Software Testing:

- Software testing is an important process in the software development lifecycle .
- It involves verifying and validating that a software application is free of bugs, meets the technical requirements set by its design and development , and satisfies user requirements efficiently and effectively.
- This process ensures that the application can handle all exceptional and boundary cases, providing a robust and reliable user experience.
- The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability.
- The process checks whether the actual software matches the expected requirements and ensures the software is bug-free.
- It mainly aims at measuring the specification, functionality, and performance of a software program or application.
- The process checks whether the actual software matches the expected requirements and ensures the software is bug-free.
- The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements.
- It mainly aims at measuring the specification, functionality, and performance of a software program or application.

Software testing can be divided into two steps

1. Verification: It refers to the set of tasks that ensure that the software correctly implements a specific function. It means “Are we building the product right?”.
2. Validation: It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. It means “Are we building the right product?”.

Objectives of Testing:

Delivering quality products is the ultimate objective of testing. Let's have a look over the various objectives of testing:

- Identification of Bugs and Errors: The first and foremost goal of software testing is to improve software quality by identifying bugs in the application.Tests that require human intervention (or cannot be automated) are conducted using manual testing.test automation frameworks (or tools) are leveraged to run tests in a CI/CD pipeline
- Delivering Quality Product: As mentioned in the earlier point, the primary intent of running software tests is to locate potential bugs in the application.These bugs are then prioritized on a severity basis, post which they are assigned to the respective developers for fixation.
- Enhanced Growth: Security tests like penetration tests, compliance tests, network security tests, etc. can be leveraged to unearth security vulnerabilities in the application. In many tests, internal testers don the hats of the hacker to identify security loopholes.
- Enhancement of scalability and reliability: The main reason for such an experience could be unprecedented load on the website, causing hiccups to the end-users of the website (or application). This is where load testing can be valuable, as it helps in validating the application's behavior when it is subjected to different kinds of load.
- Meeting Compliance Standards: Whether you are building a custom mobile application or a SaaS project or any other software product; it is essential to follow laws & principles related to software privacy, and more.This is where software testing can play a key role in ensuring that the product/application adheres to the necessary regulatory requirements.

Why do you need to perform testing?

- Helps in saving money: The testing of software has a wide array of benefits. The cost-effectiveness of the project happens to be one of the top reasons why companies go for Software Testing Services.

The testing of software comprises of a bunch of projects. In case you find any bug in the early phases, fixing them costs a reduced amount of money.

- Security: It is another crucial point why software testing should not be taken into consideration.It is considered to be the most vulnerable and sensitive part. There are a bunch of situations in which the information and details of the users are stolen and they are used for the benefits.It is considered to be the reason why people look for well tested and reliable products.
- Quality of the Product: Products should be serving the user in one way or the other. It is a must that it is going to bring value, as per the promise.Hence, it should function in a complete manner for ensuring an effective customer experience. It is also necessary to check the compatibility of the device.
- Satisfaction of the Customer: The primary objective of the owner of the products is offering the best satisfaction of the customers.The reasons why it is necessary to opt for software testing is due to the fact that it offers the prerequisite and perfect user experience.As you opt for the best project in the saturated project, you will be capable of earning the reputation of reliable clients.
- Enhancing the Development process: Quality Assurance, you can find a wide array of scenarios and errors, for the reproduction of the error.t is really simple and the developers need to fix the same in no time. In addition to this, software testers should be working with the development team parallelly, which is useful in the acceleration of the development procedure.
- Easy while adding new features: Tests counteract this calcification tendency by allowing developers to confidently add new features. As a new developer, changing older parts of your codebase can be terrifying, but with tests, you'll at least know if you've broken anything important. This helps in making your software stand ahead in the market, and beat the competition.
- Determining the Performance of the Software: Users are not going to trust any people. There are chances that the reputation of your organization is going to suffer.Thus, software testing is considered to

be an easy option as it helps in the determination of the performance of the software.

Product VS Project:

Project: If the application is developed for the specific customer based on the requirement known as Project.

Product: If the application is developed for multiple customers based on the requirement known as Product.

Error: Any incorrect human action that produces a problem system is called an error.

Defect/Bug: Deviation from the expected behaviour from actual behaviour of the system is called Defect/Bug.

Failure: The deviation identified by the end-user known as Failure.

Why Does Software Have Bugs?

- MisCommunication: To ensure that potential errors are identified and resolved it is crucial to have communication and strong relationships among team members. Misconceptions, lack of conversation, or ineffective collaboration can lead to inconsistencies.
- Deadlines and Pressure:When developers face deadlines there is a likelihood of inaccuracies in their work. This is because they may rush to complete the project within the given timeframe leaving time for evaluation of the program.
- Human Mistake:Software development heavily relies on effort which means mistakes are inevitable. Humans can introduce defects through coding issues, logical errors, or overlooking details.
- Lack of Testing: Testing plays a role in the software development process. Sometimes it can be overlooked or not fully executed due to time constraints or limited resources.

- Imperfect Document: Imperfect or unclear requirements are one of the sources of software bugs. When developers don’t receive comprehensive specifications, they may make assumptions that result in implementations.

Software Development Life Cycle:

- A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle.
- A life cycle model represents all the methods required to make a software product transit through its life cycle stages.
- Different life cycle models may plan the necessary development activities to phases in different ways.
- Thus, no element which life cycle model is followed, the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models.
- Without using an exact life cycle model, the development of a software product would not be in a systematic and disciplined manner.
- When a team is developing a software product, there must be a clear understanding among team representatives about when and what to do.
- A phase can begin only if its stage-entry criteria have been fulfilled. So without a software life cycle model, the entry and exit criteria for a stage cannot be recognized.
- Without software life cycle models, it becomes tough for software project managers to monitor the progress of the project.

The stages of SDLC are as follows:

Stage1: Planning and requirement analysis

Requirement Analysis is the most important and necessary stage in SDLC. The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.

Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.
Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.
Stage2: Defining Requirements

Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders.
This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.
Stage3: Designing the Software

The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.
Stage4: Developing the project

In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.
Stage5: Testing

After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.
During this stage, unit testing, integration testing, system testing, acceptance testing are done.
Stage6: Deployment

Once the software is certified, and no bugs or errors are stated, then it is deployed.
Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment.
After the software is deployed, then its maintenance begins.
Stage7: Maintenance
Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time.
This procedure where the care is taken for the developed product is known as maintenance.

Waterfall Model:

- Winston Royce introduced the Waterfall Model in 1970.
- The Waterfall Model was the first Process Model to be introduced.
- It is also referred to as a linear-sequential life cycle model.
- 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.
- This means that any phase in the development process begins only if the previous phase is complete.
- "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.

Stages of Waterfall model:
- 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.

Application of Waterfall Model:

- 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.

Advantages:

- 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.

Disadvantages:
- 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.

White-Box Testing;

White box testing is a software testing Technique that involves testing the internal structure and workings of a software application.
The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.
White box testing is also known as structural testing or code-based testing, and it is used to test the software’s internal logic, flow, and structure.
The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements.
The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.
Developers do white box testing. In this, the developer will test every line of the code of the program.
The white box testing contains various tests, which are as follows:
- Path testing
- Loop testing

- Condition testing
- Testing based on the memory perspective
- Test performance of the program

Path Testing: In the path testing, we will write the flow graphs and test all independent paths. Here writing the flow graph implies that flow graphs are representing the flow of the program.
Loop testing: In the loop testing, we will test the loops such as while, for, and do-while, etc. and also check for ending conditions if working correctly and if the size of the conditions is enough.
Condition testing: In this, we will test all logical conditions for both true and false values; that is, we will verify for both if and else conditions.
Testing based on the memory (size) perspective
The size of the code is increasing for the following reasons:

- The reuse of code is not there: let us take one example, where we have four programs of the same application, and the first ten lines of the program are similar. We can write these ten lines as a discrete function, and it should be accessible by the above four programs as well. And also, if any bug is there, we can modify the line of code in the function rather than the entire code.
- The developers use the logic that might be modified. If one programmer writes code and the file size is up to 250kb, then another programmer could write a similar code using the different logic, and the file size is up to 100kb.
- The developer declares so many functions and variables that might never be used in any portion of the code. Therefore, the size of the program will increase.

Test the performance (Speed, response time) of the program
The application could be slow for the following reasons:

- When logic is used.
- For the conditional cases, we will use or & and adequately.
- Switch case, which means we cannot use nested if, instead of using a switch case.

Reasons for white box testing

- It identifies internal security holes.
- To check the way of input inside the code.
- Check the functionality of conditional loops.
- To test function, object, and statement at an individual level.

Advantages of White box testing

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

Statement Coverage:
In this technique, the aim is to traverse all statements at least once. Hence, each line of code is tested. In the case of a flowchart, every node must be traversed at least once. Since all lines of code are covered, it helps in pointing out faulty code.

Branch Coverage:
In this technique, test cases are designed so that each branch from all decision points is traversed at least once. In a flowchart, all edges must be traversed at least once.

Condition Coverage:
In this technique, all individual conditions must be covered as shown in the following example:

- READ X, Y
- IF(X == 0 || Y == 0)
- PRINT ‘0’
- #TC1 – X = 0, Y = 55
- #TC2 – X = 5, Y = 0

Multiple Condition Coverage
In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once. Let’s consider the following example:

- READ X, Y
- IF(X == 0 || Y == 0)
- PRINT ‘0’
- #TC1: X = 0, Y = 0
- #TC2: X = 0, Y = 5
- #TC3: X = 55, Y = 0
- #TC4: X = 55, Y = 5

Basis Path Testing
In this technique, control flow graphs are made from code or flowchart and then Cyclomatic complexity is calculated which defines the number of independent paths so that the minimal number of test cases can be designed for each independent path.

Black-Box Testing:

- Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding.
- The primary source of black box testing is a specification of requirements that is stated by the customer.
- In this method, the tester selects a function and gives input value to examine its functionality, and checks whether the function is giving

expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed.
- The test team reports the result to the development team and then tests the next function.
- After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.

Advantages of Black Box Testing:
- The tester does not need to have more functional knowledge or programming skills to implement the Black Box Testing.
- It is efficient for implementing the tests in the larger system.
- Tests are executed from the user's or client's point of view.
- Test cases are easily reproducible.
- It is used to find the ambiguity and contradictions in the functional specifications.

Disadvantages of Black Box Testing:
- There is a possibility of repeating the same tests while implementing the testing process.
- Without clear functional specifications, test cases are difficult to implement.
- It is difficult to execute the test cases because of complex inputs at different stages of testing.
- Sometimes, the reason for the test failure cannot be detected.
- Some programs in the application are not tested.
- It does not reveal the errors in the control structure.
- Working with a large sample space of inputs can be exhaustive and consumes a lot of time.

Unit Testing:

- Unit Testing is a software testing technique in which individual units or components of a software application are tested in isolation.
- These units are the smallest pieces of code, typically functions or methods, ensuring they perform as expected.

- Unit testing helps in identifying bugs early in the development cycle, enhancing code quality, and reducing the cost of fixing issues later.
- It is an essential part of Test-Driven Development (TDD).
- Unit testing promotes modular code, ensures better test coverage, and saves time by allowing developers to focus more on coding than manual testing.
- A unit test is a small piece of code that checks if a specific function or method in is an application works correctly. It will work as the function inputs and verifying the outputs.
- These tests check that the code work as expected based on the logic the developer intended.
- In these multiple tests are written for a single function to cover different possible scenarios and these are called test cases.
- While it is ideal to cover all expected behaviors, it is not always necessary to test every scenario.
- Unit tests should run one by one, it means that they do not depend on other system parts like databases or networks.
- Instead, data stubs can be used to simulate these dependencies. Writing unit tests is easiest for simple, self-contained code blocks.

Unit testing strategies:

To create effective unit tests, follow these basic techniques to ensure all scenarios are covered:

- Logic checks: Verify if the system performs correct calculations and follows the expected path with valid inputs. Check all possible paths through the code are tested.
- Boundary checks: Test how the system handles typical, edge case, and invalid inputs.
- Error handling: Check the system properly handles errors. Does it prompt for a new input, or does it crash when something goes wrong?
- Object-oriented checks: If the code modifies objects, confirm that the object's state is correctly updated after running the code.

Advantages:

- Early Detection of Issues: Unit testing allows developers to detect and fix issues early in the development process before they become larger and more difficult to fix.
- Improved Code Quality: Unit testing helps to ensure that each unit of code works as intended and meets the requirements, improving the overall quality of the software.
- Increased Confidence: Unit testing provides developers with confidence in their code, as they can validate that each unit of the software is functioning as expected.
- Faster Development: Unit testing enables developers to work faster and more efficiently, as they can validate changes to the code without having to wait for the full system to be tested.
- Better Documentation: Unit testing provides clear and concise documentation of the code and its behavior, making it easier for other developers to understand and maintain the software.
- Facilitation of Refactoring: Unit testing enables developers to safely make changes to the code, as they can validate that their changes do not break existing functionality.
- Reduced Time and Cost: Unit testing can reduce the time and cost required for later testing, as it helps to identify and fix issues early in the development process.

Disadvantages of Unit Testing:

- Time and Effort: Unit testing requires a significant investment of time and effort to create and maintain the test cases, especially for complex systems.
- Dependence on Developers: The success of unit testing depends on the developers, who must write clear, concise, and comprehensive test cases to validate the code.
- Difficulty in Testing Complex Units: Unit testing can be challenging when dealing with complex units, as it can be difficult to isolate and test individual units in isolation from the rest of the system.

- Difficulty in Testing Interactions: Unit testing may not be sufficient for testing interactions between units, as it only focuses on individual units.
- Difficulty in Testing User Interfaces: Unit testing may not be suitable for testing user interfaces, as it typically focuses on the functionality of individual units.
- Over-reliance on Automation: Over-reliance on automated unit tests can lead to a false sense of security, as automated tests may not uncover all possible issues or bugs.
- Maintenance Overhead: Unit testing requires ongoing maintenance and updates, as the code and test cases must be kept up-to-date with changes to the software.

Integration Testing:

- Integration testing is the process of testing the interface between two software units or modules.
- It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units.
- Once all the modules have been unit-tested, integration testing is performed.
- The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other. Integration testing is typically performed after unit testing and before system testing.
- It helps to identify and resolve integration issues early in the development cycle, reducing the risk of more severe and costly problems later on.
- Integration testing is one of the basic types of software testing and there are many other basic and advanced software testing.
-

Big-Bang Integration Testing

- It is the simplest integration testing approach, where all the modules are combined and the functionality is verified after the completion of individual module testing.
- In simple words, all the modules of the system are simply put together and tested.
- This approach is practicable only for very small systems. If an error is found during the integration testing, it is very difficult to localize the error as the error may potentially belong to any of the modules being integrated.
- So, debugging errors reported during Big Bang integration testing is very expensive to fix.
- Big-bang integration testing is a software testing approach in which all components or modules of a software application are combined and tested at once.
- This approach is typically used when the software components have a low degree of interdependence or when there are constraints in the development environment that prevent testing individual components.
- The goal of big-bang integration testing is to verify the overall functionality of the system and to identify any integration problems that arise when the components are combined.
- While big-bang integration testing can be useful in some situations, it can also be a high-risk approach, as the complexity of the system and the number of interactions between components can make it difficult to identify and diagnose problems.

Advantages of Big-Bang Integration Testing

- It is convenient for small systems.
- Simple and straightforward approach.
- Can be completed quickly.
- Does not require a lot of planning or coordination.
- May be suitable for small systems or projects with a low degree of interdependence between components.

Disadvantages of Big-Bang Integration Testing

- There will be quite a lot of delay because you would have to wait for all the modules to be integrated.
- High-risk critical modules are not isolated and tested on priority since all modules are tested at once.
- Not Good for long projects.
- High risk of integration problems that are difficult to identify and diagnose.
- This can result in long and complex debugging and troubleshooting efforts.
- This can lead to system downtime and increased development costs.
- May not provide enough visibility into the interactions and data exchange between components.
- This can result in a lack of confidence in the system's stability and reliability.
- This can lead to decreased efficiency and productivity.
- This may result in a lack of confidence in the development team.
- This can lead to system failure and decreased user satisfaction.

Bottom-Up Integration Testing:

In bottom-up testing, each module at lower levels are tested with higher modules until all modules are tested. The primary purpose of this integration testing is that each subsystem tests the interfaces among various modules making up the subsystem. This integration testing uses test drivers to drive and pass appropriate data to the lower-level modules.

Advantages of Bottom-Up Integration Testing:

- In bottom-up testing, no stubs are required.
- A principal advantage of this integration testing is that several disjoint subsystems can be tested simultaneously.
- It is easy to create the test conditions.
- Best for applications that uses bottom up design approach.
- It is Easy to observe the test results.

Disadvantages of Bottom-Up Integration Testing:

- Driver modules must be produced.
- In this testing, the complexity that occurs when the system is made up of a large number of small subsystems.
- As Far modules have been created, there is no working model can be represented.

Top-Down Integration Testing:

Top-down integration testing technique is used in order to simulate the behaviour of the lower-level modules that are not yet integrated. In this integration testing, testing takes place from top to bottom. First, high-level modules are tested and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended.

Advantages of Top-Down Integration Testing:

- Separately debugged module.
- Few or no drivers needed.
- It is more stable and accurate at the aggregate level.
- Easier isolation of interface errors.
- In this, design defects can be found in the early stages.

Disadvantages of Top-Down Integration Testing:

- Needs many Stubs.
- Modules at lower level are tested inadequately.
- It is difficult to observe the test output.
- It is difficult to stub design.

Mixed Integration Testing:

A mixed integration testing is also called sandwiched integration testing. A mixed integration testing follows a combination of top down and bottom-up testing approaches. In top-down approach, testing can start only after the top-level module have been coded and unit tested. In bottom-up approach, testing can start only after the bottom level modules are ready. This sandwich or mixed approach overcomes this shortcoming of the top-down and bottom-up approaches. It is also called the hybrid integration testing. Also, stubs and drivers are used in mixed integration testing.

Advantages of Mixed Integration Testing:

- Mixed approach is useful for very large projects having several sub projects.
- This Sandwich approach overcomes this shortcoming of the top-down and bottom-up approaches.
- Parallel test can be performed in top and bottom layer tests.

Disadvantages of Mixed Integration Testing:

- For mixed integration testing, it requires a very high cost because one part has a Top-down approach while another part has a bottom-up approach.
- This integration testing cannot be used for smaller systems with huge interdependence between different modules.

Applications of Integration Testing:

-
- Identify the components: Identify the individual components of your application that need to be integrated. This could include the frontend, backend, database, and any third-party services.
- Create a test plan: Develop a test plan that outlines the scenarios and test cases that need to be executed to validate the integration points between the different components. This could include testing data flow, communication protocols, and error handling.
- Set up test environment: Set up a test environment that mirrors the production environment as closely as possible. This will help ensure that the results of your integration tests are accurate and reliable.
- Execute the tests: Execute the tests outlined in your test plan, starting with the most critical and complex scenarios. Be sure to log any defects or issues that you encounter during testing.
- Analyze the results: Analyze the results of your integration tests to identify any defects or issues that need to be addressed. This may

involve working with developers to fix bugs or make changes to the application architecture.
- Repeat testing: Once defects have been fixed, repeat the integration testing process to ensure that the changes have been successful and that the application still works as expected.

System Testing:

- System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution.
- It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users.
- This type of testing is performed after the integration testing and before the acceptance testing.
- System Testing is a type of software testing that is performed on a completely integrated system to evaluate the compliance of the system with the corresponding requirements.
- In system testing, integration testing passed components are taken as input.
- The goal of integration testing is to detect any irregularity between the units that are integrated. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested.
- System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or the context of both. System testing tests the design and behavior of the system and also the expectations of the customer.
- It is performed to test the system beyond the bounds mentioned in the SRS. System Testing is performed by a testing team that is independent of the development team and helps to test the quality of the system impartially.

- It has both functional and non-functional testing. System Testing is a black-box testing . System Testing is performed after the integration testing and before the acceptance testing.

System Testing Process:

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

Types of System Testing:
- Performance Testing: Performance Testing is a type of software testing that is carried out to test the speed, scalability, stability and reliability of the software product or application.
- Load Testing: Load Testing is a type of software Testing which is carried out to determine the behavior of a system or software product under extreme load.
- Stress Testing: Stress Testing is a type of software testing performed to check the robustness of the system under the varying loads.
- Scalability Testing: Scalability Testing is a type of software testing which is carried out to check the performance of a software application or system in terms of its capability to scale up or scale down the number of user request load.

Advantages of System Testing:

- The testers do not require more knowledge of programming to carry out this testing.
- It will test the entire product or software so that we will easily detect the errors or defects which cannot be identified during the unit testing and integration testing.
- The testing environment is similar to that of the real time production or business environment.
- It checks the entire functionality of the system with different test scripts and also it covers the technical and business requirements of clients.
- After this testing, the product will almost cover all the possible bugs or errors and hence the development team will confidently go ahead with acceptance testing
- Verifies the overall functionality of the system.
- Detects and identifies system-level problems early in the development cycle.
- Helps to validate the requirements and ensure the system meets the user needs.
- Improves system reliability and quality.
- Facilitates collaboration and communication between development and testing teams.
- Enhances the overall performance of the system.

Disadvantages of System Testing:

- This testing is time consuming process than another testing techniques since it checks the entire product or software.
- The cost for the testing will be high since it covers the testing of entire software.
- It needs good debugging tool otherwise the hidden errors will not be found.
- Can be time-consuming and expensive.
- Requires adequate resources and infrastructure.

- Can be complex and challenging, especially for large and complex systems.
- Dependent on the quality of requirements and design documents.
- Limited visibility into the internal workings of the system.
- Can be impacted by external factors like hardware and network configurations.

User Acceptance Testing:

- User Acceptance Testing (UAT) serves the purpose of ensuring that the software meets the business requirements and is ready for deployment by validating its functionality in a real-world environment.
- It allows end-users to test the software to ensure it meets their needs and operates as expected, helping to identify and fix any issues before the final release.
- UAT is crucial for quality assurance and customer satisfaction, as it ensures that the software is user-friendly, reliable, and meets all specified criteria.
- User Acceptance Testing (UAT) is a crucial phase in software testing where the software is tested in a real-world scenario by end-users to ensure it meets their requirements and functions as expected.

Alpha Testing:

- Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or the public.
- Alpha testing is commonly performed by homestead software engineers or quality assurance staff. It is the last testing stage before the software is released into the real world.
- Alpha testing is a software testing stage that takes place early in the development process, typically after the code has been written and before the final product is released to the public. Alpha testing is performed by a select group of internal stakeholders, such as developers, testers, and members of the product team.

- The purpose of alpha testing is to identify and resolve critical bugs and issues in the software before it is released to the public. Alpha testing is performed in a controlled environment, such as a lab or a test network, and is used to simulate real-world use cases and identify any potential problems.
- During alpha testing, the software is evaluated against a set of predetermined acceptance criteria and is tested for functionality, usability, performance, and stability. Alpha testing provides an opportunity to identify and fix bugs and issues before they reach end-users, ensuring that the final product is of high quality and meets the needs of the target audience.

Advantages of Alpha Testing:

- Early identification of bugs and issues: Alpha testing allows for the early identification of bugs and issues, providing an opportunity to fix them before they reach end-users.
- Improved quality: By identifying and fixing bugs and issues early in the development process, alpha testing helps to improve the overall quality of the software.
- Increased user satisfaction: Alpha testing helps to ensure that the software meets the needs of the target audience, leading to increased user satisfaction.
- Faster resolution of problems: Alpha testing allows for the rapid resolution of problems, reducing the likelihood of further issues down the line.
- Cost savings: By identifying and fixing issues early in the development process, alpha testing can help to save time and money by avoiding the need for more extensive testing and bug fixing later on.

Objective of Alpha Testing

- The objective of alpha testing is to refine the software product by finding the bugs that were not discovered during the previous tests.
- The objective of alpha testing is to refine the software product by fixing the bugs that were not discovered during the previous tests.

- The objective of alpha testing is to involve customers deep into the process of development.
- The objective of alpha testing is to give better insight into the software’s reliability at the early stages of development.
- The main objective of alpha testing is to identify and resolve critical bugs and issues in the software before it is released to the public. The goal is to assess the software’s overall quality, functionality, usability, performance, and stability in a controlled environment, and to ensure that it meets the needs and expectations of the target audience.
- During alpha testing, the software is evaluated against a set of predetermined acceptance criteria, and any issues or bugs that are identified are documented and reported back to the development team for resolution. The objective of alpha testing is to provide an early opportunity to identify and fix bugs and issues, reducing the likelihood of them affecting end-users and potentially causing damage to the software’s reputation.
- Overall, the objective of alpha testing is to improve the quality of the software, ensure that it meets the needs of the target audience, and reduce the risk of issues and bugs affecting end-users after the software has been released.

Process of Alpha Testing

- Review the design specification and functional requirements.
- Develop comprehensive test cases and test plans.
- Execute test plan
- Log defects
- Retest once the issues have been fixed.

Beta Testing:

- Beta Testing is performed by real users of the software application in a real environment.
- Beta testing is one of the types of User Acceptance Testing. A Beta version of the software, whose feedback is needed, is released to a

limited number of end-users of the product to obtain feedback on the product quality.
- Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation.
- It is the last test before shipping a product to the customers.
- One of the major advantages of beta testing is direct feedback from customers.

Characteristics of Beta Testing:
- Beta Testing is performed by clients or users who are not employees of the company.
- Reliability, security, and robustness are checked during beta testing.
- Beta Testing commonly uses black-box testing.
- Beta testing is carried out in the user’s location.
- Beta testing doesn’t require a lab or testing environment.

Criteria for Beta Testing:
- Sign off a document on Alpha testing.
- The Beta version of the software should be ready
- Environment ready to release the software application to the public
- Tool to capture real-time faults

Advantages of Beta Testing
- It reduces product failure risk via customer validation.
- Beta Testing allows a company to test post-launch infrastructure.
- It helps in improving product quality via customer feedback.
- Cost-effective compared to similar data gathering methods.
- It creates goodwill with customers and increases customer satisfaction.

Disadvantages of Beta Testing
- Sometimes, it is complex to follow the errors or bugs because the testing environment varies from user to user.
- There is a chance of having duplication of errors or bugs.

- The development team and the testing team are not having control over this real-time test environment.
- This testing is a time-consuming process since it involves real time users or clients and hence delay in the overall feedback about the entire product.
- The users who are testing the product should have enough knowledge about the working of the entire application or product. Otherwise, the testing will not be implemented efficiently.

Gamma Testing:

- Gamma testing, also known as “Release candidate testing,” is conducted after Beta Testing.
- It is concentrated on security and functionality and is conducted without in-house QA activities.
- Only a limited number of users participate.
- Gamma testing focuses on ensuring the product is ready for a broader audience.

Types:

- Feature Gamma Testing

Specific features are tested individually.

- Full Gamma Testing

Comprehensive testing of the entire product.