**MANUAL TESTING**

1. Why testing is required?

Testing is important as it uncovers a defect before it is delivered to customer ensuring quality of software. So that the defects or bugs can be identified in early stages of development; as later the stage in which bug is identified, more is the cost to rectify it. It makes software more reliable and user-friendly to operate. An untested software not only makes software error prone, it also costs the customer business failure too.

1. What type of applications we test?

Web applications (we open these applications in browser, ex: gmail.com)

Desktop/windows applications (we run from our desktop, ex: notepad, word)

Mobile applications (we run from mobile devices, android, ios, windows phone)

Web services (SOAP/REST) (we use them as part of web applications / desktop applications/mobile applications)

ETL jobs, database validations: these jobs don’t have UI and they run in back ground to load the data (informatica jobs, ssis job etc.).

Back end/batch programs/windows services.

1. What is SDLC and different phases in SDLC?

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process. The different phases are:

* Requirement gathering and analysis.
* Design.
* Implementation or coding.
* Testing.
* Deployment.
* Maintenance.

1. What is Waterfall Model?

The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model.**  It is very simple to understand and use.  In a waterfall model, each phase must be completed fully before the next phase can begin. This type of [software development model](http://istqbexamcertification.com/what-are-the-software-development-models/) is basically used for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model [software testing](http://istqbexamcertification.com/what-is-a-software-testing/) starts only after the development is complete. In **waterfall model phases**



Advantages:

* This model is simple and easy to understand and use.
* It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
* In this model phases are processed and completed one at a time. Phases do not overlap.
* Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages:

* Once an application is in the [testing](http://istqbexamcertification.com/what-is-a-software-testing/) stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.

1. What is agile method?

Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end. It works by breaking projects down into little bits of user functionality called [user stories](http://www.agilenutshell.com/user_stories), prioritizing them, and then continuously delivering them in short two week cycles called [iterations](http://www.agilenutshell.com/iterations).

1. What is Scrum Methodology?

Scrum is a subset of Agile. It is a lightweight process framework for agile development, and the most widely-used one. [Scrum](http://info.versionone.com/sutherland-power-of-scrum-agile-video.html) is an agile project management methodology or framework used primarily for software development projects with the goal of delivering new software capability every 2-4 weeks. It is one of the approaches that influenced the [Agile Manifesto](http://agilemanifesto.org/), which articulates a set of values and principles to guide decisions on how to develop higher-quality software faster.

1. What is the process of agile model?

It uses the idea of modelling the light weight fashion to deliver the documentation that is good enough for right now. Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.



1. What is daily stand up meeting? And what we discuss?

The purpose of the meeting is for each team member to answer the following three questions:

1) What did you do yesterday?

2) What will you do today?

3) Are there any impediments in your way?

Standing, rather than sitting, reinforces the idea that the meeting is intended to be short and discourages wasted time. The stand-up is not meant to be a place to solve problems, but rather to make the team aware of current status. If discussion is needed, a longer meeting with appropriate parties can be arranged.

1. What is product backlog item?

Product backlog is the work needs to be done to complete the product/project. In theory you can consider User stories, bugs, or even a change request as a product backlog item. Each PBI must have these qualities:

* Description: What the goal of the PBI is.
* Value: the [Business Value](https://www.scruminc.com/calculating-business-value/) of the PBI as determined by the Product Owner.
* Estimate: the Team needs to estimate the relative effort it will take to move the PBI to [Done](https://www.scruminc.com/definition-done/).
* Order: The Product Owner needs to prioritize PBIs by their relative value.

1. What is user story/feature/sprint back log items and tasks in user story?

A user story is a tool used in [agile software development](http://searchsoftwarequality.techtarget.com/definition/agile-software-development) to capture a description of a software feature from an end-user perspective. The user story describes the type of user, what they want and why. A user story helps to create a simplified description of a requirement.

As a <role>, I want <feature> so that <reason>.

Examples of user stories are: As a user, I want to upload photos so that I can share photos with others.

1. What is sprint planning meeting?

The Sprint Planning Meeting is typically broken into two parts. Part one of the sprint planning meeting is a review of the product backlog items the Product Owner will ask the team to forecast and deliver. This is the time for the product owner to describe what she wants to see built for the next sprint. During this part of the meeting, it is not uncommon for the team to banter back and forth with the product owner, asking clarifying questions and driving away ambiguity. By the end of sprint planning part one, the team will select a sprint goal: a one-sentence description of the overall outcome of the sprint. This helps later when questions about depth and breadth come up: if the work does not directly tie to the sprint goal, then it is not done during the sprint.

During part two of the sprint planning meeting, the team decides how the work will be built. In this meeting the team will begin decomposing the product backlog items into work tasks and estimating these in hours. The product owner must be available during this meeting but does not have to be in the room. In fact, many teams find it helpful to work without product owner during this detailed part of the meeting. Knowing that the product owner is available yet not having her observing all of the discussion about the best way to implement a feature can be freeing for many teams. Many teams find they enjoy discussing many implementation possibilities without worrying that the product owner will panic or misunderstand. If the product owner does remain in the room, the Scrum Master needs to take charge of this part of the meeting, keeping the team focused and free to explore possibilities without being limited by the product owner’s own ideas or opinions. The output of the second planning meeting will be the Sprint Backlog.

1. What is sprint review meeting?

In [Scrum](https://www.mountaingoatsoftware.com/agile/scrum), each sprint is required to deliver a potentially shippable product increment. This means that at the end of each sprint, the team has produced a coded, tested and usable piece of software.

So at the end of each sprint, a sprint review meeting is held. During this meeting, the Scrum team shows what they accomplished during the sprint. Typically this takes the form of a demo of the new features.

The sprint review meeting is intentionally kept very informal, typically with rules forbidding the use of PowerPoint slides and allowing no more than two hours of preparation time for the meeting. A sprint review meeting should not become a distraction or significant detour for the team; rather, it should be a natural result of the sprint.

Participants in the sprint review typically include the product owner, the Scrum team, the Scrum Master, management, customers and developers from other projects.

1. What is sprint retrospective?

The sprint retrospective is a meeting facilitated by the Scrum Master at which the team discusses the just-concluded sprint and determines what could be changed that might make the next sprint more productive. The sprint review looks at what the team is building, whereas the retrospective looks at how they are building it.

1. What is sprint grooming?

Backlog grooming is when the [product owner](https://www.agilealliance.org/glossary/product-owner/)and some, or all, of the rest of the team review items on the [backlog](https://www.agilealliance.org/glossary/backlog/) to ensure the backlog contains the appropriate items, that they are prioritized, and that the items at the top of the backlog are ready for delivery. This activity occurs on a regular basis and may be an officially scheduled meeting or an ongoing activity. Some of the activities that occur during this refinement of the backlog include:

* removing [user stories](https://www.agilealliance.org/glossary/user-stories/) that no longer appear relevant
* creating new user stories in response to newly discovered needs
* re-assessing the relative priority of stories
* assigning estimates to stories which have yet to receive one
* correcting estimates in light of newly discovered information
* [splitting](https://www.agilealliance.org/glossary/split/) user stories which are high priority but too coarse grained to fit in an upcoming [iteration](https://www.agilealliance.org/glossary/iteration/)

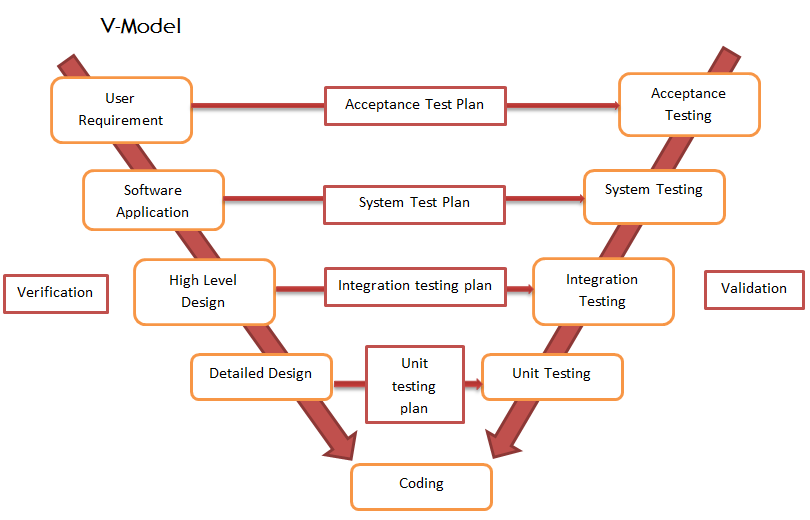
1. What is burn down chart and velocity?

Burn down chart: Progress on a Scrum project can be tracked by means of a release burn down chart. The Scrum Master should update the release burn down chart at the end of each sprint. The horizontal axis of the sprint burn down chart shows the sprints; the vertical axis shows the amount of work remaining at the start of each sprint.

Velocity: Velocity is a measure of the amount of work a Team can tackle during a single Sprint and is the key metric in Scrum. Velocity is calculated at the end of the Sprint by totaling the Points for all fully completed User Stories

1. What is user acceptance criteria test cases?
2. What is V model?

In the basic [Waterfall model](http://www.softwaretestingclass.com/what-is-software-testing-methodologies/) process seen some disadvantages or limitations in the model which started a new SDLC model. As we seen in the Waterfall model the issues found in the end of the SDLC, this is due to the testing is occurred in the end phases of the SDLC. To overcome this problem the [**V-Model**](http://www.softwaretestingclass.com/v-model/) is comes into the picture. It is always better to introduce testing in the early phase of SDLC, as in this model the testing activity gets started from the early phase of the SDLC.



1. What is STLC?

Software Testing Life Cycle refers to a testing process which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met. In STLC process, each activity is carried out in a planned and systematic way. Each phase has different goals and deliverables. Different organizations have different phases in STLC; however the basis remains the same.

Below are the phases of STLC:

1. Requirements phase
2. Planning Phase
3. Analysis phase
4. Design Phase
5. Implementation Phase
6. Execution Phase
7. Conclusion Phase
8. Closure Phase
9. What is defect?

A **defect is an error or a bug,** in the application which is created. A programmer while designing and building the software can make mistakes or error. These mistakes or errors mean that there are flaws in the software. These are called defects.

1. How to arise a defect and what we specify while logging defect?

Defect logging, a process of finding defects in the application under test or product by testing or recording feedback from customers and making new versions of the product that fix the defects or the clients feedback.

1. What is defect lifecycle?

Defect Life Cycle or Bug Life Cycle is the specific set of states that a Bug goes through from discovery to defect fixation.

The number of states that a defect goes through varies from project to project. Below lifecycle diagram, covers all possible states

* **New:** When a new defect is logged and posted for the first time. It is assigned a status NEW.
* **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to developer team
* **Open**: The developer starts analyzing and works on the defect fix
* **Fixed**: When developer makes necessary code change and verifies the change, he or she can make bug status as "Fixed."
* **Pending retest**: Once the defect is fixed the developer gives particular code for retesting the code to the tester. Since the testing remains pending from the testers end, the status assigned is "pending request."
* **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and change the status to "Re-test."
* **Verified**: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is "verified."
* **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to "reopened". Once again the bug goes through the life cycle.
* **Closed**: If the bug is no longer exists then tester assigns the status "Closed."
* **Duplicate**: If the defect is repeated twice or the defect corresponds the same concept of the bug, the status is changed to "duplicate."
* **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to "rejected."
* **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status "Deferred" is assigned to such bugs
* **Not a bug**: If it does not affect the functionality of the application then the status assigned to a bug is "Not a bug".

[](http://cdn.guru99.com/images/1-2015/012715_0802_BugLifeCycl1.png)

1. What is Unit testing?

Unit testing is a [software](http://searchsoa.techtarget.com/definition/software) development process in which the smallest testable parts of an [application](http://searchsoftwarequality.techtarget.com/definition/application), called units, are individually and independently scrutinized for proper operation. Unit testing can be done manually but is often [automated](http://searchitoperations.techtarget.com/definition/agile-test-automation-pyramid).

1. When do we use regression testing?

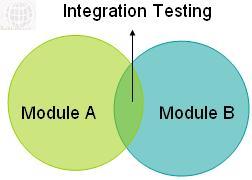
Regression means retesting the unchanged parts of the application. Test cases are re-executed in order to check whether previous functionality of application is working fine and new changes have not introduced any new bugs. This test can be performed on a new build when there is significant change in original functionality or even a single bug fix. Any time you modify an implementation within a program, you should also do regression testing. You can do so by rerunning existing tests against the modified code to determine whether the changes break anything that worked prior to the change and by writing new tests where necessary.

1. What is integration testing?

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.

Also after integrating two different [components](http://istqbexamcertification.com/what-is-component-testing/) together we do the integration [testing](http://istqbexamcertification.com/what-is-a-software-testing/). As displayed in the image below when two different modules ‘Module A’ and ‘Module B’ are integrated then the integration testing is done.

Integration testing is done by a specific integration tester or test team



1. When do we do integration testing?

Even if a software component is successfully unit tested, in an enterprise n-tier distributed application it is of little or no value if the component cannot be successfully integrated with the rest of the application.

Once unit tested components are delivered we then integrate them together.   
These “integrated” components are tested to weed out errors and bugs caused due to the integration. This is a very important step in the Software Development Life Cycle.

1. When do we use smoke testing and sanity testing?

* **Smoke testing:** Smoke testing is conducted to ensure whether the most crucial functions of a program are working, but not bothering with finer details. (Such as build verification).
* **Sanity testing:** Sanity testing is a cursory testing, it is performed whenever a cursory testing is sufficient to prove the application is functioning according to specifications. This level of testing is a subset of regression testing.

Sanity testing is to verify whether requirements are met or not, checking all features breadth-first.

1. What is UAT?

User Acceptance Testing is often the final step before rolling out the application.

Usually the end users who will be using the applications test the application before ‘accepting’ the application.

This type of testing gives the end users the confidence that the application being delivered to them meets their requirements.

This testing also helps nail bugs related to usability of the application.

1. What is alpha and beta testing?

Alpha testing: Alpha testing is a type of acceptance testing; performed to identify all possible issues/bugs before releasing the product to everyday users or public.  The focus of this testing is to simulate real users by using black box and white box techniques.

Beta testing: Beta Testing of a product is performed by "real users" of the software application in a "real environment" and can be considered as a form of external user acceptance testing. Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

1. What we will do if we don't have a time to test all stories/ execute test cases?

It is always important to prioritize the tests from the beginning of project onwards. We used to categorize the tests as P1, P2, P3 and P4. All the test cases were automated. But P1 and P2 are the tests were scheduled to execute on daily basis and P3 and P4 were added at the time of sprint closure. Those which cannot automate will be marked as Manual.

As all the test cases were automated and P1 and P2 will be executed on daily basis on the latest build, so there won’t be much problems though we need to deliver the products within short span of duration.

1. What we will do if come across any critical severity issue before release day?

You must -absolutely- not ever hold off from reporting an issue, at least to your local management structure. That would, at the very least, ruin the reputation of your team and could potentially have much more serious consequences.

The next thing to do is to determine the answer to the obvious question: "Why was this found so late?” There are many reasons why this situation could arise - your test preparation could've been too light, you could've prioritized some work wrongly, there may simply have been too much to do. As a member of the test team, you need to know what caused the issue and therefore how you can reduce the risk of it happening again.

Obviously, we don't live in a perfect world and it's possible that no action may be taken to resolve the defect before release. It's entirely possible that it makes more sense to go live with a bug and then release a quick fix, than to decide not to release at all. That's a different discussion, though.

1. When do we use automation testing?

Every software development group tests its products, yet delivered software always has defects. Test engineers strive to catch them before the product is released but they always creep in and they often reappear, even with the best manual testing processes. [Test Automation software](https://smartbear.com/product/testcomplete/overview/) is the best way to increase the effectiveness, efficiency and coverage of your software testing.

 Once automated tests are created they can easily be repeated and they can be extended to perform tasks impossible with manual testing. Because of this, savvy managers have found that automated software testing is an essential component of successful development projects.

1. What tester will do in each phase of SDLC?

**Inception Phase**

In this phase, a test engineer will get an opportunity to identify the necessities of project. Normally the data are recorded by the architecture team in the architectural reference document. Data design, information design, system design are the main issues in this phase.

**Elaboration Phase**

In this phase, a test engineer will get an opportunity to identify how the project is planned. This is a major phase, where the entire design of the project is documented in the JAD phase in the System requirement document, business requirement document, product requirement document commercial use cases. Planner, Commercial reviewer, project organization, execution, testing, maintenance of project teams etc. are attended the JAD phase to give sign-off on these completed document.

**Construction Phase**

In this phase, programmers play an important role of building the application depends on the plan acknowledged during the JAD stage. Here tester group have to follow the programming group to identify several adjustments taken by the system. There may be any kind of fault which are overlooked by programmer, misapprehend the planed records, in that time, a tester can always rise the issue to the regarding programmer to solve the issue. A testing group requires developing the high level scenarios (HLS) on basis of the elaboration phase. High level scenarios may have more than one test case. A tester should ensure that all the necessities are discovered to a test case by a quality affirmation standard. It is mandatory to record test cases on the basis of all probable references of the newest modernized data and also signed-off.

**Transition Phase**

In this phase if any fault or errors are originate then these are test again and it goes though the regression testing. With the help of regression testing, consistent systems develop. By the helping of these testing methods, any fundamental result can be converted into a tough and consistent system.

1. Difference between load and performance testing?

**Performance Testing:** Performance Testing measures the response time of an application with an expected number of users. The aim of this is to get a baseline and an indication of how an application behaves under normal conditions. Does it meet the required response time?

**Load testing:** Load testing is usually started as low numbers and gradually increased over a given period of time until it reaches the desired load on the system and then it ramps down.

1. Different types of Non-functional testing?

Types of Non Functional Testing

1. Performance Testing

2. Load Testing

3. Stress Testing

4. Volume Testing

5. Failover Testing

6. Security Testing

7. Compatibility Testing

8. Usability Testing

9. Scalability Testing

1. What is test case?

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

1. What is test planning/test strategy document?

The Test Strategy Document is a living document that is created in the project’s Requirements Definition phase, after the Requirements have been specified. The Test Strategy document describes the scope, approach, resources and schedule for the testing activities of the project. This includes defining what will be tested, who will perform testing, how testing will be managed, and the associated risks and contingencies. The Test Strategy document is maintained throughout the life of a project.

1. What are entry and exit criteria?

Entry and Exit criteria are required to start and end the testing. It is must for the success of any project.

The entry criteria defines what all need to start the testing. It is very necessary to know for tester /QA what should be start criteria for entering into testing phase. In general, entry criteria is a set of conditions that permits a task to perform, or in absence of any among these condition will not allow to perform that task is taken as the Entry Criteria of that task.

The Exit criteria is a set of conditions based on which you can say this particular task is finished.   
This can be difficult to determine. Many modern software applications are so complex, and run in such as interdependent environment, that complete testing can never be done. "When to stop testing" is one of the most difficult questions to a test engineer.

1. What is TDD and BDD (cucumber framework)?

**Test-driven development** (**TDD**) is a software development process that relies on the repetition of a very short development cycle: first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test, and finally refactors the new code to acceptable standards. Kent Beck, who is credited with having developed or ‘rediscovered’ the technique, stated in 2003 that TDD encourages simple designs and inspires confidence

In software engineering, **behavior-driven development** (abbreviated **BDD**) is a software development process based on test-driven development (TDD). Behavior-driven development combines the general techniques and principles of TDD with ideas from domain-driven design and object-oriented analysis and design to provide software development and management teams with shared tools and a shared process to collaborate on software development.

1. What is priority and severity in defect?

Severity is defined as the degree of impact a defect has on the development or operation of a component application being tested. Higher effect on the system functionality will lead to the assignment of higher severity to the bug. Quality Assurance engineer usually determines the severity level of defect.

Defect severity can be categorized into four class

**Critical**: This defect indicates complete shut-down of the process, nothing can proceed further

**Major**: It is a highly severe defect and collapse the system. However, certain parts of the system remain functional

**Medium**: It cause some undesirable behavior, but the system is still functional

**Low**: It won't cause any major break-down of the system

Priority is defined as the order in which a defect should be fixed. Higher the priority the sooner the defect should be resolved.

Defects that leave the software system unusable are given higher priority over defects that cause a small functionality of the software to fail.

Defect priority can be categorized into three class

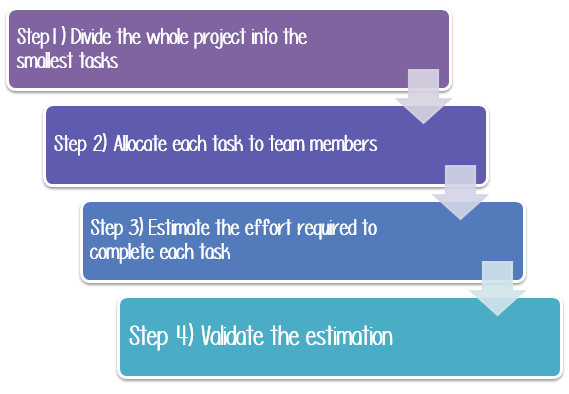
**Low:**The defect is an irritant but repair can be done once the more serious defect have been fixed

**Medium:**During the normal course of the development activities defect should be resolved. It can wait until a new version is created

**High:**The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed.

1. How to estimate the test case?

The following steps are used to estimate the test case.



Step1) Divide the whole project task into subtasks

Task is a piece of work that has been given to someone. To do this, you can use the **Work Breakdown Structure** technique.

In this technique, a complex project is divided into modules. The modules are divided into sub-modules. Each sub-module is further divided into functionality. It means divide the whole project task into the **smallest** tasks.

Step 2) Allocate each task to team member

In this step, each task is assigned to the **appropriate** member in the project team.

Step 3) Effort Estimation for Tasks

There are 2 techniques which you can apply to estimate the effort for tasks

**Functional Point Method**

**Three Point Estimation**

Step 4) Validate the estimation

Once you create an aggregate estimate for all the tasks mentioned in the WBS, you need to forward it to the **management board**, who will **review** and **approve** it.

1. What is most challenge defect u came across?

<http://www.softwaretestinghelp.com/manual-and-automation-testing-challenges/>

1. What are test design techniques?

A test design technique basically helps us to select a good set of tests from the total number of all possible tests for a given system. There are many different types of software testing technique, each with its own strengths and weaknesses. Each individual technique is good at finding particular types of defect and relatively poor at finding other types.

Each testing technique falls into one of a number of different categories. Broadly speaking there are two main categories:

[Static technique](http://istqbexamcertification.com/what-is-static-testing-technique/)

[Dynamic technique](http://istqbexamcertification.com/what-is-dynamic-testing-technique/)

Specification-based ([black-box testing](http://istqbexamcertification.com/what-is-black-box-specification-based-also-known-as-behavioral-testing-techniques/), also known as behavioral techniques)

Structure-based ([white-box testing](http://istqbexamcertification.com/what-is-white-box-or-structure-based-or-structural-testing-techniques/) or structural techniques)

[Experience- based](http://istqbexamcertification.com/what-is-experience-based-testing-technique/).

1. If we don’t have time to test call test cases what we will do?
2. How we learn the functionality of system?
3. What are the tools to manage defects/stories?
4. Who will assign the work?
5. What is requirement traceability matrix?
6. what are typical environments we have in projects
7. What are different defect metrics and measurements we prepare in testing?
8. What is development environment?

A development environment is a collection of procedures and tools for developing, testing and debugging an application or program.   
The development environment normally has three server tiers, called development, staging and production. All three tiers together are usually referred to as the DSP.

**Development Server:** Here is where the developer tests code and checks whether the application runs successfully with that code. Once the application has been tested and the developer feels that the code is working fine, the application then moves to the staging server.

**Staging Server**: This environment is made to look exactly like the production server environment. The application is tested on the staging server to check for reliability and to make sure it does not fail on the actual production server. This type of testing on the staging server is the final step before the application could be deployed on a production server. The application needs to be approved in order to deploy it on the production server.

**Production Server:** Once the approval is done, the application then becomes a part of this server.

1. What is QA environment?

A **QA environment** is where you test your upgrade procedure against data, hardware, and software that closely simulate the Production environment and where you allow intended users to test the resulting Wave set application.

1. What is staging environment?

In software, a staging environment is used to test out newer versions of software before it is moved live – into production. A staging environment is meant to have everything as closely replicated to the production environment as possible so that you can maximize your chances of finding any bugs before you release the software in production.

1. What is production environment?

Production environment is a term used mostly by developers to describe the setting where software and other products are actually put into operation for their intended uses by end users. A production environment can be thought of as a real-time setting where programs are run and hardware setups are installed and relied on for organization or commercial daily operations.

1. How to deal the production defects?