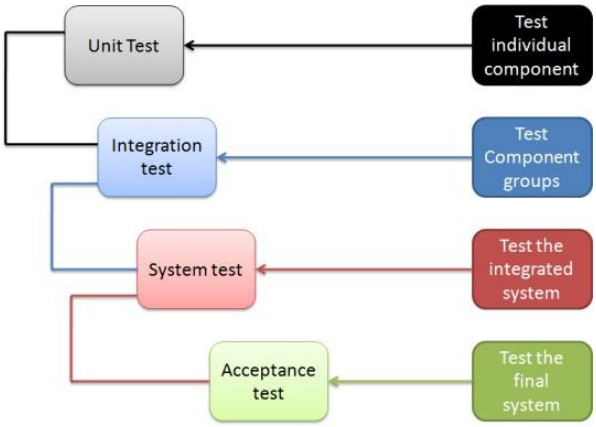
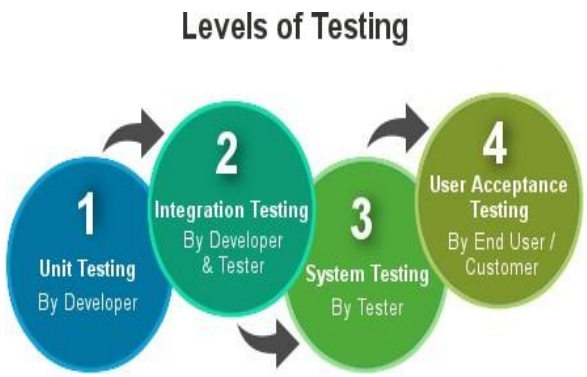
CO2 :	Prepare test cases for different types and levels of testing.

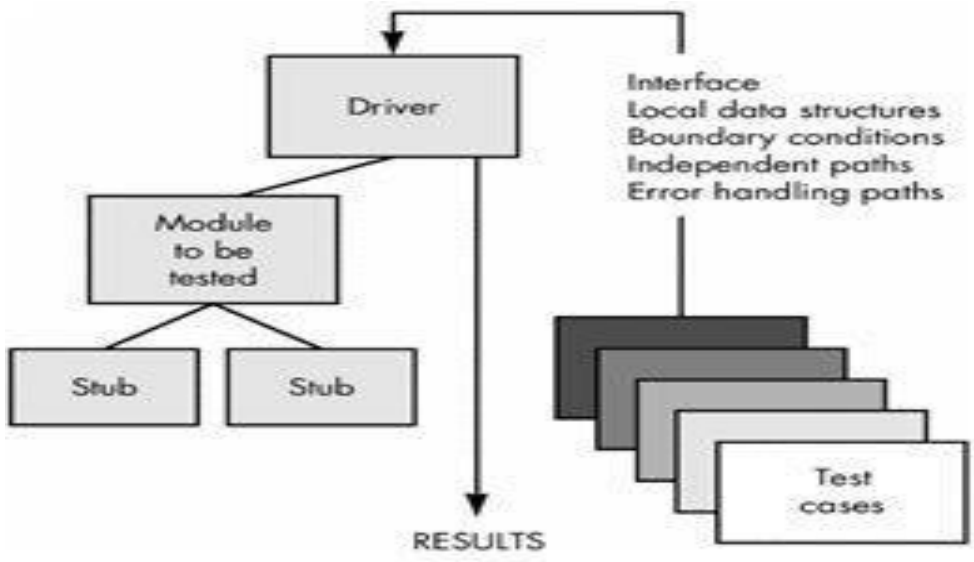
Levels of Testing





Unit Testing

- Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed.
- Unit Testing is the first level of testing and is performed prior to Integration Testing.
- A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output.
- It is executed by the Developer.
- Unit Testing is performed by using the White Box Testing method
- Example: - A function, method, Loop or statement in program is working fine.



Drivers

- Drivers are used in bottom-up integration testing approach.
- It can simulate the behavior of upper-level module that is not integrated yet.
- Drivers modules act as the temporary replacement of module and act as the actual products.
- Drivers are also used for interact with external system and usually complex than stubs.
- Driver: Calls the Module to be tested.

Now suppose you have modules B and C ready but module A which calls functions from module B and C is not ready so developer will write a dummy piece of code for module A which will return values to module B and C. This dummy piece of code is known as driver.

Stubs

- Stubs are used in top down integration testing.
- It can simulate the behavior of lower-level module that are not integrated.
- They are act as a temporary replacement of module and provide same output as actual product.
- When needs to intact with external system then also stubs are used.
- Stub: Is called by the Module under Test.

Assume you have 3 modules, Module A, Module B and module C. Module A is ready and we need to test it, but module A calls functions from Module B and C which are not ready, so developer will write a dummy module which simulates B and C and returns values to module A. This dummy module code is known as stub.

Importance of Stubs and Drivers

1. Stubs and Drivers works as a substitute for the missing or unavailable module.
2. They are specifically developed, for each module, having different functionalities.
3. Generally, developers and unit testers are involved in the development of stubs and drivers.
4. Their most common use may be seen in the integration incremental testing, where stubs are used in top bottom approach and drivers in a bottom up approach.

Benefits of Unit Testing

- Unit testing increases confidence in changing/ maintaining code. If good unit tests are written and if they are run every time any code is changed, we will be able to promptly catch any defects introduced due to the change.
- Codes are more reusable.
- Development is faster.
- The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels.
- Debugging is easy.

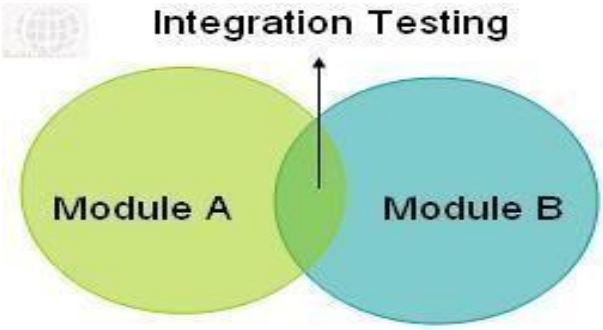
	Stub	Driver
Type	Dummy codes	Dummy codes
Description	Routines that don't actually do anything except declare themselves and the parameters they accept. The rest of the code can then take these parameters and use them as inputs	Routines that don't actually do anything except declare themselves and the parameters they accept. The rest of the code can then take these parameters and use them as inputs
Used in	Top Down Integration	Bottom Up Integration
Purpose	To allow testing of the upper levels of the code, when the lower levels of the code are not yet developed.	To allow testing of the lower levels of the code, when the upper levels of the code are not yet developed.

Integration Testing

- Integration Testing is a level of software testing where individual units are combined and tested as a group.
- In integration Testing, individual software modules are integrated logically and tested as a group.

- Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.

- As displayed in the image below when two different modules 'Module A' and 'Module B' are integrated then the integration testing is done.



1. Incremental Approach:

- In this approach, testing is done by joining two or more modules that are *logically related*. Then the other related modules are added and tested for the proper functioning. Process continues until all of the modules are joined and tested successfully.

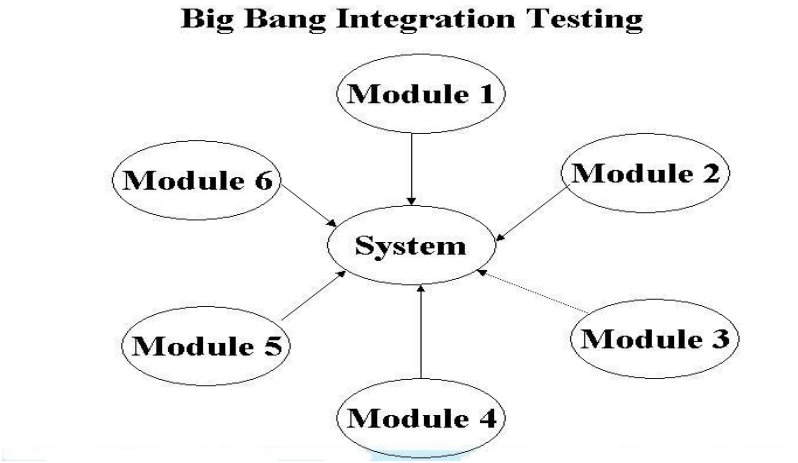
- This process is carried out by using dummy programs called Stubs and Drivers. Stubs and Drivers do not implement the entire programming logic of the software module but just simulate data communication with the calling module.

- Stub: Is called by the Module under Test.

- Driver: Calls the Module to be tested.

2. Non- Incremental Integration

- . The non-incremental approach is also known as ―Big-Bang‖ Testing.
- Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system.
- When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.
- In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole. As per the below image all the modules from 'Module 1' to 'Module 6' are integrated simultaneously then the testing is carried out.



3. Top down Integration

The strategy in top-down integration is look at the design hierarchy from top to bottom. Start with the high - level modules and move downward through the design hierarchy. In this approach testing is conducted from main module to sub module. If the sub module is not developed a temporary program called STUB is used for simulate the sub module. Modules subordinate to the top modules are integrated in the following two ways:

1. Depth first Integration: In this type, all modules on major control path of the design hierarchy are integrated first. In this example shown in fig. modules 1, 2, 4/5 will be integrated first. Next, modules 1, 3, 6 will be integrated.

2. Breadth first Integration: In this type, all modules directly subordinate at each level, moving across the design hierarchy horizontally, are integrated first. In the example shown in figure modules 2 and 3 will be integrated first. Next, modules 4,5 and 6 will be integrated .

Procedure:

The procedure for Top-Down integration process is discussed in the following steps:

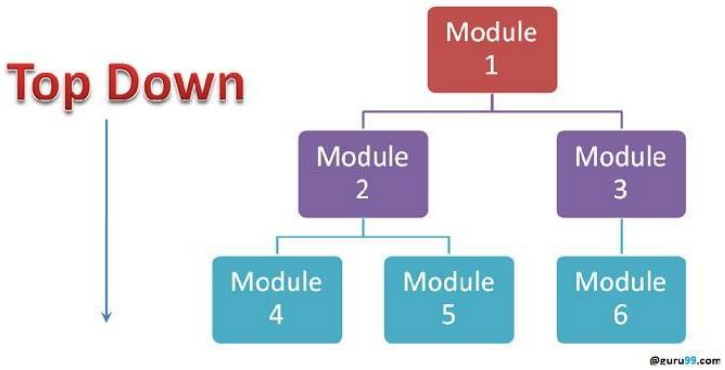
1. Start with the top or initial module in the software. Substitute the stubs for all the subordinate of the top module. Test the top module.
2. After testing the top module, stubs are replaced one at a time with the actual modules for integration.
3. Perform testing on this recent integrated environment.
4. Regression testing may be conducted to ensure that new errors have not appeared.
5. Repeat steps 2-4 for whole design hierarchy.

Advantages:

- Advantageous if major flaws occur toward the top of the program.
- Once the I/O functions are added, representation of test cases is easier.
- Early skeletal Program allows demonstrations and boosts morale.

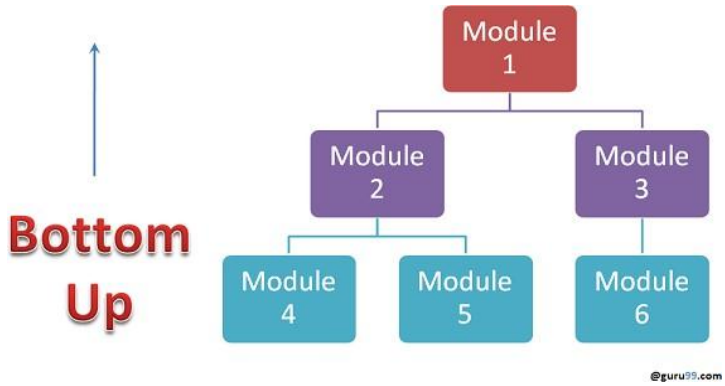
Disadvantages:

- Stub modules must be produced
- Stub Modules are often more complicated than they first appear to be.
- Before the I/O functions are added, representation of test cases in stubs can be difficult.
- Test conditions may be impossible, or very difficult, to create.
- Observation of test output is more difficult.
- Allows one to think that design and testing can be overlapped.
- Induces one to defer completion of the testing of certain modules.



4. Bottom up Integration

In this approach testing is conducted from sub module to main module, if the main module is not developed a temporary program called DRIVERS is used to simulate the main module.



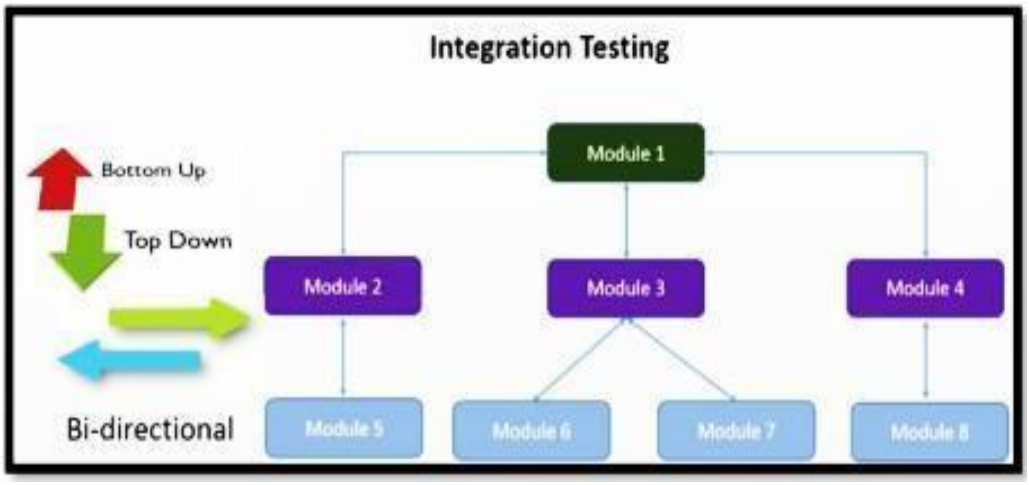
Advantages:

- Advantageous if major flaws occur toward the bottom of the program.
- Test conditions are easier to create.
- Observation of test results is easier.
- Driver Modules must be produced.
- The program as an entity does not exist until the last module is added

Disadvantages:

- Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
- Early prototype is not possible

5. Bi Direction / Sandwich Integration Testing



1. Bi-directional Integration is a kind of integration testing process that combines top-down and bottom-up testing.

2. With an experience in delivering Bi-directional testing projects custom software development services provide the best quality of the deliverables right from the development of software process.

3. Bi-directional Integration testing is a vertical incremental testing strategy that tests the bottom layers and top layers and tests the integrated system in the computer software development process.

4. Using stubs, it tests the user interface in isolation as well as tests the very lowest level functions using drivers.

5. Bi-directional Integration testing combines bottom-up and top-down testing.

6. Bottom-up testing is a process where lower level modules are integrated and then tested.

7. This process is repeated until the component of the top of the hierarchy is analyzed. It helps custom software development services find bugs easily without any problems.

8. Top down testing is a process where the top integrated modules are tested and the procedure is continued till the end of the related module.

9. Top down testing helps developers find the missing branch link easily

10. This technique is called as Sandwich Integration.

Advantages:
1. Sandwich approach is useful for very large projects having several subprojects.
2. Both Top-down and Bottom-up approach starts at a time as per development schedule.
3. Units are tested and brought together to make a system Integration is done downwards.

Disadvantages:
1. It require very high cost for testing because one part has Top-down approach while another part has bottom-up approach.
2. It cannot be used for smaller system with huge interdependence between different modules. It makes sense when the individual subsystem is as good as complete system.

Unit test	Integration test
• The idea behind Unit Testing is to test each part of the program and show that the individual parts are correct.	• The idea behind Integration Testing is to combine modules in the application and test as a group to see that they are working fine
• It is kind of White Box Testing	• It is kind of Black Box Testing
• It can be performed at any time	• It usually carried out after Unit Testing and before System Testing
• Unit Testing tests only the functionality of the units themselves and may not catch integration errors, or other system-wide issues	• Integrating testing may detect errors when modules are integrated to build the overall system
• It starts with the module specification	• It starts with the interface specification
• It pays attention to the behavior of single modules	• It pays attention to integration among modules
• Unit test does not verify whether your code works with external dependencies	• Integration tests verify that your code works with external dependencies

correctly.	correctly.
• It is usually executed by the developer	• It is usually executed by a test team
• Finding errors is easy	• Finding errors is difficult
• Maintenance of unit test is cheap	• Maintenance of integration test is expensive

Performance Testing

- Performance Testing is a type of testing to ensure software applications will perform well under their expected workload.
- A software application's performance like its response time, reliability, resource usage and scalability do matter.
- The goal of Performance Testing is not to find bugs but to eliminate performance bottlenecks.
- The focus of Performance Testing is checking a software program's
  - Speed - Determines whether the application responds quickly
  - Scalability - Determines maximum user load the software application can handle.
  - Stability - Determines if the application is stable under varying loads

Test objectives frequently include the following:

- **Response time.** For example, the product catalog must be displayed in less than 3 seconds.
- **Throughput.** For example, the system must support 100 transactions per second.
- **Resource utilization.** A frequently overlooked aspect is the amount of resources your application is consuming, in terms of processor, memory, disk input output (I/O), and network I/O.

1. **Load Testing**

- Load Testing is type of performance testing to check system with constantly increasing the load on the system until the time load is reaches to its threshold value.

- Here Increasing load means increasing number of concurrent users, transactions & check the behavior of application under test.
- It is normally carried out underneath controlled environment in order to distinguish between two different systems.
- The main purpose of load testing is to monitor the response time and staying power of application when system is performing well under heavy load.
- The successfully executed load testing is only if the specified test cases are executed without any error in allocated time.
- Load testing is testing the software under customer expected load.
- In order to perform load testing on the software you feed it all that it can handle. Operate the software with largest possible data files.
- If the software operates on peripherals such as printer, or communication ports, connect as many as you can.
- If you are testing an internet server that can handle thousands of simultaneous connections, do it. With most software it is important for it to run over long periods.
- Some software's should be able to run forever without being restarted. So Time acts as a important variable.
- Load testing can be best applied with the help of automation tools..

Simple examples of load testing:

- Testing printer by sending large job.
- Editing a very large document for testing of word processor
- Continuously reading and writing data into hard disk.
- Running multiple applications simultaneously on server.
- Testing of mail server by accessing thousands of mailboxes
- In case of zero-volume testing & system fed with zero load

2. Stress Testing

- Stress Testing is performance testing type to check the stability of software when hardware resources are not sufficient like CPU, memory, disk space etc.
- It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.
- Main parameters to focus during Stress testing are "Response Time" and "Throughput".
- Stress testing is Negative testing where we load the software with large number of concurrent users/processes which cannot be handled by the systems hardware resources. This testing is also known as **Fatigue testing**
- Stress testing is testing the software under less than ideal conditions. So subject your software to low memory, low disk space, slow cps, and slow modems and so on. Look at your software and determine what external resources and dependencies it has.

- Stress testing is simply limiting them to bare minimum. With stress testing you starve the software.
- For e.g. Word processor software running on your computer with all available memory and disk space, it works fine. But if the system runs low on resources you had a greater potential to expect a bug. Setting the values to zero or near zero will make the software execute different path as it attempt to handle the tight constraint. Ideally the software would run without crashing or losing data

3. Security Testing

- Security testing is a testing technique to determine if an information system protects data and maintains functionality as intended
- It also aims at verifying 6 basic principles as listed below:
  - Confidentiality
  - Integrity
  - Authentication
  - Authorization
  - Availability
  - Non-repudiation

❑ **Confidentiality**

A security measure which protects against the disclosure of information to parties other than the intended recipient is by no means the only way of ensuring the security.

❑ **Integrity**

Integrity of information refers to protecting information from being modified by unauthorized parties

❑ **Authentication**

This might involve confirming the identity of a person, tracing the origins of an artifact, ensuring that a product is what its packaging and labeling claims to be, or assuring that a computer program is a trusted one

❑ **Authorization**

The process of determining that a requester is allowed to receive a service or perform an operation. Access control is an example of authorization.

❑ **Availability**

Assuring information and communications services will be ready for use when expected. Information must be kept available to authorized persons when they need it.

❑ **Non-repudiation (acknowledgment)**

In reference to digital security, non-repudiation means to ensure that a transferred message has been sent and received by the parties claiming to have sent and received the message. Non-repudiation is a way to guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

Example :

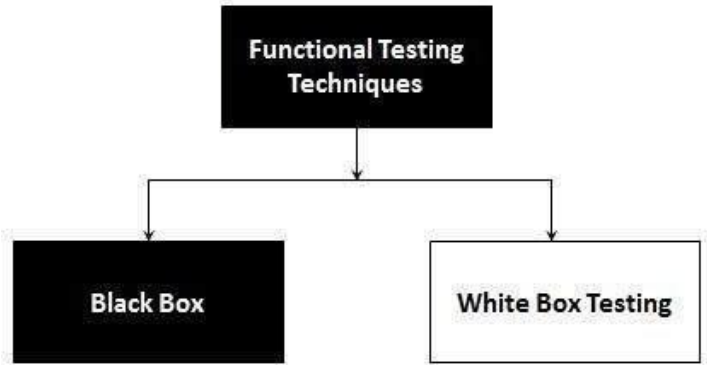
- A Student Management System is insecure if 'Admission' branch can edit the data of 'Exam' branch
- An ERP system is not secure if DEO (data entry operator) can generate 'Reports'
- An online Shopping Mall has no security if customer's Credit Card Detail is not encrypted
- A custom software possess inadequate security if an SQL query retrieves actual passwords of its users

4. Client Server Testing

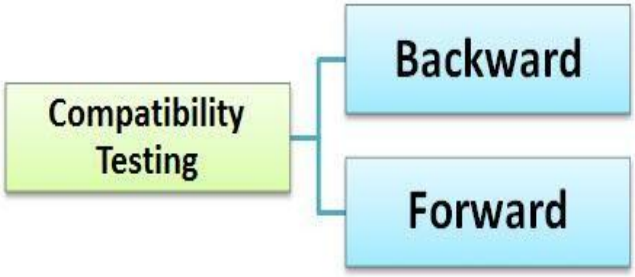
i) This type of testing usually done for 2 tier applications (usually developed for LAN) Here we will be having front-end and backend.
ii) The application launched on front-end will be having forms and reports which will be monitoring and manipulating data. E.g: applications developed in VB, VC++, Core Java, C, C++, D2K, Power Builder etc.
iii) The backend for these applications would be MS Access, SQL Server, Oracle, Sybase, MySQL, Quadbase.
iv) The tests performed on these types of applications would be– User interface testing Manual support testing– Functionality testing– Compatibility testing & configuration testing – Intersystem testing.

The approaches used for client server testing are
1. User interface testing: User interface testing, a testing technique used to identify the presence of defects is a product/software under test by using Graphical user interface [GUI].GUI Testing - Characteristics:
i) GUI is a hierarchical, graphical front end to the application, contains graphical objects with a set of properties.
ii) During execution, the values of the properties of each objects of a GUI define the GUI state.
iii)It has capabilities to exercise GUI events like key press/mouse click.
iv) Able to provide inputs to the GUI Objects.
v) To check the GUI representations to see if they are consistent with the expected ones.
vi) It strongly depends on the used technology.
2. Manual testing: Manual testing is a testing process that is carried out manually to find defects without the usage of tools or automation scripting. A test plan document is prepared that acts as a guide to the testing process to have the complete test coverage. Following are the testing techniques that are performed manually during the test life cycle are Acceptance Testing, White Box Testing, Black Box Testing, Unit Testing, System Testing, Integration Testing.
3. Functional testing: Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases.

There are two major Functional Testing techniques as shown below:



4. Compatibility testing: Compatibility testing is a non-functional testing conducted on the application to evaluate the application's compatibility within different environments. It can be of two types - forward compatibility testing and backward compatibility testing.



1. **Forward Compatibility Testing:** This type of testing verifies that the software is compatible with the newer or upcoming versions, and is thus named as forward compatible.
2. **Backward Compatibility Testing:** This type of testing helps to check whether the application designed using the latest version of an environment also works seamlessly in an older version. It is the testing performed to check the behavior of the hardware/software with the older versions of the hardware/software.

   Operating system Compatibility Testing - Linux , Mac OS, Windows
   Database Compatibility Testing - Oracle SQL Server
   Browser Compatibility Testing - IE , Chrome, Firefox
   Other System Software - Web server, networking/ messaging tool, etc.

Acceptance Testing

- **Acceptance Testing** is a level of the software testing where a system is tested for acceptability.

- The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.
- Usually, Black Box Testing method is used in Acceptance Testing.
- Acceptance Testing is performed after System Testing and before making the system available for actual use.
- The acceptance test cases are executed against the test data or using an acceptance test script and then the results are compared with the expected ones.
- The goal of acceptance **testing** is to establish confidence in the system.
- Acceptance testing is most often focused on a validation type testing.

Acceptance Criteria

Acceptance criteria are defined on the basis of the following attributes

- Functional Correctness and Completeness
- Data Integrity
- Data Conversion
- Usability
- Performance
- Timeliness
- Confidentiality and Availability
- Install ability and Upgradability
- Scalability
- Documentation

Types Of Acceptance Testing

- User Acceptance test
- Operational Acceptance test
- Contract Acceptance testing
- Compliance acceptance testing

- **User Acceptance test**

It focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user. The user acceptance test is performed by the users and application managers.

- **Operational Acceptance test**

Also known as Production acceptance test validates whether the system meets the requirements for operation. In most of the organization the operational acceptance test is performed by the system administration before the system is released. The operational acceptance test may include testing of backup/restore, disaster recovery, maintenance tasks and periodic check of security vulnerabilities.

- **Contract Acceptance testing**

It is performed against the contract's acceptance criteria for producing custom developed software. Acceptance should be formally defined when the contract is agreed.

- **Compliance acceptance testing**

It is also known as regulation acceptance testing is performed against the regulations which must be adhered to, such as governmental, legal or safety regulations.

Advantages Of Acceptance Testing

- The functions and features to be tested are known.
- The details of the tests are known and can be measured.
- The tests can be automated, which permits regression testing.
- The progress of the tests can be measured and monitored.
- The acceptability criteria are known.

Disadvantages Of Acceptance Testing

- Requires significant resources and planning.
- The tests may be a re-implementation of system tests.

It may not uncover subjective defects in the software, since you are only looking for defects you expect to find

Alpha Testing

- **Alpha Testing** is a type of testing conducted by a team of highly skilled testers at development site. Minor design changes can still be made as a result of alpha testing.
- For Alpha Testing there is a dedicated test team.
- Alpha testing is final testing before the software is released to the general public. It has two phases:
- In the **first phase** of alpha testing, the software is tested by in-house developers. They use either debugger software, or hardware-assisted debuggers. The goal is to catch bugs quickly.
- In the **second phase** of alpha testing, the software is handed over to the software QA staff, for additional testing in an environment that is similar to the intended use.
- **Pros Of Alpha Testing**

  • Helps to uncover bugs that were not found during previous testing activities
  • Better view of product usage and reliability
  • Analyze possible risks during and after launch of the product
  • Helps to be prepared for future customer support
  • Helps to build customer faith on the product
  • Maintenance Cost reduction as the bugs are identified and fixed before Beta / Production launch
  • Easy Test Management
- **Cons Of Alpha Testing**

- Not all the functionality of the product is expected to be tested
- Only Business requirements are scoped

Beta Testing

- **Beta Testing** is also known as field testing. It takes place at **customer's site**. It sends the system/software to users who install it and use it under real-world working conditions.
- A beta test is the second phase of **software testing** in which a sampling of the intended audience tries the product out
- The goal of beta testing is to place your application in the hands of real users outside of your own engineering team to discover any flaws or issues from the user's perspective that you would not want to have in your final, released version of the application.

Beta testing can be considered "pre-release testing.

Advantages of beta testing

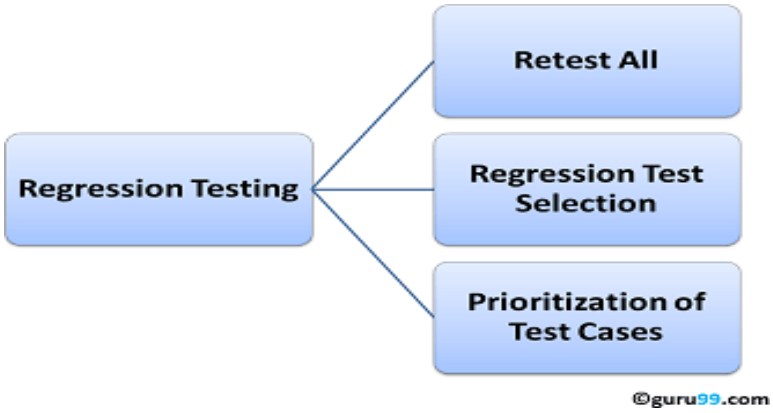
- You have the opportunity to get your application into the hands of users prior to releasing it to the general public.
- Users can install, test your application, and send feedback to you during this beta testing period.
- Your beta testers can discover issues with your application that you may have not noticed, such as confusing application flow, and even crashes.
- Using the feedback you get from these users, you can fix problems before it is released to the general public.
- The more issues you fix that solve real user problems, the higher the quality of your application when you release it to the general public.
- Having a higher-quality application when you release to the general public will increase customer satisfaction.
- These users, who are early adopters of your application, will generate excitement about your application.

Sr.No.	Alpha testing	Beta testing
1	Performed at developer's site	Performed at end user's site
2	Performed in controlled environment in developers presence	Performed in uncontrolled environment in developers absence
3	Less probability of finding errors as it is driven by developer	High probability of finding errors as it is used by end user.
4	It is done during implementation phase of software	It is done at the pre-release of the software
5	It is not considered as live application of software	It is considered as a live application of the software.
6	Less time consuming as developer can make necessary changes in given time	More time consuming as user has to report the bugs if any via appropriate channels.
7	Alpha testing involves both white box and black box testing	Beta testing typically uses black box testing only
8	Long execution cycles may be required for alpha testing	Only a few weeks of execution are required for beta testing

Special tests

1. Regression Testing

- Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
- Regression Testing is nothing but full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.
- This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that old code still works once the new code changes are done.
- Regression Testing is required when there is a
- ✓ Change in requirements and code is modified according to the requirement
- ✓ New feature is added to the software
- ✓ Defect fixing
- ✓ Performance issue fix



- **Retest All**

  This is one of the methods for Regression Testing in which all the tests in the existing test bucket or suite should be re-executed. This is very expensive as it requires huge time and resources.

- **Regression Test Selection**

  Instead of re-executing the entire test suite, it is better to select part of test suite to be run

  Test cases selected can be categorized as 1) Reusable Test Cases 2) Obsolete Test Cases.

  Re-usable Test cases can be used in succeeding regression cycles.

  Obsolete Test Cases can't be used in succeeding cycles.

- **Prioritization of Test Cases**

  Prioritize the test cases depending on business impact, critical & frequently used functionalities. Selection of test cases based on priority will greatly reduce the regression test suite.

Selecting test cases for regression testing

It was found from industry data that good number of the defects reported by customers were due to last minute bug fixes creating side effects and hence selecting the Test Case for regression testing is an art and not that easy.

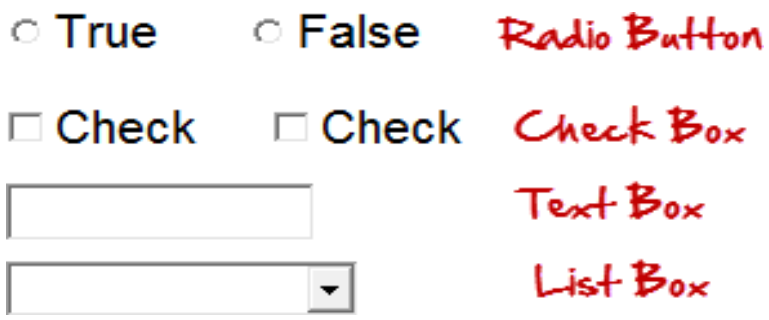
Effective Regression Tests can be done by selecting following test cases –

- ✓ Test cases which have frequent defects
- ✓ Functionalities which are more visible to the users
- ✓ Test cases which verify core features of the product
- ✓ Test cases of Functionalities which has undergone more and recent changes
- ✓ All Integration Test Cases
- ✓ All Complex Test Cases
- ✓ Boundary value test cases
- ✓ Sample of Successful test cases
- ✓ Sample of Failure test cases
- ✓ **Regression Testing Tools**
- ✓ If your software undergoes frequent changes, regression testing costs will escalate.

- ✓ In such cases, Manual execution of test cases increases test execution time as well as costs.
- ✓ Following are most important tools used for both functional and regression testing:
- ✓ **Selenium:** This is an open source tool used for automating web applications. Selenium can be used for browser based regression testing.
- ✓ **Quick Test Professional (QTP):** HP Quick Test Professional is automated software designed to automate functional and regression test cases. It uses VBScript language for automation. It is a Data driven, Keyword based tool.
- ✓ **Rational Functional Tester (RFT):** IBM's rational functional tester is a Java tool used to automate the test cases of software applications. This is primarily used for automating regression test cases and it also integrates with Rational Test Manager.

2. GUI Testing

- There are two types of interfaces for a computer application.
- Command Line Interface is where you type text and computer responds to that command.
- GUI stands for Graphical User Interface where you interact with the computer using images rather than text.
- GUI testing is the process of testing the system's Graphical User Interface of the Application Under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars - toolbar, menu bar, dialog boxes and windows, etc.
- GUI is what user sees. A user does not see the source code. The interface is visible to the user. Especially the focus is on the design structure, images that they are working properly or not.
- **GUI Testing Guidelines**
  1. Check Screen Validations
  2. Verify All Navigations
  3. Check usability Conditions
  4. Verify Data Integrity
  5. Verify the object states
  6. vi. Verify the date Field and Numeric Field Formats
- Following are the GUI elements which can be used for interaction between the user and application:



- In addition to functionality, GUI testing evaluates design elements such as layout, colors, fonts, font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links and content. GUI testing processes can be either manual or automatic, and are often performed by third -party companies, rather than developers or end users.

Example: Consider any website like MSBTE, Google, yahoo or any login form or GUI of any application to be tested. It includes following:

- All colors used for background, control colors, and font color have a major impact on users. Wrong color combinations and bright colors may increase fatigue of users.

- All words, Fonts, Alignments, scrolling pages up and down, navigations for different hyperlinks and pages, scrolling reduce usability.

- Error messages and information given to users must be usable to the user. Reports and outputs produced either on screen or printed should be readable. Also paper size on printer, font, size of screen should be consider.

- Screen layout in terms of number of instructions to users, number of controls and number of pages are defined in low level design. More controls on single page and more pages reduce usability.

- Types of control on a single page are very useful considering usability.

- Number of images on page or moving parts on screen may affect performance. These are high-priority defects. It has direct relationships with usability testing, look, and feels of an application. It affects emotions of users and can improve acceptability of an application.

Advantages of GUI Testing:

- Good GUI improves feel and look of the application; it psychologically accepts the application by the user.

- GUI represents a presentation layer of an application. Good GUI helps an application due to better experience of the users.

- Consistency of the screen layouts and designs improves usability of an application.

Disadvantages of GUI Testing:

- When number of pages is large and number of controls in a single page is huge.

- Special application testing like those made for blind people or kids below age of five may need special training for testers.

components of usability testing:-

Learnability
A usable product is easy to learn. In this busy and competitive world, no one has time to use a product which is difficult to understand.

Ease of use is the first important thing that a user feels in a product, so it needs to be a priority while making design decisions.

Efficiency
An efficient product is the one that makes it easier for a user to perform his tasks quickly and effectively. If this factor is missing in design, a beautifully designed product will also lose the trust of its users.

Keeping user's goal in mind in design stage helps to achieve this component.

Memorability
A good product is the one that do not require user to memorize it. Instead, user should be able to go through the layout and flow easily whenever he comes to use the product.

A design should meet human's psychological behavior and expectations so that he may not put extra efforts to remember its details.

Error Tolerance
A product's usability can easily be measured by the way its response to human errors. A usable product tries to resolve errors seamlessly.

If an error requires user's input, then design should handle it gracefully by providing meaningful and polite error messages. This principle also needs to consider during each stage of design process.

Satisfaction
This usability component requires to provide a pleasant and satisfactory user experience. If a user comes to use a product, he should be able to accomplish his goals in a way that he wants to come back again to the same product.

Making user satisfaction a priority in design process helps to build a good relationship between user and product.

COMPARITIBILITY TESTING:-

Checking the functionality of an application on different software, hardware platforms, network, and browsers is known as compatibility testing. Once the application is stable, we moved it to the production, it may be used or accessed by multiple users on the different platforms, s and they may face some compatibility issues, to avoid these issues, we do one round of compatibility testing.