## Manual Testing

**Manual Testing**  is a type of software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identify the bugs, issues, and defects in the software application. Manual software testing is the most primitive technique of all testing types and it helps to find critical bugs inthe software application. Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual testing concepts does not require knowledge of any testing tool. One of the Software Testing Fundamental is “**100% Automation is not possible**“. This makes Manual Testing imperative. In this Manual Testing tutorial for beginners, we cover all manual testing concepts and topics in detail.

**Software Testing**

**Software Testing** is a method to check whether the actual software product matches expected requirements and to ensure that software product is[Defect](https://www.guru99.com/defect-management-process.html)free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements. Some prefer saying Software testing definition as a [White Box](https://www.guru99.com/white-box-testing.html) and [Black Box Testing](https://www.guru99.com/black-box-testing.html). In simple terms, Software Testing means the Verification of Application Under Test (AUT). This Software Testing course introduces testing software to the audience and justifies the importance of software testing.In this Software Testing tutorial, you will learn basics of software testing like:

* [What is Software Testing?](https://www.guru99.com/software-testing-introduction-importance.html#1)
* [Why is Software Testing Important?](https://www.guru99.com/software-testing-introduction-importance.html#2)
* [What are the benefits of Software Testing?](https://www.guru99.com/software-testing-introduction-importance.html#3)
* [Testing in Software Engineering](https://www.guru99.com/software-testing-introduction-importance.html#4)
* [Types of Software Testing](https://www.guru99.com/software-testing-introduction-importance.html#5)
* [Testing Strategies in Software Engineering](https://www.guru99.com/software-testing-introduction-importance.html#6)
* [Program Testing](https://www.guru99.com/software-testing-introduction-importance.html#7)

**Why Software Testing is Important?**

**Software Testing is Important** because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

**What is the need of Testing?**

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples.

* In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
* Nissan cars recalled over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accident due to this software failure.
* Starbucks was forced to close about 60 percent of stores in the U.S and Canada due to software failure in its POS system. At one point, the store served coffee for free as they were unable to process the transaction.
* Some of Amazon’s third-party retailers saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.
* Vulnerability in Windows 10. This bug enables users to escape from security sandboxes through a flaw in the win32k system.
* In 2015 fighter plane F-35 fell victim to a software bug, making it unable to detect targets correctly.
* China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocents live
* In 1985, Canada’s Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.
* In April of 1999, a software bug caused the failure of a $1.2 billion military satellite launch, the costliest accident in history
* In May of 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.

**What are the benefits of Software Testing?**

Here are the benefits of using software testing:

* **Cost-Effective:**It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
* **Security:**It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
* **Product quality:**It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
* **Customer Satisfaction:**The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

**Testing in Software Engineering**

As per ANSI/IEEE 1059, **Testing in Software Engineering** is a process of evaluating a software product to find whether the current software product meets the required conditions or not. The testing process involves evaluating the features of the software product for requirements in terms of any missing requirements, bugs or errors, security, reliability and performance.

**Types of Software Testing:**

Typically Testing is classified into three categories.

* Functional Testing
* Non-Functional Testing or [Performance Testing](https://www.guru99.com/performance-testing.html)
* Maintenance (Regression and Maintenance)

**Types of Functional Testing:**

* [Unit Testing](https://www.simform.com/blog/functional-testing-types/#unit)
* [Component Testing](https://www.simform.com/blog/functional-testing-types/#component)
* [Smoke Testing](https://www.simform.com/blog/functional-testing-types/#smoke)
* [Integration Testing](https://www.simform.com/blog/functional-testing-types/#integration)
* [Regression Testing](https://www.simform.com/blog/functional-testing-types/#regression)
* [Sanity Testing](https://www.simform.com/blog/functional-testing-types/#sanity)
* [System Testing](https://www.simform.com/blog/functional-testing-types/#system)
* [User Acceptance Testing](https://www.simform.com/blog/functional-testing-types/#uat)-(UAT)

**Unit Testing**

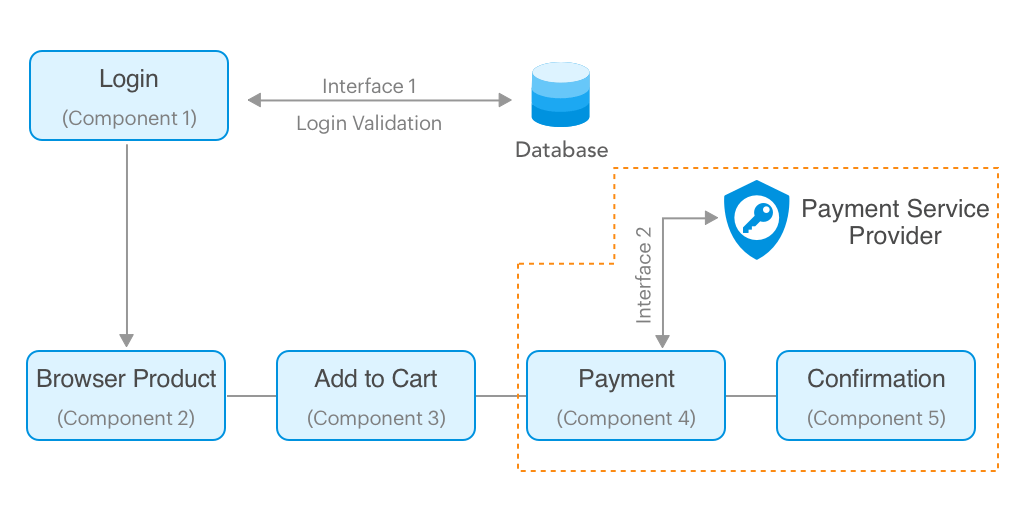
Unit testing ensures that each part of the code developed in a component delivers the desired output. In unit testing, developers only look at the interface and the specification for a component. It provides documentation of code development as each unit of the code is thoroughly tested standalone before progressing to another unit.

Unit tests support functional tests by exercising the code that is most likely to break. If you use functional tests without unit tests, you may experience several smells:

* It’s hard to diagnose failed tests
* Test fixtures work around known issues rather than diagnosing and fixing them

**Component Testing**

Testing a module or component independently to verify its expected output is called component testing. Generally, component testing is done to verify the functionality and/or usability of a component but not restricted to only these. A component can be of anything which can take input(s) and delivers some output. For example, the module of code, web page, screens and even a system inside a bigger system is a component to it.



From the above picture, Let’s see what all we can test in component 1 (login) separately:

* Testing the UI part for usability and accessibility
* Testing the Page loading to ensure performance
* Trying SQL injection through the UI components to ensure security
* Testing the login functionality with valid and invalid user credentials

**Smoke Testing**

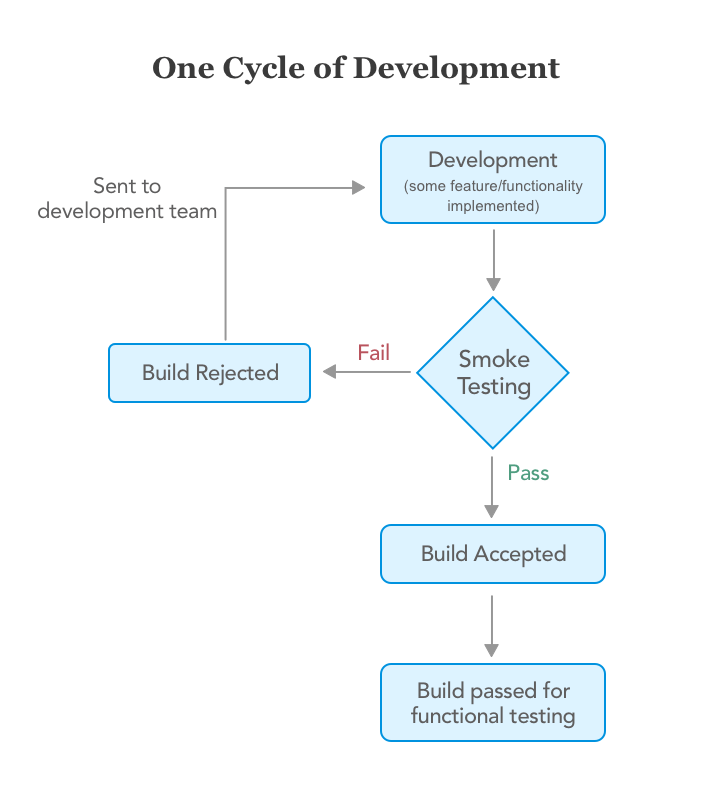
Smoke testing is performed on the ‘new’ build given by developers to QA team to verify if the basic functionalities are working or not. It is one of the important functional testing types. This should be the first test to be done on any new build. In smoke testing, the test cases chosen cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionality of the system is working fine.

If the build passes the smoke testing then it is considered as a stable build. On the stable build, QA team performs functional testing for the newly added features/functionality and then performs regression testing depending upon the situation. But if the build is not stable i.e. the smoke testing fails then the build is rejected and forwarded to the development team to fix the build issues and create a new build. Let’s understand it better with an example.

We’ve built an **Employee portal application** for our client. As we follow continuous testing we had to test each build right after its development. The client wanted us to build the portal which consists of features like leave application, leave reports, store employees’ data, etc.

First, developers build a leave application feature and passed to QA for testing. The QA team examined that the entire build required 80-100 test cases for all the scenarios:

* Login
* Show total leaves count and types
* Testing of the calendar while selecting the date
* Select date
* User should be able to fill the required information. i.e., a reason of the leave
* After applying request sent to the manager for approval
* Manager approves the leave
* Employee gets notified
* Leave gets deducted from the total count
* Logout



Here smoke testing comes in picture. Instead of testing all the functionalities, they decided to test only critical functionalities which had only 20 test cases. These test cases covered the following scenarios:

* Login
* Select date
* Fill other details
* Request sent to the manager after clicking the button

As you can see we have taken only the main features for testing which were critical. For example, if an employee can’t select the date then there’s no need for further testing. This saves the developers’ time of fixing bugs.

**Integration Testing**

Integration testing is performed to test individual components to check how they function together. In other words, it is performed to test the modules which are working fine individually and do not show bugs when integrated. It is the most common functional testing type and performed as automated testing.

Generally, developers build different modules of the system/software simultaneously and don’t focus on others. They perform extensive black and white box functional verification, commonly known as unit tests, on the individual modules. Integration tests cause data and operational commands to flow between modules which means that they have to act as parts of a whole system rather than individual components. This typically uncovers issues with UI operations, data formats, operation timing, API calls, and database access and user interface operation.

Recommended Read: [Automated Functional Testing](https://www.simform.com/automated-functional-testing/)

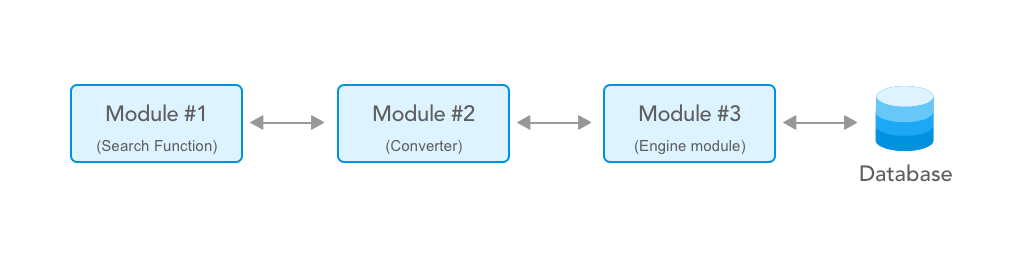
Let’s take an example of another project of search functionality in the e-commerce site where it shows the results based on the text entered by users. The complete search function works when developers build the following four modules.

**Module #1:** This is the search box visible to users where they can enter text and click the search button.

**Module #2:** It’s a converter or in simple terms program which converts entered text into XML.

**Module #3:** This is called Engine module which sends XML data to the database.

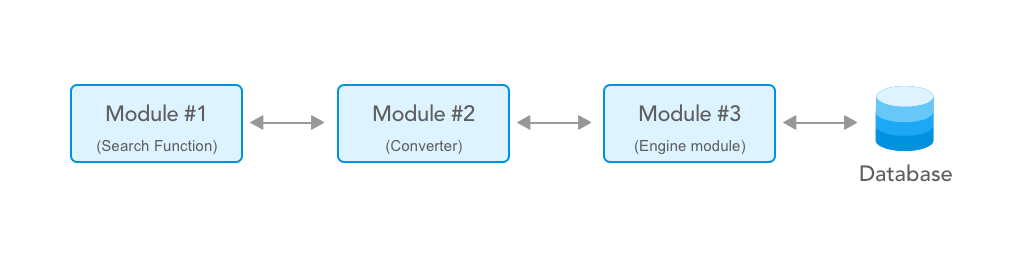
**Module #4:** Database



In our scenario, the data entered in the search function (module #1) gets converted into XML by module #2. The EN module(module #3) reads the resultant XML file generated by module 2 and extracts the SQL from it and queries into the database. The EN module also receives the result set and converts it into an XML file and returns it back to the UI module which converts the results in user readable form and displays it.So where does Integration testing comes into the picture? Well, testing whether the information/data is flowing correctly or not will be your integration testing, which in this case would be validating the XML files. Are the XML files generated correctly? Do they have the correct data? Has the data been transferred correctly from one module to another? All these things will be tested as part of Integration testing.

Checking of data transfers between two components is called as an **Interface Testing**. It is a part of integration testing. Interface testing includes testing of interfaces such as web services, APIs, connection strings that connect two components in the application. These interfaces don’t have a UI but takes an input and delivers output (do not confuse it with Unit testing).

Interface testing is done to check that the different components of the application/ system being developed are in sync with each other or not. In technical terms, interface testing helps determine that different functions like data transfer between the different elements in the system are happening according to the way they were designed to happen.



Let’s see how to test the Interface 2 in the above example considering that the interface takes an XML file as input from Component 4 and delivers a JSON file as output with a response message from the payment service provider. To test this interface we do not need to worry about the functionality of component 4. All we need is the specification of the XML file from Component 4 and the specification of JSON output. With the help of these specifications, we can create the sample input XML files and feed into the interface. The interface will pass the input to the payment service provider and returns an output JSON file. So in our example, validating the input file and the output file with the requirement is called Interface Testing.

**Regression Testing**

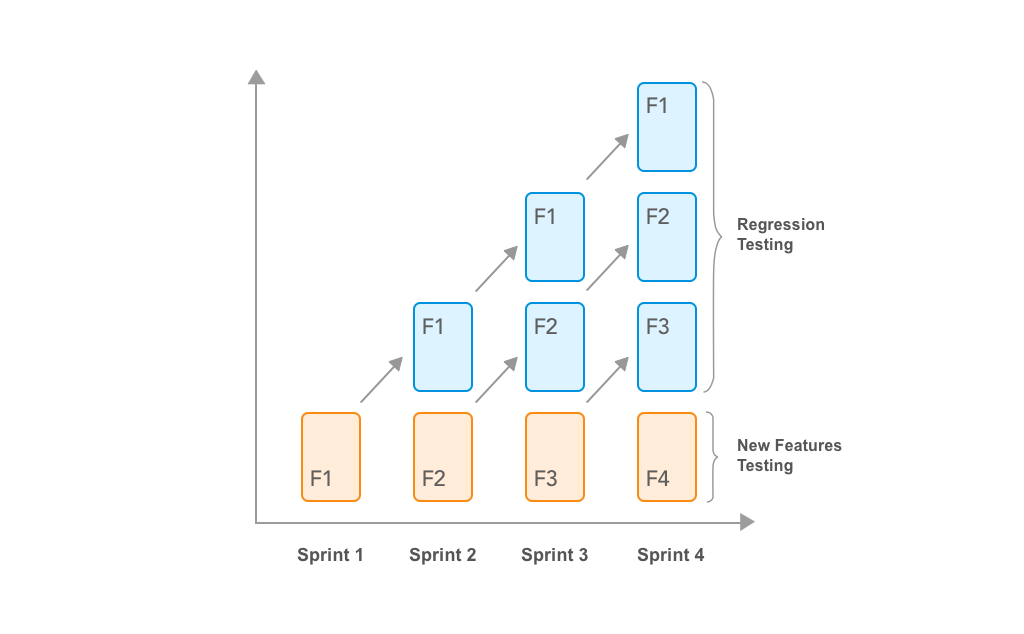
Whenever developers change or modify the functionality/feature, there’s a huge possibility that these updates may cause unexpected behaviors. Regression testing is performed to make sure that a change or addition hasn’t broken any of the existing functionality. Its purpose is to find bugs that may have been accidentally introduced into the existing build and to ensure that previously removed bugs continue to stay dead. There are many [functional testing tools](https://www.simform.com/functional-testing-tools/) available which support regression testing.

Project one 1.0 ( 2 month)

Project one 1.1

Project one 1.2. So on.

Project one 2.0 ( main Release)



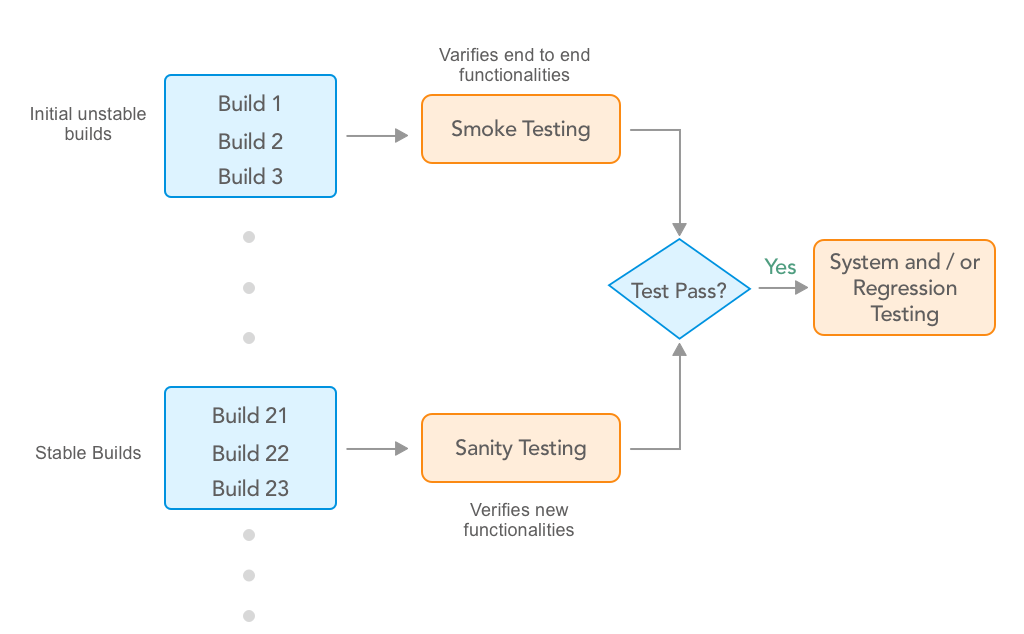
Let’s understand it by continuing our example of the leave management system. Let’s assume that developers have built a new feature(build 2) which shows the report of the employee’s leave history. Now, testers need to test this new feature by performing smoke testing with new test cases. Now, testers need to perform regression testing on build 2(Leave reports) to ensure that the code carried over from Build 1 (Leave application) behaves correctly. Here the main principle is reusing tests derived from Build 1. Also, the test case for build 2 would be a subset of build 1.

Regression testing can become a challenge for the testers as well. Here are some of the reasons:

1. The Number of test cases in the regression suite increases with each new feature.
2. Sometimes, the execution of the entire regression test suite becomes difficult due to time and budget constraints.
3. Minimizing the test suite while achieving maximum [test coverage](https://www.simform.com/test-coverage/) is not a cake walk.
4. Determination of frequency of Regression Tests after every modification or every build update or after a bunch of bug fixes is always a challenge.

**Sanity Testing**

When a new build is received with minor modifications, instead of running a thorough regression test suite we perform a sanity test. It determines that the modifications have actually fixed the issues and no further issues have been introduced by the fixes. Sanity testing is generally a subset of regression testing and a group of test cases executed that are related to the changes made to the product. Many testers get confused between sanity testing and smoke testing. Refer below image to understand the basic difference.



Let’s continue with the above example of the leave management system. Let’s assume that developers have released the build 2 with some other features. Now first we need to perform smoke testing and check whether the overall functionality is working fine. Here we are assuming that the build 2 has passed the smoke test. Now, we know that we’ve reported for “date selection” in build 1 and it has been solved in build 2. In sanity testing we’ll only test “date selection” functionality and whether it affects other functionalities.

**System Testing**

System testing is testing conducted on a complete, integrated system to evaluate its compliance with the specified requirements.

After the completion of the integration testing, the product is passed for system testing. System testing is undertaken by independent testers who haven’t played a role in developing the program. This testing is performed in an environment that closely mirrors production. System Testing is very important because it verifies that the application meets the technical, functional, and business requirements that were set by the stakeholder.

In our example, we can perform system testing when all the modules are developed and passed integration successfully. For example, the complete product may include features like leave application, reports, employee details, performance tracker, etc.

**User Acceptance Testing**

User acceptance testing (UAT) is the last phase of the software testing process. In UAT actual software/app users test the software to make sure it can handle required tasks in real-world scenarios. Generally, it is performed at the time of product delivery to stakeholders as a final checkpoint among all functional testing types.

From starting to deployment, the software/app undergoes through various types of testing by testing team and developers. The end goal of all the efforts is to deliver a working software/app that fulfills users’ requirements and the client’s expectations. Both the teams become so familiar with the application that they might become a victim of tunnel vision. They are fully aware of workarounds and may skip certain scenarios which might be critical for end users.

The users are naive about how the application works. They are focused on ‘How the application should behave?’ They use the application with a fresh mind and whether it is intuitive to follow or not. UAT is based on user stories and establishes how well it meets their requirements. Users do not use ‘Test to Break’ approach while doing user acceptance testing. Rather, UAT is a measure of how good your application performs in normal scenarios.

Conclusion

At Simform, we believe that functional testing plays a crucial role in transforming client’s understanding of customer needs into applications that meet their requirements. Releasing applications with serious functional shortcomings can lead to disastrous consequences. When all the above functional testing types are performed at the right time in the development process, functional testing ensures the delivery of a quality product. Know more about our [Software Testing Services](https://www.simform.com/services/software-testing/).

# **SDLC - Overview**

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

* SDLC is the acronym of Software Development Life Cycle.
* It is also called as Software Development Process.
* SDLC is a framework defining tasks performed at each step in the software development process.
* ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

## What is 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 following figure is a graphical representation of the various stages of a typical SDLC.



A typical Software Development Life Cycle consists of the following stages −

### **Stage 1: Planning and Requirement Analysis**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

### **Stage 2: Defining Requirements**

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

### **Stage 3: Designing the Product Architecture**

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

### **Stage 4: Building or Developing the Product-----------( testcase preparation)**

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

### **Stage 5: Testing the Product\_(execution)**

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

### **Stage 6: Deployment in the Market and Maintenance**

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

## SDLC Models

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry −

* Waterfall Model
* Iterative Model
* Spiral Model
* V-Model
* Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.etc…

# **DLC - Waterfall Model**

The Waterfall Model was the 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 before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

## Waterfall Model - Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are −

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

## Waterfall Model - Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

* Requirements are very well documented, clear and fixed.
* Product definition is stable.
* Technology is understood and is not dynamic.
* There are no ambiguous requirements.
* Ample resources with required expertise are available to support the product.
* The project is short.

## Waterfall Model - Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

* Simple and easy to understand and use
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Phases are processed and completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.
* Well understood milestones.
* Easy to arrange tasks.
* Process and results are well documented.

## Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.
* Adjusting scope during the life cycle can end a project.
* Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

# **SDLC - Iterative Model**

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

## Iterative Model - Design

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

The following illustration is a representation of the Iterative and Incremental model –

If the build passes the smoke testing then it is considered as a stable build. On the stable build, QA team performs functional testing for the newly added features/functionality and then performs regression testing depending upon the situation. But if the build is not stable i.e. the smoke testing fails then the build is rejected and forwarded to the development team to fix the build issues and create a new build. Let’s understand it better with an example.

We’ve built an **Employee portal application** for our client. As we follow continuous testing we had to test each build right after its development. The client wanted us to build the portal which consists of features like leave application, leave reports, store employees’ data, etc.First, developers build a leave application feature and passed to QA for testing. The QA team examined that the entire build required 80-100 test cases for all the scenarios:

* Login
* Show total leaves count and types
* Testing of the calendar while selecting the date
* Select date
* User should be able to fill the required information. i.e., a reason of the leave
* After applying request sent to the manager for approval
* Manager approves the leave
* Employee gets notified
* Leave gets deducted from the total count
* Logout



Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

In this incremental model, the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

The key to a successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests must be repeated and extended to verify each version of the software.

## Iterative Model - Application

Like other SDLC models, Iterative and incremental development has some specific applications in the software industry. This model is most often used in the following scenarios −

* Requirements of the complete system are clearly defined and understood.
* Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
* There is a time to the market constraint.
* A new technology is being used and is being learnt by the development team while working on the project.
* Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
* There are some high-risk features and goals which may change in the future.

## Iterative Model - Pros and Cons

The advantage of this model is that there is a working model of the system at a very early stage of development, which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.

The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

The advantages of the Iterative and Incremental SDLC Model are as follows −

* Some working functionality can be developed quickly and early in the life cycle.
* Results are obtained early and periodically.
* Parallel development can be planned.
* Progress can be measured.
* Less costly to change the scope/requirements.
* Testing and debugging during smaller iteration is easy.
* Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
* Easier to manage risk - High risk part is done first.
* With every increment, operational product is delivered.
* Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
* Risk analysis is better.
* It supports changing requirements.
* Initial Operating time is less.
* Better suited for large and mission-critical projects.
* During the life cycle, software is produced early which facilitates customer evaluation and feedback.

The disadvantages of the Iterative and Incremental SDLC Model are as follows −

* More resources may be required.
* Although cost of change is lesser, but it is not very suitable for changing requirements.
* More management attention is required.
* System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
* Defining increments may require definition of the complete system.
* Not suitable for smaller projects.
* Management complexity is more.
* End of project may not be known which is a risk.
* Highly skilled resources are required for risk analysis.
* Projects progress is highly dependent upon the risk analysis phase.

Agile methodology

Diagram

Description automatically generated

Advantages of Agile Methodology

1. Customer satisfaction is rapid, continuous development and delivery of useful software.
2. Customer, Developer, and Product Owner interact regularly to emphasize rather than processes and tools.
3. Product is developed fast and frequently delivered (weeks rather than months.)
4. A face-to-face conversation is the best form of communication.
5. It continuously gave attention to technical excellence and good design.
6. Daily and close cooperation between business people and developers.
7. Regular adaptation to changing circumstances.
8. Even late changes in requirements are welcomed.

Disadvantages of Agile methodology:

1. It is not useful for small development projects.
2. There is a lack of intensity on necessary designing and documentation.
3. It requires an expert project member to take crucial decisions in the meeting.
4. Cost of Agile development methodology is slightly more as compared to other development methodology.
5. The project can quickly go out off track if the project manager is not clear about requirements and what outcome he/she wants.

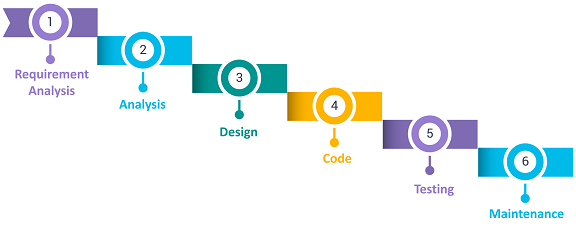
## ****Software Testing Models****

[Software Testing](https://www.edureka.co/blog/what-is-software-testing/" \t "_blank) is an integral part of the [software development life cycle](https://www.edureka.co/blog/software-testing-tutorial/). There are different models or approaches you can use in the software development process where each model has its own advantages and disadvantages. So, You must choose a particular model depending on the project deliverables and complexity of the project.

The different Software Testing Models are:

### **Waterfall Model**

This is the most basic [software development life cycle](https://www.edureka.co/blog/software-testing-life-cycle/) process which is followed broadly in the industry. In this model, the developers follow a sequence of processes downwards towards the ultimate goal. It is like a waterfall where there are various phases involved.



The different phases in the [**waterfall model**](https://www.edureka.co/blog/waterfall-vs-agile/)are:

* Requirement Analysis
* Analysis phase
* Software design
* Programmed implementation
* Testing
* Maintenance

**Advantages**

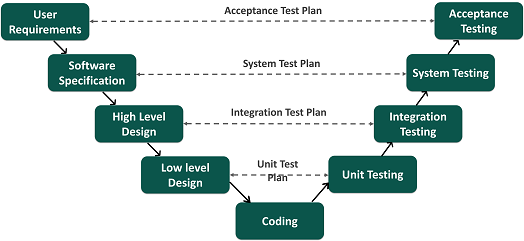
* It is easy to implement and maintain.
* The initial phase of rigorous scrutiny of requirements and systems helps in saving time later in the developmental phase.
* The requirement of resources is minimal and testing is done after each phase has been completed.

**Disadvantages**

* It is not possible to alter or update requirements.
* Once you move into the next phase you cannot make changes.
* You cannot start the next phase until the previous phase is completed.

### **V Model**

The V Model is considered superior to the waterfall model. In this model, the development and test execution activities are carried out side by side in the downhill and uphill shape. Also, testing starts at the unit level and spreads towards the integration of the entire system.



**Advantages**

* It is easy to use since testing activities like planning and test designing are done before coding.
* This model enhances the chances of success and saves time.
* Defects are mostly found at an early stage and downward flow of defects is generally avoided.

**Disadvantages**

* It is a rigid model.
* The software is developed during the implementation phase so early prototypes of the product are not available.
* If there are changes in the midway, you need to update the test document.

**The advantages of the V-Model method are as follows −**

* This is a highly-disciplined model and Phases are completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Simple and easy to understand and use( water fall model).
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

**The disadvantages of the V-Model method are as follows −**

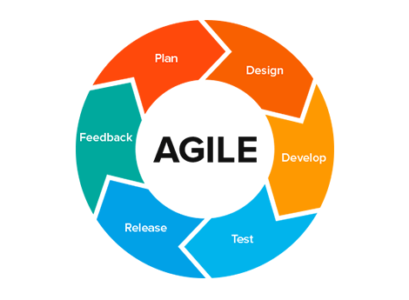
* High risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing.
* Once an application is in the testing stage, it is difficult to go back and change a functionality.
* No working software is produced until late during the life cycle.

### **Agile Model**

In the Agile model, requirements and solutions evolve through collaboration between various cross-functional teams. It is also known as an iterative and incremental model. The agile software testing model focus on process adaptability and customer satisfaction by rapid delivery of working software product and by breaking the product into small incremental builds.

**Advantages**

* It ensures customer satisfaction with rapid and continuous development of deliverables.
* The continuous interaction between the customers, developers, and testers makes it a flexible model.
* You can develop the working software quickly and adapt to changing requirements regularly.

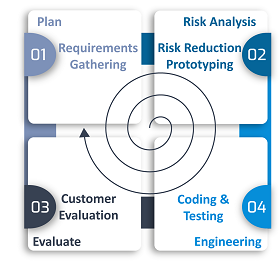


**Disadvantages**

* It is difficult to assess the effort required at the beginning of the cycle for large and complex software development cases.
* Due to continuous interaction with the customer, the project can go off track if the customer is not clear about the goals.

**Spiral Model**

This software testing model is similar to the [Agile model](https://www.edureka.co/blog/what-is-agile-testing/), but with more emphasis on risk analysis. The different phases of the spiral model include planning, risk analysis, engineering, and evaluation. In this case, you need to gather the requirements and perform the risk assessment at the base level and every upper spiral builds on it.



**Advantages**

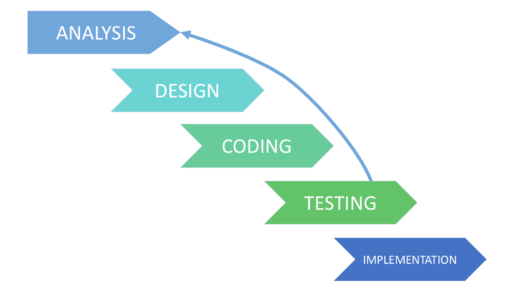
* It is suitable for complex and large systems.
* You can add functionalities depending on the changed circumstances.
* Software is produced early in the cycle.

**Disadvantages**

* It is a costly model which requires highly specialized expertise in risk analysis
* It does not work well on simpler projects.

### **Iterative Model**

The Iterative model does not need a full list of requirements before beginning the project. The development process starts with the requirements of the functional part, which can be expanded later. The process is repetitive and allows new versions of the product for every cycle. Every iteration includes the development of a separate component of the system which is added to the functional developed earlier.



**Advantages**

* It is easier to control the risks as high-risk tasks are completed first.
* The progress is easily measurable.
* Problems and risks defined within one iteration can be prevented in the next sprints.

**Disadvantages**

* Iterative model requires more resources than the waterfall model.
* The process is difficult to manage.
* The risks may not be completely determined even at the final stage of the project.

# **Iterative Model**

In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.



### **The various phases of Iterative model are as follows:**

**1. Requirement gathering & analysis:** In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfill or not. Analyst checks that need will achieve within budget or not. After all of this, the software team skips to the next phase.

**2. Design:** In the design phase, team design the software by the different diagrams like Data Flow diagram, activity diagram, class diagram, state transition diagram, etc.

**3. Implementation:** In the implementation, requirements are written in the coding language and transformed into computer programs which are called Software.

**4. Testing:** After completing the coding phase, software testing starts using different test methods. There are many test methods, but the most common are white box, black box, and grey box test methods.

**5. Deployment:** After completing all the phases, software is deployed to its work environment.

**6. Review:** In this phase, after the product deployment, review phase is performed to check the behavior and validity of the developed product. And if there are any error found then the process starts again from the requirement gathering.

**7. Maintenance:** In the maintenance phase, after deployment of the software in the working environment there may be some bugs, some errors or new updates are required. Maintenance involves debugging and new addition options.

## When to use the Iterative Model?

1. When requirements are defined clearly and easy to understand.
2. When the software application is large.
3. When there is a requirement of changes in future.

## Advantage(Pros) of Iterative Model:

1. Testing and debugging during smaller iteration is easy.
2. A Parallel development can plan.
3. It is easily acceptable to ever-changing needs of the project.
4. Risks are identified and resolved during iteration.
5. Limited time spent on documentation and extra time on designing.

## Disadvantage(Cons) of Iterative Model:

1. It is not suitable for smaller projects.
2. More Resources may be required.
3. Design can be changed again and again because of imperfect requirements.
4. Requirement changes can cause over budget.
5. Project completion date not confirmed because of changing requirements.

## What is Verification in Software Testing?

**Verification in Software Testing** is a process of checking documents, design, code, and program in order to check if the software has been built according to the requirements or not. The main goal of verification process is to ensure quality of software application, design, architecture etc. The verification process involves activities like reviews, walk-throughs and inspection.

## What is Validation in Software Testing?

**Validation in Software Engineering** is a dynamic mechanism of testing and validating if the software product actually meets the exact needs of the customer or not. The process helps to ensure that the software fulfills the desired use in an appropriate environment. The validation process involves activities like unit testing, integration testing, system testing and user acceptance testing.

Difference between the Verification And Validation

* Verification process includes checking of documents, design, code and program whereas Validation process includes testing and validation of the actual product.
* Verification does not involve code execution while Validation involves code execution.
* Verification uses methods like reviews, walkthroughs, inspections and desk-checking whereas Validation uses methods like black box testing, white box testing and non-functional testing.
* Verification checks whether the software confirms a specification whereas Validation checks whether the software meets the requirements and expectations.
* Verification finds the bugs early in the development cycle whereas Validation finds the bugs that verification cannot catch.
* Comparing validation and verification in software testing, Verification process targets on software architecture, design, database, etc. while Validation process targets the actual software product.
* Verification is done by the QA team while Validation is done by the involvement of testing team with QA team.
* Comparing Verification vs. Validation testing, Verification process comes before validation whereas Validation process comes after verification.
* Verification process includes checking of documents, design, code and program whereas Validation process includes testing and validation of the actual product.
* Verification does not involve code execution while Validation involves code execution.
* Verification uses methods like reviews, walkthroughs, inspections and desk-checking whereas Validation uses methods like black box testing, white box testing and non-functional testing.
* Verification checks whether the software confirms a specification whereas Validation checks whether the software meets the requirements and expectations.
* Verification finds the bugs early in the development cycle whereas Validation finds the bugs that verification cannot catch.
* Comparing validation and verification in software testing, Verification process targets on software architecture, design, database, etc. while Validation process targets the actual software product.
* Verification is done by the QA team while Validation is done by the involvement of testing team with QA team.
* Comparing Verification vs. Validation testing, Verification process comes before validation whereas Validation process comes after verification.

Graphical user interface, text, application, email

Description automatically generated

**What is Static Testing?**

**Static Testing** is a type of software testing in which software application is tested without code execution. Manual or automated reviews of code, requirement documents and document design are done in order to find the errors. The main objective of static testing is to improve the quality of software applications by finding errors in early stages of software development process.

Static testing involves manual or automated reviews of the documents. This review is done during an initial phase of testing to catch[Defect](https://www.guru99.com/defect-management-process.html)early in STLC. It examines work documents and provides review comments. It is also called Non-execution testing or verification testing.

Examples of Work documents-

* Requirement specifications
* Design document
* Source Code
* Test Plans
* Test Cases
* Test Scripts
* Help or User document
* Web Page content

**What is Dynamic Testing?**

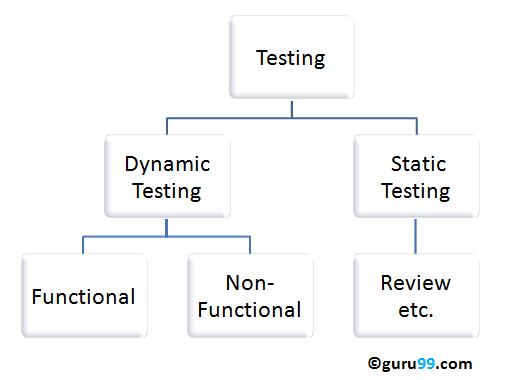
Under**Dynamic Testing**, a code is executed. It checks for functional behavior of software system, memory/cpu usage and overall performance of the system. Hence the name “Dynamic”

The main objective of this testing is to confirm that the software product works in conformance with the business requirements. This testing is also called an Execution technique or validation testing.

Dynamic testing executes the software and validates the output with the expected outcome. Dynamic testing is performed at all levels of testing and it can be either black or white box testing.

**KEY DIFFERENCE**

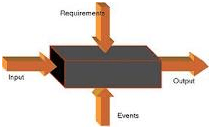
* Static testing was done without executing the program whereas dynamic testing is done by executing the program.
* Static testing checks the code, requirement documents, and design documents to find errors whereas Dynamic testing checks the functional behavior of software system, memory/CPU usage and overall performance of the system.
* Static testing is about the prevention of defects whereas dynamic testing is about finding and fixing the defects.
* Static testing does the verification process while Dynamic testing does the validation process.
* Static testing is performed before compilation whereas Dynamic testing is performed after compilation.
* Static testing techniques are structural and statement coverage while Dynamic testing techniques are Boundary Value Analysis & Equivalence Partitioning.



**Static Testing Techniques:**

* **Informal Reviews:**This is one of the type of review which doesn’t follow any process to find errors in the document. Under this technique, you just review the document and give informal comments on it.
* **Technical Reviews:**A team consisting of your peers,   review the technical specification of the software product and checks whether it is suitable for the project. They try to find any discrepancies in the specifications and standards followed. This review concentrates mainly on the technical documentation related to the software such as Test Strategy,[Test Plan](https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html)and requirement specification documents.
* **Walkthrough:**The author of the work product explains the product to his team. Participants can ask questions if any.  A meeting is led by the author*.* Scribe makes note of review comments
* **Inspection:**The main purpose is to find defects and meeting is led by a trained moderator. This review is a formal type of review where it follows a strict process to find the defects. Reviewers have a checklist to review the work products.They record the defect and inform the participants to rectify those errors.
* **Static code Review:** This is a systematic review of the software source code without executing the code. It checks the syntax of the code, coding standards, code optimization, etc. This is also termed as white box testing.This review can be done at any point during development.

**Dynamic Testing Techniques:**



* **Unit Testing:** Under Unit Testing, individual units or modules are tested by the developers. It involves testing of source code by developers.
* **Integration Testing:**Individual modules are grouped together and tested by the developers. The purpose is to determine what modules are working as expected once they are integrated.
* **System Testing:**[System Testing](https://www.guru99.com/system-testing.html) is performed on the whole system by checking whether the system or application meets the requirement specification document.

**Difference between Static and Dynamic Testing:**

|  |  |
| --- | --- |
| **Static Testing** | **Dynamic Testing** |
| Testing was done without executing the program | Testing is done by executing the program |
| This testing does the verification process | Dynamic testing does the validation process |
| Static testing is about prevention of defects | Dynamic testing is about finding and fixing the defects |
| Static testing gives an assessment of code and documentation | Dynamic testing gives bugs/bottlenecks in the software system. |
| Static testing involves a checklist and process to be followed | Dynamic testing involves test cases for execution |
| This testing can be performed before compilation | Dynamic testing is performed after compilation |
| Static testing covers the structural and statement coverage testing | Dynamic testing techniques are Boundary Value Analysis & Equivalence Partitioning. |
| Cost of finding defects and fixing is less | Cost of finding and fixing defects is high |
| Return on investment will be high as this process involved at an early stage | Return on investment will be low as this process involves after the development phase |
| More reviews  comments are highly recommended for good quality | More defects are highly recommended for good quality. |
| Requires loads of meetings | Comparatively requires lesser meetings |

**Black Box Testing**

**Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.



The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

**How to do Black Box Testing**

Here are the generic steps followed to carry out any type of Black Box Testing.

* Initially, the requirements and specifications of the system are examined.
* Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
* Tester determines expected outputs for all those inputs.
* Software tester constructs test cases with the selected inputs.
* The test cases are executed.
* Software tester compares the actual outputs with the expected outputs.
* Defects if any are fixed and re-tested.

**Types of Black Box Testing**

There are many types of Black Box Testing but the following are the prominent ones –

* **Functional testing** – This black box testing type is related to the functional requirements of a system; it is done by software testers.
* **Non-functional testing**– This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
* **Regression testing**– [Regression Testing](https://www.guru99.com/regression-testing.html) is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

**Tools used for Black Box Testing:**

Tools used for Black box testing largely depends on the type of black box testing you are doing.

* For Functional/ Regression Tests you can use – [QTP](https://www.guru99.com/quick-test-professional-qtp-tutorial.html), [Selenium](https://www.guru99.com/selenium-tutorial.html)
* For Non-Functional Tests, you can use – [Load Runner](https://www.guru99.com/loadrunner-v12-tutorials.html), [Jmeter](https://www.guru99.com/jmeter-tutorials.html" \t "_blank)

**Black Box Testing Techniques**

Following are the prominent[Test Strategy](https://www.guru99.com/how-to-create-test-strategy-document.html)amongst the many used in Black box Testing

* **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
* **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.
* **Decision Table Testing**: A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

|  |  |
| --- | --- |
| **Black Box Testing** | **White Box Testing** |
| the main focus of black box testing is on the validation of your functional requirements. | [White Box Testing](https://www.guru99.com/white-box-testing.html) (Unit Testing) validates internal structure and working of your software code |
| Black box testing gives abstraction from code and focuses on testing effort on the software system behavior. | To conduct White Box Testing, knowledge of underlying programming language is essential. Current day software systems use a variety of programming languages and technologies and its not possible to know all of them. |
| Black box testing facilitates testing communication amongst modules | White box testing does not facilitate testing communication amongst modules |

**Comparison of Black Box and White Box Testing:**

**Black Box Testing and Software Development Life Cycle (SDLC)**

Black box testing has its own life cycle called Software Testing Life Cycle ([STLC](https://www.guru99.com/software-testing-life-cycle.html)) and it is relative to every stage of Software Development Life Cycle of Software Engineering.

* **Requirement** – This is the initial stage of SDLC and in this stage, a requirement is gathered. Software testers also take part in this stage.
* **Test Planning & Analysis** – [Testing Types](https://www.guru99.com/types-of-software-testing.html) applicable to the project are determined. A[Test Plan](https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html)is created which determines possible project risks and their mitigation.
* **Design** – In this stage Test cases/scripts are created on the basis of software requirement documents
* **Test Execution**– In this stage Test Cases prepared are executed. Bugs if any are fixed and re-tested.

**White Box Testing**

**White Box Testing** is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability & Security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing. It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

The term “WhiteBox” was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software’s outer shell (or “box”) into its inner workings. Likewise, the “black box” in “[Black Box Testing](https://www.guru99.com/black-box-testing.html)” symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.  
In this white box testing tutorial, you will learn-

* [What is White Box Testing?](https://www.guru99.com/white-box-testing.html#1)
* [What do you verify in White Box Testing?](https://www.guru99.com/white-box-testing.html#2)
* [How do you perform White Box Testing?](https://www.guru99.com/white-box-testing.html#3)
* [WhiteBox Testing Example](https://www.guru99.com/white-box-testing.html#4)
* [White Box Testing Techniques](https://www.guru99.com/white-box-testing.html#5)
* [Types of White Box Testing](https://www.guru99.com/white-box-testing.html#6)
* [White Box Testing Tools](https://www.guru99.com/white-box-testing.html#9)
* [Advantages of White Box Testing](https://www.guru99.com/white-box-testing.html#7)
* [Disadvantages of WhiteBox Testing](https://www.guru99.com/white-box-testing.html#8)

**What do you verify in White Box Testing?**

White box testing involves the testing of the software code for the following:

* Internal security holes
* Broken or poorly structured paths in the coding processes
* The flow of specific inputs through the code
* Expected output
* The functionality of conditional loops
* Testing of each statement, object, and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

**How do you perform White Box Testing?**

To give you a simplified explanation of white box testing, we have divided it into **two basic steps**. This is what testers do when testing an application using the white box testing technique:

STEP 1) UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2) CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application’s source code for proper flow and structure. One way is by writing more code to test the application’s source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include [Manual Testing](https://www.guru99.com/manual-testing.html), trial, and error testing and the use of testing tools as we will explain further on in this article.

**WhiteBox Testing Example**

Consider the following piece of code

Printme (int a, int b) { ------------ Printme is a function

int result = a+ b;

If (result> 0)

Print ("Positive", result)

Else

Print ("Negative", result)

} ----------- End of the source code

The goal of WhiteBox testing in software engineering is to verify all the decision branches, loops, statements in the code.

To exercise the statements in the above white box testing example, WhiteBox test cases would be

* A = 1, B = 1
* A = -1, B = -3

**White Box Testing Techniques**

A major White box testing technique is Code Coverage analysis. Code Coverage analysis eliminates gaps in a [Test Case](https://www.guru99.com/test-case.html) suite. It identifies areas of a program that are not exercised by a set of test cases. Once gaps are identified, you create test cases to verify untested parts of the code, thereby increasing the quality of the software product

There are automated tools available to perform Code coverage analysis. Below are a few coverage analysis techniques a box tester can use:

**Statement Coverage**:- This technique requires every possible statement in the code to be tested at least once during the testing process of [software engineering](https://www.guru99.com/what-is-software-engineering.html).

**Branch Coverage –**This technique checks every possible path (if-else and other conditional loops) of a software application.

Apart from above, there are numerous coverage types such as Condition Coverage, Multiple Condition Coverage, Path Coverage, Function Coverage etc. Each technique has its own merits and attempts to test (cover) all parts of software code. **Using Statement and Branch coverage you generally attain 80-90% code coverage which is sufficient.**

Following are important WhiteBox Testing Techniques:

* Statement Coverage
* Decision Coverage
* Branch Coverage
* Condition Coverage
* Multiple Condition Coverage
* Finite State Machine Coverage
* Path Coverage
* Control flow testing
* Data flow testing

## Grey Box Testing

**Grey Box Testing** or Gray box testing is a software testing technique to test a software product or application with partial knowledge of internal structure of the application. The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of applications.In this process, context-specific errors that are related to web systems are commonly identified. It increases the testing coverage by concentrating on all of the layers of any complex system.Gray Box Testing is a software testing method, which is a combination of both [White Box Testing](https://www.guru99.com/white-box-testing.html) and Black Box Testing method.

* In White Box testing internal structure (code) is known
* In Black Box testing internal structure (code) is unknown
* In Grey Box Testing internal structure (code) is partially known

In Software Engineering, Gray Box Testing gives the ability to test both sides of an application, presentation layer as well as the code part. It is primarily useful in [Integration Testing](https://www.guru99.com/integration-testing.html) and [Penetration Testing](https://www.guru99.com/learn-penetration-testing.html).

**Example of Gray Box Testing:** While testing websites feature like links or orphan links, if tester encounters any problem with these links, then he can make the changes straightaway in HTML code and can check in real time.

## Why Gray Box Testing

Gray Box Testing is performed for the following reason,

* It provides combined benefits of both black box testing and white box testing both
* It combines the input of developers as well as testers and improves overall product quality
* It reduces the overhead of long process of testing functional and non-functional types
* It gives enough free time for a developer to fix defects
* Testing is done from the user point of view rather than a designer point of view

## Gray Box Testing Strategy

To perform Gray box testing, it is not necessary that the tester has the access to the source code. A test is designed based on the knowledge of algorithm, architectures, internal states, or other high -level descriptions of the program behavior.

To perform Gray box Testing-

* It applies a straightforward technique of black box testing
* It is based on requirement test case generation, as such, it presets all the conditions before the program is tested by assertion method.

##### Techniques used for Grey box Testing are-

* **Matrix Testing:**This testing technique involves defining all the variables that exist in their programs.
* **Regression Testing**: To check whether the change in the previous version has regressed other aspects of the program in the new version. It will be done by testing strategies like retest all, retest risky use cases, retest within a firewall.
* **Orthogonal Array Testing or OAT**: It provides maximum code coverage with minimum test cases.
* **Pattern Testing:** This testing is performed on the historical data of the previous system defects. Unlike black box testing, gray box testing digs within the code and determines why the failure happened

Usually, Grey box methodology uses automated software testing tools to conduct the testing. Stubs and module drivers are created to relieve tester to manually generate the code.

##### Steps to perform Grey box Testing are:

* **Step 1**: Identify inputs
* **Step 2**: Identify the outputs
* **Step 3**: Identify the major paths
* **Step 4**: Identify Subfunctions
* **Step 5**: Develop inputs for Subfunctions
* **Step 6**: Develop outputs for Subfunctions
* **Step 7**: Execute test case for Subfunctions
* **Step 8**: Verify the correct result for Subfunctions
* **Step 9**: Repeat steps 4 & 8 for other Subfunctions
* **Step 10**: Repeat steps 7 & 8 for other Subfunctions

The test cases for grey box testing may include, GUI related, Security related, Database related, Browser related, Operational system related, etc.

## Gray Box Testing Challenges

* When a component under test encounter a failure of some kind may lead to abortion of the ongoing operation
* When test executes in full but the content of the result is incorrect.

**Summary:**

* The overall cost of system defects can be reduced and prevented from passing further with Grey box testing
* Grey box testing is suited more for GUI,[Functional Testing](https://www.guru99.com/functional-testing.html), security assessment, web applications, web-services, etc.
* Techniques used for Grey box Testing
  + Matrix Testing
  + [Regression Testing](https://www.guru99.com/regression-testing.html)
  + OAT or Orthogonal Array Testing
  + Pattern Testing

# **Test Plan**

A test plan is a detailed document which describes software testing areas and activities. It outlines the test strategy, objectives, test schedule, required resources (human resources, software, and hardware), test estimation and test deliverables.The test plan is a base of every software's testing. It is the most crucial activity which ensures availability of all the lists of planned activities in an appropriate sequence.

The test plan is a template for conducting software testing activities as a defined process that is fully monitored and controlled by the testing manager. The test plan is prepared by the Test Lead (60%), Test Manager(20%), and by the test engineer(20%).

## Types of Test Plan

There are three types of the test plan

* Master Test Plan
* Phase Test Plan
* Testing Type Specific Test Plans

### **Master Test Plan**

Master Test Plan is a type of test plan that has multiple levels of testing. It includes a complete test strategy.

### **Phase Test Plan**

A phase test plan is a type of test plan that addresses any one phase of the testing strategy. For example, a list of tools, a list of test cases, etc.

### **Specific Test Plans**

Specific test plan designed for major types of testing like security testing, load testing, performance testing, etc. In other words, a specific test plan designed for non-functional testing.

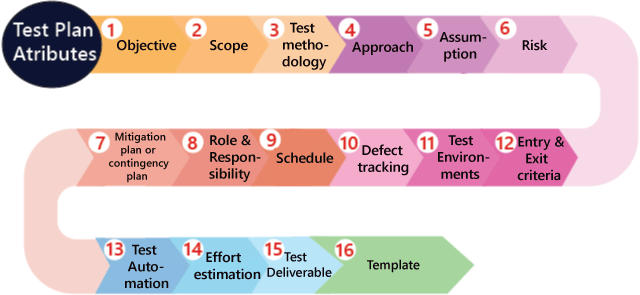
## How to write a Test Plan

Making a test plan is the most crucial task of the test management process. According to IEEE 829, follow the following seven steps to prepare a test plan.

* First, analyze product structure and architecture.
* Now design the test strategy.
* Define all the test objectives.
* Define the testing area.
* Define all the useable resources.
* Schedule all activities in an appropriate manner.
* Determine all the Test Deliverables.

## Test plan components or attributes

The test plan consists of various parts, which help us to derive the entire testing activity.



**Objectives:** It consists of information about modules, features, test data etc., which indicate the aim of the application means the application behavior, goal, etc.

**Scope:** It contains information that needs to be tested with respective of an application. The Scope can be further divided into two parts:

* In scope
* Out scope

**In scope:** These are the modules that need to be tested rigorously (in-detail).

**Out scope:** These are the modules, which need not be tested rigorously.

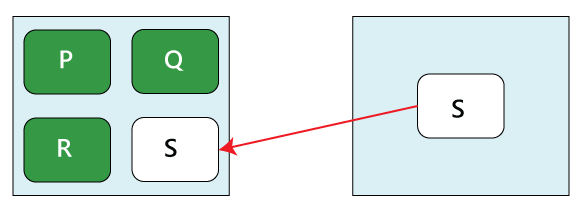
**For example**, Suppose we have a Gmail application to test, where **features to be tested** such as **Compose mail, Sent Items, Inbox, Drafts** and the **features which not be tested** such as **Help**, and so on which means that in the planning stage, we will decide that which functionality has to be checked or not based on the time limit given in the product.

Now **how we decide which features not to be tested?**

We have the following aspects where we can decide which feature not to be tested:

* As we see above that **Help** features is not going to be tested, as it is written and developed by the technical writer and reviewed by another professional writer.
* Let us assume that we have one application that have **P, Q, R, and S** features, which need to be developed based on the requirements. But here, the S feature has already been designed and used by some other company. So the development team will purchase S from that company and integrate with additional features such as P, Q, and R.

Now, we will not perform functional testing on the S feature because it has already been used in real-time. But we will do the integration testing, and system testing between P, Q, R, and S features because the new features might not work correctly with S feature as we can see in the below image:



* Suppose in the first release of the product, the elements that have been developed, such as **P, Q, R, S, T, U, V, W…..X, Y, Z**. Now the client will provide the requirements for the new features which improve the product in the second release and the new features are **A1, B2, C3, D4, and E5.**

After that, we will write the scope during the test plan as

**Scope**

**Features to be tested**

A1, B2, C3, D4, E5 (new features)

P, Q, R, S, T

**Features not to be tested**

W…..X, Y, Z

Therefore, we will check the new features first and then continue with the old features because that might be affected after adding the new features, which means it will also affect the impact areas, so we will do one round of regressing testing for P, Q, R…, T features.

### **Test methodology:**

It contains information about performing a different kind of testing like Functional testing, Integration testing, and System testing, etc. on the application. In this, we will decide what type of testing; we will perform on the various features based on the application requirement. And here, we should also define that what kind of testing we will use in the testing methodologies so that everyone, like the management, the development team, and the testing team can understand easily because the testing terms are not standard.

**For example,** for standalone application such as **Adobe Photoshop**, we will perform the following types of testing:

Smoke testing→ Functional testing → Integration testing →System testing →Adhoc testing → Compatibility testing → Regression testing→ Globalization testing → Accessibility testing → Usability testing → Reliability testing → Recovery testing → Installation or Uninstallation testing And suppose we have to test the  application, so we will perform following types of testing:

### **Approach**

This attribute is used to describe the flow of the application while performing testing and for the future reference.

We can understand the flow of the application with the help of below aspects:

* **By writing the high-level scenarios**
* **By writing the flow graph**

#### **By writing the high-level scenarios**

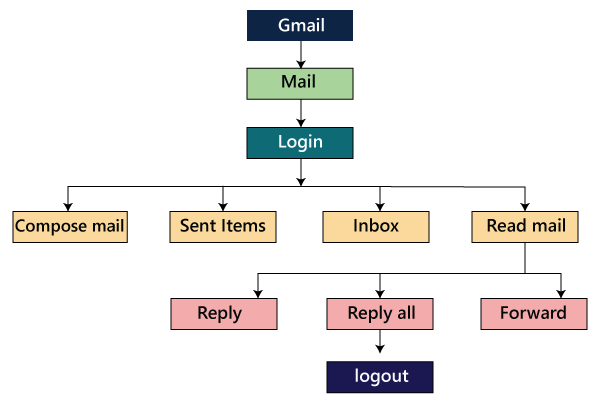
**For example**, suppose we are testing the **Gmail** application:

* Login to Gmail- sends an email and check whether it is in the Sent Items page
* Login to …….

We are writing this to describe the approaches which have to be taken for testing the product and only for the critical features where we will write the high-level scenarios. Here, we will not be focusing on covering all the scenarios because it can be decided by the particular test engineer that which features have to be tested or not.

#### **By writing the flow graph**

The flow graph is written because writing the high-level scenarios are bit time taking process, as we can see in the below image:



We are creating flow graphs to make the following benefits such as:

* Coverage is easy
* Merging is easy

The approach can be classified into two parts which are as following:

* Top to bottom approach
* Bottom to top approach

### **Assumption**

It contains information about a problem or issue which maybe occurred during the testing process and when we are writing the test plans, the assured assumptions would be made like resources and technologies, etc.

### **Risk**

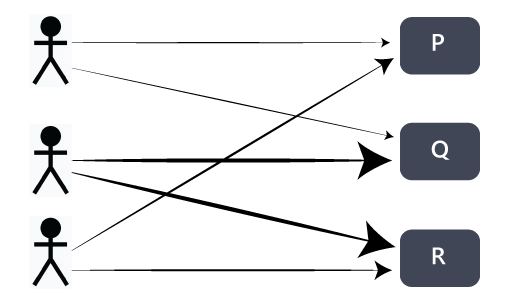
These are the challenges which we need to face to test the application in the current release and if the assumptions will fail then the risks are involved.

**For example,** the effect for an application, release date becomes postponed.

### **Mitigation Plan or Contingency Plan**

It is a back-up plan which is prepared to overcome the risks or issues.

Let us see one example for assumption, risk, and the contingency plan together because they are co-related to each other.



In any product, the **assumption** we will make is that the all 3 test engineers will be there until the completion of the product and each of them is assigned different modules such as P, Q, and R. In this particular scenario, the **risk** could be that if the test engineer left the project in the middle of it.

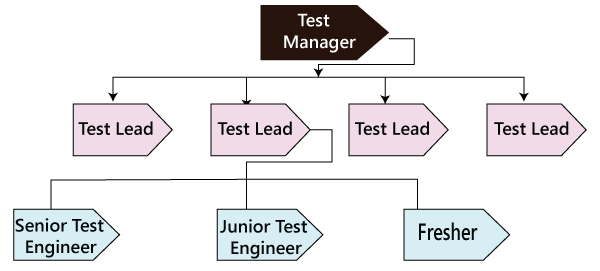
Therefore, the **contingency plan** will be assigned a primary and subordinate owner to each feature. So if the one test engineer will leave, the subordinate owner takes over that specific feature and also helps the new test engineer, so he/she can understand their assigned modules.

The assumptions, risk, and mitigation or contingency plan are always precise on the product itself. The various types of risks are as follows:

* Customer perspective
* Resource approach
* Technical opinion

### **Role & Responsibility**

It defines the complete task which needs to be performed by the entire testing team. When a large project comes, then the **Test Manager** is a person who writes the test plan. If there are 3-4 small projects, then the test manager will assign each project to each Test Lead. And then, the test lead writes the test plan for the project, which he/she is assigned.



Let see one example where we will understand the roles and responsibility of the Test manager, test lead, and the test engineers.

**Role: Test Manager**

**Name: Ryan**

**Responsibility:**

* Prepare( write and review) the test plan
* Conduct the meeting with the development team
* Conduct the meeting with the testing team
* Conduct the meeting with the customer
* Conduct one monthly stand up meeting
* Sign off release note
* Handling Escalations and issues

**Role: Test Lead**

**Name: Harvey**

**Responsibility:**

* Prepare( write and review) the test plan
* Conduct daily stand up meeting
* Review and approve the test case
* Prepare the RTM and Reports
* Assign modules
* Handling schedule

**Role: Test Engineer 1, Test Engineer 2 and Test Engineer 3**

**Name: Louis, Jessica, Donna**

**Assign modules: M1, M2, and M3**

**Responsibility:**

* Write, Review, and Execute the test documents which consists of test case and test scenarios
* Read, review, understand and analysis the requirement
* Write the flow of the application
* Execute the test case
* RTM for respective modules
* Defect tracking
* Prepare the test execution report and communicate it to the Test Lead.

### **Schedule**

It is used to explain the timing to work, which needs to be done or this attribute covers when exactly each testing activity should start and end? And the exact data is also mentioned for every testing activity for the particular date.

2.6 ( Done)

2.6.1 ( 15 Days)-Features ( A)-10 August- High Level-1 ( 2 days 8-9-10)

2.6.2( after 2.6.1 15 days) (B) 25 August(20-25 5 days) 2.6.1( 2.6.2)

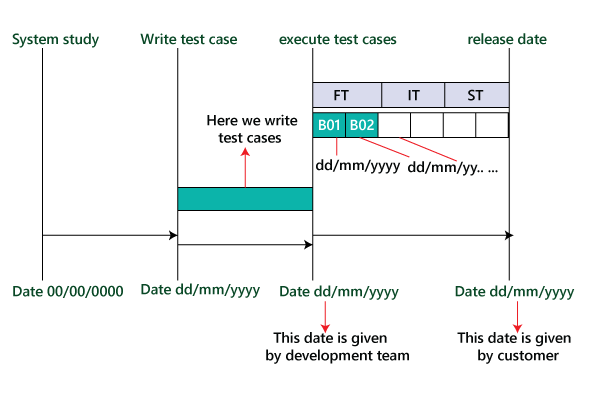
2.7( A, B, CDE) 15 Oct. 5 oct( 2.61,2.6.2,2.7) 10 days 5 -15

(QA)

Therefore as we can see in the below image that for the particular activity, there will be a starting date and ending date; for each testing to a specific build, there will be the specified date.

**For example**

* Writing test cases
* Execution process



### **Defect tracking**

It is generally done with the help of tools because we cannot track the status of each bug manually. And we also comment about how we communicate the bugs which are identified during the testing process and send it back to the development team and how the development team will reply. Here we also mention the priority of the bugs such as high, medium, and low.

Following are various aspects of the defect tracking:

* **Techniques to track the bug**
* **Bug tracking tools**  
  Some of the most commonly used bug tracking tools are Jira, Bugzilla, Mantis, and Trac, etc.
* **Severity**  
  The severity could be as following:  
  **Blocker or showstopper**  
    
  **For example**, there will be a defect in the module; we cannot go further to test other modules because if the bug is blocked, we can proceed to check other modules.

**Critical**  
Explain it with an example in the test plan  
In this situation, the defects will affect the business.

**Major**  
(Explain it with an example in the test plan)  
**Minor**  
  
These defects are those, which affect the look and feel of the application.

* **Priority**  
  **High-P1**  
  **Medium-P2**  
  **Low-P3**  
  **P4**

Therefore, based on the priority of bugs like high, medium, and low, we will categorize it as P1, P2, P3, and P4.

### **Test Environments**

These are the environments where we will test the application, and here we have two types of environments, which are of **software** and **hardware** configuration.

The **software configuration** means the details about different **Operating Systems** such as **Windows, Linux, UNIX, and Mac** and various **Browsers** like **Google Chrome, Firefox, Opera, Internet Explorer**, and so on.

And the **hardware configuration** means the information about different sizes of **RAM, ROM, and the Processors**.

**For example**

* The **Software** includes the following:

|  |  |
| --- | --- |
| **Operating system** | Linux |
| **Webserver** | Apache Tomcat |
| **Application server** | Websphere |
| **Database server** | Oracle or MS-SQL Server |

**Server**

**Client**

|  |  |
| --- | --- |
| **Operating System** | Window XP, Vista 7 |
| **Browsers** | Mozilla Firefox, Google Chrome, Internet Explorer, Internet Explorer 7, and Internet Explorer 8 |

* The **Hardware** includes the following:

**Server**: Sun StarCat 1500

This particular server can be used by the testing team to test their application.

**Client:**

It has the following configuration, which is as follows

|  |  |
| --- | --- |
| **Processor** | Intal2GHz |
| **RAM** | 2GB |

* **Process to install the software**  
  The development team will provide the configuration of how to install the software. If the development team will not yet provide the process, then we will write it as Task-Based Development (TBD) in the test plan.

### **Entry and Exit criteria**

It is a necessary condition, which needs to be satisfied before starting and stopping the testing process.

#### **Entry Criteria**

The entry criteria contain the following conditions:

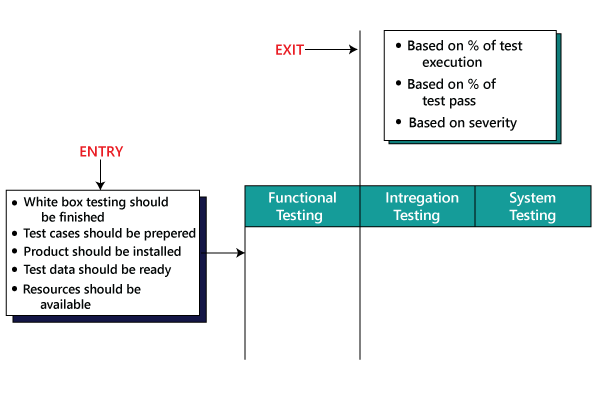
* White box testing should be finished.
* Understand and analyze the requirement and prepare the test documents or when the test documents are ready.
* Test data should be ready.
* Build or the application must be prepared
* Modules or features need to be assigned to the different test engineers.
* The necessary resource must be ready.

#### **Exit Criteria**

The exit criteria contain the following conditions:

* When all the test cases are executed.
* Most of the test cases must be **passed**.
* Depends on severity of the bugs which means that there must not be any blocker or major bug, whereas some minor bugs exist.

Before we start performing functional testing, all the above **Entry Criteria** should be followed. After we performed functional testing and before we will do integration testing, then the **Exit criteria of** the functional testing should be followed because the % of exit criteria are decided by the meeting with both development and test manager because their collaboration can achieve the percentage. But if the exit criteria of functional testing are not followed, then we cannot proceed further to integration testing.



Here **based on the severity** of the bug's means that the testing team would have decided that to proceed further for the next phases.

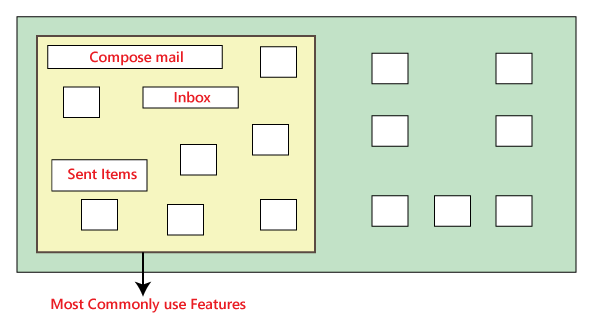
### **Test Automation**

In this, we will decide the following:

* Which feature has to be automated and not to be automated?
* Which test automation tool we are going to use on which automation framework?

We automate the test case only after the first release.

Here the question arises that on what basis **we** will **decide which features have to be tested?**



In the above image, as we can see that the most commonly used features need to test again and again. Suppose we have to check the Gmail application where the essential features are **Compose mail, Sent Items, and Inbox**. So we will test these features because while performing manual testing, it takes more time, and it also becomes a monotonous job.

Now, **how we decide which features are not going to be tested?**

Suppose **the Help** feature of the Gmail application is not tested again and again because these features are not regularly used, so we do not need to check it frequently.

But **if some features are unstable and have lots of bugs**, which means that we will not test those features because it has to be tested again and again while doing manual testing.

If **there is a feature that has to be tested frequently**, but we are expecting the requirement change for that feature, so we do not check it because changing the manual test cases is more comfortable as compared to change in the automation test script.

### **Effort estimation**

In this, we will plan the effort need to be applied by every team member.

### **Test Deliverable**

These are the documents which are the output from the testing team, which we handed over to the customer along with the product. It includes the following:

* **Test plan-Team lead**
* **Test Cases-Test engineer**
* **Test Scripts-test engineer-automation-devloper**
* **RTM(Requirement Traceability Matrix)-QA**
* **Defect Report-QA**
* **Test Execution Report-QA**
* **Graphs and metrics-(Performance)**
* **Release Notes ( Test lead)**

#### **Graphs and Metrics**

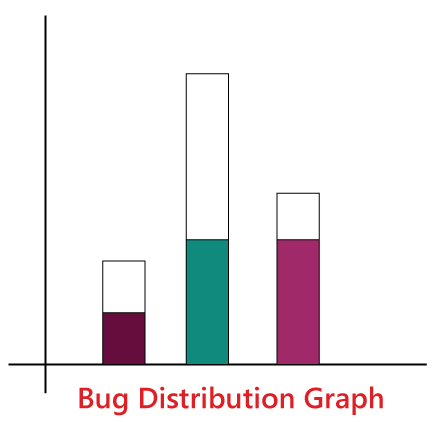
**Graph**

In this, we will discuss about the types of **graphs** we will send, and we will also provide a sample of each graph.

As we can see, we have five different graphs that show the various aspects of the testing process.

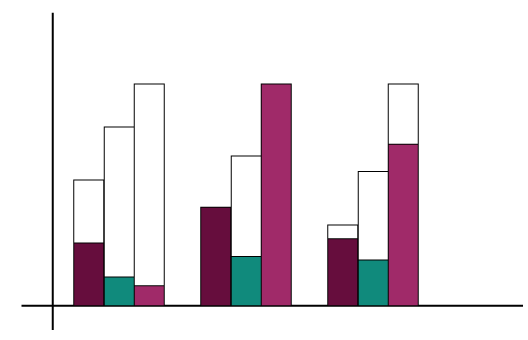
**Graph1:**

 In this, we will show how many defects have been identified and how many defects have been fixed in every module.



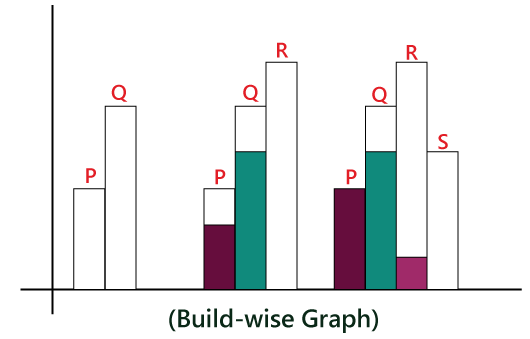
**Graph 2:**

Figure one shows how many critical, major, and minor defects have been identified for every module and how many have been fixed for their respective modules.



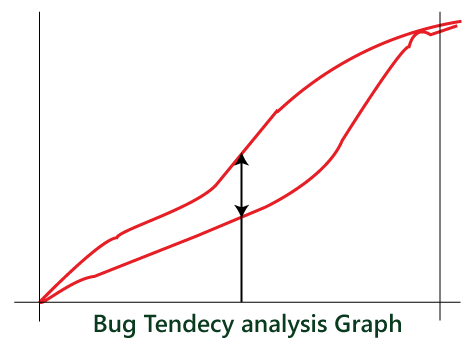
**Graph3:**

In this particular graph, we represent the **build wise graph**, which means that in every builds how many defects have been identified and fixed for every module. Based on the module, we have determined the bugs. We will add **R** to show the number of defects in P and Q, and we also add **S** to show the defects in P, Q, and R.



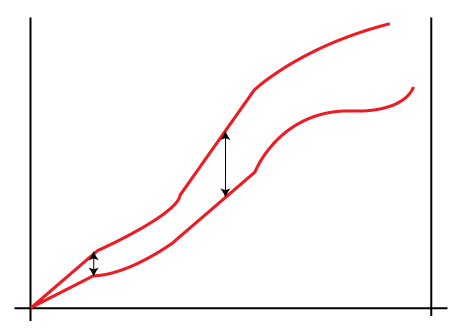
**Graph4:**

The test lead will design the **Bug Trend analysis** graph which is created every month and send it to the Management as well. And it is just like prediction which is done at the end of the product. And here, we can also **rate the bug fixes** as we can observe that **arc** has an **upward tendency** in the below image.

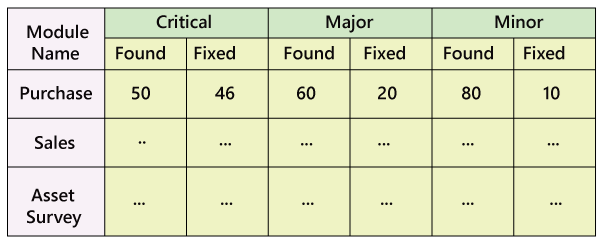


**Graph5:**

The **Test Manager** has designed this type of graph. This graph is intended to understand the gap in the assessment of bugs and the actual bugs which have been occurred, and this graph also helps to improve the evaluation of bugs in the future.

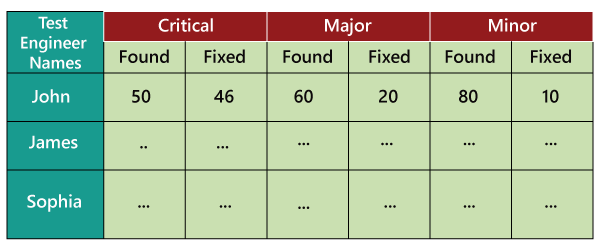


**Metrics**



As above, we create the bug distribution graph, which is in the figure 1, and with the help of above mention data, we will design the metrics as well.

**For example**



In the above figure, we retain the records of all the test engineers in a particular project and how many defects have been identified and fixed. We can also use this data for future analysis. When a new requirement comes, we can decide whom to provide the challenging feature for testing based on the number of defects they have found earlier according to the above metrics. And we will be in a better situation to know who can handle the problematic features very well and find maximum numbers of defects.

**Release Note:** It is a document that is prepared during the release of the product and signed by the Test Manager.

In the below image, we can see that the final product is developed and deployed to the customer, and the latest release name is **Beta**.

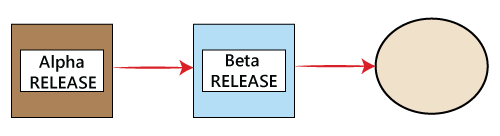


The **Release note** consists of the following:

* User manual.
* List of pending and open defects.
* List of added, modified, and deleted features.
* List of the platform (Operating System, Hardware, Browsers) on which the product is tested.
* Platform in which the product is not tested.
* List of bugs fixed in the current release, and the list of fixed bugs in the previous release.
* Installation process
* Versions of the software

**For Example**

Suppose that **Beta** is the second release of the application after the first release **Alpha** is released. Some of the defect identified in the first released and that has been fixed in the later released. And here, we will also point out the list of newly added, modified, and deleted features from alpha release to the beta release.



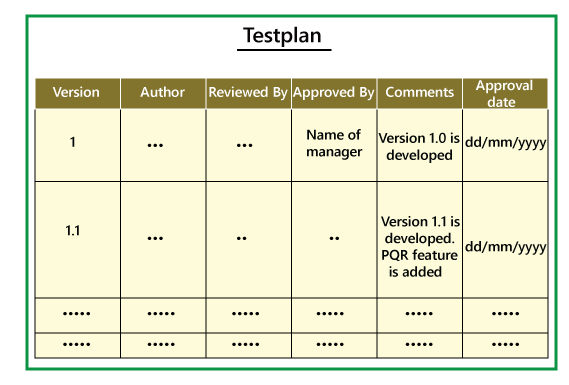
### **Template**

This part contains all the templates for the documents that will be used in the product, and all the test engineers will use only these templates in the project to maintain the consistency of the product. Here, we have different types of the template which are used during the entire testing process such as:

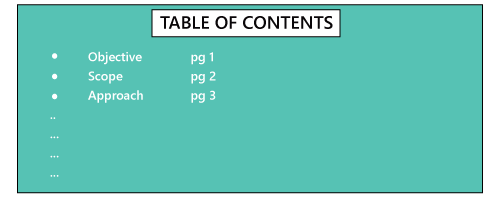
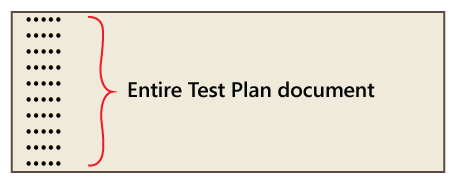
* Test case template
* Test case review template
* RTM Template
* Bug Report Template
* Test execution Report

Let us see one sample of test plan document

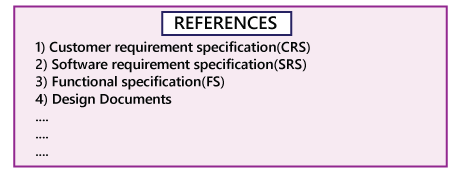
**Page-1**



**Page3-page18**

**Page-20**



In-Page 1, we primarily fill only the **Versions, Author, Comments, and Reviewed By** fields, and after the manager approves it, we will mention the details in the **Approved By and Approval Date** fields.

Mostly the test plan is approved by the Test Manager, and the test engineers only reviews it. And when the new features come, we will modify the test plan and do the necessary modification in **Version** field, and then it will be sent again for further review, update, and approval of the manager. The test plan must be updated whenever any changes have occurred. On page 20, the **References** specify the details about all the documents which we are going to use to write the test plan document.

**Note:**

**Who writes the test plan?**

* Test Lead→60%
* Test Manager→20%
* Test Engineer→20%

Therefore, as we can see from above that in 60% of the product, the test plan is written by the Test Lead.

**Who reviews the Test Plan?**

* Test Lead
* Test Manager
* Test engineer
* Customer
* Development team

The Test Engineer review the Test plan for their module perspective and the test manager review the Test plan based on the customer opinion.

**Who approve the test Plan?**

* Customer
* Test Manager

**Who writes the test case?**

* Test Lead
* Test Engineer

**Who review the test case?**

* Test Engineer
* Test Lead
* Customer
* Development Team

**Who approves the Test cases?**

* Test Manager
* Test Lead
* Customer

## Test Plan Guidelines

* Collapse your test plan.
* Avoid overlapping and redundancy.
* If you think that you do not need a section that is already mentioned above, then delete that section and proceed ahead.
* Be specific. For example, when you specify a software system as the part of the test environment, then mention the software version instead of only name.
* Avoid lengthy paragraphs.
* Use lists and tables wherever possible.
* Update plan when needed.
* Do not use an outdated and unused document.

## Importance of Test Plan

* The test plan gives direction to our thinking. This is like a rule book, which must be followed.
* The test plan helps in determining the necessary efforts to validate the quality of the software application under the test.
* The test plan helps those people to understand the test details that are related to the outside like developers, business managers, customers, etc.
* Important aspects like test schedule, test strategy, test scope etc are documented in the test plan so that the management team can review them and reuse them for other similar projects.

# **Different Testing Types With Details**

|  |  |
| --- | --- |
| 1) Alpha Testing | 2) Acceptance Testing |
| 3)Ad-hoc Testing | 4) Accessibility Testing |
| 5) Beta Testing | 6) Back-end Testing |
| [7) Browser Compatibility Testing](https://www.softwaretestinghelp.com/types-of-software-testing/#7_Browser_Compatibility_Testing) | 8) Backward Compatibility Testing |
| 9) Black Box Testing | 10) Boundary Value Testing |
| 11) Branch Testing | 12) Comparison Testing |
| 13) Compatibility Testing | 14) Component Testing |
| 15) End-to-End Testing | 16) Equivalence Partitioning |
| 17) Example Testing | [18) Exploratory Testing](https://www.softwaretestinghelp.com/types-of-software-testing/#18_Exploratory_Testing) |
| 20) Functional Testing | 21) Graphical User Interface (GUI) Testing |
| 22) Gorilla Testing | 23) Happy Path Testing |
| 24) Incremental Integration Testing | 25) Install/Uninstall Testing |
| 26) Integration Testing | 27) Load Testing |
| 28) Monkey Testing | 29) Mutation Testing |
| 30) Negative Testing | 31) Non-Functional Testing |
| 32) Performance Testing | 33) Recovery Testing |
| 34) Regression Testing | 35) Risk-Based Testing (RBT) |
| 36) Sanity Testing | 37) Security Testing |
| 38) Smoke Testing | 39) Static Testing |
| 40) Stress Testing | 41) System Testing |
| 42) Unit Testing | 43) Usability Testing |
| 44) Vulnerability Testing | 45) Volume Testing |
| 46) White Box Testing |  |

## Different Types Of Software Testing

Given below is a list of some common types of Software Testing:

|  |  |
| --- | --- |
| **Functional Testing types include:**   * Unit Testing * Integration Testing * System Testing * Sanity Testing * Smoke Testing * Interface Testing * Regression Testing * Beta/Acceptance Testing * Retesting | **Non-functional Testing types include:**   * Performance Testing * Load Testing * Stress Testing * Volume Testing * Security Testing * Compatibility Testing * Install Testing * [Recovery Testing](https://www.softwaretestinghelp.com/recovery-testing-tutorial/) * Reliability Testing * Usability Testing * Compliance Testing * Localization Testing |

**Let’s explore more about these Testing types:**

### **1) Alpha Testing**

It is the most commonly used testing in the Software industry. The objective of this testing is to identify all possible issues or defects before releasing it into the market or to the user. Alpha Testing will be carried out at the end of the software development phase but before the Beta Testing. Still, minor design changes may be made as a result of such testing. [Alpha Testing](https://www.softwaretestinghelp.com/what-is-alpha-testing-beta-testing/) will be conducted at the developer’s site. An in-house virtual user environment can be created for this type of testing.

### **2) Acceptance Testing**

An [Acceptance Test](https://www.softwaretestinghelp.com/what-is-acceptance-testing/) is performed by the client and it verifies whether the end to end flow of the system is as per the business requirements or not and if it is as per the needs of the end-user.

Client accepts the software only when all the features and functionalities work as expected. This is the last phase of testing, after which the software goes into production. This is also called User Acceptance Testing (UAT).

### **3) Ad-hoc Testing**

The name itself suggests that this testing is performed on an [ad-hoc](https://www.softwaretestinghelp.com/ad-hoc-testing/) basis i.e., with no reference to the test case and also without any plan or documentation in place for this type of testing.

The objective of this testing is to find the defects and break the application by executing any flow of the application or any random functionality.

Ad-hoc Testing is an informal way of finding defects and can be performed by anyone in the project. It is difficult to identify defects without a test case but sometimes it is possible that defects found during ad-hoc testing might not have been identified using the existing test cases.

### **4) Accessibility Testing**

Recommend the aim of [Accessibility Testing](https://www.softwaretestinghelp.com/what-is-web-accessibility-testing/) is to determine whether the software or application is accessible for disabled people or not.

Here, disability means deafness, color blindness, mentally disabled, blind, old age and other disabled groups. Various checks are performed such as font size for visually disabled, color and contrast for color blindness, etc.

### **5) Beta Testing**

[Beta Testing](https://www.softwaretestinghelp.com/beta-testing/) is a formal type of Software Testing which is carried out by the customer. It is performed in **the Real Environment**before releasing the product to the market for the actual end-users. Beta Testing is carried out to ensure that there are no major failures in the software or product and it satisfies the business requirements from an end-user perspective. Beta Testing is successful when the customer accepts the software.

Usually, this testing is typically done by the end-users or others. This is the final testing done before releasing the application for commercial purposes. Usually, the Beta version of the software or product released is limited to a certain number of users in a specific area. So the end-user actually uses the software and shares the feedback with the company. The company then takes necessary action before releasing the software worldwide.

### **6) Back-end Testing**

Whenever an input or data is entered on the front-end application, it is stored in the database and the testing of such database is known as Database Testing or Backend Testing. There are different databases like SQL Server, MySQL, and Oracle, etc. Database Testing involves testing of table structure, schema, stored procedure, data structure and so on. In Back-end Testing, GUI is not involved, the testers are directly connected to the database with proper access and testers can easily verify data by running a few queries on the database. There can be issues identified like data loss, deadlock, data corruption etc. during this back-end testing and these issues are critical to fix before the system goes live into the production environment

### **7) Browser Compatibility Testing**

This is a sub-type of Compatibility Testing (which is explained below) and is performed by the testing team. [Browser Compatibility Testing](https://www.softwaretestinghelp.com/how-is-cross-browser-testing-performed/) is performed for web applications and ensures that the software can run with a combination of different browsers and operating systems. This type of testing also validates whether a web application runs on all versions of all browsers or not.

### **8) Backward Compatibility Testing**

It is a type of testing which validates whether the newly developed software or updated software works well with the older version of the environment or not. Backward Compatibility Testing checks whether the new version of the software works properly with the file format created by an older version of the software; it also works well with data tables, data files, and data structure created by the older version of that software. If any of the software is updated then it should work well on top of the previous version of that software.

### **9) Black Box Testing**

Internal system design is not considered in this type of testing. Tests are based on the requirements and functionality.

### **10) Boundary Value Testing**

This type of testing checks the behavior of the application at boundary level. [Boundary Value Testing](https://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/) is performed to check if defects exist at boundary values. Boundary Value Testing is used for testing a different range of numbers. There is an upper and lower boundary for each range and testing is performed on these boundary values. If testing requires a test range of numbers from 1 to 500 then Boundary Value Testing is performed on values at 0, 1, 2, 499, 500 and 501.

### **11) Branch Testing**

This is a type of White box Testing and is carried out during Unit Testing. Branch Testing, the name itself, suggests that the code is tested thoroughly by traversing at every branch.

### **12) Comparison Testing**

Comparison of a product’s strengths and weaknesses with its previous versions or other similar products is termed as Comparison Testing.

### **13) Compatibility Testing**

This is a testing type in which it validates how software behaves and runs in a different environment, web servers, hardware, and network environment. [Compatibility testing](https://www.softwaretestinghelp.com/software-compatibility-testing/) ensures that software can run on a different configuration, different databases, different browsers, and their versions. Compatibility testing is performed by the testing team.

### **14) Component Testing**

This is mostly performed by developers after the completion of unit testing. [Component Testing](https://www.softwaretestinghelp.com/what-is-component-testing-or-module-testing/) involves testing of multiple functionalities as a single code and its objective is to identify if any defect exists after connecting those multiple functionalities with each other.

### **15) End-to-End Testing**

Similar to system testing, [End-to-End Testing](https://www.softwaretestinghelp.com/what-is-end-to-end-testing/) involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

### **16) Equivalence Partitioning**

It is a testing technique and a type of Black Box Testing. During this [Equivalence Partitioning](https://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/), a set of groups are selected and a few values or numbers are picked up for testing. It is understood that all values from that group generate the same output.

The aim of this testing is to remove redundant test cases within a specific group which generate the same output but not any defect. Password 8-16 chars,7 char, 8 char, 12 char,15 char, 16 char, 17 char Suppose the application accepts values between -10 and +10, then using equivalence partitioning the values picked for testing are zero, one positive value, and one negative value. So the Equivalence Partitioning for this testing is  -10 to -1, 0, and 1 to 10.

### **17) Example Testing**

This means real-time testing. Example Testing includes real-time scenarios; it also involves scenarios based on the experience of the testers.

### **18) Exploratory Testing**

Exploratory Testing is informal testing performed by the testing team. The objective of this testing is to explore the application and look for defects that exist in the application. Sometimes it may happen that during this testing a major defect discovered can even cause a system failure. During Exploratory Testing, it is advisable to keep a track of what flow you have tested and what activity you did before the start of a specific flow. [Exploratory Testing techniques](https://www.softwaretestinghelp.com/what-is-exploratory-testing/) are performed without documentation or test cases.

### **20) Functional Testing**

This type of testing ignores the internal parts and focuses only on the output to check if it is as per the requirement or not. This is a black-box type testing that is geared towards the functional requirements of an application.

### **21) Graphical User Interface (GUI) Testing**

The objective of this GUI Testing is to validate the GUI as per the business requirement. The expected GUI of the application is mentioned in the Detailed Design Document and GUI mockup screens. GUI Testing includes the size of the buttons and input fields present on the screen, alignment of all text, tables, and content in the tables. It also validates the menu of the application, after selecting different menu and menu items, it validates that the page does not fluctuate and the alignment remains the same after hovering the mouse on the menu or sub-menu.

### **22) Gorilla Testing**

Gorilla Testing is a testing type performed by a tester and sometimes by the developer as well.In Gorilla Testing, one module or the functionality in the module is tested thoroughly and heavily. The objective of this testing is to check the robustness of the application.

### **23) Happy Path Testing**

The objective of Happy Path Testing is to test an application successfully on a positive flow.It does not look for negative or error conditions. The focus is only on valid and positive inputs through which the application generates the expected output.

### **24) Incremental Integration Testing**

[Incremental Integration Testing](https://www.softwaretestinghelp.com/incremental-testing/) is a Bottom-up approach for testing i.e continuous testing of an application when a new functionality is added. Application functionality and modules should be independent enough to test separately. This is done by programmers or by testers.

### **25) Install/Uninstall Testing**

[Installation and Uninstallation Testing](https://www.softwaretestinghelp.com/software-installationuninstallation-testing/) is done on full, partial, or upgraded install/uninstall processes on different operating systems under different hardware or software environments.

### **26) Integration Testing**

Testing of all integrated modules to verify the combined functionality after integration is termed as[Integration Testing](https://www.softwaretestinghelp.com/what-is-integration-testing/). Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to the client/server and distributed systems.

### **27) Load Testing**

It is a type of Non-Functional Testing and the objective of Load Testing is to check how much load or maximum workload a system can handle without any performance degradation. [Load Testing](https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/) helps to find the maximum capacity of the system under specific load and any issues that cause software performance degradation. Load testing is performed using tools like[JMeter](https://www.softwaretestinghelp.com/jmeter-tutorials/), Load Runner, Web Load, Silk performer, etc.

### **28) Monkey Testing**

[Monkey Testing](https://www.softwaretestinghelp.com/what-is-monkey-testing-in-software-testing/) is carried out by a tester assuming that if the monkey uses the application then how random input and values will be entered by the Monkey without any knowledge or understanding of the application.

The objective of Monkey Testing is to check if an application or system gets crashed by providing random input values/data. Monkey Testing is performed randomly and no test cases are scripted and it is not necessary to be aware of the full functionality of the system.

### **29) Mutation Testing**

[Mutation Testing](https://www.softwaretestinghelp.com/what-is-mutation-testing/) is a type of white box testing in which the source code of one of the programs is changed and verifies whether the existing test cases can identify these defects in the system.

The change in the program source code is very minimal so it does not impact the entire application, only the specific area having the impact and the related test cases should be able to identify those errors in the system.

### **30) Negative Testing**

Testers have the mindset of “attitude to break” and using Negative Testing they validate if the system or application breaks. [Negative Testing technique](https://www.softwaretestinghelp.com/what-is-negative-testing/) is performed using incorrect data, invalid data or input. It validates if the system throws an error of invalid input and behaves as expected.

### **31) Non-Functional Testing**

This is the type of testing for which every organization has a separate team which is usually called a Non-Functional Test (NFT) team or Performance team. [Non-Functional Testing](https://www.softwaretestinghelp.com/what-is-non-functional-testing/)involves testing of non-functional requirements such as Load Testing, Stress Testing, Security, Volume, Recovery Testing, etc. The objective of NFT testing is to ensure whether the response time of software or application is quick enough as per the business requirement. It should not take much time to load any page or system and should be sustained during peak load.

### **32) Performance Testing**

This term is often used interchangeably with ‘stress’ and ‘load’ testing. [Performance Testing](https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/) is done to check whether the system meets the performance requirements. Different performance and load tools are used to do this testing.

**33) Recovery Testing**

It is a type of testing which validates how well the application or system recovers from crashes or disasters. Recovery Testing determines if the system is able to continue its operation after a disaster. Assume that the application is receiving data through a network cable and suddenly that network cable has been unplugged. Sometime later, plug in the network cable; then the system should start receiving data from where it lost the connection due to network cable being unplugged.

### **34) Regression Testing**

Testing an application as a whole for the modification of any module or functionality is termed as Regression Testing. It is difficult to cover all the systems in [Regression Testing](https://www.softwaretestinghelp.com/regression-testing-tools-and-methods/), so typically [Automation Testing Tools](https://www.softwaretestinghelp.com/automation-testing-tutorial-1/) are used for these types of testing.

### **35) Risk-Based Testing (RBT)**

For [Risk-Based Testing](https://www.softwaretestinghelp.com/risk-management-during-test-planning-risk-based-testing/), the functionalities or requirements are tested based on their priority. Risk-Based Testing includes testing of highly critical functionality, which has the highest impact on business and in which the probability of failure is very high.

Priority decisions are based on business needs, so once priority is set for all functionalities, then high priority functionality or test cases are executed first followed by medium and then low priority functionalities.

Low priority functionality may be tested or not tested based on the available time. Risk-Based Testing is carried out if there is insufficient time available to test the entire software and the software needs to be implemented on time without any delay. This approach is followed only by the discussion and approval of the client and senior management of the organization.

### **36) Sanity Testing**

[Sanity Testing](https://www.softwaretestinghelp.com/smoke-testing-and-sanity-testing-difference/) is done to determine if a new software version is performing well enough to accept it for a major testing effort or not. If an application is crashing for initial use then the system is not stable enough for further testing. Hence a build or an application is assigned to fix it. Sanity Testing is done during the release phase to check for the main functionalities of the application without going deeper. It is also called as a subset of Regression testing. It is done at the “release level”. At times due to release time constraints rigorous [regression testing](https://www.softwaretestingmaterial.com/difference-between-regression-and-retesting/) can’t be done to the build, sanity testing does that part by checking main functionalities.

### **37) Security Testing**

It is a type of testing performed by a special team of testers. A system can be penetrated by any hacking method. [Security Testing](https://www.softwaretestinghelp.com/how-to-test-application-security-web-and-desktop-application-security-testing-techniques/) is done to check how the software, application or website is secure from internal and external threats. This testing includes how much software is secure from malicious programs, viruses and how secure & strong the authorization and authentication processes are. It also checks how software behaves for any hackers attack & malicious programs and how software is maintained for data security after such a hacker attack.

### **38) Smoke Testing**

Whenever a new build is provided by the development team, then the Software Testing team validates the build and ensures that no major issue exists. The testing team will ensure that the build is stable and a detailed level of testing will be carried out further. [Smoke Testing](https://www.softwaretestinghelp.com/smoke-testing-and-sanity-testing-difference/) checks for no show stopper defects exist in the build which will prevent the testing team from testing the application in detail. If the testers find that the major critical functionality is broken down at the initial stage itself then the testing team can reject the build and inform accordingly to the development team. Smoke Testing is carried out to a detailed level of any Functional or Regression Testing.

### **39) Static Testing**

Static Testing is a type of testing which is executed without any code. The execution will be performed on the documentation during the testing phase. It involves reviews, walkthroughs, and inspection of the deliverables of the project. Static Testing does not execute the code instead the code syntax and naming conventions are checked. [Static Testing](https://www.softwaretestinghelp.com/static-testing-and-dynamic-testing-difference/) is also applicable for test cases, test plans, and design documents. We need to perform static testing with the testing team as the defects identified during this type of testing are cost-effective from a project perspective.

### **40) Stress Testing**

This testing is done when a system is stressed beyond its specifications in order to check how and when it fails. This is performed under heavy load like putting large number beyond storage capacity, complex database queries and continuous input to the system or database load.

### **41) System Testing**

Under [System Testing technique](https://www.softwaretestinghelp.com/system-testing/), the entire system is tested as per the requirements. It is a Black-box type testing that is based on the overall requirement specifications and covers all the combined parts of the system.

### **42) Unit Testing**

Testing of an individual software component or module is termed as [Unit Testing](https://www.softwaretestinghelp.com/unit-testing/). It is typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. It may also require developing test driver modules or test harnesses.

### **43) Usability Testing**

Under [Usability Testing](https://www.softwaretestinghelp.com/usability-testing-guide/), the User-Friendliness Check is done. The application flow is tested to see if a new user can understand the application easily or not. Proper help is documented if a user gets stuck at any point. Basically, system navigation is checked in this testing.

### **44) Vulnerability Testing**

The testing, which involves identifying weaknesses in the software, hardware and the network, is known as Vulnerability Testing. In malicious programs, the hacker can take control of the system, if it is vulnerable to such kind of attacks, viruses, and worms. We need to check if those systems undergo Vulnerability Testing before production. It may identify critical defects and flaws in security.

### **45) Volume Testing**

[Volume Testing](https://www.softwaretestinghelp.com/what-is-volume-testing/) is a type of Non-Functional Testing performed by the Performance Testing team. The software or application undergoes a huge amount of data and Volume Testing checks the system behavior and response time of the application when the system came across such a high volume of data. This high volume of data may impact the system’s performance and speed of processing time.

### **46) White Box Testing**

[White Box Testing](https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/) is based on the knowledge about the internal logic of an application’s code.

It is also known as Glass box Testing. Internal software and code work should be known to perform this type of testing. Under this, tests are based on the coverage of code statements, branches, paths, conditions, etc.

### **40. Is there any difference between bug leakage and bug release?**

**Bug leakage:** Bug leakage is something, when the bug is discovered by the end user/customer and missed by the testing team to detect while testing the software. It is a defect that exists in the application and not detected by the tester, which is eventually found by the customer/end user.

**Bug release:** A bug release is when a particular version of the software is released with a set of known bug(s). These bugs are usually of low severity/priority. It is done when a software company can afford the existence of bugs in the released software but not the time/cost for fixing it in that particular version

**1. What is Software Testing?**

According to ANSI/IEEE 1059 standard – A process of analyzing a software item to detect the differences between existing and required conditions (i.e., defects) and to evaluate the features of the software item.

**2. What are the best practices for writing test cases?**

* Write test cases with end-users perspective
* Write test steps in a simple way that anyone can follow them easily
* Make the test cases reusable
* Set the priority
* Provide a test case description, test data, expected result, precondition, post condition.
* Write invalid test cases along with valid test cases
* Follow proper naming conventions
* Review the test cases regularly and update them if necessary.

#### **3. What is configuration management?**

Configuration management (CM) is a process of systems engineering to maintain system resources, computer systems, servers, software, and product’s performance in a consistent state. It helps to record all the changes made in the system and ensures that the system performs as expected even though changes are made over time.

**4. Name some popular configuration management tools?**

Some of the popular configuration management tools are Ansible, Chef, Puppet, Terraform, Salt stack, etc.

#### **5. What if the software is so buggy it can’t really be tested at all?**

If the software is so buggy, the first thing we need to do is to report the bugs and categories them based on Severity. If the bugs are critical bugs then it severely affects schedules and indicates deeper problems in the software development process. So you need to let the manager know about the bugs with proper documentation as evidence.

#### **6. What are Quality Assurance and Quality Control?**

**Quality Assurance:**Quality Assurance involves in process-oriented activities. It ensures the prevention of defects in the process used to make Software Applications. So the defects don’t arise when the Software Application is being developed.

**Quality Control:**Quality Control involves in product-oriented activities. It executes the program or code to identify the defects in the Software Application.

#### **7. What is the difference between SDET, Test Engineer, and Developer**

| **Test Engineer** | **SDET** | **Developer** |
| --- | --- | --- |
| Test Engineer thinks only in the terms of pass or fail of a test case and how to break the software | SDET knows system functional objectives as well as quality objectives | Developer thinks how to develop a system and make a functionality work |
| Test Engineer works only for test life cycle, like design of test cases, and execution | SDET is involved in Designing, development, and testing | Developer is limited to Coding part and release to testing team |
| No coding knowledge is required | Dynamic skill sets, like knowledge of quality and testing and good in coding too | Only coding knowledge is required |
| Test Engineers know where repetitive work or simple data entry is present but they are not expected to minimize the repetitive tasks | SDET understands automation needs, they can code and provide a solution to the team where repetitive kind of work is killing the time. They can design framework which can help testing team to reduce repetitive test cycle or simple data entry task. | Developers don’t deal with such tasks |
| Test Engineers are not expected to reach up to code level and tune the performance | Well aware of Performance tuning and security threats , they can suggest and reach to the code and suggest where application is poor in performance, plus they can optimize the code | Developers are only expected to code the functionality which is expected by customer |

#### **8. What is Verification in software testing?**

Verification is the process, to ensure that whether we are building the product right i.e., to verify the requirements which we have and to verify whether we are developing the product accordingly or not. Activities involved here are Inspections, Reviews, and Walk-throughs.

#### **9. What is Validation in software testing?**

Validation is the process, whether we are building the right product i.e., to validate the product which we have developed is right or not. Activities involved in this is Testing the software application.

#### **10. What is Static Testing?**

Static Testing involves reviewing the documents to identify the defects in the early stages of SDLC. In static testing, we do code reviews, walkthroughs, peer reviews, and static analysis of a source code by using tools like Style Cop, ESLint, etc.,

#### **11. What is Dynamic Testing?**

Dynamic testing involves the execution of code. It validates the output with the expected outcome.

#### **12. What is White Box Testing?**

White Box Testing is also called as Glass Box, Clear Box, and Structural Testing. It is based on applications internal code structure. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. This testing usually was done at the unit level.

Various white-box testing techniques are:

1. Statement Coverage
2. Decision Coverage
3. Condition Coverage
4. Multiple Condition Coverage

#### **13. What is Black Box Testing?**

Black Box Testing is a [software testing](https://www.softwaretestingmaterial.com/software-testing/) method in which testers evaluate the functionality of the software under test without looking at the internal code structure. This can be applied to every level of software testing such as Unit, Integration, System and Acceptance Testing

#### **14. What is Grey Box Testing?**

Grey box is the combination of both White Box and Black Box Testing. The tester who works on this type of testing needs to have access to design documents. This helps to create better test cases in this process.

#### **15. What is Positive and Negative Testing?**

**Positive Testing:** It is to determine what system supposed to do. It helps to check whether the application is justifying the requirements or not.

**Negative Testing:** It is to determine what system not supposed to do. It helps to find the defects from the software.

#### **16. What is Test Strategy?**

Test Strategy is a high-level document (static document) and usually developed by the project manager. It is a document that captures the approach on how we go about testing the product and achieve the goals. It is normally derived from the Business Requirement Specification (BRS). Documents like Test Plan are prepared by keeping this document as a base.

#### **17. What is Test Plan and contents available in a Test Plan?**

Test plan document is a document which contains the plan for all the testing activities to be done to deliver a quality product. Test Plan document is derived from the Product Description, SRS, or Use Case documents for all future activities of the project. It is usually prepared by the Test Lead or Test Manager.

1. Test plan identifier
2. References
3. Introduction
4. Test items (functions)
5. Software risk issues
6. Features to be tested
7. Features not to be tested
8. Approach
9. Items pass/fail criteria
10. Suspension criteria and resolution requirements
11. Test deliverables
12. Remaining test tasks
13. Environmental needs
14. Staff and training needs
15. Responsibility
16. Schedule
17. Plan risks and contingencies
18. Approvals
19. Glossaries

#### **18. What is Test Suite?**

Test Suite is a collection of test cases. The test cases which are intended to test an application.

#### **19. What is Test Scenario?**

Test Scenario gives the idea of what we have to test. Test Scenario is like a high-level test case.

#### **20. What is Test Case?**

Test cases are the set of positive and negative executable steps of a test scenario which has a set of pre-conditions, test data, expected result, post-conditions and actual results

#### **21. What is Test Bed?**

An environment configured for testing. Test bed consists of hardware, software, network configuration, an application under test, other related software.

#### **22. What is Test Environment?**

Test Environment is the combination of hardware and software on which Test Team performs testing.

* Application Type: Web Application
* OS: Windows
* Web Server: IIS
* Web Page Design: Dot Net
* Client Side Validation: JavaScript
* Server Side Scripting: ASP Dot Net
* Database: MS SQL Server
* Browser: IE/FireFox/Chrome

#### **23. What is Test Data?**

Test data is the data that is used by the testers to run the test cases. Whilst running the test cases, testers need to enter some input data. To do so, testers prepare test data. It can be prepared manually and also by using tools. For example, To test a basic login functionality having a user id, password fields. We need to enter some data in the user id and password fields. So we need to collect some test data.

#### **24. What is Test Harness?**

A test harness is the collection of software and test data configured to test a program unit by running it under varying conditions which involves monitoring the output with the expected output. It contains the Test Execution Engine & Test Script Repository

#### **25. What is Test Closure?**

Test Closure is the note prepared before test team formally completes the testing process. This note contains the total no. of test cases, total no. of test cases executed, total no. of defects found, total no. of defects fixed, total no. of bugs not fixed, total no of bugs rejected etc.,

#### **26. What are the tasks of Test Closure activities in Software Testing?**

Test Closure activities fall into four major groups.

**Test Completion Check:** To ensure all tests should be either run or deliberately skipped and all known defects should be either fixed, deferred for a future release or accepted as a permanent restriction.

**Test Artifacts handover:** Tests and test environments should be handed over to those responsible for maintenance testing. Known defects accepted or deferred should be documented and communicated to those who will use and support the use of the system.

**Lessons learned:** Analyzing lessons learned to determine changes needed for future releases and projects. In retrospective meetings, plans are established to ensure that good  
practices can be repeated and poor practices are not repeated.

Archiving results, logs, reports, and other documents and work products in the CMS (configuration management system).

#### **27. What is test coverage?**

Test coverage helps in measuring the amount of testing performed by a set of tests.  
Test coverage can be done on both functional and non-functional activities. It assists testers to create tests that cover areas which are missing.

#### **28. What is Code coverage?**

Code coverage is different from Test coverage. Code coverage is about unit testing practices that must target all areas of the code at least once. It is usually done by developers or unit testers.

#### **29. List out Test Deliverables?**

1. Test Strategy
2. Test Plan
3. Effort Estimation Report
4. Test Scenarios
5. Test Cases/Scripts
6. Test Data
7. Requirement Traceability Matrix (RTM)
8. Defect Report/Bug Report
9. Test Execution Report
10. Graphs and Metrics
11. Test summary report
12. Test incident report
13. Test closure report
14. Release Note
15. Installation/configuration guide
16. User guide
17. Test status report
18. Weekly status report (Project manager to client)

#### **30. What are the most common components of a defect report?**

The most common components of a defect report format include the following

* Project Name
* Module Name
* Defect ID
* Defect detected on
* Defect detected by
* Priority
* Severity
* Defect resolved on
* Defect resolved by

#### **31. What are the levels of testing?**

In software testing, there are four testing levels.

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing

#### **32. What is Unit Testing?**

Unit Testing is also called Module Testing or Component Testing. It is done to check whether the individual unit or module of the source code is working properly. It is done by the developers in the developer’s environment. Learn more about [Unit Testing](https://www.softwaretestingmaterial.com/unit-testing/) in detail.

#### **33. What is Integration Testing?**

Integration Testing is the process of testing the interface between the two software units. Integration testing is done in three ways. Big Bang Approach, Top-Down Approach, Bottom-Up Approach. Learn more about [Integration Testing](https://www.softwaretestingmaterial.com/integration-testing/) in detail.

#### **34. What is System Testing?**

Testing the fully integrated application to evaluate the system’s compliance with its specified requirements is called System Testing AKA End to End testing. Verifying the completed system to ensure that the application works as intended or not

| **INTEGRATION TESTING** | **SYSTEM TESTING** |
| --- | --- |
| It is a low level testing | It is a high level testing |
| It is followed by System Testing | It is followed by Acceptance Testing |
| It is performed after unit testing | It is performed after integration testing |
| Different types of integration testing are: • Top bottom integration testing • Bottom top integration testing • Big bang integration testing • Sandwich integration testing | Different types of system testing are: • Regression testing • Sanity testing • Usability testing • Retesting • Load testing • Performance testing • Maintenance testing |
| Testers perform functional testing to validate the interaction of two modules | Testers perform both functional as well as non-functional testing to evaluate the functionality, usability, performance testing etc., |
| Performed to test whether two different modules interact effectively with each other or not | Performed to test whether the product is performing as per user expectations and the required specifications |
| It can be performed by both testers and developers | It is performed by testers |
| Testing takes place on the interface of two individual modules | Testing takes place on complete software application |

Difference between the

#### **35. What is Big Bang Approach?**

Combining all the modules once and verifying the functionality after completion of individual module testing.

Top-down and bottom-up are carried out by using dummy modules known as Stubs and Drivers. These Stubs and Drivers are used to stand in for missing components to simulate data communication between modules.

#### **36. What is Top-Down Approach?**

Testing takes place from top to bottom. High-level modules are tested first and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended. Stubs are used as a temporary module if a module is not ready for integration testing.

#### **37. What is Bottom-Up Approach?**

It is a reciprocate of the Top-Down Approach. Testing takes place from bottom to up. Lowest level modules are tested first and then high-level modules and finally integrating the high-level modules to a low level to ensure the system is working as intended. Drivers are used as a temporary module for integration testing.

#### **39. What is End-To-End Testing?**

In simple words, end-to-end testing is the process of testing software from start to end. Check this [End-To-End Testing](https://www.softwaretestingmaterial.com/end-to-end-testing-tutorial/) guide for more information. Also, refer [System Testing](https://www.softwaretestingmaterial.com/system-testing/) tutorial.

#### **40. What is Functional Testing?**

In simple words, what the system actually does is functional testing. To verify that each function of the software application behaves as specified in the requirement document. Testing all the functionalities by providing appropriate input to verify whether the actual output is matching the expected output or not. It falls within the scope of black box testing and the testers need not concern about the source code of the application.

Learn more about [Functional Testing](https://www.softwaretestingmaterial.com/functional-testing/) here

#### **41. What is Non-Functional Testing?**

In simple words, how well the system performs is non-functionality testing. Non-functional testing refers to various aspects of the software such as performance, load, stress, scalability, security, compatibility etc., Main focus is to improve the user experience on how fast the system responds to a request.

#### **42. What is the difference between functional and non-functional testing?**

Functional Testing vs Non-functional testing

| **Functional Testing** | **Non-functional Testing** |
| --- | --- |
| What the system actually does is functional testing | How well the system performs is non-functionality testing |
| To ensure that your product meets customer and business requirements and doesn’t have any major bugs | To ensure that the product stands up to customer expectations |
| To verify the accuracy of the software against expected output | To verify the behavior of the software at various load conditions |
| It is performed before non-functional testing | It is performed after functional testing |
| Example of functional test case is to verify the login functionality | Example of non-functional test case is to check whether the homepage is loading in less than 2 seconds |
| Testing types are • Unit testing • Smoke testing • User Acceptance • Integration Testing • Regression testing • Localization • Globalization • Interoperability | Testing types are • Performance Testing • Volume Testing • Scalability • Usability Testing • Load Testing • Stress Testing • Compliance Testing • Portability Testing • Disaster Recover Testing |
| It can be performed either manual or automated way | It can be performed efficiently if automated |

#### **43. What is Acceptance Testing?**

It is also known as pre-production testing.  This is done by the end-users along with the testers to validate the functionality of the application. After successful acceptance testing. Formal testing conducted to determine whether an application is developed as per the requirement. It allows the customer to accept or reject the application. Types of acceptance testing are Alpha, Beta & Gamma.

#### **44. On what basis is the acceptance plan prepared?**

The acceptance test plan is prepared using the following inputs.

* **Requirement Document:** The requirement document specifies what exactly is needed and not needed in the existing project from the customer’s perspective.
* **Input from customer:** Input from the customer will be in the format of formal emails, informal talks, discussions, etc.,
* **Project plan:** Project plan document prepared by the project manager.

All the above three inputs act as good inputs to prepare the acceptance test plan.

#### **45. What is Alpha Testing?**

Alpha testing is done by the in-house developers (who developed the software) and testers before we ship the software to the customers. Sometimes alpha testing is done by the client or outsourcing team with the presence of developers or testers. It is a part of [User Acceptance Testing](https://www.softwaretestingmaterial.com/user-acceptance-testing-uat/). The purpose of doing this is to find bugs before the customers start using the software.

#### **46. What is Beta Testing?**

Beta testing is done by a limited number of end-users before delivery. It is done after the Alpha Testing. Usually, it is done in the client’s place. Learn more about [Beta Testing](https://www.softwaretestingmaterial.com/beta-testing-a-detailed-guide/) here.

#### **47. What is Gamma Testing?**

Gamma testing is done when the software is ready for release with specified requirements. It is done at the client place. It is done directly by skipping all the in-house testing activities.

#### **48. What is Smoke Testing?**

Smoke Testing is done to make sure if the build we received from the development team is testable or not. It is also called as “Day 0” check. It is done at the “build level”. It helps not to waste the testing time to simply testing the whole application when the key features don’t work or the key bugs have not been fixed yet.

#### **49. What is Sanity Testing?**

Sanity Testing is done during the release phase to check for the main functionalities of the application without going deeper. It is also called as a subset of [Regression testing](https://www.softwaretestingmaterial.com/regression-testing/). It is done at the “release level”. At times due to release time constraints rigorous regression testing can’t be done to the build, sanity testing does that part by checking main functionalities.

#### **50. What is the difference between Sanity and Smoke Testing?**

Sanity vs Smoke Testing

| **SMOKE TESTING** | **SANITY TESTING** |
| --- | --- |
| Smoke Test is done to make sure if the build we received from the development team is testable or not | Sanity Test is done during the release phase to check for the main functionalities of the application without going deeper |
| Smoke Testing is performed by both Developers and Testers | Sanity Testing is performed by Testers alone |
| Smoke Testing exercises the entire application from end to end | Sanity Testing exercises only the particular component of the entire application |
| Smoke Testing, build may be either stable or unstable | Sanity Testing, build is relatively stable |
| It is done on initial builds. | It is done on stable builds. |
| It is a part of basic testing. | It is a part of regression testing. |
| Usually it is done every time there is a new build release. | It is planned when there is no enough time to do in-depth testing. |

#### **51. What is Retesting?**

To ensure that the defects which were found and posted in the earlier build were fixed or not in the current build. Say, Build 1.0 was released. Test team found some defects (Defect Id 1.0.1, 1.0.2) and posted. Build 1.1 was released, now testing the defects 1.0.1 and 1.0.2 in this build is retesting.

#### **52. What is Regression Testing?**

Repeated testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the changes in the software being tested or in another related or unrelated software components.

Usually, we do regression testing in the following cases:

1. New functionalities are added to the application
2. Change Requirement (In organizations, we call it as CR)
3. Defect Fixing
4. Performance Issue Fix
5. Environment change (E.g., Updating the DB from MySQL to Oracle)

#### **53. What do you mean by regression and confirmation testing?**

**Regression Testing:** Testing team re-execute the tests against the modified application to make sure whether the modified code breaks anything which was working earlier.

**Confirmation Testing:** Usually testers report a bug when a test fails. Dev Team releases a new version of the software after the defect is fixed. Now the testing team will retest to make sure the reported bug is actually fixed or not.

#### **54. What is GUI Testing?**

Graphical User Interface Testing is to test the interface between the application and the end user.

#### **55. What is Recovery Testing?**

Recovery testing is performed in order to determine how quickly the system can recover after the system crash or hardware failure. It comes under the type of non-functional testing.

#### **56. What is Globalization Testing?**

Globalization is a process of designing a software application so that it can be adapted to various languages and regions without any changes.

#### **57. What is Internationalization Testing (I18N Testing)?**

Refer Globalization Testing.

#### **58. What is Localization Testing (L10N Testing)?**

Localization is a process of adapting globalization software for a specific region or language by adding local specific components.

#### **59. What is Installation Testing?**

It is to check whether the application is successfully installed and it is working as expected after installation.

#### **60. What is Formal Testing?**

It is a process where the testers test the application by having pre-planned procedures and proper documentation.

#### **61. What is Risk Based Testing?**

Identify the modules or functionalities which are most likely cause failures and then testing those functionalities.

#### **62. What is Compatibility Testing?**

It is to deploy and check whether the application is working as expected in a different combination of environmental components.

#### **63. What is Exploratory Testing?**

Usually, this process will be carried out by domain experts. They perform testing just by exploring the functionalities of the application without having the knowledge of the requirements. Check our detailed guide on [Exploratory Testing](https://www.softwaretestingmaterial.com/exploratory-testing-tutorial/) and also don’t miss these popular [Exploratory Testing Tools](https://www.softwaretestingmaterial.com/exploratory-testing-tools/).

#### **64. What is Monkey Testing?**

Perform abnormal action on the application deliberately in order to verify the stability of the application. Check our in-depth guide on [Monkey Testing](https://www.softwaretestingmaterial.com/monkey-testing/).

#### **65. What is Usability Testing?**

To verify whether the application is user-friendly or not and was comfortably used by an end-user or not. The main focus in this testing is to check whether the end-user can understand and operate the application easily or not. An application should be self-exploratory and must not require training to operate it. Check this guide to [learn how to perform Usability Testing](https://www.softwaretestingmaterial.com/usability-testing/).

#### **66. What is Security Testing?**

Security testing is a process to determine whether the system protects data and maintains functionality as intended.

#### **67. What is Soak Testing?**

Running a system at high load for a prolonged period of time to identify the performance problems is called Soak Testing.

#### **68. What is Endurance Testing?**

Endurance testing is a non-functional testing type. It is also known as Soak Testing. Refer Soak testing.

#### **69. What is Performance Testing?**

This type of testing determines or validates the speed, scalability, and/or stability characteristics of the system or application under test. Performance is concerned with achieving response times, throughput, and resource-utilization levels that meet the performance objectives for the project or product.

Complete Tutorial: [Performance Testing](https://www.softwaretestingmaterial.com/performance-testing-tutorial/)

#### **70. What is Load Testing?**

It is to verify that the system/application can handle the expected number of transactions and to verify the system/application behavior under both normal and peak load conditions.

#### **71. What is Volume Testing?**

It is toverify that the system/application can handle a large amount of data

#### **72. What is Stress Testing?**

It is to verify the behavior of the system once the load increases more than its design expectations.

#### **73. What is Scalability Testing?**

Scalability testing is a type of non-functional testing. It is to determine how the application under test scales with increasing workload.

#### **74. What is Concurrency Testing?**

Concurrency testing means accessing the application at the same time by multiple users to ensure the stability of the system. This is mainly used to identify deadlock issues.

#### **75. What is Fuzz Testing?**

Fuzz testing is used to identify coding errors and security loopholes in an application. By inputting a massive amount of random data to the system in an attempt to make it crash to identify if anything breaks in the application.

### **Software Testing Interview Questions – 76-100:**

#### **76. What is Adhoc Testing?**

Ad-hoc testing is quite opposite to the formal testing. It is an informal testing type. In Adhoc testing, testers randomly test the application without following any documents and test design techniques. This testing is primarily performed if the knowledge of testers in the application under test is very high. Testers randomly test the application without any test cases or any business requirement document.

#### **77. What is Interface Testing?**

Interface testing is performed to evaluate whether two intended modules pass data and communicate correctly to one another.

#### **78. What is Reliability Testing?**

Perform testing on the application continuously for long period of time in order to verify the stability of the application

#### **79. What is Bucket Testing?**

Bucket testing is a method to compare two versions of an application against each other to determine which one performs better.

#### **80. What is A/B Testing?**

Refer Bucket Testing.

#### **81. What is Split Testing?**

Refer Bucket Testing.

#### **82. What are the principles of Software Testing?**

1. Testing shows presence of defects
2. Exhaustive testing is impossible
3. Early testing
4. Defect clustering
5. Pesticide Paradox
6. Testing is context depending
7. Absence of error fallacy

#### **83. What is Exhaustive Testing?**

Testing all the functionalities using all valid and invalid inputs and preconditions is known as Exhaustive testing.

#### **84. What is Early Testing?**

Defects detected in early phases of SDLC are less expensive to fix. So conducting early testing reduces the cost of fixing defects.

#### **85. What is Defect clustering?**

Defect clustering in software testing means that a small module or functionality contains most of the bugs or it has the most operational failures.

#### **86. What is Pesticide Paradox?**

Pesticide Paradox in software testing is the process of repeating the same test cases, again and again, eventually, the same test cases will no longer find new bugs. So to overcome this Pesticide Paradox, it is necessary to review the test cases regularly and add or update them to find more defects.

#### **87. What is Defect Cascading in Software Testing?**

Defect cascading in Software testing means triggering of other defects in an application. When a defect is not identified or goes unnoticed while testing, it invokes other defects. It leads to multiple defects in the later stages and results in an increase in a number of defects in the application.

For example, if there is a defect in an accounting system related to negative taxation then the negative taxation defect affects the ledger which in turn affects other reports such as Balance Sheet, Profit & Loss etc.,

#### **88. What is the difference between Outsourced Testing and Crowdsourced Testing**

Outsourced Testing vs Crowdsourced Testing

| **Outsource Testing** | **Crowdsourced testing** |
| --- | --- |
| A dedicated team is present to handle your testing Needs we can say it’s a third party which is unknown to you, test your application or product with a fresh set of mind. | A completely unknown pool of testing resources test your application, you can judge the quality of your product on the basis of number of bugs reported. |
| Payment is done on the basis of hours spent in testing, this estimation is done prior to the testing cycle. As an example testing outsourcing costs around 20 to 40$ per hour. | Payment is done on the basis of bug reported, no of severe bugs and low priority bugs. For example severe bug cost is 15$ and low priority bug is 3$ whereas medium priority bugs are costing 5$. |
| Application data is kept confidential and this is one of the code of ethics of every testing provider company. | Since there are n number of testers working on your application and they are not legally bound with Crowd source provider company, they are not bound to keep application data confidential. There are chances of data leakage if crowd source testing is done and no assurance of data privacy. |
| Communication is quite easy, because there is one representative always present to handle to share testing status, quality of your product. | Communication is bit tricky, because you have to understand the product quality on the basis of bugs logged by testers, you have to understand the bug by talking to the tester individually. |
| Quality is not compromised, since the objective is to identify all the bugs, within time and within budget. Entire team works to achieve this milestone, They present potential and valid bugs and organization is confident enough to fix only those bugs and get assured about their product quality. | Since there is no team concept, here focus is more on Quantity rather than quality. There are chances that your application is tested by 1000 of testers of different experiences. They may log 5K bugs of different severity. So its organization’s responsibility to identify the real bugs and fix them. |
| Testing platform and environment is completely owned by outsourced company, they are well settled with all useful software, tools, management tools, OS and Devices. | Testing environment is totally dependent on individual tester, some testers are on testing on MAC machine or some are testing on Windows, some are testing on Android or some are testing on Apple. |
| Skilled testers are in the team, there are fixed no of testers in the team. Each tester is well skilled in a particular area like mobile testing, performance testing, automation testing, functional testing. | Huge no of testers, with different expertise and different years of experiences, so chances of quality bugs depends of expertise of testers. Which may be surprisingly good or bad too. |
| One team, one time zone, restricted deadline, and planned budget, in this way testing cycles are complete. | No team concept, Different time zones, no deadlines but bugs are reported very fast. |
| Bugs reported are generally predictive in nature because testers work within a scope of testing. They don’t touch few areas because it may not in their budget. | Here no limitations of testing scope, N no of testers, n no of directions of breaking the system, Due to this testing cycle goes through a real scenario, for example n number of users are accessing application they might get some security flaw in the application. |
| High paid in comparison to Crowd sourced testing but lesser than Inhouse Testing team. | Budget friendly, quick results some time real and unexpected issues are identified. |

#### **89. What is Walk Through?**

A walkthrough is an informal meeting conducts to learn, gain understanding, and find defects. The author leads the meeting and clarifies the queries raised by the peers in the meeting.

#### **90. What is Inspection?**

Inspection is a formal meeting lead by a trained moderator, certainly not by the author. The document under inspection is prepared and checked thoroughly by the reviewers before the meeting. In the inspection meeting, the defects found are logged and shared with the author for appropriate actions. Post inspection, a formal follow-up process is used to ensure a timely and corrective action.

#### **91. Who are all involved in an inspection meeting?**

Author, Moderator, Reviewer(s), Scribe/Recorder and Manager.

#### **92. What is a Defect?**

The variation between the actual results and expected results is known as a defect. If a developer finds an issue and corrects it by himself in the development phase then it’s called a defect

#### **93. What is a Bug?**

If testers find any mismatch in the application/system in testing phase then they call it as Bug

#### **94. What is an Error?**

We can’t compile or run a program due to a coding mistake in a program. If a developer unable to successfully compile or run a program then they call it as an error.

#### **95. What is a Failure?**

Once the product is deployed and customers find any issues then they call the product as a failure product. After release, if an end user finds an issue then that particular issue is called as a failure

#### **96. What is Bug Severity?**

Bug/Defect severity can be defined as the impact of the bug on customer’s business. It can be Critical, Major or Minor. In simple words, how much effect will be there on the system because of a particular defect.

#### **97. What is Bug Priority?**

Defect priority can be defined as how soon the defect should be fixed. It gives the order in which a defect should be resolved. Developers decide which defect they should take up next based on the priority. It can be High, Medium or Low. Most of the times the priority status is set based on the customer requirement.

#### **98. Tell some examples of Bug Severity and Bug Priority?**

**High Priority & High Severity:**Submit button is not working on a login page and customers are unable to login to the application

**Low Priority & High Severity:**Crash in some functionality which is going to deliver after couple of releases

**High Priority & Low Severity:**Spelling mistake of a company name on the homepage

**Low Priority & Low Severity:**FAQ page takes a long time to load

#### **99. What is a Critical Bug?**

A critical bug is a show stopper which means a large piece of functionality or major system component is completely broken and there is no workaround to move further.  
For example, Due to a bug in one module, we cannot test the other modules because that blocker bug has blocked other modules. Bugs which affects the customers business are considered as critical.

**Example:**

1. “Sign In” button is not working on Gmail App and Gmail users are blocked to login to their accounts.  
2. An error message pops up when a customer clicks on transfer money button in a Banking website.

#### **100. What is the difference between a Standalone application, Client-Server application and Web application?**

**Standalone application:**

Standalone applications follow one-tier architecture. Presentation, Business, and Database layer are in one system for a single user.

**Client-Server Application:**

Client-server applications follow two-tier architecture. Presentation and Business layer are in a client system and Database layer on another server. It works majorly in Intranet.

**Web Application:**

Web server applications follow three-tier or n-tier architecture. The presentation layer is in a client system, a Business layer is in an application server and Database layer is in a Database server. It works both in Intranet and Internet.

#### **101. What is Bug Life Cycle?**

**Bug life cycle** is also known as **Defect life cycle**. In Software Development process, the bug has a life cycle. The bug should go through the life cycle to be closed. Bug life cycle varies depends upon the tools (QC, JIRA etc.,) used and the process followed in the organization.

#### **102. What is Bug Leakage?**

A bug which is actually missed by the testing team while testing and the build was released to the Production. If now that bug (which was missed by the testing team) was found by the end user or customer then we call it as Bug Leakage.

#### **103. What is Bug Release?**

Releasing the software to the Production with the known bugs then we call it as Bug Release. These known bugs should be included in the release note.

#### **104. What is Defect Age?**

Defect age can be defined as the time interval between date of defect detection and date of defect closure.

Defect Age = Date of defect closure – Date of defect detection

Assume, a tester found a bug and reported it on 1 Jan 2016 and it was successfully fixed on 5 Jan 2016. So the defect age is 5 days.

#### **105. What is Error Seeding?**

Error seeding is a process of adding known errors intendedly in a program to identify the rate of error detection. It helps in the process of estimating the tester skills of finding bugs and also to know the ability of the application (how well the application is working when it has errors.)

#### **106. What is Error Guessing?**

Error guessing is also a method of test case design similar to error seeding. In error guessing, testers design test cases by guessing the possible errors that might occur in the software application. The intention is to catch the errors immediately.

#### **107. What is Showstopper Defect?**

A showstopper defect is a defect which won’t allow a user to move further in the application. It’s almost like a crash.

Assume that login button is not working. Even though you have a valid username and valid password, you could not move further because the login button is not functioning.

#### **108. What is HotFix?**

A bug that needs to handle as a high priority bug and fix it immediately.

#### **109. What are the different strategies for rollout to end-users?**

There are four strategies to be followed for the rollout of any software testing project are as follows:

* Pilot
* Gradual Implementation
* Phased Implementation
* Parallel Implementation

#### **110. What is Boundary Value Analysis?**

Boundary value analysis (BVA) is based on testing the boundary values of valid and invalid partitions. The Behavior at the edge of each equivalence partition is more likely to be incorrect than the behavior within the partition, so boundaries are an area where testing is likely to yield defects. Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition. A boundary value for a valid partition is a valid boundary value. Similarly, a boundary value for an invalid partition is an invalid boundary value.

#### **111. What is Equivalence Class Partition?**

Equivalence Partitioning is also known as Equivalence Class Partitioning. In equivalence partitioning, inputs to the software or system are divided into groups that are expected to exhibit similar behavior, so they are likely to be proposed in the same way. Hence selecting one input from each group to design the test cases.

#### **112. What is Decision Table testing?**

Decision Table is aka Cause-Effect Table. This test technique is appropriate for functionalities which has logical relationships between inputs (if-else logic). In the Decision table technique, we deal with combinations of inputs. To identify the test cases with a decision table, we consider conditions and actions. We take conditions as inputs and actions as outputs.

#### **113. What is State Transition?**

Using state transition testing, we pick test cases from an application where we need to test different system transitions. We can apply this when an application gives a different output for the same input, depending on what has happened in the earlier state.

#### **114. What is an entry criteria?**

The prerequisites that must be achieved before commencing the testing process.

#### **115. What is an exit criteria?**

The conditions that must be met before testing should be concluded. [.](https://www.softwaretestingmaterial.com/entry-and-exit-criteria/)

#### **116. What is SDLC?**

Software Development Life Cycle (SDLC) aims to produce a high-quality system that meets or exceeds customer expectations, works effectively and efficiently in the current and planned information technology infrastructure, and is inexpensive to maintain and cost-effective to enhance.

#### **117. What are the different available models of SDLC?**

1. [Waterfall](https://www.softwaretestingmaterial.com/waterfall-model-in-sdlc/)
2. [Spiral](https://www.softwaretestingmaterial.com/spiral-model-in-sdlc/)
3. [V Model](https://www.softwaretestingmaterial.com/v-model-in-sdlc/)
4. Prototype
5. [Agile](https://www.softwaretestingmaterial.com/agile-scrum-methodology/)

#### **118. Can you do System testing at any stage of SDLC?**

We can do System Testing only when all the units are in place and working properly. It can only be done before User Acceptance Testing (UAT).

#### **119. What is the procedure of manual testing?**

Manual testing is crucial for testing software applications more thoroughly. The procedure of manual testing comprises of the following.  
1. Planning and Control  
2. Analysis and Design  
3. Implementation and Execution  
4. Evaluating and Reporting  
5. Test Closure activities

#### **120. What is STLC?**

STLC (Software Testing Life Cycle) identifies what test activities to carry out and when to accomplish those test activities. Even though testing differs between Organizations, there is a testing life cycle.

#### **121. What is RTM?**

Requirements Traceability Matrix (RTM) is used to trace the requirements to the tests that are needed to verify whether the requirements are fulfilled. Requirement Traceability Matrix AKA Traceability Matrix or Cross Reference Matrix.

#### **122. What is Test Metrics?**

Software test metrics is to monitor and control process and product. It helps to drive the project towards our planned goals without deviation. Metrics answer different questions. It’s important to decide what questions you want answers to.

#### **123. When to stop testing? (Or) How do you decide when you have tested enough?**

There are many factors involved in real-time projects to decide when to stop testing.

1. Requirement coverage reaches a specified point
2. Testing deadlines or release deadlines
3. When the complete testing budget is exhausted
4. By reaching the decided pass percentage of test cases
5. The risk in the project is under an acceptable limit
6. All the high priority bugs, blockers are fixed
7. When acceptance criteria is met
8. After the Alpha and Beta testing period ends
9. Depends on Management decision

Don’t miss: [ISTQB Quiz](https://www.softwaretestingmaterial.com/istqb-quiz/)

**124. What is API Testing?**

API testing is a type of [software testing](https://www.softwaretestingmaterial.com/software-testing/) that involves testing APIs directly and also as a part of integration testing to check whether the API meets expectations in terms of functionality, reliability, performance, and security of an application. In API Testing our main focus will be on a Business logic layer of the [software architecture](https://www.softwaretestingmaterial.com/software-architecture/). API testing can be performed on any software system which contains multiple APIs. API testing won’t concentrate on the look and feel of the application. API testing is entirely different from GUI Testing.

Learn [API Testing](https://www.softwaretestingmaterial.com/api-testing)

#### **125. Which test cases are written first white boxes or black box?**

The simple answer is black-box test cases are written first.

Let’s see why black-box test cases are written first compared to white box test cases.  
Prerequisites to start writing black-box test cases are Requirement documents or design documents. These documents will be available before initiating a project.  
Prerequisites to start writing white box test cases are the internal architecture of the application. The internal architecture of the application will be available in the later part of the project i.e., designing.

#### **126. What is the workbench concept in Software Testing?**

Workbench is a practice of documenting how a specific activity must be performed. It is often referred to as phases, steps, and tasks.

In every workbench there will be five tasks such as Input, Execute, Check, Output, and rework.

#### **127. What is Random testing?**

In random testing is a form of black-box software testing technique where the application is testing by generating random data.

### 1. What is Exploratory Testing?

Exploratory testing is 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 use 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 afterward.

### 2. What is “use case testing”?

In order to identify and execute the functional requirement of an application from start to finish “use case” is used and the techniques used to do this is known as “Use Case Testing.”

### 3. What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?

SDLC deals with development/coding of the software while STLC deales with validation and verification of the software

### 4. What is traceability matrix?

The relationship between test cases and requirements is shown with the help of a document. This document is known as a traceability matrix.

### 5. What is Equivalence partitioning testing?

Equivalence partitioning testing is a software testing technique which divides the application input test data into each partition at least once of equivalent data from which test cases can be derived. By this testing method, it reduces the time required for software testing.

### 6. What is white box testing and list the types of white box testing?

White box testing technique involves selection of test cases based on an analysis of the internal structure (Code coverage, branches coverage, paths coverage, condition coverage, etc.) of a component or system. It is also known as Code-Based testing or Structural testing. Different types of white box testing are

1. Statement Coverage
2. Decision Coverage

### 7. In white box testing, what do you verify?

In white box testing following steps are verified.

1. Verify the security holes in the code
2. Verify the incomplete or broken paths in the code
3. Verify the flow of structure according to the document specification
4. Verify the expected outputs
5. Verify all conditional loops in the code to check the complete functionality of the application
6. Verify the line by line coding and cover 100% testing

### 8. What is black box testing? What are the different black box testing techniques?

Black box testing is the software testing method which is used to test the software without knowing the internal structure of code or program. This testing is usually done to check the functionality of an application. The different black box testing techniques are

1. Equivalence Partitioning
2. Boundary value analysis
3. Cause-effect graphing

### 9. What is the difference between static and dynamic testing?

Static testing: During Static testing method, the code is not executed, and it is performed using the software documentation.

Dynamic testing: To perform this testing the code is required to be in an executable form.

### 10. What are verification and validation?

Verification is a process of evaluating software at the development phase. It helps you to decide whether the product of a given application satisfies the specified requirements. Validation is the process of evaluating software at the after the development process and to check whether it meets the customer requirements.

### 11. What are the different test levels?

There are four test levels

1. Unit/component/program/module testing
2. Integration testing
3. System testing
4. Acceptance testing

### 12. What is Integration testing?

[Integration testing](https://www.guru99.com/integration-testing.html)is a level of software testing process, where individual units of an application are combined and tested. It is usually performed after unit and functional testing.

### 13. What Test Plans consists of?

Test design, scope, test strategies, approach are various details that Test plan document consists of.

1. Test case identifier
2. Scope
3. Features to be tested
4. Features not to be tested
5. Test strategy & Test approach
6. Test deliverables
7. Responsibilities
8. Staffing and training
9. Risk and Contingencies

### 14. What is the difference between UAT (User Acceptance Testing) and System testing?

System Testing: System testing is finding defects when the system undergoes testing as a whole; it is also known as end-to-end testing. In such type of testing, the application suffers from beginning till the end.

UAT: User Acceptance Testing (UAT) involves running a product through a series of specific tests which determines whether the product will meet the needs of its users.

### 15. Mention the difference between Data Driven Testing and Retesting?

**Retesting:** It is a process of checking bugs that are actioned by the development team to verify that they are fixed.

**Data Driven Testing (DDT):**In data driven testing process, the application is tested with multiple test data. The application is tested with a different set of values.

## Advanced Manual Software Testing Interview Questions for 3/5/10 Years Experience

### 16. What are the valuable steps to resolve issues while testing?

* Record: Log and handle any problems which have happened
* Report: Report the issues to higher level manager
* Control: Define the issue management process

### 17. What is the difference between test scenarios, test cases, and test script?

Difference between test scenarios and test cases is that

**Test Scenarios:** A Test Scenario is any functionality that can be tested. It is also called Test Condition or Test Possibility.

**Test Cases:** It is a document that contains the steps that have to be executed; it has been planned earlier.

**Test Script:**It is written in a programming language and it’s a short program used to test part of the functionality of the software system. In other words a written set of steps that should be performed manually.

### 18. What is Latent defect?

**Latent defect:**This defect is an existing defect in the system which does not cause any failure as the exact set of conditions has never been met

### 19. What are the two parameters which can be useful to know the quality of test execution?

To know the quality of test execution, we can use two parameters

* Defect reject ratio
* Defect leakage ratio

Parameters for quality of test execution

### 20. What is the function of the software testing tool “phantom”?

Phantom is a freeware and is used for windows GUI automation scripting language. It allows us to take control of windows and functions automatically. It can simulate any combination of keystrokes and mouse clicks as well as menus, lists and more.

### 21. Explain what Test Deliverables is?

Test Deliverables are a set of documents, tools and other components that have to be developed and maintained in support of testing.

There are different test deliverables at every phase of the software development lifecycle

* Before Testing
* During Testing
* After the Testing

### 22. What is mutation testing?

Mutation testing is a technique to identify if a set of test data or test case is useful by intentionally introducing various code changes (bugs) and retesting with original test data/ cases to determine if the bugs are detected.

### 23. What all things you should consider before selecting automation tools for the AUT?

* Technical Feasibility
* Complexity level
* Application stability
* Test data
* Application size
* Re-usability of automated scripts
* Execution across environment

### 24. How will you conduct Risk Analysis?

For the risk analysis following steps need to be implemented

1. Finding the score of the risk
2. Making a profile for the risk
3. Changing the risk properties
4. Deploy the resources of that test risk
5. Making a database of risk

### 25. What are the categories of debugging?

Categories for debugging

1. Brute force debugging
2. Backtracking
3. Cause elimination
4. Program Slicing
5. Fault tree analysis

### 26. What is fault masking explain with example?

When the presence of one defect hides the presence of another defect in the system, it is known as fault masking.

Example: If the “Negative Value” cause a firing of unhandled system exception, the developer will prevent the negative values input. This will resolve the issue and hide the defect of unhandled exception firing.

### 27. Explain what Test Plan is? What is the information that should be covered in Test Plan?

A test plan can be defined as a document describing the scope, approach, resources, and schedule of testing activities and a test plan should cover the following details.

* Test Strategy
* Test Objective
* Exit/Suspension Criteria
* Resource Planning
* Test Deliverables

### 28. How can you eliminate the product risk in your project?

It helps you to eliminate product risk in your project, and there is a simple yet crucial step that can reduce the product risk in your project.

* Investigate the specification documents
* Have discussions about the project with all stakeholders including the developer
* As a real user walk around the website

### 29. What is the common risk that leads to project failure?

The common risk that leads to a project failure are

* Not having enough human resource
* Testing Environment may not be set up properly
* Limited Budget
* Time Limitations

### 30. On what basis you can arrive at an estimation for your project?

To estimate your project, you have to consider the following points

* Divide the whole project into the smallest tasks
* Allocate each task to team members
* Estimate the effort required to complete each task
* Validate the estimation

### 31. Explain how you would allocate a task to team members?

|  |  |
| --- | --- |
| **Task** | **Member** |
| * Analyze software requirement specification | * All the members |
| * Create the test specification | * Tester/Test Analyst |
| * Build up the test environment | * Test administrator |
| * Execute the test cases | * Tester, a Test administrator |
| * Report defects | * Tester |

### 32. Explain what is testing type and what are the commonly used testing type?

To get an expected test outcome, a standard procedure is followed which is referred to as Testing Type.

Commonly used testing types are

* Unit Testing: Test the smallest code of an application
* API Testing: Testing API created for the application
* Integration Testing: Individual software modules are combined and tested
* System Testing: Complete testing of the system
* Install/UnInstall Testing: Testing done from the point of client/customer view
* Agile Testing: Testing through Agile technique

### 33. While monitoring your project what all things you have to consider?

The things that have to be taken in considerations are

* Is your project on schedule
* Are you over budget
* Are you working towards the same career goal
* Have you got enough resources
* Are there any warning signs of impending problems
* Is there any pressure from management to complete the project sooner

### 34. What are the common mistakes which create issues?

* Matching resources to wrong projects
* Test manager lack of skills
* Not listening to others
* Poor Scheduling
* Underestimating
* Ignoring the small problems
* Not following the process

### 35. What does a typical test report contain? What are the benefits of test reports?

A test report contains the following things:

* Project Information
* Test Objective
* Test Summary
* Defect

The benefits of test reports are:

* Current status of project and quality of product are informed
* If required, stakeholder and customer can take corrective action
* A final document helps to decide whether the product is ready for release

### 36. What is test management review and why it is important?

Management review is also referred to as [Software Quality Assurance](https://www.guru99.com/software-quality-assurance-test-audit-review-makes-your-life-easy.html) or SQA. SQA focusses more on the software process rather than the software work products. It is a set of activities designed to make sure that the project manager follows the standard process. SQA helps test manager to benchmark the project against the set standards.

### 37. What are the best practices for software quality assurance?

The best practices for an effective SQA implementation is

* Continuous Improvement
* Documentation
* Tool Usage
* Metrics
* Responsibility by team members
* Experienced SQA auditors

### 38. When is RTM (Requirement Traceability Matrix) prepared?

RTM is prepared before test case designing. Requirements should be traceable from review activities.

### 39. What is the difference between Test matrix and Traceability matrix?

**Test Matrix**: Test matrix is used to capture actual quality, effort, the plan, resources and time required to capture all phases of software testing

**Traceability Matrix**: Mapping between test cases and customer requirements is known as Traceability Matrix

### 40. In manual testing what are stubs and drivers?

Both stubs and drivers are part of incremental testing. In incremental testing, there are two approaches namely bottom-up and top-down approach. Drivers are used in bottom-up testing and stub is used for a top-down approach. In order to test the main module, the stub is used, which is a dummy code or program.

### 41. What is the step you would follow once you find the defect?

Once a defect is found you would follow the step

a) Recreate the defect

b) Attach the screenshot

c) Log the defect

### 42. Explain what is “Test Plan Driven” or “Key Word Driven” method of testing?

This technique uses the actual test case document developed by testers using a spreadsheet containing special “key Words”. The key words control the processing.

### 43. What is the DFD (Data Flow Diagram)?

When a “flow of data” through an information system is graphically represented, then it is known as Data Flow Diagram. It is also used for the visualization of data processing.

### 44. Explain what LCSAJ is?

LCSAJ stands for ‘linear code sequence and jump.’ It consists of the following three items

a) Start of the linear sequence of executable statements

b) End of the linear sequence

c) The target line to which control flow is transferred at the end of the linear sequence

### 45. Explain what N+1 testing is?

The variation of regression testing is represented as N+1. In this technique, the testing is performed in multiple cycles in which errors found in test cycle ‘N’ are resolved and re-tested in test cycle N+1. The cycle is repeated unless there are no errors found.

### 46. What is Fuzz testing and when it is used?

Fuzz testing is used to detect security loopholes and coding errors in software. In this technique, random data is added to the system in an attempt to crash the system. If vulnerability persists, a tool called fuzz tester is used to determine potential causes. This technique is more useful for bigger projects but only detects a major fault.

### 47. Mention what the main advantages of statement coverage metric of software testing are?

The benefit of statement coverage metric is that

a) It does not require processing source code and can be applied directly to object code

b) Bugs are distributed evenly through the code, due to which percentage of executable statements covered reflects the percentage of faults discovered

### 48. How to generate test cases for “replace a string” method?

a) If characters in new string > characters in the previous string. None of the characters should get truncated

b) If characters in new string< characters in the previous string. Junk characters should not be added

c) Spaces after and before the string should not be deleted

d) String should be replaced only for the first occurrence of the string

### 49. How will you handle a conflict amongst your team members?

* I will talk individually to each person and note their concerns
* I will find a solution to the common problems raised by team members
* I will hold a team meeting, reveal the solution and ask people to co-operate

### 50. Mention what are the categories of defects?

Mainly there are three defect categories

* **Wrong**: When a requirement is implemented incorrectly
* **Missing**: It is a variance from the specification, an indication that a specification was not implemented or a requirement of the customer is not met
* **Extra**: A requirement incorporated into the product that was not given by the end customer. It is considered as a defect because it is a variance from the existing requirements

### 51. Explain how does a test coverage tool work?

The code coverage testing tool runs parallel while performing testing on the actual product. The code coverage tool monitors the executed statements of the source code. When the final testing is done, we get a complete report of the pending statements and also get the coverage percentage.

### 52. Mention what the difference between a “defect” and a “failure” in software testing is?

In simple terms when a defect reaches the end customer, it is called a failure while the defect is identified internally and resolved; then it is referred to as a defect.

### 53. Explain how to test documents in a project that span across the software development lifecycle?

The project span across the software development lifecycle in the following manner

* Central/Project test plan: It is the main test plan that outlines the complete test strategy of the project. This plan is used till the end of the software development lifecycle
* Acceptance test plan: This document begins during the [requirement phase](https://www.guru99.com/lifecycle-of-requirement.html) and is completed at the final delivery
* System test plan: This plan starts during the design plan and proceeds until the end of the project
* Integration and Unit test plan: Both these test plans start during the execution phase and last until the final delivery

### 54. Explain which test cases are written first black boxes or white boxes?

Black box test cases are written first as to write black box test cases; it requires project plan and requirement document all these documents are easily available at the beginning of the project. While writing white box test cases requires more architectural understanding and is not available at the start of the project.

### 55. Explain what the difference between latent and masked defects is?

* **Latent defect:** A latent defect is an existing defect that has not caused a failure because the sets of conditions were never met
* **Masked defect:** It is an existing defect that has not caused a failure because another defect has prevented that part of the code from being executed

### 56. Mention what bottom-up testing is?

Bottom-up testing is an approach to integration testing, where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.

### 57. Mention what the different types of test coverage techniques are?

Different types of test coverage techniques include

* **Statement Coverage:** It verifies that each line of source code has been executed and tested
* **Decision Coverage:** It ensures that every decision in the source code is executed and tested
* **Path Coverage:** It ensures that every possible route through a given part of the code is executed and tested

### 58. Mention what the meaning of breath testing is?

Breath testing is a test suite that exercises the full functionality of a product but does not test features in detail

### 59. Explain what the meaning of Code Walk Through is?

Code Walk Through is the informal analysis of the program source code to find defects and verify coding techniques

### 60. Mention what the basic components of defect report format are?

The essential components of defect report format include

* Project Name
* Module Name
* Defect detected on
* Defect detected by
* Defect ID and Name
* Snapshot of the defect
* Priority and Severity status
* Defect resolved by
* Defect resolved on

### 61. Mention what the purpose behind doing end-to-end testing is?

End-to-end testing is done after functional testing. The purpose behind doing end-to-end testing is that

* To validate the [software requirements](https://www.guru99.com/learn-software-requirements-analysis-with-case-study.html) and integration with external interfaces
* Testing application in real-world environment scenario
* Testing of interaction between application and database

### 62. Explain what it means by test harness?

A test harness is configuring a set of tools and test data to test an application in various conditions, and it involves monitoring the output with expected output for correctness.

### 63. Explain in a testing project what testing activities would you automate?

In testing project testing activities, you would automate are

* Tests that need to be run for every build of the application
* Tests that use multiple data for the same set of actions
* Identical tests that need to be executed using different browsers
* Mission critical pages
* A transaction with pages that do not change in a short time

### 64. What is the MAIN benefit of designing tests early in the life cycle?

It helps prevent defects from being introduced into the code.

### 65. What is risk-based testing?

Risk-based[Testing](https://www.guru99.com/software-testing.html)is the term used for an approach to creating a[Test Strategy](https://www.guru99.com/how-to-create-test-strategy-document.html)that is based on prioritizing tests by risk. The basis of the approach is a detailed risk analysis and prioritizing of risks by risk level. Tests to address each risk are then specified, starting with the highest risk first.

### 66. What is the KEY difference between preventative and reactive approaches to testing?

Preventative tests are designed early; reactive tests are designed after the software has been produced.

### 67. What is the purpose of exit criteria?

The purpose of exit criteria is to define when a test level is completed.

### 68. What determines the level of risk?

The likelihood of an adverse event and the impact of the event determine the level of risk.

### 69. When is used Decision table testing?

Decision table testing is used for testing systems for which the specification takes the form of rules or cause-effect combinations. In a decision table, the inputs are listed in a column, with the outputs in the same column but below the inputs. The remainder of the table explores combinations of inputs to define the outputs produced.

### 70. Why we use decision tables?

The techniques of equivalence partitioning and boundary value analysis are often applied to specific situations or inputs. However, if different combinations of inputs result in different actions being taken, this can be more difficult to show using equivalence partitioning and boundary value analysis, which tend to be more focused on the user interface. The other two specification-based techniques, decision tables, and state transition testing are more focused on business logic or business rules. A decision table is a good way to deal with combinations of things (e.g., inputs). This technique is sometimes also referred to as a ’cause-effect’ table. The reason for this is that there is an associated logic diagramming technique called ’cause-effect graphing’ which was sometimes used to help derive the decision table

### 71. What is the MAIN objective when reviewing a software deliverable?

To identify defects in any software work product.

### 72. Which of the following defines the expected results of a test? Test case specification or test design specification.

Test case specification defines the expected results of a test.

### 73. What is the benefit of test independence?

It avoids author bias in defining effective tests.

### 74. As part of which test process do you determine the exit criteria?

The exit criteria are determined on the bases of ‘Test Planning’.

### 75. What is Alpha testing?

Pre-release testing by end user representatives at the developer’s site.

### 76. What is beta testing?

Testing performed by potential customers at their own locations.

### 77. Mention what the difference between Pilot and Beta testing is?

The difference between a pilot and beta testing is that pilot testing is actually done using the product by the group of users before the final deployment, and in beta testing, we do not input real data, but it is installed at the end customer to validate if the product can be used in production.

### 78. Given the following fragment of code, how many tests are required for 100% decision coverage?

if width > length

thenbiggest\_dimension = width

if height > width

thenbiggest\_dimension = height

end\_if

elsebiggest\_dimension = length

if height > length

thenbiggest\_dimension = height

end\_if

end\_if

4

### 79. You have designed test cases to provide 100% statement and 100% decision coverage for the following fragment of code. if width > length then biggest\_dimension = width else biggest\_dimension = length end\_if The following has been added to the bottom of the code fragment above. print “Biggest dimension is ” &biggest\_dimensionprint “Width: ” & width print “Length: ” & length How many more test cases are required?

None, existing test cases can be used.

### 80. What is the difference between Testing Techniques and Testing Tools?

Testing technique: – Is a process for ensuring that some aspects of the application system or unit functions properly there may be few techniques but many tools.

Testing Tools: – Is a vehicle for performing a test process. The tool is a resource to the tester, but itself is insufficient to conduct testing

### 81. We use the output of the requirement analysis, the requirement specification as the input for writing …

User Acceptance Test Cases

### 82. Repeated Testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the changes in the software being tested or in another related or unrelated software component:

Regression Testing

### 83. A wholesaler sells printer cartridges. The minimum order quantity is 5. There is a 20% discount for orders of 100 or more printer cartridges. You have been asked to prepare test cases using various values for the number of printer cartridges ordered. Which of the following groups contain three test inputs that would be generated using Boundary Value Analysis?

4, 5, 99

### 84. What is component testing?

Component testing, also known as unit, module, and program testing, searches for defects in and verifies the functioning of software (e.g., modules, programs, objects, classes, etc.) that are separately testable. Component testing may be done in isolation from the rest of the system depending on the context of the development life cycle and the system. Most often stubs and drivers are used to replace the missing software and simulate the interface between the software components simply. A stub is called from the software component to be tested; a driver calls a component to be tested.

### 85. What is functional system testing?

Testing the end to end functionality of the system as a whole is defined as a functional system testing.

### 86. What are the benefits of Independent Testing?

Independent testers are unbiased and identify different defects at the same time.

### 87. In a REACTIVE approach to testing when would you expect the bulk of the test design work to be begun?

The bulk of the test design work begun after the software or system has been produced.

### 88. What are the different Methodologies in Agile Development Model?

There are currently seven different agile methodologies that I am aware of:

1. Extreme Programming (XP)
2. Scrum
3. Lean Software Development
4. Feature-Driven Development
5. Agile Unified Process
6. Crystal
7. Dynamic Systems Development Model (DSDM)

### 89. Which activity in the fundamental test process includes evaluation of the testability of the requirements and system?

A ‘Test Analysis’ and ‘Design’ includes evaluation of the testability of the requirements and system.

### 90. What is typically the MOST important reason to use risk to drive testing efforts?

Because testing everything is not feasible.

### 91. What is random/monkey testing? When is it used?

Random testing is often known as monkey testing. In such type of testing data is generated randomly often using a tool or automated mechanism. With this randomly generated input, the system is tested, and results are analyzed accordingly. These testing are less reliable; hence it is normally used by the beginners and to see whether the system will hold up under adverse effects.

### 92. Which of the following are valid objectives for incident reports?

1. Provide developers and other parties with feedback about the problem to enable identification, isolation, and correction as necessary.
2. Provide ideas for test process improvement.
3. Provide a vehicle for assessing tester competence.
4. Provide testers with a means of tracking the quality of the system under test.

### 93. Consider the following techniques. Which are static and which are dynamic techniques?

1. Equivalence Partitioning.
2. Use Case Testing.
3. Data Flow Analysis.
4. Exploratory Testing.
5. Decision Testing.
6. Inspections.

Data Flow Analysis and Inspections are static; Equivalence Partitioning, Use Case Testing, Exploratory Testing and Decision Testing are dynamic.

### 94. Why are static testing and dynamic testing described as complementary?

Because they share the aim of identifying defects but differ in the types of defect they find.

### 95. What are the phases of a formal review?

In contrast to informal reviews, formal reviews follow a formal process. A typical formal review process consists of six main steps:

1. Planning
2. Kick-off
3. Preparation
4. Review meeting
5. Rework
6. Follow-up.

### 96. What is the role of moderator in the review process?

The moderator (or review leader) leads the review process. He or she determines, in co-operation with the author, the type of review, approach and the composition of the review team. The moderator performs the entry check and the follow-up on the rework, in order to control the quality of the input and output of the review process. The moderator also schedules the meeting, disseminates documents before the meeting, coaches other team members, paces the meeting, leads possible discussions and stores the data that is collected.

### 97. What is an equivalence partition (also known as an equivalence class)?

An input or output ranges of values such that only one value in the range becomes a test case.

### 98. When should configuration management procedures be implemented?

During test planning.

### 99. A Type of Functional Testing, which investigates the functions relating to the detection of threats, such as virus from malicious outsiders?

Security Testing

### 100. Testing wherein we subject the target of the test, to varying workloads to measure and evaluate the performance behaviors and the ability of the target and the test to continue to function properly under these different workloads?

Load Testing

### 101. Testing activity which is performed to expose defects in the interfaces and in the interaction between integrated components is?

Integration Level Testing

### 102. What are the Structure-based (white-box) testing techniques?

Structure-based testing techniques (which are also dynamic rather than static) use the internal structure of the software to derive test cases. They are commonly called ‘white-box’ or ‘glass-box’ techniques (implying you can see into the system) since they require knowledge of how the software is implemented, that is, how it works. For example, a structural technique may be concerned with exercising loops in the software. Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.

### 103. When should “Regression Testing” be performed?

After the software has changed or when the environment has changed [Regression testing](https://www.guru99.com/regression-testing.html) should be performed.

### 104. What is negative and positive testing?

A negative test is when you put in an invalid input and receives errors. While positive testing is when you put in a valid input and expect some action to be completed in accordance with the specification.

### 105. What is the purpose of a test completion criterion?

The purpose of test completion criterion is to determine when to stop testing

### 106. What can static analysis NOT find?

For example memory leaks.

### 107. What is the difference between re-testing and regression testing?

Re-testing ensures the original fault has been removed; regression testing looks for unexpected side effects.

### 108. What are the Experience-based testing techniques?

In experience-based techniques, people’s knowledge, skills, and background are a prime contributor to the test conditions and test cases. The experience of both technical and business people is important, as they bring different perspectives to the test analysis and design process. Due to previous experience with similar systems, they may have insights into what could go wrong, which is very useful for testing.

### 109. What type of review requires formal entry and exit criteria, including metrics?

Inspection

### 110. Could reviews or inspections be considered part of testing?

Yes, because both help detects faults and improves quality.

### 111. An input field takes the year of birth between 1900 and 2004 what the boundary values for testing this field are?

1899,1900,2004,2005

### 112. Which of the following tools would be involved in the automation of regression test? a. Data tester b. Boundary tester c. Capture/Playback d. Output comparator.

d. Output comparator

### 113. To test a function, what has to write a programmer, which calls the function to be tested and pass test data.

Driver

### 114. What is the one Key reason why developers have difficulty testing their own work?

Lack of Objectivity

### 115. “How much testing is enough?”

The answer depends on the risk for your industry, contract and special requirements.

### 116. When should testing be stopped?

It depends on the risks for the system being tested. There are some criteria based on which you can stop testing.

1. Deadlines (Testing, Release)
2. Test budget has been depleted
3. Bug rate fall below a certain level
4. Test cases completed with certain percentage passed
5. Alpha or beta periods for testing ends
6. Coverage of code, functionality or requirements are met to a specified point

### 117. Which of the following is the primary purpose of the integration strategy for integration testing in the small?

The primary purpose of the integration strategy is to specify which modules to combine when and how many at once.

### 118. What are semi-random test cases?

Semi-random test cases are nothing, but when we perform random test cases and do equivalence partitioning to those test cases, it removes redundant test cases, thus giving us semi-random test cases.

### 119. Given the following code, which statement is true about the minimum number of test cases required for full statement and branch coverage?

**Read p**

**Read q**

**IF p+q> 100**

**THEN Print “Large”**

**ENDIF**

**IF p > 50**

**THEN Print “p Large”**

**ENDIF**

1 test for statement coverage, 2 for branch coverage

### 120. Which review is normally used to evaluate a product to determine its suitability for the intended use and to identify discrepancies?

Technical Review.

### 121. Faults found should be originally documented by whom?

By testers.

### 122. Which is the current formal world-wide recognized documentation standard?

There isn’t one.

### 123. Which of the following is the review participant who has created the item to be reviewed?

Author

### 124. A number of critical bugs are fixed in software. All the bugs are in one module, related to reports. The test manager decides to do regression testing only on the reports module.

Regression testing should be done on other modules as well because fixing one module may affect other modules.

### 125. Why does the boundary value analysis provide good test cases?

Because errors are frequently made during programming of the different cases near the ‘edges’ of the range of values.

### 126. What makes an inspection different from other review types?

It is led by a trained leader, uses formal entry and exit criteria and checklists.

### 127. Why can be tester dependent on configuration management?

Because configuration management assures that we know the exact version of the testware and the test object.

### 128. What is V-Model?

A software development model that illustrates how testing activities integrate with software development phases

### 129. What is maintenance testing?

Triggered by modifications, migration or retirement of existing software

### 130. What is test coverage?

Test coverage measures in some specific way the amount of testing performed by a set of tests (derived in some other way, e.g., using specification-based techniques). Wherever we can count things and can tell whether or not each of those things has been tested by some test, then we can measure coverage.

### 131. Why is incremental integration preferred over “big bang” integration?

Because incremental integration has better early defects screening and isolation ability

### 132. What is called the process starting with the terminal modules?

Bottom-up integration

### 133. During which test activity could fault be found most cost-effectively?

During test planning

### 134. The purpose of the requirement phase is

To freeze requirements, to understand user needs, to define the scope of testing

### 135. Why we split testing into distinct stages?

We split testing into distinct stages because of the following reasons,

1. Each test stage has a different purpose
2. It is easier to manage to test in stages
3. We can run different test into different environments
4. Performance and quality of the testing is improved using phased testing

### 136. What is DRE?

In order to measure test effectiveness, a powerful metric is used to measure test effectiveness known as DRE (Defect Removal Efficiency) From this metric we would know how many bugs we have found from the set of test cases. The formula for calculating DRE is

DRE=Number of bugs while a testing/number of bugs while testing + number of bugs found by a user

### 137. Which of the following is likely to benefit most from the use of test tools providing test capture and replay facilities? a) Regression testing b) Integration testing c) System testing d) User acceptance testing

Regression testing

### 138. How would you estimate the amount of re-testing likely to be required?

Metrics from previous similar projects and discussions with the development team

### 139. What studies data flow analysis?

The use of data on paths through the code.

### 140. What is failure?

Failure is a departure from specified behavior.

### 141. What are Test comparators?

Is it really a test if you put some inputs into some software, but never look to see whether the software produces the correct result? The essence of testing is to check whether the software produces the correct result and to do that, and we must compare what the software produces to what it should produce. A test comparator helps to automate aspects of that comparison.

### 142. Who is responsible for document all the issues, problems and open point that were identified during the review meeting

Scribe

### 143. What is the main purpose of Informal review

An inexpensive way to get some benefit

### 144. What is the purpose of test design technique?

Identifying test conditions and Identifying test cases

### 145. When testing a grade calculation system, a tester determines that all scores from 90 to 100 will yield a grade of A, but scores below 90 will not. This analysis is known as:

Equivalence partitioning

### 146. A test manager wants to use the resources available for the automated testing of a web application. The best choice is

Tester, test automater, web specialist, DBA

### 147. During the testing of a module tester, ‘X’ found a bug and assigned it to a developer. But developer rejects the same, saying that it’s not a bug. What ‘X’ should do?

Send the detailed information of the bug encountered and check the reproducibility

### 148. A type of integration testing in which software elements, hardware elements, or both are combined all at once into a component or an overall system, rather than in stages.

Big-Bang Testing

### 149. In practice, which Life Cycle model may have more, fewer or different levels of development and testing, depending on the project and the software product. For example, there may be component integration testing after component testing, and system integration testing after system testing.

V-Model

### 150. Which technique can be used to achieve input and output coverage? It can be applied to human input, input via interfaces to a system, or interface parameters in integration testing.

Equivalence partitioning

### 151. “This life cycle model is driven by schedule and budget risks” This statement is best suited for.

V-Model

### 152. In which order should tests be run?

The most important one must be tested first

### 153. The later in the development life cycle a fault is discovered, the more expensive it is to fix. Why?

The fault has been built into more documentation, code, tests, etc

### 154. What is Coverage measurement?

It is a partial measure of test thoroughness.

### 155. What is Boundary value testing?

Test boundary conditions on, below and above the edges of input and output equivalence classes. For instance, let say a bank application where you can withdraw maximum Rs.20,000 and a minimum of Rs.100, so in boundary value testing we test only the exact boundaries, rather than hitting in the middle. That means we test above the maximum limit and below the minimum limit.

### 156. What does COTS represent?

Commercial Off The Shelf.

### 157. The purpose of which is to allow specific tests to be carried out on a system or network that resembles as closely as possible the environment where the item under test will be used upon release?

Test Environment

### 158. What can be thought of as being based on the project plan, but with greater amounts of detail?

Phase Test Plan

### 159. What is Rapid Application Development?

Rapid Application Development (RAD) is formally a parallel development of functions and subsequent integration. Components/functions are developed in parallel as if they were mini projects, the developments are time-boxed, delivered, and then assembled into a working prototype. This can very quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements. Rapid change and development of the product are possible using this methodology. However the product specification will need to be developed for the product at some point, and the project will need to be placed under more formal controls before going into production.

### **1. What do you understand by software testing?**

Software testing is a validation process which confirms that a system works as per the business requirements. It qualifies a system on various aspects such as usability, accuracy, completeness, efficiency, etc. ANSI/IEEE 1059 is the global standard that defines the basic principles of testing.

### **2. When should you stop the testing process?**

The testing activity ends when the testing team completes the following milestones.

**Test case execution**

The successful completion of a full test cycle after the final bug fix marks the end of the testing phase.

**Testing deadline**

The end date of the validation stage also declares the closure of the validation if no critical or high-priority defects remain in the system.

**Code Coverage(CC) ratio**

It is the amount of code concealed via automated tests. If the team achieves the intended level of code coverage (CC) ratio, then it can choose to end the validation.

**Mean Time Between Failure (MTBF) rate**  
Mean time between failure (MTBF) refers to the average amount of time that a device or product functions before failing. This unit of measurement includes only operational time between failures and does not include repair times, assuming the item is repaired and begins functioning again. MTBF figures are often used to project how likely a single unit is to fail within a certain period of time

### **3. What do verification and validation mean in software testing?**

In software testing, verification is a process to confirm that product development is taking place as per the specifications and using the standard development procedures. The process comprises the following activities:

* Inspections
* Reviews
* Walk-throughs
* Demos

### Validation is a means to confirm that the developed product doesn’t have any bugs and is working as expected. It comprises the following activities:

* Functional testing
* Non-functional testing

### **4. What is static testing? When does it start and what does it cover?**

Static testing is a white-box testing technique that directs developers to verify their code with the help of a checklist to find errors in it. Developers can start the static testing without actually finalizing the application or program. Static testing is more cost-effective than dynamic testing as it more areas than dynamic testing in a shorter time.

### **5. Define Black-box testing.**

It is a standard software testing approach that requires testers to assess the functionality of the software as per the business requirements. The software is treated as a black box and validated as per the end user’s point of view.

### **6. What is a test plan and what does it include?**

A test plan stores all possible testing activities to ensure a quality product. It gathers data from the product description, requirement, and use case documents.

The test plan document includes the following:

* Testing objectives
* Test scope
* Testing the frame
* Environment
* Reason for testing
* Criteria for entrance and exit
* Deliverables
* Risk factorsTop of Form





### **8. Is it possible to achieve 100% testing coverage? How would you ensure it?**

It’s considered not possible to perform 100% testing of any product. But you can follow the below steps to come closer.

* Set a hard limit on the following factors:
  + Percentage of test cases passed
  + Number of bugs found
* Set a red flag if:
  + Test budget is depleted
  + Deadlines are breached
* Set a green flag if:
  + The entire functionality gets covered in test cases
  + All critical and major bugs must have a ‘CLOSED’ status

### **9. What are unit testing and integration testing?**

Unit testing has many names such as module testing or component testing.

Many times, it is the developers who test individual units or modules to check if they are working correctly.

Whereas, integration testing validates how well two or more units of software interact with each other.

**There are three ways to validate integration:**

* Big Bang approach
* Top-down approach
* Bottom-up approach

### **10. Can we do system testing at any stage?**

No. System testing should start only if all modules are in place and they work correctly. However, it should be performed before UAT (user acceptance testing).

### **11. Mention the different types of software testing.**

**Various testing types used by manual testers are as follows:**

* Unit testing
* Integration testing
* Regression testing
* Shakeout testing
* Smoke testing
* Functional testing
* Performance testing
  + Load testing
  + Stress testing
  + Endurance testing
* White-box and Black-box testing
* Alpha and Beta testing
* System testing

### **12. What is the difference between a test driver and a test stub?**

The **test driver** is a section of code that calls a software component under test. It is useful in testing that follows the bottom-up approach.

The **test stub** is a dummy program that integrates with an application to complete its functionality. It is relevant for testing that uses the top-down approach.

For example:

1. Let’s assume a scenario where we have to test the interface between Modules A and B. We have developed only Module A. Here, we can test Module A if we have the real Module B or a dummy module for it. In this case, we call Module B as the test stub.
2. Now, Module B can’t send or receive data directly from Module A. In such a scenario, we’ve to move data from one module to another using some external features called test driver.

### **13. What is agile testing and why is it important?**

**Agile testing** is a software testing process that evaluates software from the customers’ point of view. It is favorable as it does not require the development team to complete coding for starting QA. Instead, both coding and testing go hand in hand. However, it may require continuous customer interaction.

### **14. What do you know about data flow testing?**

It is one of the white-box testing techniques.

Data flow testing emphasizes for designing test cases that cover control flow paths around variable definitions and their uses in the modules. It expects test cases to have the following attributes:

1. The input to the module
2. The control flow path for testing
3. A pair of an appropriate variable definition and its use
4. The expected outcome of the test case

### **15. What is the purpose of the end-to-end testing?**

End-to-end testing is a testing strategy to execute tests that cover every possible flow of an application from its start to finish. The objective of performing end-to-end tests is to discover software dependencies and to assert that the correct input is getting passed between various software modules and sub-systems.

**16. The probability that a server-class application hosted on the cloud is up and running for six long months without crashing is 99.99 percentage. To analyze this type of a scenario, what test you will perform?**

Reliability testing

### **17. What will you do when a bug turns up during testing?**

When a bug occurs, we can follow the below steps.

* We can run more tests to make sure that the problem has a clear description.
* We can also run a few more tests to ensure that the same problem doesn’t exist with different inputs.
* Once we are certain of the full scope of the bug, we can add details and report it.

### **18. Why is it impossible to test a program thoroughly?**

Here are the two principal reasons that make it impossible to test a program entirely.

* Software specifications can be subjective and can lead to different interpretations.
* A software program may require too many inputs, outputs, and path combinations.

### **19. How do you test a product if the requirements are yet to be freezed?**

If the required specifications are not available for a product, then a test plan can be created based on the assumptions made about the product. But we should get all assumptions well-documented in the test plan.

### **20. If a product is in the production stage and one of its modules gets updated, then is it necessary to ret**

It is suggested to perform a regression testing and run tests for all the other modules as well. Finally, the QA should also carry out a system testing.

### **21. How will you overcome the challenges faced due to the unavailability of proper documentation for testing?**

If the standard documents like System Requirement Specification or Feature Description Document are not available, then QAs may have to rely on the following references, if available.

* Screenshots
* A previous version of the application
* Wireframes

Another reliable way is to have discussions with the developer and the business analyst. It helps in solving the doubts, and it opens a channel for bringing clarity on the requirements. Also, the emails exchanged could be useful as a testing reference.

Smoke testing is yet another option that would help verify the main functionality of the application. It would reveal some very basic bugs in the application. If none of these work, then we can just test the application from our previous experiences.

### **22. Is there any difference between retesting and regression testing?**

Possible differences between retesting and regression testing are as follows:

* We perform **retesting** to verify the defect fixes. But, the regression testing assures that the bug fix does not break other parts of the application.
* **Regression**test cases verify the functionality of some or all modules.
* **Regression** testing ensures the re-execution of passed test cases. Whereas, **retesting** involves the execution of test cases that are in a failed state.
* **Retesting** has a higher priority over **regression**. But in some cases, both get executed in parallel.

### **23. As per your understanding, list down the key challenges of software testing.**

Following are some of the key challenges of software testing:

* The lack of availability of standard documents to understand the application
* Lack of skilled testers
* Understanding the requirements: Testers require good listening and understanding capabilities to be able to communicate with the customers the application requirements.
* The decision-making ability to analyze when to stop testing
* Ability to work under time constraints
* Ability to decide which tests to execute first
* Testing the entire application using an optimized number of test cases

### **24. What are the different types of functional testing?**

Functional testing covers the following types of validation techniques:

* Unit testing
* Smoke testing
* UAT
* Sanity testing
* Interface testing
* Integration testing
* System testing
* Regression testing

### **25. What are functional test cases and non-functional test cases?**

* Functional testing: It is testing the ‘functionality’ of a software or an application under test. It tests the behavior of the software under test. Based on the requirement of the client, a document called a software specification or requirement specification is used as a guide to test the application.
* Non-functional testing: In software terms, when an application works as per the user’s expectation, smoothly and efficiently under any condition, then it is stated as a reliable application. Based on quality, it is very critical to test these parameters. This type of testing is called non-functional testing.

### **26. What do you understand by STLC?**

Software testing life cycle (STLC) proposes the test execution in a planned and systematic manner. In the STLC model, many activities occur to improve the quality of the product.

**The STLC model lays down the following steps:**

1. Requirement Analysis
2. Test Planning
3. Test Case Development
4. Environment Setup
5. Test Execution
6. Test Cycle Closure

### **27. In software testing, what does a fault mean?**

Fault is a condition that makes the software fail to execute while performing the considered function.

### **28. Difference between Bug, Defect, and Error.**

A slip in coding is indicated as an error. The error spotted by a manual tester becomes a defect. The defect which the development team admits is known as a bug. If a built code misses on the requirements, then it is a functional failure.

### **29. How do severity and priority relate to each other?**

**Severity:** It represents the gravity/depth of a bug. It describes the application point of view.

**Priority:** It specifies which bug should get fixed first. It defines the user’s point of view.

### **30. List the different types of severity.**

The criticality of a bug can be low, medium, or high depending on the context.

* User interface defects – Low
* Boundary related defects – Medium
* Error handling defects – Medium
* Calculation defects – High
* Misinterpreted data – High
* Hardware failures – High
* Compatibility issues – High
* Control flow defects – High
* Load conditions – High

## Advanced Interview Questions

### **31. What do you mean by defect detection percentage in software testing?**

Defect detection percentage (DDP) is a type of testing metric. It indicates the effectiveness of a testing process by measuring the ratio of defects discovered before the release and reported after the release by customers.

For example, let’s say, the QA has detected 70 defects during the testing cycle and the customer reported 20 more after the release. Then, DDP would be: 70/(70 + 20) = 72.1%

### **32. What does defect removal efficiency mean in software testing?**

Defect removal efficiency (DRE) is one of the testing metrics. It is an indicator of the efficiency of the development team to fix issues before the release.

It gets measured as the ratio of defects fixed to total the number of issues discovered.

For example, let’s say, there were 75 defects discovered during the test cycle while 62 of them got fixed by the development team at the time of measurement. The DRE would be 62/75 = 82.6%

Go through the [**Manual Testing Training**](https://intellipaat.com/software-testing-training-course/) to get clear understanding of Weak AI and Strong AI.

### **33. What is the average age of a defect in software testing?**

Defect age is the time elapsed between the day the tester discovered a defect and the day the developer got it fixed.

While estimating the age of a defect, consider the following points:

* The day of birth of a defect is the day it got assigned and accepted by the development team.
* The issues which got dropped are out of the scope.
* Age can be both in hours or days.
* The end time is the day the defect got verified and closed, not just the day it got fixed by the development team.

### **34. How do you perform automated testing in your environment?**

Automation testing is a process of executing tests automatically. It reduces the human intervention to a great extent. We use different test automation tools like QTP, Selenium, and WinRunner. Testing tools help in speeding up the testing tasks. These tools allow you to create test scripts to verify the application automatically and also to generate the test reports.

### **35. Is there any difference between quality assurance, quality control, and software testing. If so, what is it?**

Quality Assurance (QA) refers to the planned and systematic way of monitoring the quality of the process which is followed to produce a quality product. QA tracks the test reports and modifies the process to meet the expectation.

Quality Control (QC) is relevant to the quality of the product. QC not only finds the defects but suggests improvements too. Thus, a process that is set by QA is implemented by QC. QC is the responsibility of the testing team.

Software testing is the process of ensuring that the product which is developed by developers meets the users’ requirements. The aim of performing testing is to find bugs and make sure that they get fixed. Thus, it helps to maintain the quality of the product to be delivered to the customer.

### **36. Tell me about some of the essential qualities an experienced QA or Test Lead must possess.**

**A QA or Test Lead should have the following qualities:**

1. Well-versed in software testing processes
2. Ability to accelerate teamwork to increase productivity
3. Improve coordination between QA and Dev engineers
4. Provide ideas to refine QA processes
5. Skill to conduct RCA meetings and draw conclusions
6. Excellent written and interpersonal communication skills
7. Ability to learn fast and to groom the team members

### **37. What is a Silk Test and why should you use it?**

**Here are some facts about the Silk Test tool:**

1. Skill tool is developed for performing regression and functionality testing of an application.
2. It is used when we are testing Window-based, Java, web, and the traditional client/server applications.
3. Silk Test helps in preparing the test plan and managing it to provide direct accessing of the database and validation of the field.

### **38. On the basis of which factors you would consider choosing automated testing over manual testing?**

**Choosing automated testing over manual testing depends on the following factors:**

1. Tests require periodic execution.
2. Tests include repetitive steps.
3. Tests execute in a standard runtime environment.
4. Automation is expected to take less time.
5. Automation is increasing reusability.
6. Automation reports are available for every execution.
7. Small releases like service packs include a minor bug fix. In such cases, executing the regression test is sufficient for validation.

### **39. Tell me the key elements to consider while writing a bug report.**

**An ideal bug report should consist of the following key points:**

* A unique ID
* Defect description: A short description of the bug
* Steps to reproduce: They include the detailed test steps to emulate the issue. They also provide the test data and the time when the error has occurred
* Environment: Add any system settings that could help in reproducing the issue
* Module/section of the application in which the error has occurred
* Severity
* Screenshots
* Responsible QA: This person is a point of contact in case you want to follow-up regarding this issue

### **41. What is the difference between performance testing and monkey testing?**

Performance testing checks the speed, scalability, and/or stability characteristics of a system. Performance is identified with achieving response time, throughput, and resource-utilization levels that meet the performance objectives for a project or a product.

Monkey testing is a technique in software testing where the user tests the application by providing random inputs, checking the behavior of the application (or trying to crash the application).

### **42. What is exploratory testing?**

Exploratory testing is an approach to software testing, wherein testers learn simultaneously about the test design and test execution. In other words, it is a hands-on approach where testers are involved more in the test execution part than in planning.

### **43. What is meant by system testing?**

System testing is a black-box testing technique, used on a complete integrated system, where it will test the system compliance as per the requirement.

### **44. What are the benefits of test reports?**

Test reports will help us find the current status of a project and its quality. This can help stakeholders and customers take necessary actions. The complete documentation of test reports will help analyze different phases of the project.

### **45. What is meant by latent defect?**

A latent defect is a hidden defect in an application/software, which cannot be identified by a user. However, this will not cause any failure to the application because the conditions will never be met.