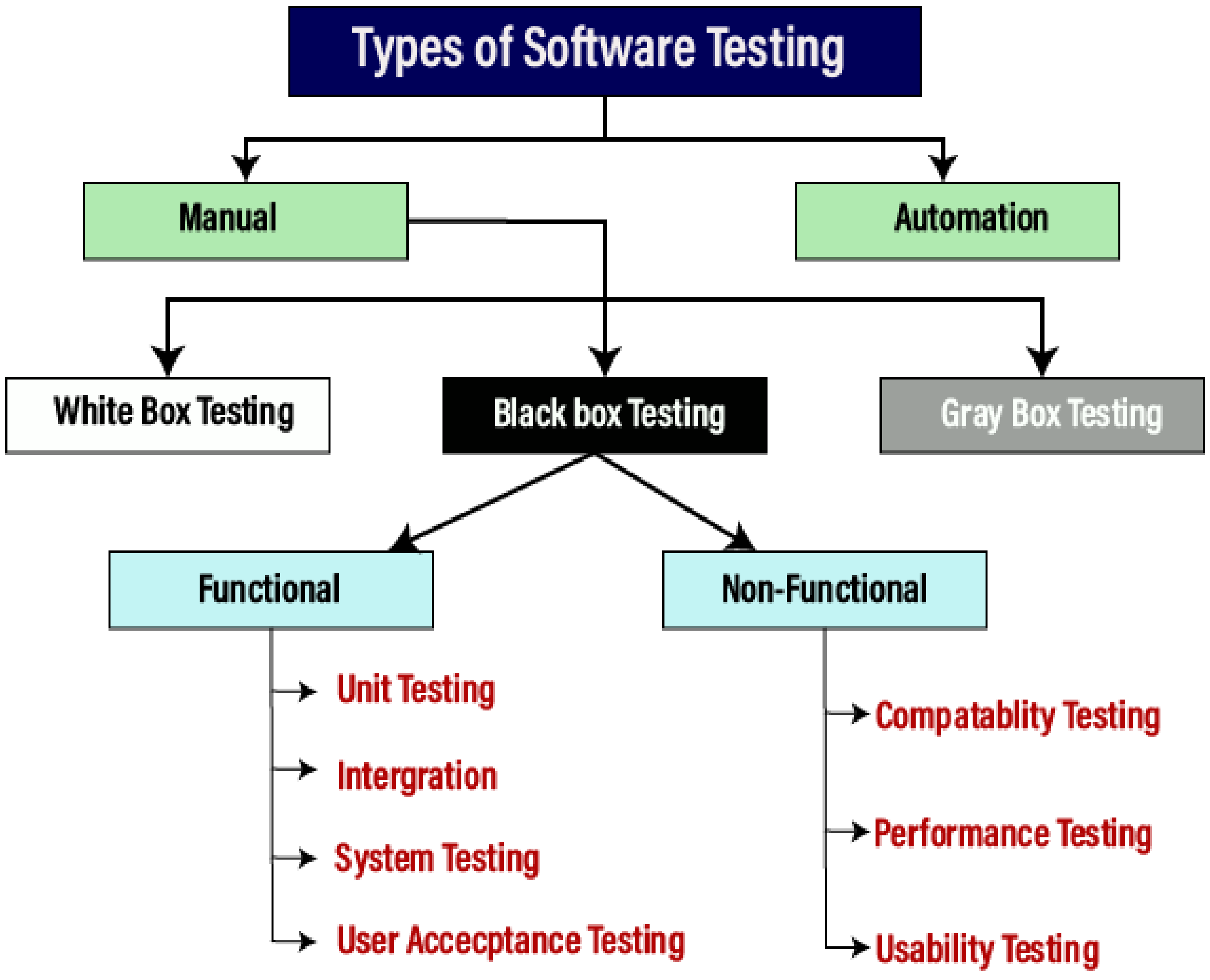
- Requirement Analysis
- Test planning
- Test case Development
- Test Environment Setup
- Test Execution

SOFTWARE TESTING METHODS:

A group activity of testing at testing a component or system focused on a specific test objective, i.e., functional test, usability test, regression test etc. A test type may take place on one or more test levels or test phases.

- Manual
- Automation

Software Testing (METHODS)Types Flowchart:



Software Testing Tools:

Software Testing tools are the tools that are used for the testing of software. Software testing tools are often used to assure firmness, thoroughness, and performance in testing software products. Unit testing and subsequent integration testing can be performed by software testing tools. These tools are used to fulfill all the requirements of planned testing activities. These tools also work as commercial software testing tools. The quality of the software is evaluated by software testers with the help of various testing tools.

Types of Testing Tools:

Software testing is of two types, static testing, and dynamic testing.

Testing tools can be categorized into two types which are as follows:

1. **Static Test Tools:** Static test tools are used to work on the static testing processes.

Static test tools consist of the following:

Flow analyzers: Flow analyzers provides flexibility in the data flow from input to output.

Path Tests: It finds the not used code and code with inconsistency in the software.

Coverage Analyzers: All rationale paths in the software are assured by the coverage analyzers.

Interface Analyzers: They check out the consequences of passing variables and data in the modules.

2. **Dynamic Test Tools:** Dynamic testing process is performed by the dynamic test tools. These tools test the software with existing or current data. Dynamic test tools comprise the following:

Test driver: The test driver provides the input data to a module-under-test (MUT).

Test Beds: It displays source code along with the program under execution at the same time.

Emulators: Emulators provide the response facilities which are used to imitate parts of the system not yet developed.

Mutation Analyzers: They are used for testing the fault tolerance of the system by knowingly providing the errors in the code of the software.

Top 10 Software Testing Tools:

1. **Test Management Tools:**

   Test management tools are used to store information on how testing is to be done, help to plan test activities, and report the status of quality assurance activities. For example, JIRA, Redmine, Selenium, etc.

2. **Automated Testing Tools:**

   Automated testing tools helps to conduct testing activities without human intervention with more accuracy and less time and effort. For example, Appium, Cucumber, Ranorex, etc.

3. **Performance Testing Tools:**

   Performance testing tools helps to perform effectively and efficiently performance testing which is a type of non-functional testing that checks the application for parameters like stability, scalability, performance, speed, etc. For example, WebLOAD, Apache JMeter, Neo Load, etc.

4. **Cross-browser Testing Tools:**

   Cross-browser testing tools helps to perform cross-browser testing that lets the tester check whether the website works as intended when accessed through different browser-OS combinations. For example, Testsigma, Testim, Perfecto, etc.

5. **Integration Testing Tools:**

   Integration testing tools are used to test the interface between the modules and detect the bugs. The main purpose here is to check whether the specific modules are working as per the client's needs or not. For example, Citrus, FitNesse, TESSY, etc.

6. **Unit Testing Tools:**

   Unit testing tools are used to check the functionality of individual modules and to make sure that all independent modules work as expected. For example, Jenkins, PHPUnit, JUnit, etc.

7. **Mobile Testing Tools:**

   Mobile testing tools are used to test the application for compatibility on different mobile devices. For example, Appium, Robotium, Test IO, etc.

8. **GUI Testing Tools:**

   GUI testing tools are used to test the graphical user interface of the software. For example, EggPlant, Squish, AutoIT, etc.

9. **Bug Tracking Tools:**

Bug tracking tool helps to keep track of various bugs that come up during the application lifecycle management. It helps to monitor and log all the bugs that are detected during software testing. For example, Trello, JIRA, GitHub, etc.

10. **Security Testing Tools:**

Security testing is used to detect the vulnerabilities and safeguard the application against the malicious attacks. For example, NetSparker, Vega, ImmuniWeb, etc.

I selected two **SOFTWARE TESTINGS METHODS** and its **TOOLS** with examples and unique features and also its advantages and disadvantages of each Testing.

- ❑ **Functional Testing**
- ❑ **Performance Testing**

Functional Testing:

Functional testing is a type of software testing that evaluates the functionality of a software application by testing it against the specified requirements. It verifies that each function of the software operates in accordance with the functional requirements.

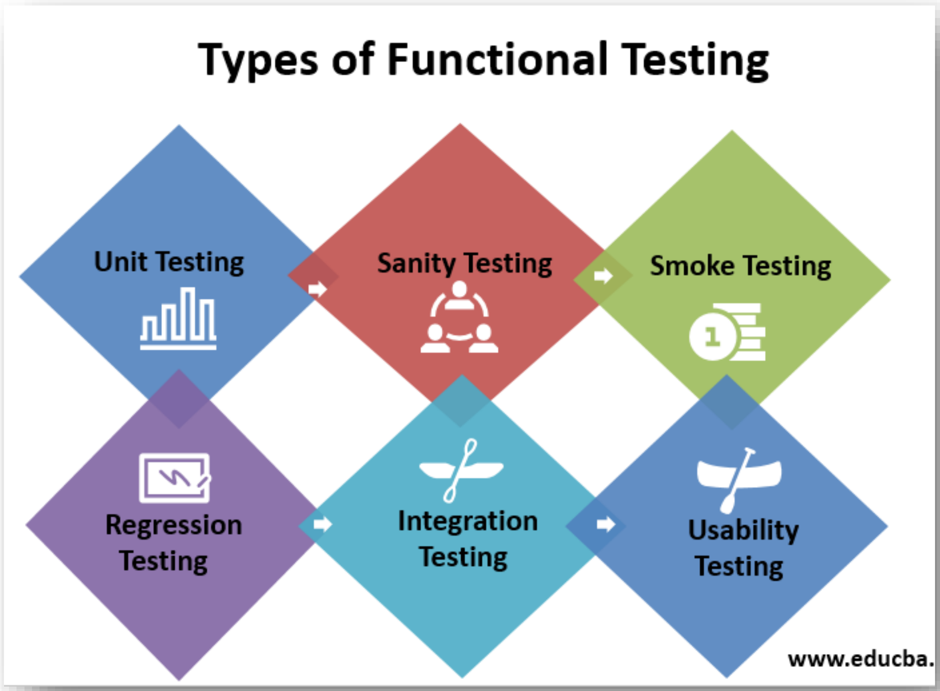
In essence, this testing process involves testing the individual features or functions of the software by providing input and checking if the output matches the expected results. It ensures that the application behaves as expected, performs its intended functions, and meets the user's requirements.

Functional testing can include various techniques such as unit testing, integration testing, system testing, and acceptance testing, among others, to thoroughly examine different aspects of the software's functionality. The goal is to identify any discrepancies between the expected behavior and the actual behavior of the software.

Functional testing types:

Types of Functional Testing and Examples,

1. Unit testing.
2. Component testing.
3. Smoke testing.
4. Sanity testing.
5. Regression testing.
6. Integration testing.
7. API testing.
8. UI testing.



Here are some types of functional testing along with their definitions and examples:

1. Unit Testing:

Definition: It tests individual units or components of the software to ensure they function correctly.

Example: Testing a specific function or method within a code module to verify its behavior.

2. Integration Testing:

Definition: Verifies interactions between different components/modules to ensure they work together as expected.

Example: Testing the interaction between a database and the application to ensure data retrieval and storage function correctly.

3. System Testing:

Definition: Tests the complete and integrated software system to evaluate its compliance with specified requirements.

Example: Testing the entire application to validate its functionality, user interfaces, and system interactions.

4. Acceptance Testing:

Definition: Validates if the system meets the acceptance criteria and user requirements.

Example: Conducting User Acceptance Testing (UAT) where end-users or stakeholders perform tests to ensure the software meets their needs.

5. Smoke Testing:

Definition: A preliminary test to check if the basic functionalities of the software work without encountering critical errors.

Example: Testing the login functionality of a web application to ensure basic user access.

6. Regression Testing:

Definition: Verifies that new changes or updates to the software have not adversely affected existing functionalities.

Example: Re-running test cases for previously working features after implementing new code to ensure they still function as expected.

7. Functional/Black Box Testing:

Definition: Tests the software based on external specifications without knowledge of internal code structure.

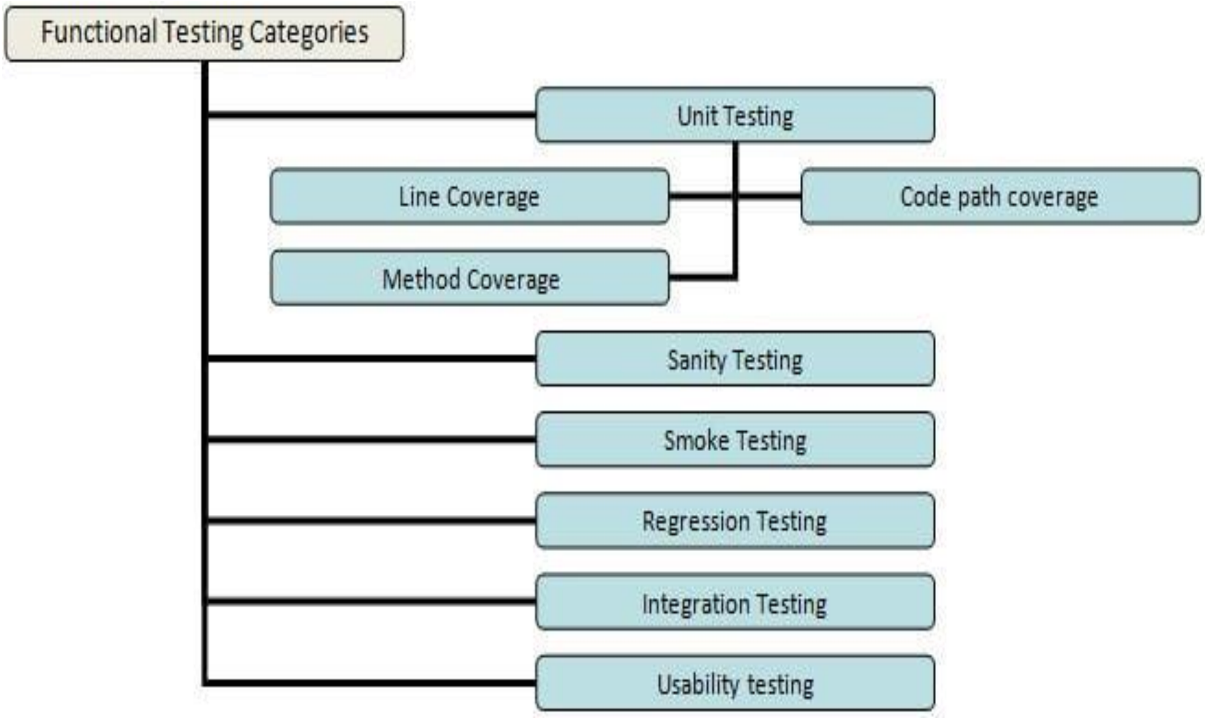
Example: Inputting specific data into a form and verifying that the output matches the expected result.

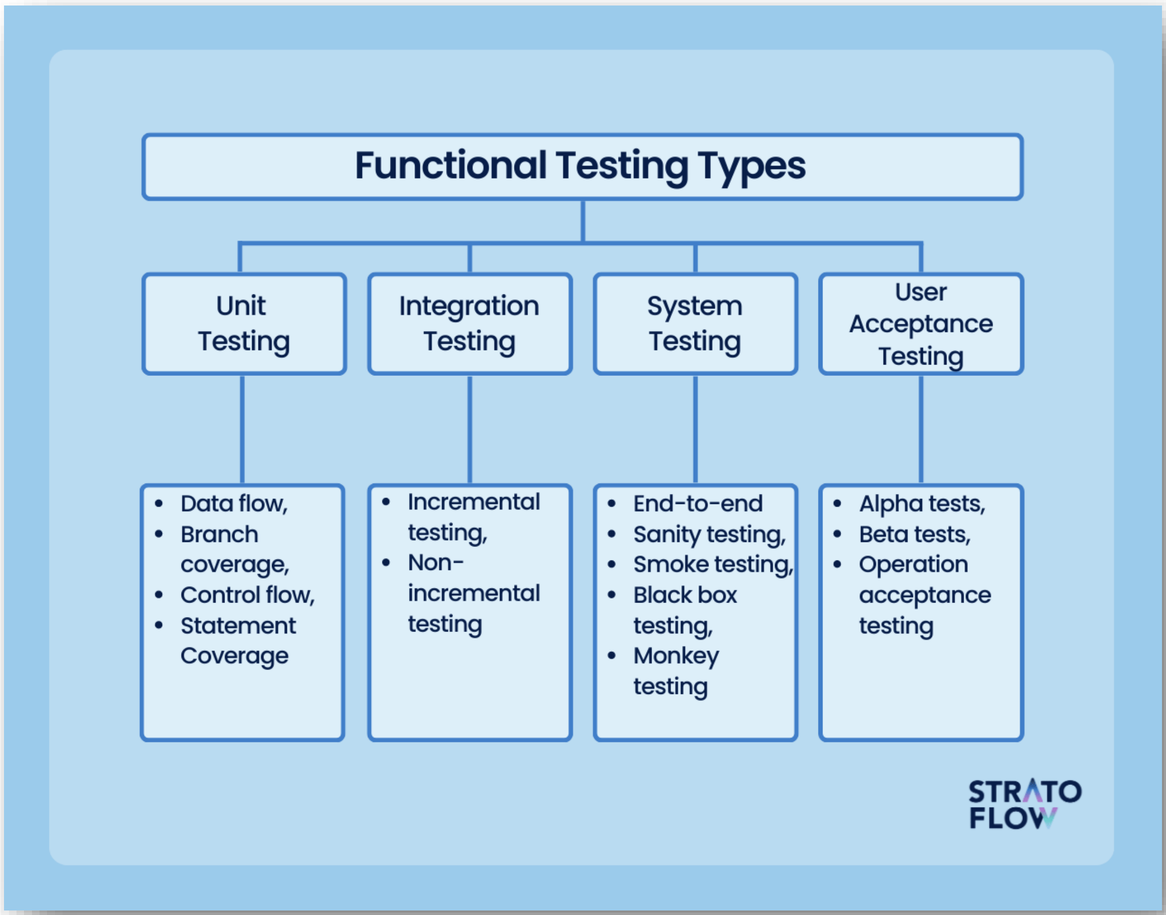
8. User Interface (UI) Testing:

Definition: Evaluates the graphical interface to ensure it meets design specifications and is user-friendly.

Example: Checking button placements, font sizes, and color schemes on a website to ensure consistency and usability.

Each type of functional testing serves a unique purpose in ensuring the software or system functions correctly, meeting the specified requirements and user expectations.


Functional Testing Categories
Unit Testing
Line Coverage
Code path coverage
Method Coverage
Sanity Testing
Smoke Testing
Regression Testing
Integration Testing
Usability testing


Functional Testing Types
Unit Testing
Integration Testing
System Testing
User Acceptance Testing
• Data flow,
• Branch coverage,
• Control flow,
• Statement Coverage
• Incremental testing,
• Non-incremental testing
• End-to-end
• Sanity testing,
• Smoke testing,
• Black box testing,
• Monkey testing
• Alpha tests,
• Beta tests,
• Operation acceptance testing
STRATO FLOW

Functional Testing Tools With Definition And Example:

Functional testing tools are software applications used to automate the validation of a system's functionality against specified requirements. They simulate user actions, input data, and expected outcomes to ensure that the software operates as intended.

Here are a few examples of functional testing tools along with brief definitions:

1. Selenium:

An open-source testing framework primarily used for web applications. It supports multiple programming languages and allows testers to write scripts to automate browser actions.

Example**:** Writing Selenium scripts in Python to automate the testing of web application functionalities such as form submissions, button clicks, and navigation.

2. HP Unified Functional Testing (UFT):

Formerly known as QTP (Quick Test Professional), UFT is a commercial tool used for functional and regression testing. It supports various technologies, including web, mobile, and API testing.

Example**:** Creating automated test scripts in UFT to validate the functionality of a banking application across different platforms and devices.

3. Appium:

An open-source tool used for testing mobile applications across different platforms (iOS, Android, and Windows). It enables automation of native, hybrid, and mobile web apps.

***Example*:** Developing Appium scripts in Java to automate interactions with an e-commerce mobile app, such as adding items to the cart and checking out.

4. SoapUI:

Primarily used for testing SOAP and RESTful web services. It allows testers to create, manage, and execute automated functional, compliance, and security tests.

***Example*:** Creating test suites in SoapUI to validate the functionality and performance of APIs by sending requests and verifying responses.

5. Postman:

A popular tool for API testing that offers features for designing, testing, and debugging APIs. It allows testers to create and automate API requests.

***Example*:** Using Postman collections to create automated test suites that validate endpoints, request payloads, and responses of a social media platform's API.

These tools help streamline the testing process, improve test coverage, and reduce the time required for repetitive manual testing tasks. They play a crucial role in ensuring the reliability and quality of software applications before they are deployed to production environments.

UNIQUE FEATURES OF FUNCTIONAL TESTING:

Functional testing involves evaluating a software application's functionality to ensure that it operates in accordance with requirements. Here are some unique features of functional testing:

1. **Requirement Validation:**

   It verifies whether the software meets specified requirements and functions as intended. Each feature or component is tested against the defined specifications.

2. **Black Box Testing:**

   Functional testing primarily involves black-box testing methods, where the internal logic, structure, or code of the application isn't known to the tester. Tests are conducted from a user's perspective.

3. **Test Cases Creation:**

   Test cases are designed based on various inputs, expected outputs, and specific functionalities of the software. These cases cover different scenarios to ensure comprehensive testing.

4. **Validation of User Expectations:**

   It aims to validate that the application meets user expectations and delivers the desired functionalities, features, and user experience. **5.Functional Coverage:** It focuses on covering all the functionalities of the application to ensure that each part performs as expected under different conditions.

5. **Regression Testing:**

   Functional testing often involves regression testing to ensure that new changes or features haven't adversely affected existing functionalities.

6. **Data Handling:**

   It verifies how the software handles data—input, storage, manipulation, and output. This includes checking for data integrity, accuracy, and security.

7. **UI/UX Validation:**

   Functional testing evaluates the user interface (UI) and user experience (UX) elements of the application to ensure they are intuitive, responsive, and user-friendly.

ADVANTAGES OF FUNCTIONAL TESTING:

Functional testing offers numerous advantages in the software development life cycle:

1. Validating Requirements
2. Detecting Bugs Early
3. Improving Quality
4. Enhancing User Experience
5. Mitigating Risks
6. Increasing Reliability
7. Regulatory Compliance
8. Facilitating Maintenance

Functional Testing Benefits:

1. Bug Identification
2. Ensuring Correct Functionality
3. Improved Quality
4. Validation of User Expectations
5. Reduction of Risks
6. Compliance and Standards
7. Enhanced Maintenance
8. Cost-Effectiveness

DISADVANTAGES OF FUNCTIONAL TESTING:

Functional testing, like any testing method, has its limitations and drawbacks. Here are some disadvantages:

1. Limited scope
2. Incomplete coverage
3. Dependency on specifications
4. Time-consuming
5. Not comprehensive for complex systems
6. Difficulty in handling dynamic systems
7. Inability to detect non-visible defects

Limitations Of Functional Testing:

Functional testing, while vital, does have its limitations. Here are some of them:

1. Incomplete Test Coverage
2. Limited Scope
3. Dependency on Documentation
4. Inability to Detect Integration Issues
5. Difficulty in Reproducing Bugs
6. Resource and Time Constraints
7. Difficulty in Handling Dynamic Data

PERFORMANCE TESTING

What is
Performance Testing?
A testing type for evaluation of a system's behavior under varying workload conditions.
SPACE TECHNOLOGIES

Performance testing is a non-functional software testing technique that determines how the stability, speed, scalability, and responsiveness of an application holds up under a given workload.

Performance testing is a method used to assess how a system performs under specific conditions. It evaluates the speed, responsiveness, stability, and scalability of software, applications, or devices under various workloads. The primary goal is to ensure that the system can handle the expected user load without compromising its functionality.

What is performance testing with example?

For example, system performance testing evaluates the system's overall speed and efficiency in responding to user inputs. Similarly, web application performance tests assess a web application's response time and resource consumption when exposed to increased load from multiple sources of traffic.

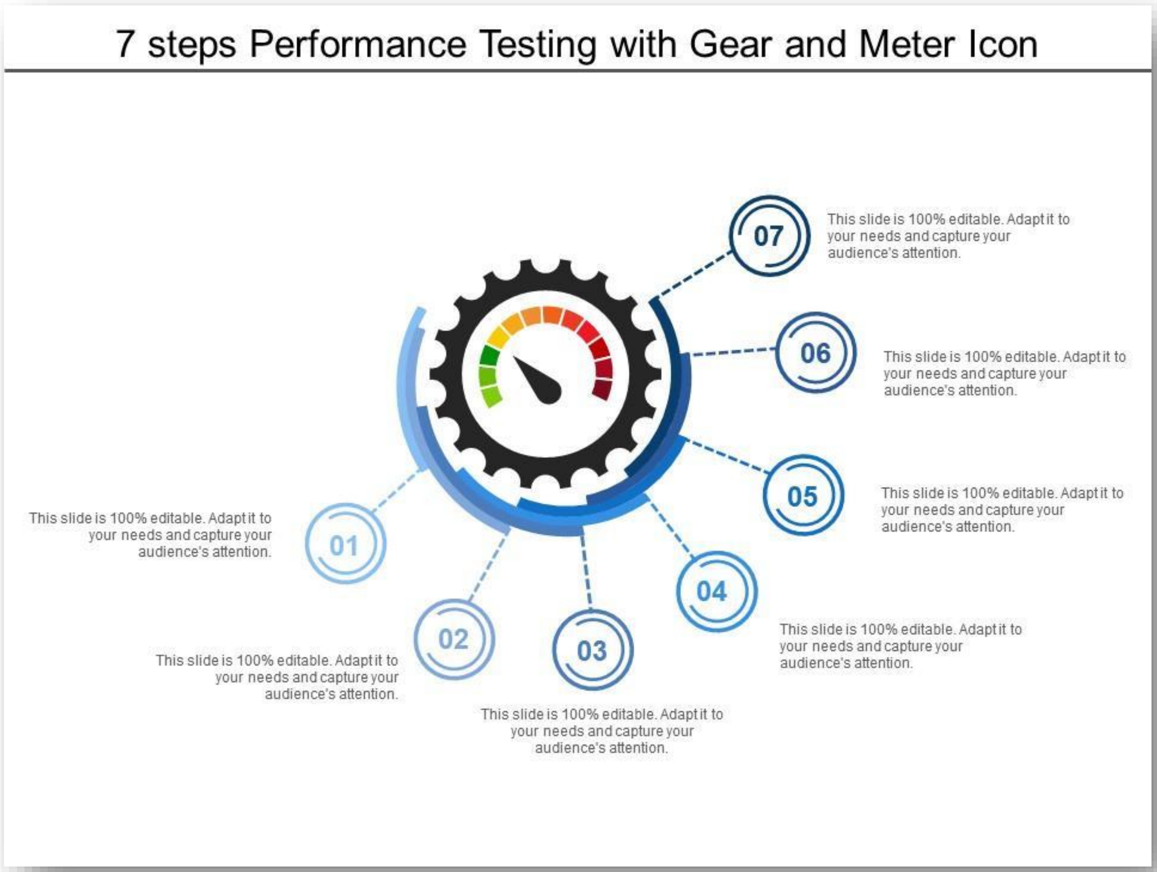
Performance Testing Metrics

- Throughput
- Memory
- Response time
- Bandwidth
- Average latency
- Central processing unit interrupts per second
- Average load time
- Peak response time
- Error
- Disk time

Seven Performance Testing Steps:

Plan and design performance tests. Configure the test environment. Implement your test design. Execute tests. Analyze, report, retest.29 Aug 2022

1. Identify the Testing Environment....
2. Identify Performance Metrics. ...
3. Plan and Design Performance Tests....
4. Configure the Test Environment….
5. Implement Your Test Design....
6. Execute Tests....
7. Analyze, Report, Retest….



7 steps Performance Testing with Gear and Meter Icon

01 This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

02 This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

03 This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

04 This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

05 This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

06 This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

07 This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

Types Of Performance Testing:

Performance testing involves assessing the speed, responsiveness, and stability of a software application under various conditions. There are several types of performance testing, each serving a specific purpose in evaluating different aspects of an application's performance:

1. **Load Testing:**
   - **Definition:** Load testing examines how a system performs under anticipated user loads. It helps determine the system's behavior under normal and peak load conditions.
   - **Example:** Simulating concurrent user interactions on an e-commerce website to assess how the system handles the traffic during regular days and holiday sales.
2. **Stress Testing:**

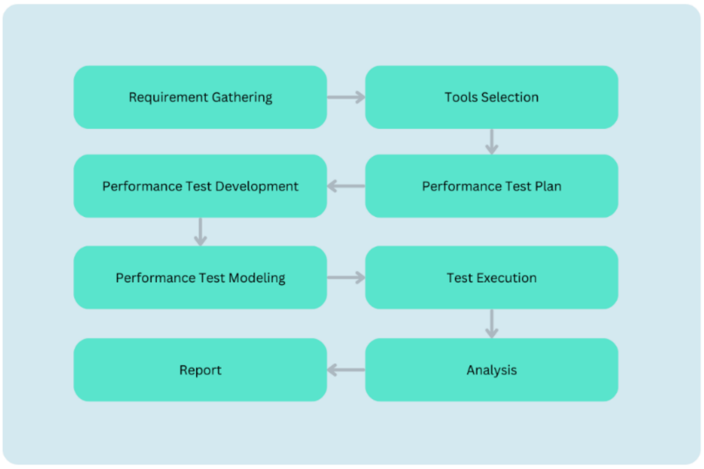
- **Definition:** Stress testing involves evaluating the system's behavior beyond its normal operational capacity, pushing it to its limits or beyond to identify breaking points.
- **Example:** Applying excessive load or initiating extreme conditions on a server to determine at what point it crashes or fails to respond.

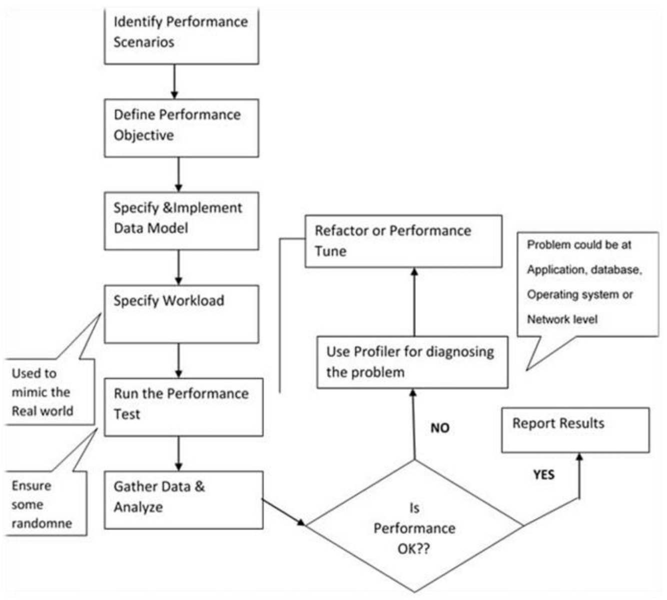
3. **Soak Testing (Endurance Testing):**
    - **Definition:** Soak testing involves subjecting the system to a sustained load over an extended period to check for memory leaks or performance degradation.
    - **Example:** Running an application continuously for 72 hours to ensure it operates correctly without memory leaks or performance decline.
4. **Scalability Testing:**
    - **Definition:** Scalability testing assesses an application's ability to handle increased workload by adding resources like CPU, memory, or servers.
    - **Example:** Testing how an online streaming service scales by adding more servers when the number of users exceeds a certain limit.
5. **Volume Testing:**
    - **Definition:** Volume testing checks the system's performance while handling a large amount of data, verifying its scalability concerning data size.
    - **Example:** Testing a database's performance with millions of records to ensure it functions efficiently without data corruption or slow response times.
6. **Concurrency Testing:**
    - **Definition:** Concurrency testing evaluates how well an application performs when multiple users access it simultaneously.
    - **Example:** Checking an online collaboration tool's performance when multiple users edit the same document or access shared resources concurrently.
7. **Compatibility Testing:**

- **Definition:** Compatibility testing verifies the application's performance across different environments, devices, operating systems, and browsers.
- **Example:** Testing a web application's performance on various browsers like Chrome, Firefox, Safari, and Edge to ensure consistent functionality.

Each type of performance testing aims to identify specific aspects of an application's performance under different conditions, helping to enhance its overall reliability, speed, and stability.

Performance Testing Process:

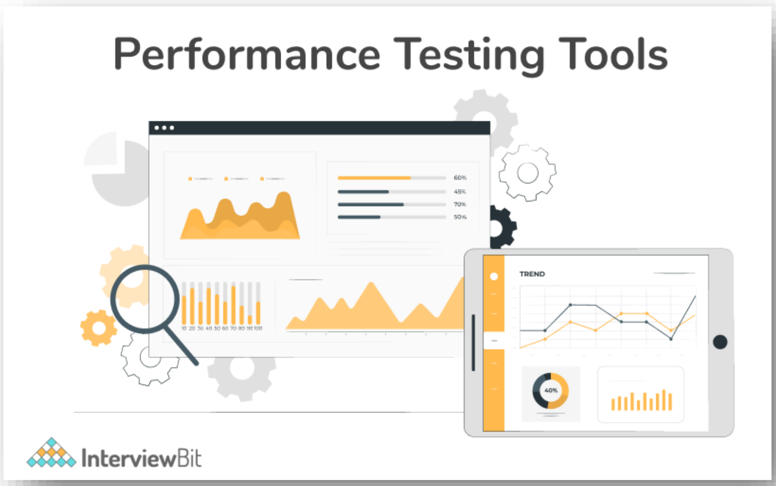



Identify Performance Scenarios
Define Performance Objective
Specify &Implement Data Model
Specify Workload
Run the Performance Test
Gather Data & Analyze
Used to mimic the Real world
Ensure some randomne
Is Performance OK??
NO
YES
Use Profiler for diagnosing the problem
Refactor or Performance Tune
Problem could be at Application, database, Operating system or Network level
Report Results

Performance Testing Tools with Definition and Examples:

Performance testing tools are software applications designed to assess the speed, responsiveness, stability, and scalability of a system under various conditions. These tools help identify performance bottlenecks, measure key performance metrics, and analyze how a system behaves under load.

Here are a few types of performance testing tools along with examples:

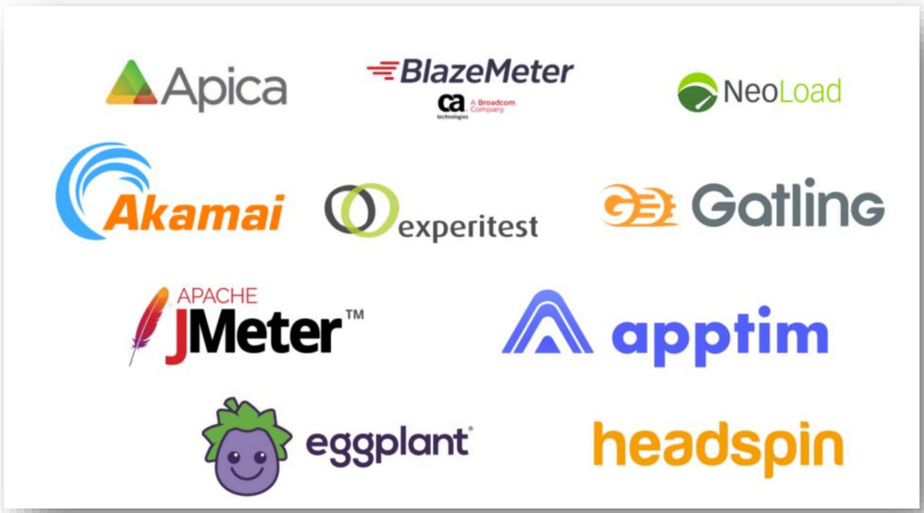


1. **Load Testing Tools:** These tools simulate high user traffic to evaluate system behavior under expected load conditions.
   - ***JMeter***: Apache JMeter is an open-source tool used for load testing and performance measurement.
   - ***LoadRunner***: Developed by Micro Focus, LoadRunner is a popular commercial tool for load testing.
2. **Stress Testing Tools:** These tools assess system robustness and stability under extreme conditions, pushing it beyond normal capacity to find its breaking point.
   - ***Gatling***: An open-source tool designed for stress testing web applications.
   - ***Siege***: Another open-source tool used to stress-test web servers.
3. **Performance Monitoring Tools:** These tools observe and collect real-time data on various system metrics to identify performance issues.

- ***New Relic***: A SaaS-based performance monitoring tool that provides insights into application performance.
- ***AppDynamics***: Offers real-time monitoring and analytics to detect and diagnose performance issues.

4. **Benchmarking Tools:** These tools compare the performance of a system against industry standards or competitors.
    - ***Passmark performance***: Allows benchmarking and comparison of different hardware components.
    - ***Geekbench***: Benchmarking tool for assessing CPU and GPU performance across different devices.
5. **Network Performance Tools:** These tools evaluate the performance of network infrastructure.
    - ***Wireshark***: A widely used network protocol analyzer for network troubleshooting and analysis.
    - ***PingPlotter*:** Helps in visualizing network performance and identifying issues like packet loss and latency.

Each tool has its strengths and suitability for different types of performance testing scenarios. The choice of tool often depends on factors like the type of application being tested, the scale of testing required, ease of use, and specific features needed for the testing process.



UNIQUE FEATURES OF PERFORMANCE TESTING:

Performance testing is a critical aspect of software development that ensures applications perform well under different conditions. Here are some unique features and aspects of performance testing:

1. Scalability Testing
2. Load Testing
3. Stress Testing
4. Endurance Testing
5. Volume Testing
6. Response Time Testing
7. Concurrency Testing

ADVANTAGES OF PERFORMANCE TESTING:

Performance testing offers numerous advantages that contribute significantly to the overall quality and reliability of software applications:

1. Identifying Performance Bottlenecks
2. Improving Response Time
3. Enhancing Scalability
4. Assuring Reliability and Stability
5. Cost Reduction
6. Optimizing Resource Utilization
7. Enhancing User Satisfaction

DISADVANTAGES OF PERFORMANCE TESTING:

1. Complexity in Simulation
2. Time-Consuming
3. Difficulty in Reproducing Issues
4. Dependency on Tools and Technology
5. Risk of Over-Optimization
6. Limited Scope of Testing
7. Cost and Resource Intensiveness