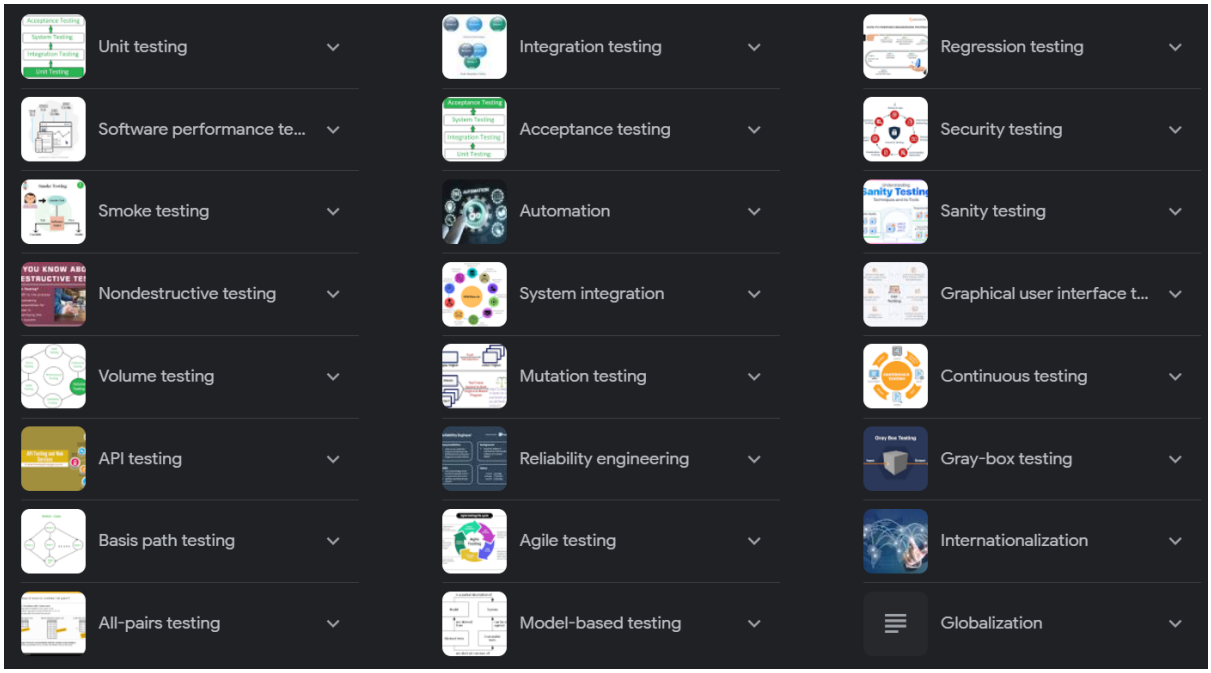
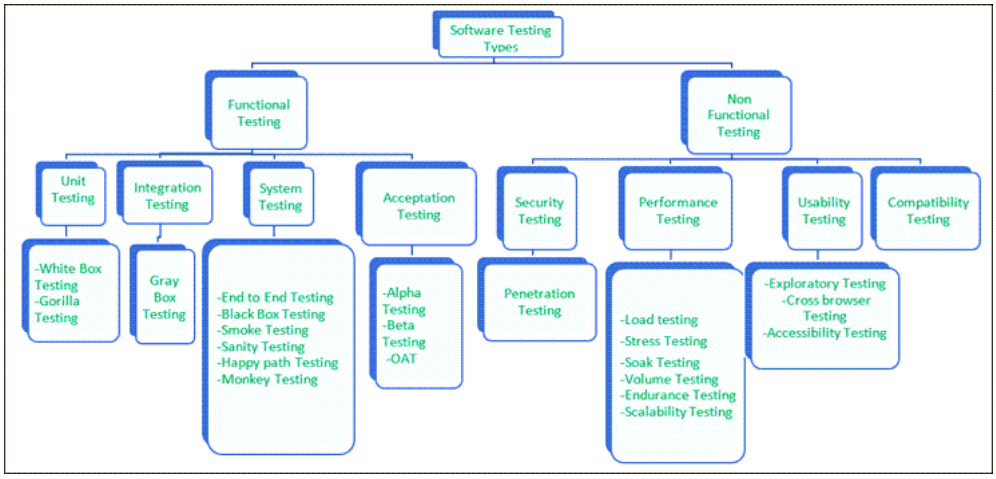
Testing notes





Testing is the process of executing a program to find errors. To make our software perform well it should be error-free. If testing is done successfully it will remove all the errors from the software.

Principles of Testing:-

(i) All the tests should meet the customer requirements.

(ii) To make our software testing should be performed by a third party.

(iii) Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.

(iv) All the tests to be conducted should be planned before implementing it

(v) It follows the Pareto rule(80/20 rule) which states that 80% of errors come from 20% of program components.

(vi) Start testing with small parts and extend it to large parts.

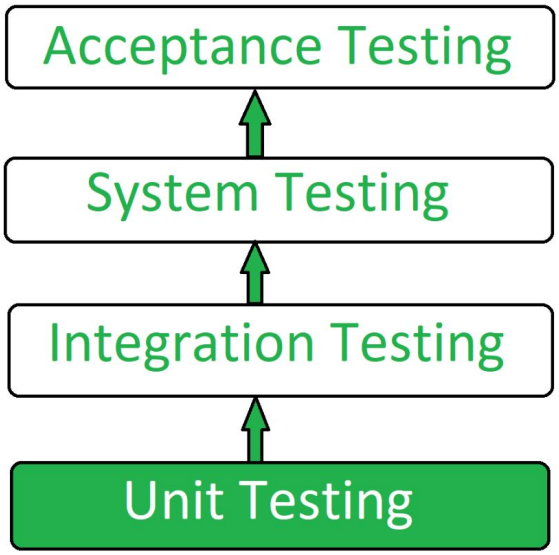
Types of Testing:-

1. Unit Testing

Unit testing is a method of testing individual units or components of a software application. It is typically done by developers and is used to ensure that the individual units of the software are working as intended. Unit tests are usually automated and are designed to test specific parts of the code, such as a particular function or method. Unit testing is done at the lowest level of the software development process, where individual units of code are tested in isolation.

The main advantages of unit testing include:

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



Some popular frameworks and tools that are used for unit testing include JUnit, NUnit, and xUnit.

It's important to keep in mind that unit testing is only one aspect of software testing and it should be used in combination with other types of testing such as integration testing, functional testing, and acceptance testing to ensure that the software meets the needs of its users.

It focuses on the smallest unit of software design. In this, we test an individual unit or group of interrelated units. It is often done by the programmer by using sample input and observing its corresponding outputs.

Example:

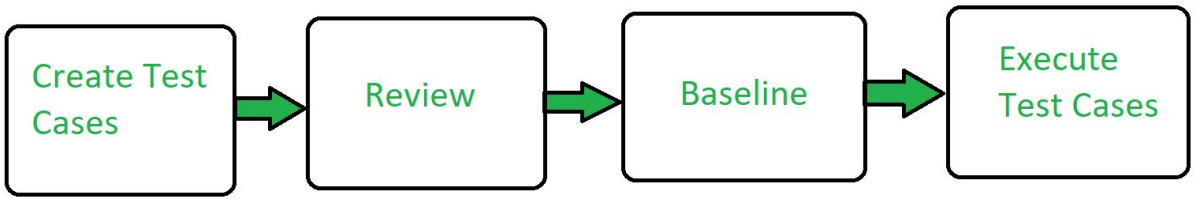
```
a) In a program we are checking if the loop, method, or
   function is working fine
b) Misunderstood or incorrect, arithmetic precedence.
c) Incorrect initialization
```

Unit Testing Techniques:

There are 3 types of Unit Testing Techniques. They are

1. **Black Box Testing:** This testing technique is used in covering the unit tests for input, user interface, and output parts.
2. **White Box Testing:** This technique is used in testing the functional behavior of the system by giving the input and checking the functionality output including the internal design structure and code of the modules.
3. **Gray Box Testing:** This technique is used in executing the relevant test cases, test methods, test functions, and analyzing the code performance for the modules.

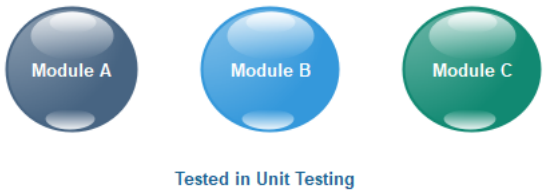
Workflow of Unit Testing:



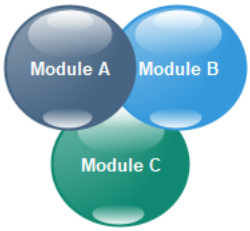
2. Integration Testing

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules



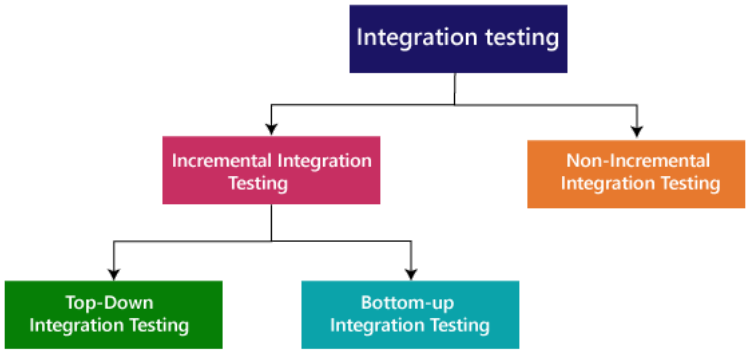
Tested in Unit Testing



Under Integration Testing

Integration testing can be performed in different ways, such as:

1. Top-down integration testing: It starts with the highest level modules and differentiate them with lower-level modules.
2. Bottom-up integration testing: It starts with the lowest-level modules and integrates them with higher-level modules.
3. Big-Bang integration testing: It combines all the modules and integrates them all at once.
4. Incremental integration testing: It integrates the modules in small groups, testing each group as it is added.



Integration testing is of four types: (i) Top-down (ii) Bottom-up (iii) Sandwich (iv) Big-Bang

Example:

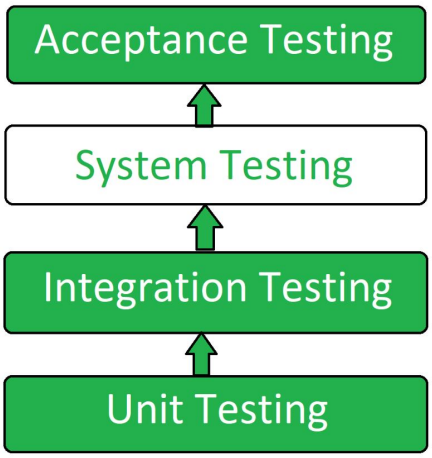
```
(a) Black Box testing:- It is used for validation.
In this, we ignore internal working mechanisms and
focus onwhat is the output?.

(b) White box testing:- It is used for verification.
In this, we focus on internal mechanisms i.e.
how the output is achieved?
```

3. System Testing

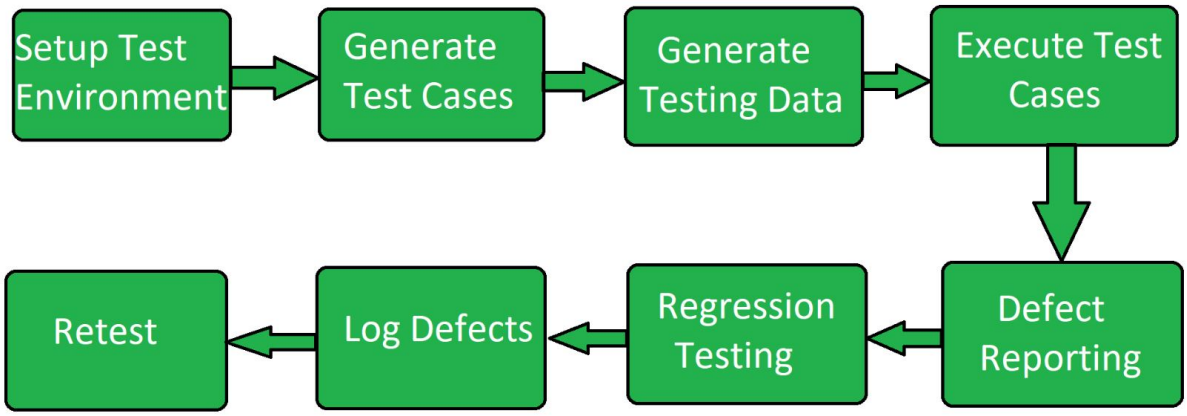
This type of testing validates the entire system to ensure that it meets all of its requirements.

System Testing is a type of **software testing** that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested. **System Testing** is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the **software requirements specification (SRS)**. System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing. **System Testing is a black-box testing**. System Testing is performed after the integration testing and before the acceptance testing.



System Testing Process: System Testing is performed in the following steps:

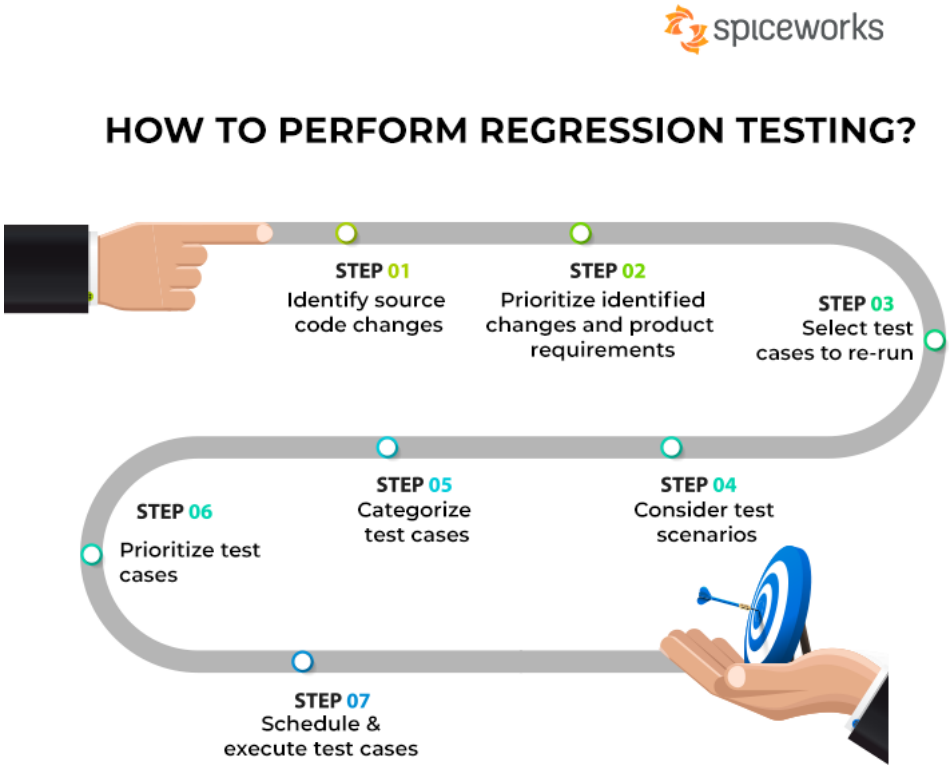
- **Test Environment Setup:** Create testing environment for the better quality testing.
- **Create Test Case:** Generate test case for the testing process.
- **Create Test Data:** Generate the data that is to be tested.
- **Execute Test Case:** After the generation of the test case and the test data, test cases are executed.
- **Defect Reporting:** Defects in the system are detected.
- **Regression Testing:** It is carried out to test the side effects of the testing process.
- **Log Defects:** Defects are fixed in this step.
- **Retest:** If the test is not successful then again test is performed.



4. Regression Testing

Regression testing is a method of testing that is used to ensure that changes made to the software do not introduce new bugs or cause existing functionality to break. It is typically done after changes have been made to the code, such as bug fixes or new features, and is used to verify that the software still works as intended.

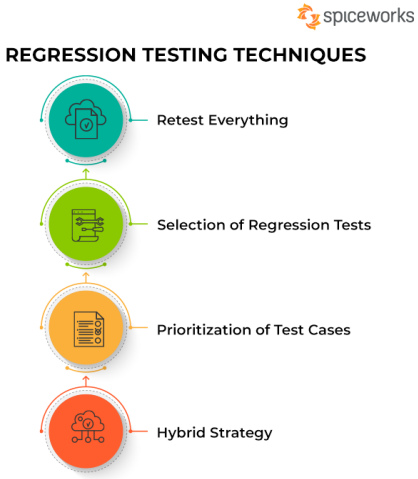
Regression testing refers to a software testing technique that re-runs non-functional and functional tests to ensure that a software application works as intended after any code changes, updates, revisions, improvements, or optimizations. It is an integral part of the software development cycle as it allows developers to detect unexpected faults in an application that may arise due to tweaks, enhancements, or extending of the existing codebase.



Regression testing can be performed in different ways, such as:

1. **Retesting**: This involves testing the entire application or specific functionality that was affected by the changes.

2. **Re–execution**: This involves running a previously executed test suite to ensure that the changes did not break any existing functionality.
3. **Comparison**: This involves comparing the current version of the software with a previous version to ensure that the changes did not break any existing functionality.



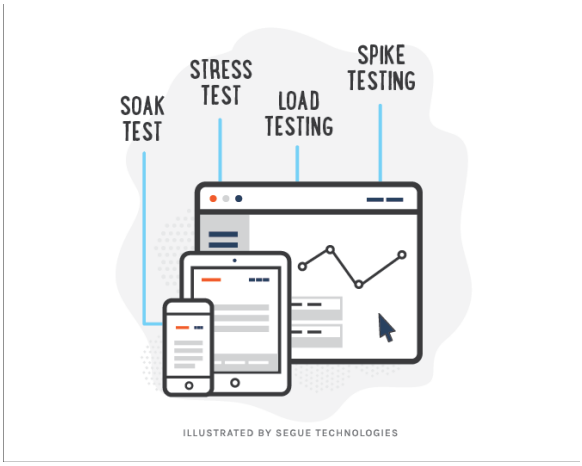
Every time a new module is added leads to changes in the program. This type of testing makes sure that the whole component works properly even after adding components to the complete program.

Example

```
In school, record suppose we have module staff, students
and finance combining these modules and checking if on
integration of these modules works fine in regression testing
```

5.Software performance testing

Software performance testing is the practice of determining whether a given application has the capacity to perform in terms of scalability and responsiveness under a specified workload.



Load Tests

Load testing is used to study the behavior of the application under specified loads. It also shows how an application will function when the majority of its users are logged in. Load testing is mainly done to measure response times, resource utilization levels, and throughput rates.

Stress Tests

A stress test is performed to determine the upper limit of the application capacity and how the application performs when the current load exceeds the expected maximum. The primary focus of performing testing is to identify application bugs that occur in high load conditions. This test determines the maximum load that a given application can support.

Soak Tests

Soak tests are performed with the objective of determining how the application endures under a continuous expected load. For example, a soak test can be performed to monitor memory utilization and detect memory leaks and other performance problems that can occur. The objective of performing this type of test is to determine the application's performance in sustained use.

Spike Tests

Spike testing is performed to determine whether a given application has the capacity to sustain the workload. The test is accomplished by increasing the number of end-users by a large amount and assessing the performance of the application overall.

Configuration and Isolation Tests

While Microsoft states that configuration and isolation testing are useful, performing these tests is typically uncommon. Configuration testing is performed to determine the impacts of configuration changes to components of the application on the application's behavior and performance. Isolation testing is performed to isolate the fault domain.

Using External Resources for Performance Testing

Setting up the environment for performance testing for various applications (web, desktop, mobile) can be time consuming and expensive. We decided to use Neustar's Web performance service for our testing needs. Please read this case study to learn more about our decision to use Neustar and our experience working with them.

The quality of your website performance is important. People will visit a website less often if it is slower than a close competitor by more than 250 milliseconds (NYTIMES). When you are accessing the quality of your application, be sure to include how well it performs. Your web application's performance could be the factor that draws a customer to your services instead of a competitor's.

4. Smoke Testing

Smoke testing involves checking the basic functionality of the application to determine if major features of the system are working as expected. Especially useful after a new build, smoke tests help in determining if more expensive tests need to be run on the application in the newly deployed environment. They also help reveal if failures are severe enough to reject a prospective software release.

Example:

```
If the project has 2 modules so before going to the module
make sure that module 1 works properly
```

5. Sanity Testing

It helps to check whether added new functionality is working according to requirements or not. If a newly added functionality is not working according to requirement, then, in that case, it fails. If a newly added functionality to a system and web application is working according to a requirement, then the Sanity test passes. When the Sanity test passes, complete system testing is carried out to check that newly added functionally to an application will not affect the previously present system and application components.

Sanity testing is a form of regression testing that helps testers determine if new code changes or feature updates perform well enough to undergo a major testing effort. If the application crashes, it means the system is not stable enough for further testing, and an additional build is assigned to fix the problems. The objective of sanity testing is not to thoroughly verify the functionality of the software, but instead to measure if the complete build of the software is proper.

Sanity Testing vs Regression Testing vs Smoke Testing

Regression, Smoke, and Sanity testing difference is the most misunderstood topic inSoftware Testing. The differences are stated below.

Sanity Testing vs. Smoke Testing

- Sanity test has the goal of checking reasonability, whereas the smoke test has the goal of checking stability.
- Testers do sanity tests, and both developers and testers do Smoke tests.
- Sanity test verifies new features such as bug fixes, and Smoke tests verify the critical functionality of the system.
- Sanity tests are a subset of regression tests, and Smoking tests are a subset of acceptance tests.

Sanity Testing vs. Regression Testing:

- It is part of Regression Testing, whereas Regression Testing is independent testing.
- It is often performed manually, and Regression testing is preferred for automation.
- It is run to verify the stability of new features or code changes in existing builds. Regression testing is performed to verify the stability of all areas affected by functional or code changes

5. Alpha Testing

This is a type of validation testing. It is a type of *acceptance testing* which is done before the product is released to customers. It is typically done by QA people.

Example:

```
When software testing is performed internally within
the organization
```

6. Beta Testing

The beta test is conducted at one or more customer sites by the end-user of the software. This version is released for a limited number of users for testing in a real-time environment

Example:

```
When software testing is performed for the limited
number of people
```

7. Acceptance Testing

Acceptance testing is done by the customers to check whether the delivered products perform the desired tasks or not, as stated in requirements.

We use this OOT, for discussing test plans and for executing the projects.

This article is contributed by **Kritka**. If you like GeeksforGeeks and would like to contribute, you can also write an article using **write.geeksforgeeks.org** or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

8. Monkey test

Monkey testing is a technique in software testing where the user tests the application by providing random inputs and checking the behavior (or trying to crash the application). Mostly this technique is done automatically where the user enters any random invalid inputs and checks the behavior.

As said earlier, there are no rules; this technique does not follow any predefined test cases or strategy and thus works on the tester’s mood and gut feeling.

……

short notes from google:

1.

- **Unit testing:** This is the lowest level of testing, and it involves testing individual units of code to ensure that they are working correctly.
- **Integration testing:** This type of testing ensures that different units of code work together correctly.
- **System testing:** This type of testing validates the entire system to ensure that it meets all of its requirements.
- **Acceptance testing:** This type of testing is performed by the customer or end-user to ensure that the system meets their needs.
- **Performance testing:** This type of testing ensures that the system can handle the expected workload.
- **Security testing:** This type of testing ensures that the system is secure from unauthorized access or malicious attacks.
- **Usability testing:** This type of testing ensures that the system is easy to use and navigate.
- **Localization testing:** This type of testing ensures that the system is localized for different languages and cultures.

2.

1. Functional Testing: This type of testing verifies that the software functions as intended, based on its requirements and specifications. It involves testing individual features and their interactions to ensure they work correctly.
2. Regression Testing: Regression testing is performed to ensure that changes or enhancements to the software do not introduce new defects or break existing functionality. It involves retesting previously tested functionalities to uncover any unexpected issues.
3. Performance Testing: Performance testing evaluates the software's responsiveness, scalability, and stability under different load conditions. It helps identify performance bottlenecks, such as slow response times, high resource utilization, or system crashes.
4. Security Testing: Security testing aims to identify vulnerabilities and weaknesses in the software's security mechanisms. It involves assessing the software's ability to protect sensitive information, resist unauthorized access, and withstand malicious attacks.
5. Usability Testing: Usability testing focuses on assessing the software's user-friendliness and user experience. It involves evaluating how easily users can interact with the software, complete tasks, and understand its features, menus, and documentation.
6. Compatibility Testing: Compatibility testing ensures that the software functions correctly across different platforms, devices, browsers, and operating systems. It verifies that the software is compatible with the intended environment and does not exhibit any unexpected behavior.
7. Integration Testing: Integration testing verifies the proper interaction and communication between different software modules or components. It tests how well the integrated parts work together and ensures that data flows correctly between them.
8. Acceptance Testing: Acceptance testing involves testing the software from the end-user's perspective to determine whether it meets the specified requirements and is ready for deployment. It often includes user acceptance testing (UAT) where real users or representatives validate the software against their needs.
9. Exploratory Testing: Exploratory testing is a technique where testers dynamically explore the software to identify defects, without relying on predefined test cases. Testers use their domain knowledge, experience, and intuition to uncover issues that might have been missed through scripted testing.
10. Localization Testing: Localization testing ensures that the software is adapted and functions correctly in different languages, regions, and cultural contexts. It verifies the correct translation of text, formatting, date and time representations, and adherence to local regulations.

3.

functional and non functional:

1. Functional Testing: Functional testing focuses on verifying that the software behaves as expected and meets the functional requirements. It involves testing the individual features and functions of the software to ensure they work correctly and produce the desired results. Functional testing answers questions like "Does the software perform the tasks it's supposed to do?"

Examples of functional testing techniques include:

- Unit Testing: Testing individual units or components of the software in isolation to ensure they function as intended.
- Integration Testing: Testing the interaction and communication between different components or modules to verify their integration.
- System Testing: Testing the entire system as a whole to validate its behavior against the functional requirements.
- User Acceptance Testing (UAT): Involving end-users or stakeholders to test the software and ensure it meets their expectations and requirements.

1. Non-Functional Testing: Non-functional testing focuses on evaluating the software's attributes and characteristics that are not directly related to specific functionality. It aims to assess the quality attributes of the software, such as performance, security, usability, reliability, and maintainability. Non-functional testing answers questions like "How well does the software perform beyond its core functionality?"

Examples of non-functional testing techniques include:

- Performance Testing: Evaluating the responsiveness, scalability, and efficiency of the software under various load and stress conditions.
- Security Testing: Assessing the software's ability to resist unauthorized access, protect data, and withstand malicious attacks.
- Usability Testing: Evaluating the software's user-friendliness, ease of use, and overall user experience.
- Reliability Testing: Testing the software's stability, availability, and ability to recover from failures or errors.
- Compatibility Testing: Verifying that the software functions correctly across different platforms, devices, and configurations.

Non-functional testing is crucial to ensure that the software not only works correctly but also meets the user's expectations in terms of performance, security, usability, and other important aspects.

Advantages of software testing:

1. Improved software quality and reliability
2. Early identification and fixing of defects
3. Improved customer satisfaction
4. Increased stakeholder confidence
5. Reduced maintenance costs

Disadvantages of software testing:

1. Time-consuming and adds to project cost
2. Can slow down development process
3. Not all defects can be found
4. Can be difficult to fully test complex systems
5. Potential for human error during testing process