Testing Life Cycle

**Re-Testing & Regression Testing**

**Re-Testing**: After a defect is detected and fixed, the software should be retested to confirm that the original defect has been successfully removed. This is called Confirmation Testing or Re-Testing

**Regression testing**:  Testing your software application when it undergoes a code change to ensure that the new code has not affected other parts of the software.

**Retesting:**

In this type of testing, the tester re tests the application which was earlier reported as a bug and now fixed by the developer. This bug can be due to functionality issues as well as design issues. Retesting the functionality of a button which was previously not working or testing the alignment issues of a page etc can be listed as examples for retesting.

Regression testing is done to test whether the changes made on the site have any impact on other modules which were earlier working fine. New requirement change, implementation of new functionality are some of the scenarios where regression testing is effective. Regression testing can be automated using effective automated tools such as Selenium, Quick Test Professional etc.

**Regression testing:**

Refer the given example for further understanding:

For example, we have a image uploading field which supports only certain image formats say JPG, PNG etc. Suppose one of the image formats failed (say JPG) and the developer fixed it. Testing the functionality, after fixing this bug is retesting. Here, testing and ensuring whether this new change (fixes) has not altered the working of other fields on that page is regression testing.

In normal cases, regression testing is carried out only after retesting of the application. But there are exceptions where both testing are carried out simultaneously. While testing a website, make sure that the application have undergone both retesting and regression testing.

### Smoke Vs Sanity Testing

Smoke and Sanity testing are the most misunderstood topics in Software Testing. There is enormous amount of literature on the subject, but most of them are confusing. The following article makes an attempt to address the confusion.

## what is a Software Build?

 If you are developing a simple computer program which consists of only one source code file, you merely need to compile and link this one file, to produce an executable file. This process is very simple.

Usually this is not the case. A typical Software Project consists of hundreds or even thousands of source code files. Creating an executable program from these source files is a complicated and time-consuming task.

You need to use "build" software to create an executable program and the process is called "***Software Build***"

**what is Smoke Testing?**

Smoke Testing is performed after software build to **ascertain that the critical functionalities of the program is working fine**.It is executed **"before"** any detailed functional or regression tests are executed on the software build.The **purpose is to reject a badly broken application**, so that the QA team does not waste time installing and testing the software application.

In Smoke Testing, the **test cases chosen cover the most important functionality** or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system is working fine.

For Example a typical smoke test would be - Verify that the application launches successfully, Check that the GUI is responsive ... etc.

## what is Sanity Testing?

After receiving a **software build, with minor changes in code, or functionality, Sanity testing is performed to ascertain that the bugs have been fixed and no further issues are introduced due to these changes**.The goal is to determine that the proposed functionality works roughly as expected. **If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing**.

The **objective is "not" to verify thoroughly the new functionality**, but to determine that the developer has applied some rationality (sanity) while producing the software. For instance, if your scientific calculator gives the result of 2 + 2 =5! Then, there is no point testing the advanced functionalities like sin 30 + cos 50.

### What is Exploratory testing in software testing?

* As its name implies, exploratory testing is about exploring, finding out about the software, what it does, what it doesn’t do, what works and what doesn’t work. The tester is constantly making decisions about what to test next and where to spend the (limited) time. This is an approach that is most useful when there are no or poor specifications and when time is severely limited.
* Exploratory testingis a hands-on approach in which testers are involved in minimum planning and maximum test execution.
* The planning involves the creation of a test charter, a short declaration of the scope of a short (1 to 2 hour) time-boxed test effort, the objectives and possible approaches to be used.
* The test design and test execution activities are performed in parallel typically without formally documenting the test conditions, test cases or test scripts. This does not mean that other, more formal testing techniques will not be used. For example, the tester may decide to us boundary value analysis but will think through and test the most important boundary values without necessarily writing them down. Some notes will be written during the exploratory-testing session, so that a report can be produced afterwards.
* Test logging is undertaken as test execution is performed, documenting the key aspects of what is tested, any defects found and any thoughts about possible further testing.
* It can also serve to complement other, more formal testing, helping to establish greater confidence in the software. **In** this way, exploratory testing can be used as a check on the formal test process by helping to ensure that the most serious defects have been found.

**Exploratory testing -**

* Is not random testing but it is adhoc testing with purpose of find bugs
* Is structured and rigorous
* Is cognitively (thinking) structured as compared to procedural structure of scripted testing. This structure comes from Charter, time boxing etc.
* Is highly teachable and manageable
* Is not a technique but it is an approach. What actions you perform next is governed by what you are doing currently

**Exploratory Test Preparation:**

Exploratory test preparation goes through following 5 stages detailed below and it is also called session based test management (SBTM Cycle):

1. Create a Bug Taxonomy (classification )
   * Categorize common types of faults found in the past projects
   * Analyze the root cause analysis of the problems or faults
   * Find the risks and develop ideas to test the application.
2. Test Charter
   * Test Charter should suggest
     1. what to test
     2. how it can be tested
     3. What needs to be looked
   * Test ideas are the starting point of exploration testing
   * Test charter helps determine how the end user could use the system
3. Time Box
   * This method includes pair of testers working together not less than 90 minutes
   * There should not be any interrupted time in those 90 minutes session
   * Time box can be extended or reduced by 45 minutes
   * This session encourages testers to react on the response from the system and prepare for the correct outcome
4. Review Results:
   * Evaluation of the defects
   * Learning from the testing
   * Analysis of coverage areas
5. Debriefing:
   * Compilation of the output results
   * Compare the results with the charter
   * Check whether any additional testing is needed

During e**xploratory execution**, following needs to be done:

* Mission of testing should be very clear
* Keep notes on what needs to be tested, why it needs to be tested and the assessment of the product quality
* Tracking of questions and issues raised during exploratory testing
* Better to pair up the testers for effective testing
* The more we test, more likely to execute right test cases for the required scenarios

It is very important to take document and monitor the following

* Test Coverage - Whether we have taken notes on the coverage of test cases and improve the quality of the software
* Risks - Which risks needs to be covered and which are all important ones?
* Test Execution Log - Recordings on the test execution
* Issues / Queries - Take notes on the question and issues on the system

### Adhoc Testing

## Ad hoc Testing**, also known as Random Testing or Monkey Testing, is a method of software testingwithout any planning and documentation. The tests are conducted informally and randomly without any formal expected results.**

Adhoc testing is an informal testing type with an aim to break the system. This testing is usually an unplanned activity . It does not follow any test design techniques to create test cases. In fact is does not create test cases altogether! This testing is primarily performed if the knowledge of testers in the system under test is very high. Testers randomly test the application without any test cases or any business requirement document.

 Adhoc Testing does not follow any structured way of testing and it is randomly done on any part of application. Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the testing technique called **Error Guessing.** Error guessing can be done by the people having enough experience on the system to "guess" the most likely source of errors.

 This testing requires no documentation/ planning /process to be followed. Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. Hence, sometimes, it is very difficult to reproduce the defects as there are no test-steps or requirements mapped to it.

**When execute Adhoc Testing?**

Adhoc testing can be performed when there is limited time to do elaborative testing. Usually adhoc testing is performed after the formal test execution. And if time permits, adhoc testing can be done on the system).Adhoc testing will be effective only if the tester is knowledgeable of the System Under Test.

**Types of adhoc testing**

There are different types of Adhoc testing and they are listed as below:

|  |  |
| --- | --- |
| **Buddy Testing** | Two buddies mutually work on identifying defects in the same module. Mostly one buddy will be from development team and another person will be from testing team. Buddy testing helps the testers develop better test cases and development team can also make design changes early. This testing usually happens after unit testing completion. |
| **Pair testing** | Two testers are assigned modules, share ideas and work on the same machines to find defects. One person can execute the tests and another person can take notes on the findings. Roles of the persons can be a tester and scriber during testing.  **Comparison Buddy and Pair Testing**  Buddy testing is combination of unit and system testing together with developers and testers but Pair testing is done only with the testers with different knowledge levels.(Experienced and non-experienced to share their ideas and views) |
| **Monkey Testing** | Randomly test the product or application without test cases **with a goal to**break the system. |

**Best practices of Adhoc testing**

 Following best pratices can ensure effective Adhoc Testing.

### Good business knowledge

Testers should have good knowledge of the business and clear understanding of the requirements- Detailed knowledge of the end to end business process will help find defects easily. Experienced testers find more defects as they are better at error guessing.

### Test Key Modules

Key business modules should be identified and targeted for adhoc testing.. Business critical modules should be tested first to gain confidence on the quality of the system.

### Record Defects

All defects need to be recorded or written in a notepad . Defects must be assigned to developers for fixing. For each valid defect ,corresponding test cases must be written & must be added to planned test cases.

These defect findings should be made as lesson learned and these should be reflected in our next system while we are planning for test cases.

[[https://resources.blogblog.com/img/icon18_email.gif](https://www.blogger.com/email-post.g?blogID=196205546010114473&postID=4188577277735675427)](https://www.blogger.com/email-post.g?blogID=196205546010114473&postID=4188577277735675427)

[Email This](https://www.blogger.com/share-post.g?blogID=196205546010114473&postID=4188577277735675427&target=email)[BlogThis!](https://www.blogger.com/share-post.g?blogID=196205546010114473&postID=4188577277735675427&target=blog)[Share to Twitter](https://www.blogger.com/share-post.g?blogID=196205546010114473&postID=4188577277735675427&target=twitter)[Share to Facebook](https://www.blogger.com/share-post.g?blogID=196205546010114473&postID=4188577277735675427&target=facebook)[Share to Pinterest](https://www.blogger.com/share-post.g?blogID=196205546010114473&postID=4188577277735675427&target=pinterest)

### User Acceptance Testing (UAT)

##### What is User Acceptance Testing?

**User Acceptance testing** is the software testing process where system tested for acceptability & validates the end to end business flow. Such type of testing executed by client in separate environment (similar to production environment) & confirm whether system meets the requirements as per requirement specification or not.

UAT is performed after System Testing is done and all or most of the major defects have been fixed. This testing is to be conducted in the final stage of Software Development Life Cycle (SDLC) prior to system being delivered to a live environment. UAT users or end users are concentrating on end to end scenarios & typically involves running a suite of tests on the completed system.

The **Acceptance testing** is “black box” tests, means UAT users doesn’t aware of internal structure of the code, they just specify the input to the system & check whether systems respond with correct result.

**User Acceptance testing** also known as Customer Acceptance testing (CAT), if the system is being built or developed by an external supplier. The CAT or UAT are the final confirmation from the client before the system is ready for production. The business customers are the primary owners of these UAT tests. These tests are created by business customers and articulated in business domain languages. So ideally it is collaboration between business customers, business analysts, testers and developers. It consists of test suites which involve multiple test cases & each test case contains input data (if required) as well as the expected output. The result of test case is either a pass or fail.

Acceptance Testing Definition:

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

##### Prerequisites of User Acceptance Testing:

Prior to start the UAT following checkpoints to be considered:

* The Business Requirements should be available.
* The development of software application should be completed & different levels of testing like Unit Testing, Integration Testing & System Testing is completed.
* All High Severity, High Priority defects should be verified. No any Showstoppers defects in the system.
* Check if all reported defects should be verified prior to UAT starts.
* Check if Traceability matrix for all testing should be completed.
* Before UAT starts error like cosmetic error are acceptable but should be reported.
* After fixing all the defects regression Testing should be carried out to check fixing of defect not breaking the other working area.
* The separate UAT environment similar to production should be ready to start UAT.
* The Sign off should be given by System testing team which says that Software application ready for UAT execution.

##### TYPES OF ACCEPTANCE TEST

**Why Alpha and Beta Testing?**

The software application may use different users on different way & it impossible to developer or tester to predict what all possible scenarios or test data end user will use & how customer actually use the software application. So most of software venders are use the term like Alpha testing and Beta Testing which help to uncover the errors that may occurs in the actual test environment. In this testing method the software application release over limited end users rather than testing professionals to get feedback from them.

**A] Alpha Testing**

Alpha testing is conducted by Customer at the developer’s site, it is performed by potential users like developer, end users or organization users before it is released to external customers & report the defects found while Alpha testing.

This software product testing is not final version of software application, after fixing all reported bug (after bug triage) the new version of software application will release. Sometimes the Alpha Testing is carried out by client or an outsider with the attendance of developer and tester. The version of the release on which Alpha testing is perform is called “**Alpha Release**”.

**B] Beta Testing**

Most if times we have the sense of hearing term “Beta release/version”, so it is linked to Beta Testing.

Basically the beta testing is to be carried out without any help of developers at the end user’s site by the end users &, so it is performed under uncontrolled environment. Beta testing is also known as Field testing. This is used to get feedback from the market.

This testing is conducted by limited users & all issues found during this testing are reported on continuous basis which helps to improve the system. Developers are taking actions on all issues reported in beta testing after bug triage & then the software application is ready for the final release. The version release after beta testing is called “**Beta Release**“.

This is last stage of the testing where product is sent outside the company or for trial offer to download.

##### What to Test in User Acceptance Testing?

* Based on the Requirements definition stage use cases the Test cases are created.
* Also the Test cases are created considering the real world scenarios for the application.
* The actual testing is to be carried out in environments that copy of the production environment. So in the type of testing is concentrating on the exact real world use of application.
* Test cases are designed such that all area of application is covered during testing to ensure that an effective User Acceptance Testing.

##### What are the key deliverable of User Acceptance Testing?

The completion of User Acceptance Testing is the significant milestone for traditional testing method. The following key deliverable of User Acceptance Testing phase:

* **Test Plan**: This outlines the Testing Strategy
* **UAT Test cases**: The Test cases help the team to effectively test the application in UAT environment.
* **Test Results and Error Reports**: This is a log of all the test cases executed and the actual results.
* **User Acceptance Sign-off**: This is the system, documentation, and training materials have passed all tests within acceptable margins.
* **Installation Instructions**: This is document which helps to install the system in production environment.
* **Documentation Materials**: Tested and updated user documentation and training materials are finalized during user acceptance testing

### Gray Box Testing

## **Gray Box Testing is a software testing method which is a combination of Black Box Testing method and**[**White Box Testing**](http://softwaretestingfundamentals.com/white-box-testing/)**method. In Black Box Testing, the internal structure of the item being tested is unknown to the tester and in White Box Testing the internal structure in known. In Gray Box Testing, the internal structure is partially known. This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level.**

Gray Box Testing is named so because the software program, in the eyes of the tester is like a gray/ semi-transparent box; inside which one can partially see.  
 **EXAMPLE**

An example of Gray Box Testing would be when the codes for two units/ modules are studied (White Box Testing method) for designing test cases and actual tests are conducted using the exposed interfaces (Black Box Testing method).

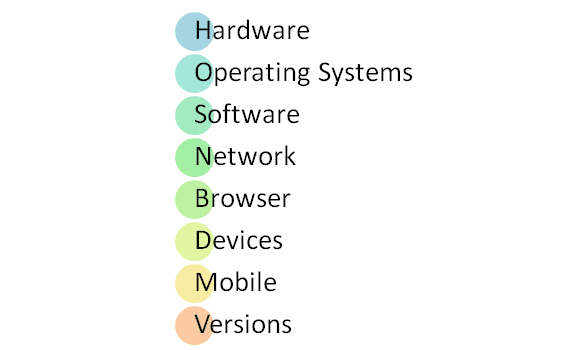
**LEVELS APPLICABLE TO**

Though Gray Box Testing method may be used in other levels of testing, it is primarily useful in Integration Testing.

### What is Compatibility testing in software testing?

* It is a type of non-functional testing.
* Compatibility testing is a type of software testing used to ensure compatibility of the system/application/website built with various other objects such as other web browsers, hardware platforms, users (in case if it’s very specific type of requirement, such as a user who speaks and can read only a particular language), operating systems etc. This type of testing helps find out how well a system performs in a particular environment that includes hardware, network, operating system and other software etc.
* It is basically the testing of the application or the product built with the computing environment.
* It tests whether the application or the software product built is compatible with the hardware, operating system, database or other system software or not.

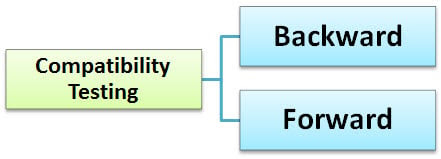
#### Types of compatibility tests

[](http://www.guru99.com/images/c2.png)

**Compatibility testing types**

* **Hardware** : It checks software to be compatible with different hardware configurations .
* **Operating Systems**: It checks your software to be compatible with different Operating Systems like Windows , Unix , Mac OS etc.
* **Software**: It checks your developed software to be compatible with other software's.For example: MS Word application should be compatible with other softwares like MS Outlook,MS Excel , VBA etc.
* **Network:**Evaluation of performance of system In network with varying parameters such as Bandwidth, Operating speed, Capacity. It also checks application  in different networks with all parameters mentioned earlier.
* **Browser**: It checks compatibility of your website with different browsers like Firefox , Google Chrome , Internet Explorer etc.
* **Devices** : It checks compatibility of your software with different devices like USB port Devices, Printers and Scanners, Other media devices and Blue tooth.
* **Mobile**: Checking you software is compatible with mobile platforms like Android , iOS etc.
* **Versions of the software:**It is verifying you software application to be compatible with different versions of software.For instance checking your Microsoft Word to be compatible with Windows 7, Windows 7 SP1 , Windows 7 SP 2 , Windows 7 SP 3.

**There are two types of version checking**



**Backward compatibility Testing**is to verify the behavior of the developed hardware/software with the **older** **versions** of the hardware/software.

**Forward compatibility Testing**is to verify the behavior of the developed hardware/software with the **newer** **versions** of the hardware/software.

**Tools for compatibility testing**

**Adobe Browser Lab - Browser Compatibility Testing:**

This tool helps us to check your application in different browsers.

**Secure Platform - Hardware Compatibility tool**

This tools includes necessary drivers for a specific hardware platform and it provides information on tool to check for CD burning process with CD burning tools.

**Virtual Desktops - Operating System Compatibility**:

This is used to run the applications in multiple operating systems as virtual machines. N Number of systems can be connected and compare the results.

**Compatibility testing process**

1. Initial phase of compatibility testing is to define the set of environments or platforms the application is expected to work on.
2. Tester should have enough knowledge on the platforms / software / hardware to understand the expected application behavior under different configurations.
3. Environment needs to be set-up for testing with different platforms, devices, networks to check whether your application runs well under different configurations.
4. Report the bugs .Fix the defects. Re-test to confirm defect fixing.

**What is Security testing in software testing?**

* It is a type of non-functional testing.
* Security testing is basically a type of software testing that’s done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization.
* It is a process to determine that an information system protects data and maintains functionality as intended.
* The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software’s and hardware’s and firewall etc.
* Software security is about making software behave in the presence of a malicious attack.
* The six basic security concepts that need to be covered by security testing are: confidentiality, integrity, authentication, availability, authorization and non-repudiation.

**What is Recovery testing in software?**

* It is a type of non-functional testing.
* Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc.
* Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed. For example, when an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application’s ability to continue receiving data from the point at which the network connection was broken. Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.

[[https://resources.blogblog.com/img/icon18_email.gif](https://www.blogger.com/email-post.g?blogID=196205546010114473&postID=7415272366177989263)](https://www.blogger.com/email-post.g?blogID=196205546010114473&postID=7415272366177989263)

[Email This](https://www.blogger.com/share-post.g?blogID=196205546010114473&postID=7415272366177989263&target=email)[BlogThis!](https://www.blogger.com/share-post.g?blogID=196205546010114473&postID=7415272366177989263&target=blog)[Share to Twitter](https://www.blogger.com/share-post.g?blogID=196205546010114473&postID=7415272366177989263&target=twitter)[Share to Facebook](https://www.blogger.com/share-post.g?blogID=196205546010114473&postID=7415272366177989263&target=facebook)[Share to Pinterest](https://www.blogger.com/share-post.g?blogID=196205546010114473&postID=7415272366177989263&target=pinterest)

**What is Performance testing in software?**

* It is  a type of non-functional testing.
* Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload.
* It can serve different purposes like it can demonstrate that the system meets performance criteria.
* It can compare two systems to find which performs better. Or it can measure what part of the system or workload causes the system to perform badly.
* This process can involve quantitative tests done in a lab, such as measuring the response time or the number of MIPS (millions of instructions per second) at which a system functions.
* Very simply we can put the difference between Volume, Load and stress testing as:

Volume Testing = Large amounts of data  
Load Testing = Large amount of users  
Stress Testing = Too many users, too much data, too little time and too little room.

**What is Non-functional testing ?**

In non-functional testing the quality characteristics of the component or system is tested. Non-functional refers to aspects of the software that may not be related to a specific function or user action such as scalability or security. Eg. How many people can log in at once? Non-functional testing is also performed at all levels like functional testing.

**Non-functional testing includes:**

* Functionality testing
* Reliability testing
* Usability testing
* Efficiency testing
* Maintainability testing
* Portability testing
* Baseline testing
* Compliance testing
* Documentation testing
* Endurance testing
* Load testing
* Performance testing
* Compatibility testing
* Security testing
* Scalability testing
* Volume testing
* Stress testing
* Recovery testing
* Internationalization testing and Localization testing
  + **Functionality testing:**Functionality testing is performed to verify that a software application performs and functions correctly according to design specifications. During functionality testing we check the core application functions, text input, menu functions and installation and setup on localized machines, etc.
  + **Reliability testing:**Reliability Testing is about exercising an application so that failures are discovered and removed before the system is deployed. The purpose of reliability testing is to determine product reliability, and to determine whether the software meets the customer’s reliability requirements.
  + **Usability testing:**In usability testing basically the testers tests the ease with which the user interfaces can be used. It tests that whether the application or the product built is user-friendly or not.

 Usability testing includes the following five components:

* + 1. **Learnability:**How easy is it for users to accomplish basic tasks the first time they encounter the design?
    2. **Efficiency:** How fast can experienced users accomplish tasks?
    3. **Memorability:** When users return to the design after a period of not using it, does the user remember enough to use it effectively the next time, or does the user have to start over again learning everything?
    4. **Errors:** How many errors do users make, how severe are these errors and how easily can they recover from the errors?
    5. **Satisfaction:** How much does the user like using the system?
  1. **Efficiency testing:**Efficiency testing test the amount of code and testing resources required by a program to perform a particular function. Software Test Efficiency is number of test cases executed divided by unit of time (generally per hour).
  2. **Maintainability testing:**It basically defines that how easy it is to maintain the system. This means that how easy it is to analyze, change and test the application or product.
  3. **Portability testing:**It refers to the process of testing the ease with which a computer software component or application can be moved from one environment to another, e.g. moving of any application from Windows 2000 to Windows XP. This is usually measured in terms of the maximum amount of effort permitted. Results are measured in terms of the time required to move the software and complete the and documentation updates.
  4. **Baseline testing:**It refers to the validation of documents and specifications on which test cases would be designed. The requirement specification validation is baseline testing.
  5. **Compliance testing:**It is related with the IT standards followed by the company and it is the testing done to find the deviations from the company prescribed standards.
  6. **Documentation testing:**As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.
  7. **Endurance testing:**Endurance testing involves testing a system with a significant load extended over a significant period of time, to discover how the system behaves under sustained use. For example, in software testing, a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.
  8. **Load testing:**A load test is usually conducted to understand the behavior of the application under a specific expected load. Load testing is performed to determine a system’s behavior under both normal and at peak conditions. It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation. E.g. If the number of users are in creased then how much CPU, memory will be consumed, what is the network and bandwidth response time
  9. **Performance testing:**Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload. It can serve different purposes like it can demonstrate that the system meets performance criteria. It can compare two systems to find which performs better. Or it can measure what part of the system or workload causes the system to perform badly.
  10. **Compatibility testing:**Compatibility testing is basically the testing of the application or the product built with the computing environment. It tests whether the application or the software product built is compatible with the hardware, operating system, database or other system software or not.
  11. **Security testing:**Security testing is basically to check that whether the application or the product is secured or not. Can anyone came tomorrow and hack the system or login the application without any authorization. It is a process to determine that an information system protects data and maintains functionality as intended.
  12. **Scalability testing:**It is the testing of a software application for measuring its capability to scale up in terms of any of its non-functional capability like load supported, the number of transactions, the data volume etc.
  13. **Volume testing:**Volume testing refers to testing a software application or the product with a certain amount of data. E.g., if we want to volume test our application with a specific database size, we need to expand our database to that size and then test the application’s performance on it.
  14. **Stress testing:**It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. It is a form of testing that is used to determine the stability of a given system. It  put  greater emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behavior under normal circumstances. The goals of such tests may be to ensure the software does not crash in conditions of insufficient computational resources (such as memory or disk space).
  15. **Recovery testing:**Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc. Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed. For example, when an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application’s ability to continue receiving data from the point at which the network connection got disappeared. Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.
  16. **Internationalization testing and Localization testing:**Internationalization is a process of designing a software application so that it can be adapted to various languages and regions without any changes. Whereas Localization is a process of adapting internationalized software for a specific region or language by adding local specific components and translating text.

### Cookies & Sessions testing

**COOKIE:**

A cookie keep information in the user’s browser until it is deleted from there. We can store almost anything in browser cookies. The problem with cookies are that anyone can block or delete cookie content at any time, so if any application is dependent on browser cookie then that application would not work if cookie is deleted or blocked.

**SESSION:**

Sessions are implemented by using cookies, the actual data is not in the browser; it is stored in the user's session record on the server. They work instead like a token allowing access and passing information while the user has their browser open. The main problem with session is that as soon as we close the browser we lose the session and all the information get lost.

**Key Differences Between Cookie And Session:**

|  |  |
| --- | --- |
| **Cookie** | **Session** |
| Cookies are stored in user’s browser. | Sessions are not stored in user’s browser. |
| Cookie data are stored at client side. | Session data are stored at server side. |
| Cookie data are easy to modify as they are stored at client side. | Session data are not easy to modify as they are stored at server side. |
| Cookie data is available in our browser up to expiration date. | Session data is available for the browser run. After closing the browser we lose the session information. |
| Cookies saves local user data as website specific on local computer on text file. | Session is application specific and keeps information until the browser is open. |

**Example of Cookies:** If we open any e-commerce site the purchase few items then those items are stored in cookies. Any time if we again go to that site and log in then it will show our user name and those items will be shown in that site’s cart.

**Example of Sessions:** When we log in as a member in any website it creates sessions until we log out of the website.

**Why Cookies are used?**  
Cookies are nothing but the user’s identity and used to track where the user navigated throughout the web site pages. The communication between web browser and web server is stateless.

For example if you are accessing domain http://www.example.com/1.html then web browser will simply query to example.com web server for the page 1.html. Next time if you type page as http://www.example.com/2.html then new request is send to example.com web server for sending 2.html page and web server don’t know anything about to whom the previous page 1.html served.

What if you want the previous history of this user communication with the web server? You need to maintain the user state and interaction between web browser and web server somewhere. This is where cookie comes into picture. Cookies serve the purpose of maintaining the user interactions with web server.

**How cookies work?**  
The HTTP protocol used to exchange information files on the web is used to maintain the cookies. There are two types of HTTP protocol. Stateless HTTP and Stateful HTTP protocol. Stateless HTTP protocol does not keep any record of previously accessed web page history. While Stateful HTTP protocol do keep some history of previous web browser and web server interactions and this protocol is used by cookies to maintain the user interactions.

Whenever user visits the site or page that is using cookie, small code inside that HTML page (Generally a call to some language script to write the cookie like cookies in JAVAScript, PHP, Perl) writes a text file on users machine called cookie.  
Here is one example of the code that is used to write cookie and can be placed inside any HTML page:

Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN\_NAME;

When user visits the same page or domain later time this cookie is read from disk and used to identify the second visit of the same user on that domain. Expiration time is set while writing the cookie. This time is decided by the application that is going to use the cookie.

Generally two types of cookies are written on user machine.

**1) Session cookies:** This cookie is active till the browser that invoked the cookie is open. When we close the browser this session cookie gets deleted. Some time session of say 20 minutes can be set to expire the cookie.  
**2) Persistent cookies:** The cookies that are written permanently on user machine and lasts for months or years.

**Where cookies are stored?**  
When any web page application writes cookie it get saved in a text file on user hard disk drive. The path where the cookies get stored depends on the browser. Different browsers store cookie in different paths. E.g. Internet explorer store cookies on path **“C:\Documents and Settings\Default User\Cookies”**  
Here the “Default User” can be replaced by the current user you logged in as. Like “Administrator”, or user name like “Vijay” etc.  
The cookie path can be easily found by navigating through the browser options. In Mozilla Firefox browser you can even see the cookies in browser options itself. Open the Mozila browser, click on Tools->Options->Privacy and then “Show cookies” button.

**How cookies are stored?**  
Lets take example of cookie written by rediff.com on Mozilla Firefox browser:  
On Mozilla Firefox browser when you open the page rediff.com or login to your rediffmail account, a cookie will get written on your Hard disk. To view this cookie simply click on “Show cookies” button mentioned on above path. Click on Rediff.com site under this cookie list. You can see different cookies written by rediff domain with different names.

**Site:** Rediff.com Cookie name: RMID  
**Name:**RMID (Name of the cookie)  
**Content:** 1d11c8ec44bf49e0… (Encrypted content)  
**Domain:** .rediff.com  
**Path:** / (Any path after the domain name)  
**Send For:** Any type of connection  
**Expires:** Thursday, December 31, 2020 11:59:59 PM

**Applications where cookies can be used:**

**1) To implement shopping cart:**  
Cookies are used for maintaining online ordering system. Cookies remember what user wants to buy. What if user adds some products in their shopping cart and if due to some reason user don’t want to buy those products this time and closes the browser window? When next time same user visits the purchase page he can see all the products he added in shopping cart in his last visit.

**2) Personalized sites:**  
When user visits certain pages they are asked which pages they don’t want to visit or display. User options are get stored in cookie and till the user is online, those pages are not shown to him.

**3) User tracking:**  
To track number of unique visitors online at particular time.

**4) Marketing:**  
Some companies use cookies to display advertisements on user machines. Cookies control these advertisements. When and which advertisement should be shown? What is the interest of the user? Which keywords he searches on the site? All these things can be maintained using cookies.

**5) User sessions:**  
Cookies can track user sessions to particular domain using user ID and password.

**Drawbacks of cookies:**

**1)**Even writing Cookie is a great way to maintain user interaction, if user has set browser options to warn before writing any cookie or disabled the cookies completely then site containing cookie will be completely disabled and can not perform any operation resulting in loss of site traffic.

**2) Too many Cookies:**  
If you are writing too many cookies on every page navigation and if user has turned on option to warn before writing cookie, this could turn away user from your site.

**3) Security issues:**  
Some times users personal information is stored in cookies and if someone hack the cookie then hacker can get access to your personal information. Even corrupted cookies can be read by different domains and lead to security issues.

**4) Sensitive information:**  
Some sites may write and store your sensitive information in cookies, which should not be allowed due to privacy concerns.

**Some Major Test cases for web application cookie testing:**

The first obvious test case is to test if your application is writing cookies properly on disk. You can use the Cookie Tester application also if you don’t have any web application to test but you want to understand the cookie concept for testing.

**Test cases:**

**1)** As a Cookie **privacy policy** make sure from your design documents that no personal or sensitive data is stored in the cookie.

**2)**If you have no option than saving **sensitive data** in cookie make sure data stored in cookie is stored in encrypted format.

**3)** Make sure that there is **no overuse of cookies** on your site under test. Overuse of cookies will annoy users if browser is prompting for cookies more often and this could result in loss of site traffic and eventually loss of business.

**4) Disable the cookies** from your browser settings: If you are using cookies on your site, your sites major functionality will not work by disabling the cookies. Then try to access the web site under test. Navigate through the site. See if appropriate messages are displayed to user like “For smooth functioning of this site make sure that cookies are enabled on your browser”. There should not be any page crash due to disabling the cookies. (Please make sure that you close all browsers, delete all previously written cookies before performing this test)

**5) Accepts/Reject some cookies:**The best way to check web site functionality is, not to accept all cookies. If you are writing 10 cookies in your web application then randomly accept some cookies say accept 5 and reject 5 cookies. For executing this test case you can set browser options to prompt whenever cookie is being written to disk. On this prompt window you can either accept or reject cookie. Try to access major functionality of web site. See if pages are getting crashed or data is getting corrupted.

**6) Delete cookie:**Allow site to write the cookies and then close all browsers and manually delete all cookies for web site under test. Access the web pages and check the behavior of the pages.

**7) Corrupt the cookies:** Corrupting cookie is easy. You know where cookies are stored. Manually edit the cookie in notepad and change the parameters to some vague values. Like alter the cookie content, Name of the cookie or expiry date of the cookie and see the site functionality. In some cases corrupted cookies allow to read the data inside it for any other domain. This should not happen in case of your web site cookies. Note that the cookies written by one domain say rediff.com can’t be accessed by other domain say yahoo.com unless and until the cookies are corrupted and someone trying to hack the cookie data.

**8 ) Checking the deletion of cookies from your web application page:** Some times cookie written by domain say rediff.com may be deleted by same domain but by different page under that domain. This is the general case if you are testing some ‘action tracking’ web portal. Action tracking or purchase tracking pixel is placed on the action web page and when any action or purchase occurs by user the cookie written on disk get deleted to avoid multiple action logging from same cookie. Check if reaching to your action or purchase page deletes the cookie properly and no more invalid actions or purchase get logged from same user.

**9) Cookie Testing on Multiple browsers:**This is the important case to check if your web application page is writing the cookies properly on different browsers as intended and site works properly using these cookies. You can test your web application on Major used browsers like Internet explorer (Various versions), Mozilla Firefox, Netscape, Opera etc.

**10)** If your web application is using **cookies to maintain the logging state of any user** then log in to your web application using some username and password. In many cases you can see the logged in user ID parameter directly in browser address bar. Change this parameter to different value say if previous user ID is 100 then make it 101 and press enter. The proper access message should be displayed to user and user should not be able to see other users account.

**Database Testing**

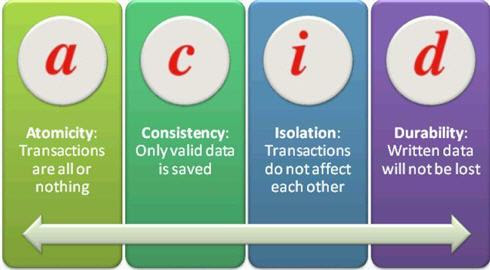
**Why do we test a database?**

**Below, we will briefly see why the following aspects of a DB should be validated:**

**1) Data Mapping:** In the software systems, data often travels back and forth from the UI (user interface) to the backend DB and vice versa. So following are the aspects to look for:

* To check whether the fields in the UI/Front end forms and mapped consistently with the corresponding DB table (and also the fields within).  Typically this mapping information is defined in the requirements documents.
* Whenever a certain action is performed in the front end of an application, a corresponding CRUD (Create, Retrieve, Update and delete) action gets invoked at the back end. A tester will have to check if the right action is invoked and the invoked action in itself is successful or not.

**2) ACID properties validation:**  atomicity, consistency, isolation and durability. Every transaction a DB performs has to adhere to these four properties. (image credit)

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2013/08/DB-Testing.jpg)

* **Atomicity** means that a transaction either fails or passes. This means that even if a single part of transaction fails- it means that the entire transaction has failed. Usually this is called the “all-or nothing” rule.
* **Consistency**: A transaction will always result in a valid state of the DB
* **Isolation**: If there are multiple transactions and they are executed all at once, the result/state of the DB should be the same as if they were executed one after the other.
* **Durability**: Once a transaction is done and committed, no external factors like power loss or crash should be able to change it

**3) Data integrity:**

This means that following any of the CRUD operations, the updated and most recent values/Status of shared data should appear on all the forms and screens. A value should not be updated on one screen and display an older value on another one. So devise your DB test cases in a way to include checking the data in all the places it appears to see if it is consistently the same.

**4) Business rule conformity:**More complex databases means more complicated components like relational constraints, triggers, stored procedures, etc. So testers will have to come up with appropriate SQL queries in order to validate these complex objects.

**How to test – Database Testing Process**

The general test process for DB testing is not very different from any other application. The following are the steps:

**Step #1)** Prepare the environment

**Step #2) Run a test**

**Step #3) Check test result**

**Step #4) Validate according to the expected results**

**Step #5) Report the findings to the respective stakeholders**

Usually SQL queries are used to develop the tests. The most commonly used command is the “Select”.

*Select \* from where*

Apart from select, SQL has 3 important types of commands:

1. DDL : Data definition language
2. DML: Data manipulation language
3. DCL: data control language

Let us see the syntax for the most commonly used statements.

**Data Definition language:**Uses CREATE, ALTER, RENAME, DROP and TRUNCATE to handle tables (and indexes).

* Create table: *Create table (field1 datatype(field size) ,……………..fieldn datatype(field size))*
* Delete entire table*: Drop table .* – this command cannot be rolled back

**Data Manipulation language:**Includes statements to add, update and delete records.

* To insert a row into a DB: INSERT INTO (field1, field2, field3)  VALUES  (‘val1’, ‘val2’…’valn’);

|  |
| --- |
|  |

* Delete specific row/rows from a table: *DELETE FROM TABLENAME WHERE .*
* Update rows:*UPDATE SET field1 = ‘updated value’ WHERE field2 = ‘N’;*

**Data control language:** Deals with giving the authorization to users for manipulation and access to the data. Grant and Revoke are the two statements used.

Grant syntax:

*Grant select/update*

*On*

|  |
| --- |
|  |

*To ;*

Revoke syntax:

*Revokeselect/update*

*on*

|  |
| --- |
|  |

*from;*

**What to test – different components**

**1) Transactions:**

When testing transactions it is important to make sure that they satisfy the ACID properties.

The following are the statements commonly used:

* BEGIN TRANSACTION TRANSACTION#
* END TRANSACTION TRANSACTION#

Rollback statement ensures that the database lies in a consistent state.

* ROLLBACK TRANSACTION#

After these statements are executed, use a select to make sure if the changes have been reflected.

* SELECT \* FROM TABLENAME

**2) Database schema:**

Database schema is nothing but a formal definition of the how the data is going to be organized into a DB. To test it:

------------

* Identify the requirements based on which the database operates. Sample requirements:
  + Primary keys to be created before any other fields are created.
  + Foreign keys should be completely indexed for easy retrieval and searching.
  + Field names starting or ending with certain characters.
  + Fields with a constraint that certain values can or cannot be inserted.
* Use one of the following ways according to the relevance:
  + SQL Query *DESC<table name>* to validate the schema.
  + Regular expressions for validating the names of the individual fields and their values
  + Tools like SchemaCrawler

**3) Trigger:**

When a certain event takes places on a certain table, a piece of code (a trigger) can be auto-instructed to be executed.

**For example**, a new student joined a school. The student is taking 2 classes; math and science. The student is added to the “student table”.  A trigger could be adding the student to the corresponding subject tables once he is added to the student table.

The common method to test is to execute SQL query embedded in the trigger independently first and record the result. Follow this up with executing the trigger as a whole. Compare the results.

These are tested during both the black box and white box testing phases.

* **White box testing**:  Stubs and drivers are to insert or update or delete data that would result in the trigger being invoked. The basic idea is to just test the DB alone even before the integration with the front end (UI) is made.
* **Black box testing**:

**a)** Since the UI and DB integration is now available; we can insert/delete/update data from the front end in a way that the trigger gets invoked. Following that select statements can be used to retrieve the DB data to see if the trigger was successful in performing the intended operation.

**b)** Second way to test this is to directly load the data that would invoke the trigger and see if it works as intended.

**4) Stored Procedures:**

Stored procedures are more or less similar to user defined functions. These can be invoked by a call procedure/execute procedure statements and the output is usually in the form of result sets.

These are stored in the RDBMS and are available for applications.

These are also tested during:

* **White box testing:** Stubs are used to invoke the stored procedures and then the results are validated against the expected values.
* **Black box testing:** Perform an operation from the frontend(UI) of the application and check for the execution of the stored procedure and its results.

**5. Field constraints – Default value, unique value and foreign key:**

* Perform a front end operation which overruns the database object condition
* Validate the results with a SQL Query.

Checking the default value for a certain field is quite simple. It is a part of business rule validation. You can do it manually or you can use tools like QTP to do so. Manually, you can perform an action that will add a value other than the default value into the field from the front end and see if it results in an error.

**Error handling testing**

* The main objective of the Error handling testing techniques is to ensure that the applications are capable to handling incorrect Transactions.
* The application is smart enough to identify the incorrect transactions and approximate decision need to take and intimate the end user.
* Normal testing is basically test the application is working properly as per the requirements but error handling testing is exactly the opposite manner.
* Error handling testing is basically done by Experience persons who had the full knowledge of the domain.
* Error handling testing can generate in such that the improper transactions is done in-between the proper transactions and check how the system is behaved during improper transactions and identify the problems.
* Use the improper master data and check the application whether it is capable to identify the problem and informed to the Users.

### Input domain testing (ECP, BVA)

**Boundary value analysis** and **Equivalence Class Partitioning** both are test case design techniques in black box testing. In this article we are covering “What is Boundary value analysis and equivalence partitioning & its simple examples”.

## What is Equivalence Class Partitioning?

Equivalence partitioning is a Test Case Design Technique to divide the input data of software into different equivalence data classes. Test cases are designed for equivalence data class. The equivalence partitions are frequently derived from the requirements specification for input data that influence the processing of the test object. A use of this method reduces the time necessary for testing software using less and effective test cases.

Equivalence Partitioning = Equivalence Class Partitioning = ECP

It can be used at any level of software for testing and is preferably a good technique to use first. In this technique, only one condition to be tested from each partition. Because we assume that, all the conditions in one partition behave in the same manner by the software. In a partition, if one condition works other will definitely work. Likewise we assume that, if one of the condition does not work then none of the conditions in that partition will work.

Equivalence partitioning is a testing technique where input values set into classes for testing.

* Valid Input Class = Keeps all valid inputs.
* Invalid Input Class = Keeps all Invalid inputs.

## Example of Equivalence Class Partitioning?

* A text field permits only numeric characters
* Length must be 6-10 characters long

Partition according to the requirement should be like this:

While evaluating Equivalence partitioning, values in all partitions are equivalent that’s why 0-5 are equivalent, 6 – 10 are equivalent and 11- 14 are equivalent.

At the time of testing, test 4 and 12 as invalid values and 7 as valid one.

It is easy to test input ranges 6–10 but harder to test input ranges 2-600. Testing will be easy in the case of lesser test cases but you should be very careful. Assuming, valid input is 7. That means, you belief that the developer coded the correct valid range (6-10).

## What is Boundary value analysis:

**Boundary value analysis** is a test case design technique to test boundary value between partitions (both valid boundary partition and invalid boundary partition). A boundary value is an input or output value on the border of an equivalence partition, includes minimum and maximum values at inside and outside boundaries. Normally Boundary value analysis is part of stress and negative testing.

Using Boundary Value Analysis technique tester creates test cases for required input field. For example; an Address text box which allows maximum 500 characters. So, writing test cases for each character once will be very difficult so that will choose boundary value analysis.

## Example for Boundary Value Analysis:

**Example 1**

Suppose you have very important tool at office, accepts valid User Name and Password field to work on that tool, and accepts minimum 8 characters and maximum 12 characters. Valid range 8-12, Invalid range 7 or less than 7 and Invalid range 13 or more than 13.



Write Test Cases for Valid partition value, Invalid partition value and exact boundary value.

* Test Cases 1: Consider password length less than 8.
* Test Cases 2: Consider password of length exactly 8.
* Test Cases 3: Consider password of length between 9 and 11.
* Test Cases 4: Consider password of length exactly 12.
* Test Cases 5: Consider password of length more than 12.

**Example 2**

Test cases for the application whose input box accepts numbers between 1-1000. Valid range 1-1000, Invalid range 0 and Invalid range 1001 or more.



Write Test Cases for Valid partition value, Invalid partition value and exact boundary value.

* Test Cases 1: Consider test data exactly as the input boundaries of input domain i.e. values 1   and 1000.
* Test Cases 2: Consider test data with values just below the extreme edges of input domains i.e. values 0 and 999.
* Test Cases 3: Consider test data with values just above the extreme edges of input domain i.e. values 2 and 1001.

### Over to you on Boundary Value Analysis and Equivalence Class Partitioning:

Till now we have seen about What is Boundary value analysis and equivalence partitioning & its simple examples. Here we have covered very basic and simple example to understand the most commonly used test case design techniques. There is no such hard and fast rule to  take only one input from each partition. Based on your needs and previous experience you can decide the inputs.

**What is functionality testing in software?**

Functionality testing is performed to verify that a software application performs and functions correctly according to design specifications. During functionality testing we check the core application functions, text input, menu functions and installation and setup on localized machines, etc.

 The following is needed to be checked during the functionality testing:

* Installation and setup on localized machines running localized operating systems and local code pages.
* Text input, including the use of extended characters or non-Latin scripts.
* Core application functions.
* String handling, text, and data, especially when interfacing with non-Unicode applications or modules.
* Regional settings defaults.
* Text handling (such as copying, pasting, and editing) of extended characters, special fonts, and non-Latin scripts.
* Accurate hot-key shortcuts without any duplication.

Functionality testing verifies that an application is still fully functional after localization. Even applications which are professionally internationalized according to world-readiness guidelines require functionality testing.

### GUI Testing

**GUI Testing** is nothing but testing the design of the application.

Verifying the User interface of the application means how the front end design, colors,  fonts, font sizes, Labels, Text format, Text boxes, Captions, Radio Buttons, buttons, list boxes, Alignments, icons, links, popup, content, tabbing is provided to move between the objects, background, foreground etc  are displayed as specified.  
GUI testing is functional testing - ensuring that all interactions, navigation, links, pop-ups, content, etc all work as required.

**CONTENTS:-**

**Section 1 - Windows Compliance Standards**

1.1. Application  
1.2. For Each Window in the Application  
1.3. Text Boxes  
1.4. Option (Radio Buttons)  
1.5. Check Boxes  
1.6. Command Buttons  
1.7. Drop Down List Boxes  
1.8. Combo Boxes  
1.9. List Boxes

**Section 2 - Tester's Screen Validation Checklist**

2.1. Aesthetic Conditions  
2.2. Validation Conditions  
2.3. Navigation Conditions  
2.4. Usability Conditions  
2.5. Data Integrity Conditions  
2.6. Modes (Editable Read-only) Conditions  
2.7. General Conditions  
2.8. Specific Field Tests  
2.8.1. Date Field Checks  
2.8.2. Numeric Fields  
2.8.3. Alpha Field Checks

**Section 3 - Validation Testing - Standard Actions**

3.1. On every Screen  
3.2. Shortcut keys / Hot Keys  
3.3. Control Shortcut Keys  
Section 4 - Origin & Inspiration  
4.1. Document origin  
4.2. Sources of Inspiration & information  
4.3. Contacting the author.

**Compliance for each application**

·         Application should be started by double clicking on the icon.

·         Loading message should have information about application name, version number, icon etc.

·         Main window of application should have same caption as the icon in the program manager.

·         Closing of the application should result in “Are you sure?” message.

·         Behavior for starting application more than once must be specified.

·         Try to start application while it is loading

·         On every application, if application is busy it should show hour glass or some other mechanism to notify user that it is processing.

·         Normally F1 button is used for help. If your product has help integrated, it should come by pressing F1 button.

·         Minimize and restoring functionality should work properly

·         Compliance for each window in the application

·         Window caption for every application should have application name and window name. Specially, error messages.

·         Title of the window and information should make sense to the user.

·         If screen has control menu, use the entire control menu like move, close, resize etc.

·         Text present should be checked for spelling and grammar.

·         If tab navigation is present, TAB should move focus in forward direction and SHIFT+TAB in backward direction.

·         Tab order should be left to right and top to bottom within a group box.

·         If focus is present on any control, it should be presented by dotting lines around it.

·         User should not be able to select greyed or disabled control. Try this using tab as well as mouse.

·         Text should be left justified

·         In general, all the operations should have corresponding key board shortcut key for this.

·         All tab buttons should have distinct letter for it.

**Text boxes**

·         Move mouse to textbox and it should be changed to insert bar for editable text field and should remain unchanged for non-editable text field.

·         Test overflowing textbox by inserting as many characters as you can in the text field. Also test width of the text field by entering all capital W.

·         Enter invalid characters, special characters and make sure that there is no abnormality.

·         User should be able to select text using Shift + arrow keys.

·         Selection should be possible using mouse and double click should select entire text in the text box.

**Radio Buttons**

·         Only one should be selected from the given option.

·         User should be able to select any button using mouse or key board

·         Arrow key should set/unset the radio buttons.

**Check boxes**

·         User should be able to select any combination of checkboxes

·         Clicking mouse on the box should set/unset the checkbox.

·         Spacebar should also do the same

**Push Buttons**

·         All buttons except OK/Cancel should have a letter access to them. This is indicated by a letter underlined in the button text. The button should be activated by pressing ALT

·         Clicking each button with mouse should activate it and trigger required action.

·         Similarly, after giving focus SPACE or RETURN button should also do the same.

·         If there is any Cancel button on the screen, pressing Esc should activate it.

**Drop down list boxes**

·         Pressing the arrow should give list of options available to the user. List can be scrollable but user should not be able to type in.

·         Pressing Ctrl-F4 should open the list box.

·         Pressing a letter should bring the first item in the list starting with the same letter.

·         Items should be in alphabetical order in any list.

·         Selected item should be displayed on the list.

·         There should be only one blank space in the dropdown list.

**Combo Box**

·         Similar to the list mentioned above, but user should be able to enter text in it.

**List Boxes**

·         Should allow single select, either by mouse or arrow keys.

·         Pressing any letter should take you to the first element starting with that letter

·         If there are view/open button, double clicking on icon should be mapped to these behavior.

·         Make sure that all the data can be seen using scroll bar.

### Black Box Testing

## Black Box Testing**, also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.**

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structures or external database access
* Behavior or performance errors
* Initialization and termination errors

Definition by ISTQB

* **black box testing:**Testing, either functional or non-functional, without reference to the  
  internal structure of the component or system.
* **black box test design technique:**Procedure to derive and/or select test cases based on an  
  analysis of the specification, either functional or non-functional, of a component or system  
  without reference to its internal structure.

**EXAMPLE**

A tester, without knowledge of the internal structures of a website, tests the web pages by using a browser; providing inputs (clicks, keystrokes) and verifying the outputs against the expected outcome.

**LEVELS APPLICABLE TO**

Black Box Testing method is applicable to the following levels of software testing:

* Integration Testing
* System Testing
* Acceptance Testing

The higher the level, and hence the bigger and more complex the box, the more black box testing method comes into use.

**BLACK BOX TESTING TECHNIQUES**

Following are some techniques that can be used for designing black box tests.

* *Equivalence partitioning:*It is a software test design technique that involves dividing input values into valid and invalid partitions and selecting representative values from each partition as test data.
* *Boundary Value Analysis:*It is a software test design technique that involves determination of boundaries for input values and selecting values that are at the boundaries and just inside/ outside of the boundaries as test data.
* *Cause Effect Graphing:*It is a software test design technique that involves identifying the cases (input conditions) and effects (output conditions), producing a Cause-Effect Graph, and generating test cases accordingly.

**BLACK BOX TESTING ADVANTAGES**

* Tests are done from a user’s point of view and will help in exposing discrepancies in the specifications.
* Tester need not know programming languages or how the software has been implemented.
* Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias.
* Test cases can be designed as soon as the specifications are complete.

**BLACK BOX TESTING DISADVANTAGES**

* Only a small number of possible inputs can be tested and many program paths will be left untested.
* Without clear specifications, which is the situation in many projects, test cases will be difficult to design.
* Tests can be redundant if the software designer/ developer has already run a test case.
* Ever wondered why a soothsayer closes the eyes when foretelling events? So is almost the case in Black Box Testing.

**White Box Testing**

**White Box Testing**(also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

This method is named so because the software program, in the eyes of the tester, is like a white/ transparent box; inside which one clearly sees.

Definition by ISTQB

* **white-box testing:** Testing based on an analysis of the internal structure of the component or  
  system.
* **white-box test design technique:**Procedure to derive and/or select test cases based on an  
  analysis of the internal structure of a component or system.

**EXAMPLE**

A tester, usually a developer as well, studies the implementation code of a certain field on a webpage, determines all legal (valid and invalid) AND illegal inputs and verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.

White Box Testing is like the work of a mechanic who examines the engine to see why the car is not moving.

**LEVELS APPLICABLE TO**

White Box Testing method is applicable to the following levels of software testing:

* Unit Testing: For testing paths within a unit.
* Integration Testing: For testing paths between units.
* System Testing: For testing paths between subsystems.

However, it is mainly applied to Unit Testing.

**WHITE BOX TESTING ADVANTAGES**

* Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
* Testing is more thorough, with the possibility of covering most paths.

**WHITE BOX TESTING DISADVANTAGES**

* Since tests can be very complex, highly skilled resources are required, with thorough knowledge of programming and implementation.
* Test script maintenance can be a burden if the implementation changes too frequently.
* Since this method of testing it closely tied with the application being testing, tools to cater to every kind of implementation/platform may not be readily available.

**What are Software Testing Levels?**

Testing levels are basically to identify missing areas and prevent overlap and repetition between the development life cycle phases. In software development life cycle models there are defined phases like requirement gathering and analysis, design, coding or implementation, testing and deployment.  Each phase goes through the testing. Hence there are various levels of testing. The various levels of testing are:

1. **Unit testing**: It is basically done by the developers to make sure that their code is working fine and meet the user specifications. They test their piece of code which they have written like classes, functions, interfaces and procedures.
2. **Component testing:** It is also called as module testing. The basic difference between the unit testing and component testing is in unit testing the developers test their piece of code but in component testing the whole component is tested. For example, in a student record application there are two modules one which will save the records of the students and other module is to upload the results of the students. Both the modules are developed separately and when they are tested one by one then we call this as a component or module testing.
3. **Integration testing**: Integration testing is done when two modules are integrated, in order to test the behavior and functionality of both the modules after integration. Below are few types of integration testing:
   * Big bang integration testing
   * Top down
   * Bottom up
   * Functional incremental
4. **Component integration testing:** In the example above when both the modules or components are integrated then the testing done is called as Component integration testing. This testing is basically done to ensure that the code should not break after integrating the two modules.
5. **System integration testing:** System integration testing (SIT) is a testing where testers basically test that in the same environment all the related systems should maintain data integrity and can operate in coordination with other systems.
6. **System testing:** In system testing the testers basically test the compatibility of the application with the system.
7. **Acceptance testing:** Acceptance testing are basically done to ensure that the requirements of the specification are met.
8. **Alpha testing:**Alpha testing is done at the developers site. It is done at the end of the development process
9. **Beta testing:** Beta testing is done at the customers site. It is done just before the launch of the product.