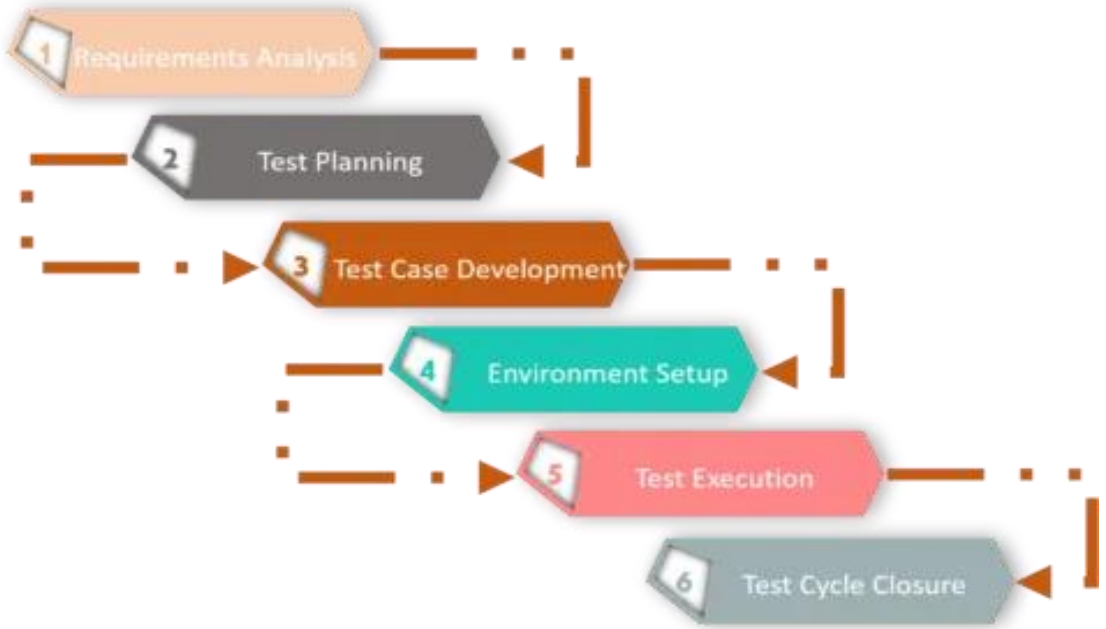
20th_feb: Testing

What is software Testing?

Software Testing is a process of evaluating the functionality of a software application to find any software bugs.

It checks whether the developed software met the specified requirements and identifies any defect in the software in order to produce a quality product.

It is also stated as the process of verifying and validating a software product.



Software Testing Stages:

Requirement Analysis – Understand and analyze software requirements for test planning.

Test Planning – Define testing scope, strategy, tools, schedule, and resources.

Test Case Design – Create detailed test cases based on requirements and use cases.

Test Environment Setup – Prepare hardware, software, and network configurations for testing.

Test Execution – Run test cases and log defects for failed scenarios.

Defect Reporting & Tracking – Identify, report, and track defects until resolution.

Test Closure – Review test results, generate reports, and finalize the testing process.

Automation and Manual Testing:

Manual Testing:

- No tools are used for the testing

-Manual testing is conducted to discover bugs in the developed software application.

-The tester checks all the essential features of the application.

-The tester executes test cases and generates test reports without any help from the automation tools.

-It is conducted by the experienced tester to accomplish the testing process.

When to Perform Manual Testing?

To check for Functionalities.

For User Interface.

For Website Behavior.

For Application Behavior.

For User Acceptance.

For User Experience.

Automation testing:

-Tools are used to perform the testing

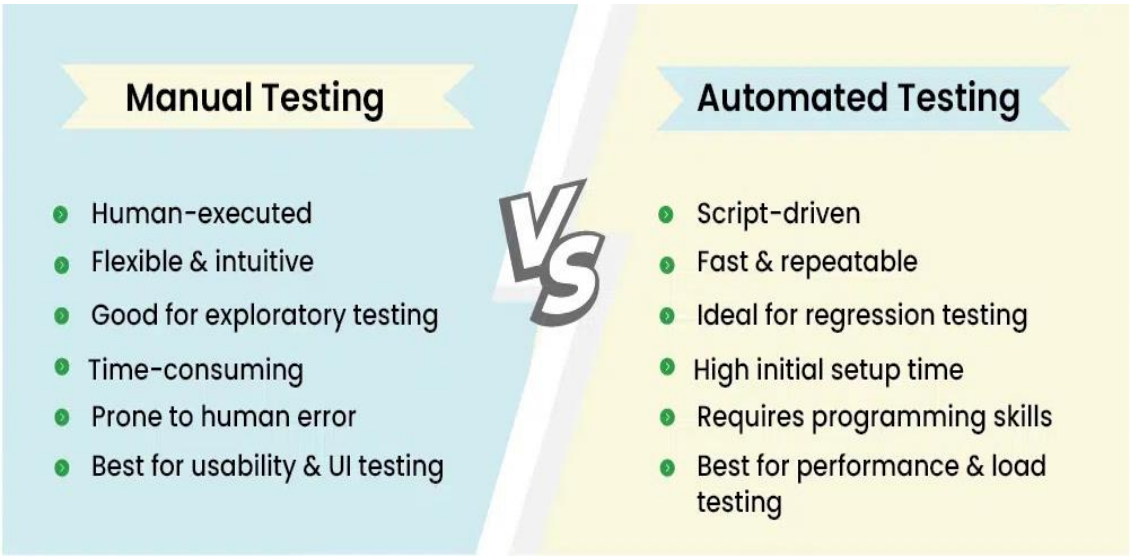
-It is fast and there is no chance of human errors

- It relies entirely on pre-scripted test which runs automatically to compare actual results with expected results.

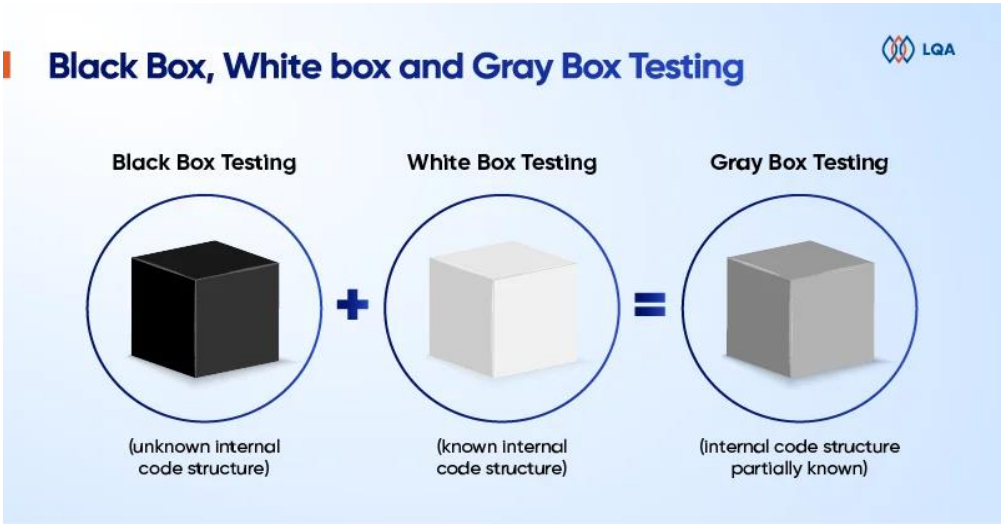
-Automation testing helps the tester determine whether the application performs as expected or not.

-It allows the execution of repetitive tasks and regression tests.

Automation requires manual effort to create initial testing scripts.



Types of testing:



1. Black box Testing:

Black Box Testing is a software testing technique where the tester evaluates the application's functionality **without knowing its internal code or implementation**. It focuses on **inputs and expected outputs** rather than how the system processes them internally.

Key Characteristics:

No knowledge of internal code or logic is required.
Focuses on **functional and non-functional** testing.
Based on **user perspective** and system requirements.

- Blck Box testing is best for the End Users prescptive.
- Mainly used Manual amd Automation Testing for Functinal and Non-Functional Testing.

Example of Black Box Testing:
Login Page Testing:

- Enter a **valid** username & password → Expect successful login.
- Enter an **invalid** password → Expect an error message.
- Leave fields blank → Expect a validation message.

2. White Box Testing:
White Box Testing (also called **Clear Box, Open Box, or Glass Box Testing**) is a software testing technique where the tester has **full knowledge of the internal structure, code, and logic** of the application. It is used to test the **code implementation, logic, and flow of the program**.

Key Characteristics:

- Requires **knowledge of coding and system architecture**.
- Focuses on **code structure, logic, and data flow**.
- Helps in identifying **logical errors, security flaws, and code inefficiencies**.

-**White Box Testing** is best for testing **code logic, security, and performance**
-It is widely used in **unit testing, integration testing, and security testing**

Example:
Login Function Testing (Valid or Invalid User Credentials)

3. Grey Box Testing:

Grey Box Testing is a **hybrid** testing technique that combines elements of both **Black Box** and **White Box Testing**. In this approach, the tester has **partial knowledge of the internal structure** while still focusing on functionality and user experience.

Key Characteristics:

- Tester has **limited knowledge** of the internal code or architecture.
- Focuses on **both functional and structural aspects**.
- Helps detect **security flaws, integration issues, and boundary defects**.
- Suitable for **API Testing, Database Testing, and Security Testing**.

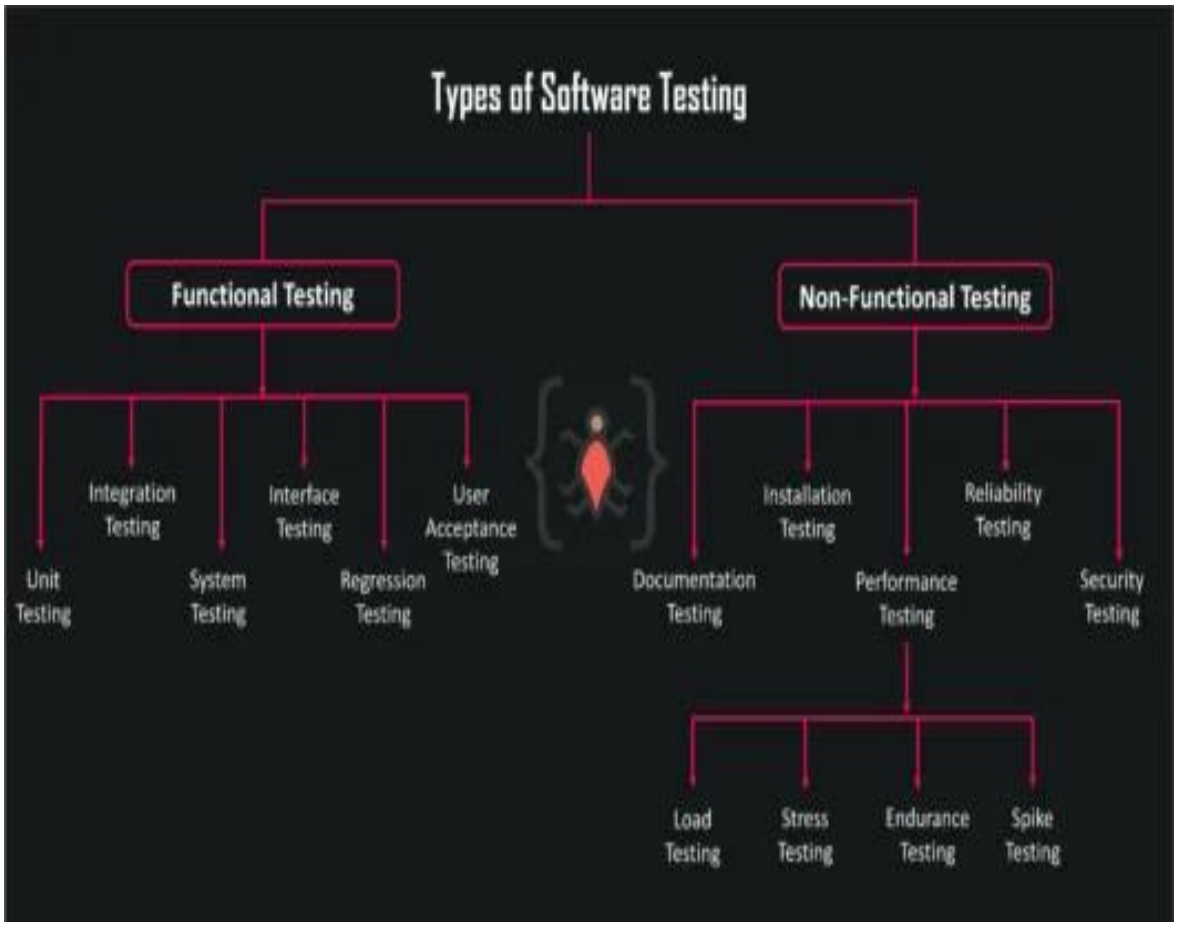
-**Grey Box Testing** is ideal for **API testing, security testing, and integrtation Testing.**

- It helps detect **database errors, security vulnerabilities, and functional bugs**.

Example of Grey Box Testing: E-commerce Website Order Processing

- A tester knows how the **database stores order details** but tests from a user's perspective.
- Enters an order and **checks if the correct data is saved** in the database.
- Runs SQL queries to verify if the order is recorded correctly.

Testing Levels:



What Exactly is the Functional Testing?

Functional testing is a type of(Black Box) software testing where the system is tested against the functional requirements or specifications like the technical details, data manipulation and processing, and other specific functionalities.

This type of testing checks whether the application works as per the user need/requirement or not.

Functions (features) are tested by feeding the input and examining the output of the system.

Types of Functional Testing:

- **Unit testing**
- **Integration testing**
- **System testing**
- **Interface testing**
- **Regression testing**
- **User-acceptance testing**
- **Alpha Testing**
- **Beta Testing**

What is Non-Functional Testing?

Non-functional testing is another type of software testing which is used to check the non-functional aspects like performance, usability, reliability, etc. of a software application which is not tested using functional testing.

Basically, non-functional testing is done to check and evaluate all the non-functional parameters, such as speed, scalability, security, reliability, and efficiency of an application

It makes an application robust and prepares it against certain vulnerabilities.

Types of Non-Functional Testing

- Documentation testing
- Installation testing
- Performance testing
- Reliability testing
- Security testing

1. What is Unit Test?

Unit testing is the process of testing the smallest parts of your code, like individual functions or methods, to make sure they work correctly. It's a key part of software development that improves code quality by testing each unit in isolation.

Note: Unit Testing basically Included in both White Box Testing and Black Box Testing.

Advantages of Unit Testing:

It helps to identify bugs early in the development process before they become more difficult and expensive to fix.
It helps to ensure that changes to the code do not introduce new bugs.
It makes the code more modular and easier to understand and maintain.
It helps to improve the overall quality and reliability of the software.

2. Integration Testing

It is a method of testing how different units or components of a software application interact with each other. It is used to identify and resolve any issues that may arise when different units of the software are combined. Integration testing is typically done after unit testing and before functional testing and is used to verify that the different units of the software work together as intended.

Different Ways of Performing Integration Testing:

Top-down integration testing: It starts with the highest-level modules and differentiates them from lower-level modules.
Bottom-up integration testing: It starts with the lowest-level modules and integrates them with higher-level modules.

Big-Bang integration testing: It combines all the modules and integrates them all at once.
Incremental integration testing: It integrates the modules in small groups, testing each group as it is added.

3. System Testing

System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. This type of testing is performed after the integration testing and before the acceptance testing.

4. Acceptance Testing

It is formal testing according to user needs, requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the users, customers, or other authorized entities to determine whether to accept the system or not.

5. Regression Testing

Ensures **new code changes do not break existing functionality**.
Typically done **after bug fixes, enhancements, or code updates**.
Example: After updating an e-commerce website's payment gateway, testers check whether other existing features still work properly.

6. Alpha Testing

Conducted before the product is released to real users.
Performed by internal testers or employees in a controlled environment.
Aims to identify bugs before external testing.
Example: A software company tests a new app internally before launching the beta version for users.

7. Beta Testing

Conducted after alpha testing but before the official release.
Performed by real users in a real-world environment.
Helps gather feedback, uncover unexpected issues, and improve user experience.

Example: A company releases a beta version of a new mobile app to selected users for feedback**.**

Non-Functional Testing:

1. Performance Testing: Measures the speed, responsiveness, and stability of an application under normal and peak load conditions.
Ensuring that the system performs well in terms of speed, resource usage, and scalability.
Example:
Testing the time it takes for an e-commerce website to load a product page with multiple high-resolution images.

2. Stress Testing: Determines the system's behavior under extreme conditions, such as high traffic or data volume, to find the **breaking point** of the system.
Identifying system limitations and weaknesses under overload situations.

Example:
Simulating a sudden traffic spike on an online banking system during an account update and observing the response and recovery time.

3. Load Testing: Verifies that the application can handle a specific number of **simultaneous users** or **transactions** without performance degradation.
Identifying system limitations and weaknesses under overload situations.
Measuring system performance under expected or peak loads.

Example:
Testing a food delivery app's ability to process **100 orders per minute** during a promotional event.

4. Portability Testing: Ensures that the application works on various platforms, devices, and operating systems.
Verifying the application's **compatibility** and **adaptability** across different environments (e.g., mobile, browsers, operating systems).

Example:
Testing whether a shopping app is compatible with **various mobile devices** (Android, iOS) and **screen resolutions**.