Unit 1: Testing for Applications

1.1 Testing life cycle, Test Exit criteria
1.2 Component level Testing
1.3 Navigation Testing
1.4 Configuration Testing
    1.4.1 Server-side issues
    1.4.2 Client-side issues
1.5 Security Testing
1.6 Performance Testing
    1.6.1 Performance testing objectives
    1.6.2 Load Testing
    1.6.3 Stress Testing

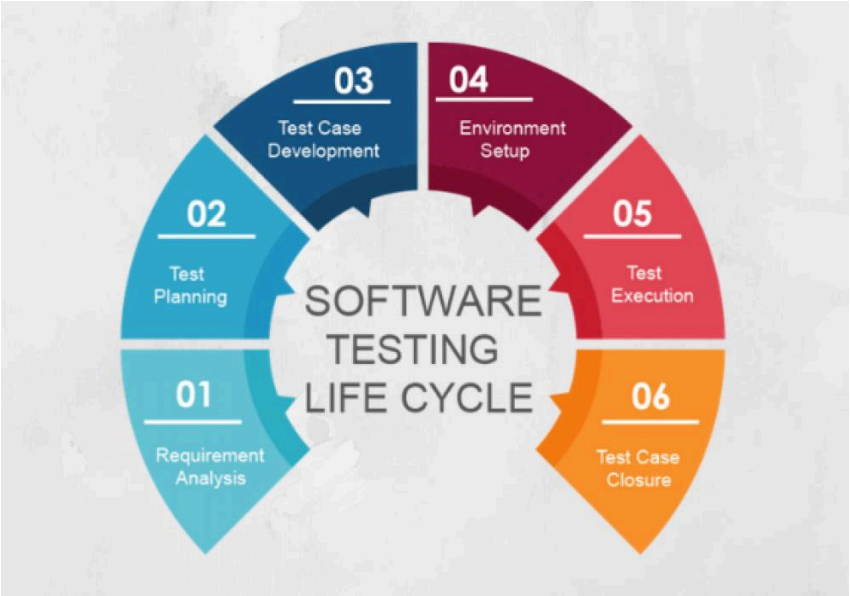
What is Software Testing

- Software testing is a process of identifying the correctness of software by considering its all attributes (Reliability, Scalability, Portability, Re-usability, Usability) and evaluating the execution of software components to find the software bugs or errors or defects.
- Software testing provides an independent view and objective of the software and gives surety of fitness of the software. It involves testing of all components under the required services to confirm whether it is satisfying the specified requirements or not. The process is also providing the client with information about the quality of the software.

- Testing is mandatory because it will be a dangerous situation if the software fails any time due to lack of testing. So, without testing software cannot be deployed to the end user.

Software Testing Life Cycle (STLC)

- The procedure of software testing is also known as STLC (Software Testing Life Cycle) which includes phases of the testing process.The testing process is executed in a well-planned and systematic manner. All activities are done to improve the quality of the software product.
- **Software testing life cycle contains the following steps:**
- **Requirement Analysis**
- **Test Planning**
- **Test case development**
- **Test Environment setup**
- **Test Execution**
- **Test Cycle closure**



- **Requirement Analysis:**
- The first step of the manual testing procedure is requirement analysis. In this phase, **the tester analyses the requirement document of SDLC (Software Development Life Cycle) to examine requirements stated by the client. After examining the requirements, the tester makes a test plan to check whether the software is meeting the requirements or not.**

Entry Criteria	Activities	Exit Criteria

For the **planning of test plan requirement specification**, **application architecture document** and well-defined acceptance criteria should be available.	Prepare the list of all requirements and queries, and get resolved from Technical Manager/Lead, System Architecture, Business Analyst and Client. **Make a list of all types of tests** (Performance, Functional and security) to be performed. **Make a list of test environment details, which should contain all the necessary tools to execute test cases.**	List of all the necessary tests for the **testable requirements andTest environment details**

- **Test Planning**
- Test plan creation is the crucial phase of STLC where **all the testing strategies are defined. Tester determines the estimated effort and cost of the entire project.** This phase takes place after the successful completion of the Requirement Analysis Phase. Testing strategy and effort estimation documents provided by this phase. **Test case execution can be started after the successful completion of Test Plan Creation.**

Entry Criteria	Activities	Exit Criteria
Requirement Document	Define Objective as well as the scope of the software. **List down methods involved in testing**. Overview of the testing process. Settlement of testing environment. Preparation of **the test schedules and control procedures**. **Determination of roles and responsibilities**. **List down testing deliverables, define risk if any**.	**Test strategy document.** **Testing Effort estimation documents are the deliverables of this phase.**

- **Test case development**
- Test case Execution takes place after the successful completion of test planning. **In this phase, the testing team starts case development and execution activity. The testing team writes down the detailed test cases, and also prepares the test data if required. The prepared test cases are reviewed by peer members of the team or Quality Assurance leader.**

- **RTM (Requirement Traceability Matrix) is also prepared in this phase. Requirement Traceability Matrix is an industry level format, used for tracking requirements.** Each test case is mapped with the requirement specification. Backward & forward traceability can be done via RTM.

Entry Criteria	Activities	Exit Criteria
Requirement Document	**Creation of test cases.** **Execution of test cases.** Mapping of test cases according to requirements.	**Test execution result.** **List of functions with the detailed explanation of defects.**

- **Test Environment setup**
- **Setup of the test environment is an independent activity and can be started along with Test Case Development.** This is an essential part of the manual testing procedure as without environment testing is not possible. **Environment setup requires a group of essential software and hardware to create a test environment.** The testing team is not involved in setting up the testing environment, its senior developers who create it.

Entry Criteria	Activities	Exit Criteria
Test strategy and test plan document. Test case document. Testing data.	Prepare the list of software and hardware by analyzing requirement specifications. After the setup of the test environment, execute the smoke test cases to check the readiness of the test environment.	Execution report. Defect report.

- **Test Execution**
- Testers and developers evaluate the completion criteria of the software based on test coverage, quality, time consumption, cost, and critical business objectives. This phase determines the characteristics and drawbacks of the software. Test cases and bug reports are analyzed in depth to detect the type of defect and its severity.
- Defect logging analysis mainly works to find out defect distribution depending upon severity and types.If any defect is detected, then the software is returned to the development team to fix the defect, then the software is re-tested on all aspects of the testing.
- Once the test cycle is fully completed then test closure report, and test metrics are prepared.

Entry Criteria	Activities	Exit Criteria
Test case execution report. Defect report	It evaluates the completion criteria of the software based on test coverage, quality, time consumption, cost, and critical business objectives. Defect logging analysis finds out defect distribution by categorizing in types and severity.	Closure report Test metrics

- **Test Cycle closure**
- The test cycle closure report includes all the documentation related to software design, development, testing results, and defect reports.
- This phase evaluates the strategy of development, testing procedure, and possible defects in order to use these practices in the future if there is software with the same specification.

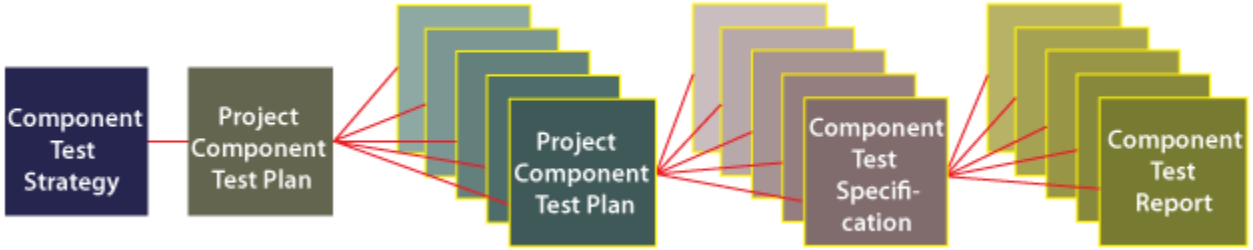
Entry Criteria	Activities	Exit Criteria

All documents and reports related to software.	Evaluates the strategy of development, testing procedure, possible defects to use these practices in the future if there is a software with the same specification	Test closure report

Component level Testing

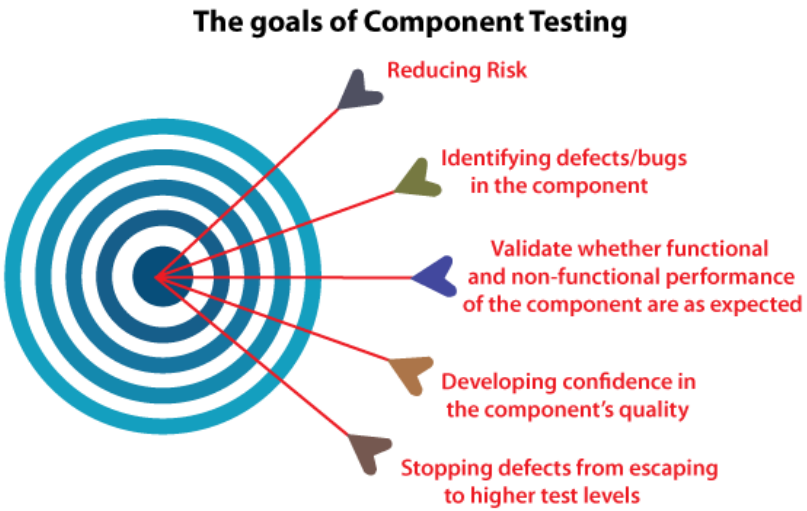
Reference Link: https://www.javatpoint.com/component-testing

- It is used to test **all the components separately** as well as the usability testing; interactive valuation is also done for each specific component. It is further known as **Module Testing or Program Testing and Unit Testing.**
- In order to implement the component testing, all the components or modules require to be in the individual state and manageable state. And all the related components of the software should be user-understandable.
- This type of testing provides a way to find defects, which happen in all the modules. And also helps in certifying the working of each component of the software.



- Components testing is one of the most repeated types of **black-box testing executed by the Quality Assurance Team.**
- It can be performed individually, that is, in separation from the remaining system. However, it has relied on the model of the preferred life cycle.
- The **debugging** or **test structure tools** can be used in the component testing.

- In component testing bugs can be fixed as soon as possible when they are identified without keeping any records.
- In simple words, we can say that the execution of component testing makes sure that all the application components are working correctly and according to the requirements. Component testing is executed before handing out the integration testing.
- **The Goals of Component Testing**
- The primary purpose of executing **the component testing is to validate the input/output performance of the test object**. And also make sure the specified test object's functionality is working fine as per needed requirement or specification.'
- **Reducing Risk**
- **Identifying defects/bugs in the component**
- **Validate whether the functional and non-functional performance of the component is as expected**
- **Developing confidence in the component's quality**
- **Stopping defects from escaping to higher test levels**

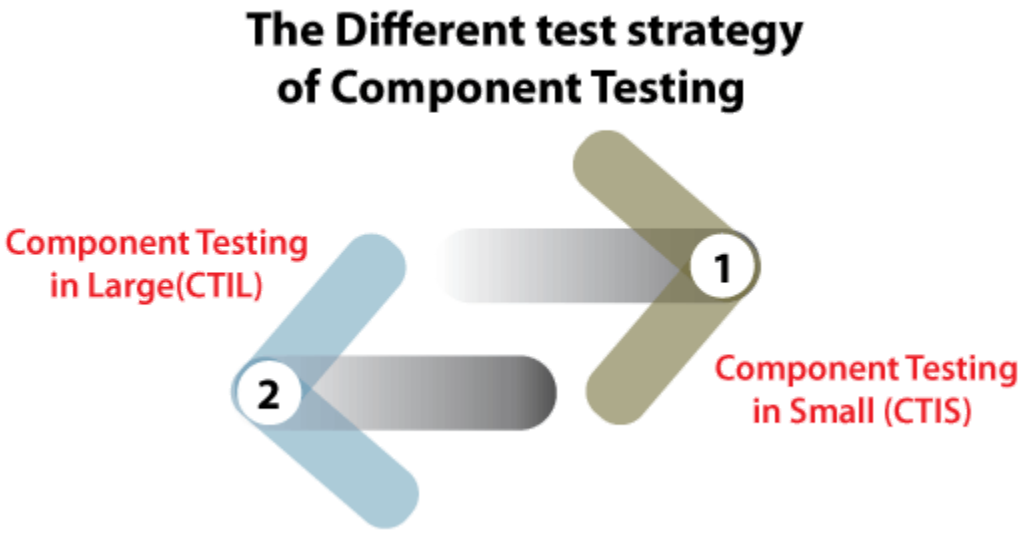


- **Reducing Risk**
- The implementation of components testing validates every single unit of the application. And helps the developers in order to identify the bugs in the code and fix them. Thus, we can say that component testing diminishes the possibilities of risk at a fundamental level.
- **Identifying defects/bugs in the component**
- Another essential purpose of performing the component testing is to identify errors in the source code. Furthermore, it also validates **control flow, functions, data structure,** etc., which are used in the program.
- **Validate whether functional and non-functional performances of the component are as expected**

- The purpose of executing the component testing is to verify that the functional and non-functional features of the component are functioning correctly or not.
- In other words, we can describe that the execution of component testing guarantees their design and specifications are performing as expected.
- It may involve functional features such as the **correctness of calculations** and non-functional features like exploring memory leaks.
- **Developing confidence in the component's quality**
- As we understood from the definition of the component testing, it has occurred on the unit level, and most of the errors are identified and fixed while coding itself.
- The components testing plays a crucial role in building confidence in the component, which means fewer bugs or defects in the additional testing.
- **Stopping defects from escaping to higher test levels**
- Conclusively, in Component testing, the coding errors are identified and fixed by the developers. And as an outcome, component testing diminishes the existence of errors in the higher level of testing.
- **Who performs Component Testing?**
- The component testing is performed by **test engineers or the developers** who write the codes with IDE help. As we already knew, Unit Testing is executed by the developers to perform the testing of the particular features or methods.
- **The Different test strategy of Component Testing**
- The Component Testing is separated into the following two types, based on the testing level's complexity.

1. **Component Testing in Small (CTIS)**

2. Component Testing in Large (CTIL)



- **Component Testing in Small (CTIS)**
- Some component testing may be performed with or without other modules in the particular application or the software under a test. If the component testing is executed in the segregation with other modules is signified as ***component testing in small,*** which is also denoted as **CTIS.**
- Let see one sample example, where we can clearly understand how Component testing in small works.
- **For Example:** Suppose we have one website, which includes five different web pages. Therefore, testing each web page individually and with the isolation of other components is signified as **Component testing in Small.**
- **Component Testing in Large (CTIL)**

- The component testing in large is testing, where we execute the component testing without segregation with having other modules of the software.
- **This type of testing has occurred when there is a dependency on the performance flow of the modules, and consequently, we can't separate them.** Those components on which we have the dependency are not created yet, and then we use the replicate modules instead of the actual modules or components.
- **For Example**
- Let us assume that we have a web application containing three different modules: Module P, Module Q, and Module R.
- Here, the developer has created Module Q, which needs to be tested.
- However, to test Module Q entirely, some of its features are reliant on Module P, and some of its features are reliant on Module R.
- So that, performance flow of the particular modules are as follows:
- Module P → Module Q → Module R
- This performance flow implies that there is a dependency to Module Q from both Module P and Module R. As we can see in the below image, we have Stubs and Driver where stubs are called as the called function, and the drivers are called as the calling function.
- But the Module P and Module R have not been established until now. So, in this scenario, if we want to test Module Q entirely.
- Thus, we can interchange Module P and Module R with the help of Stub and Drivers as per the requirement.
- Hence, Module P and Module R are primarily changed by stub and Driver that are performed as a replica object until they are created.

- **Example of Component Testing**
- Suppose we have a web application containing three different modules as Login, Home, and User.
- The first module (Login) is installed in the testing environment, but the other two modules, Home and User, need to be called by the Login module that is yet to be finished.
- For testing purposes, the developer will add a segment of code that replicates the call methods of the remaining modules. And this specific segment of code is known as Stubs which is considered a top-down approach.
- If the second module (Home) and third module (User) are ready, but the Login module is yet to be developed, from where both components get their return values, the developer will add a code that replicates the Login module.
- And this specific segment of code is known as a Drivers, which is considered a bottom-up approach.

Navigation Testing

- **Navigation testing involves verifying that all links, menus, and buttons within an application work as intended.** It ensures that users are guided efficiently through their journey and are not confused or stuck while interacting with the site or app.
- Typical use cases include e-commerce platforms, blogs, social media websites, and apps where users must navigate through various pages to complete tasks such as purchasing, finding content, or interacting with features.
- This type of testing is relevant for QA engineers, UI/UX designers, product managers, and web developers, as they aim to improve user experiences and minimize errors caused by broken or confusing navigation.
- **What is the Importance of Navigation Testing?**
- Navigation testing is essential for the following reasons:

- **Improving User Experience:** Seamless navigation helps users find what they need without frustration, leading to higher satisfaction rates and fewer drop-offs. A positive user experience encourages users to return and engage further.
- **Reducing Errors:** Faulty or broken navigation links can lead to dead ends, preventing users from completing desired actions. This can cause frustration, leading to a loss of trust and potential business. Navigation testing ensures all navigation paths are functional.

- **Improving SEO:** Good navigation positively affects search engine optimization (SEO). Search engines rank websites based on how easily they can crawl and index pages. Proper navigation helps

search engine bots navigate and index content better, improving a site's visibility in search engine results.

- **What are the Different Methods of Navigation Testing?**

- Below are the various methods of conducting navigation testing:
- **Tree Testing:** This method examines the structure of your website or app to see how users find their way. Participants are given specific tasks to complete, and by analyzing how they navigate through the tree "branches" (menus and links), you can identify any confusing or unclear labels or pathways.
- **Alpha Testing:** Alpha testing is like stepping into the early stages of a digital world, where users navigate through a prototype. By assigning tasks and observing how participants interact with the prototype's navigation, you can identify potential issues such as unclear elements or confusing usability. This approach helps catch navigation problems early on before the product is fully developed.
- **Usability Testing:** In usability testing, you let real users take the wheel and explore your website or app. By observing how they perform tasks and navigate through the interface, you can spot areas where they might struggle or get frustrated. This method provides valuable insights into how users naturally interact with your navigation, helping you improve the overall experience.

- **A/B Testing:** A/B testing puts two different versions of your navigation to the test. It's like offering users two different paths to the same destination and observing which one works better. By comparing different designs, labels, or menu placements, you can determine which navigation option leads to the best user experience, improving efficiency and satisfaction.
- **The Different Types of Navigation Tests**

- **Categorization Testing:** Categorization testing focuses on how well content is organized within the website or application. It assesses whether users can easily find information based on how categories and subcategories are structured. Effective categorization ensures users can quickly navigate to their desired sections, enhancing the overall experience.

- **Information Architecture Testing:** Information architecture (IA) testing evaluates the organization and labeling of information on a site. It examines whether the structure makes sense to users and whether they can intuitively navigate the content they seek. A well-designed IA reduces cognitive load and helps users find relevant information without confusion.
- **Layout and Functionality Testing:** This type of testing assesses the visual arrangement and functionality of navigation elements, such as menus, buttons, and links. It ensures that users can interact effectively with navigation items and respond appropriately to user actions. Proper layout and functionality contribute to a seamless user experience, making it easier for users to explore the site.

- **Step-by-Step: How To Conduct Navigation Testing**
- **Step 1: Know Your Goals**
- Clearly understand what you want to achieve with navigation testing. Are you aiming for smoother menu navigation, quicker button responses, or an overall user-friendly journey? Define your objectives.
- **Step 2: Pick Your Testing Style**
- Choose a testing method that suits your goals. It could be as simple as asking users to follow a tree structure or navigating a prototype. Select the method that aligns with what you want to learn.
- **Step 3: Craft Realistic script**
- Develop scripts that mimic real-life user tasks. Whether finding information or completing actions, ensure your script covers various navigation aspects. Keep it practical and relatable.
- **Step 4: Assemble Your Test Team**
- Recruit a diverse group of participants resembling your actual user base. Include both seasoned users and those new to your application. A mix provides valuable perspectives.
- **Step 5: Set Up Your Testing Space**
- Arrange your testing environment, whether a room or a digital platform. Ensure your scenarios are ready, tools are set, and everything is prepared for a smooth testing experience.

- **Step 6: Conduct the Tests**
- Begin the testing session by explaining the process to participants. Encourage them to verbalize their thoughts as they navigate through the site. This will provide valuable insights into their experience and help identify any confusion or frustration.
- **Step 7: Observe and Record**
- Observe participants' interactions with the navigation elements as they perform their tasks. Take detailed notes on their behaviors. Recording sessions (with consent) can also provide visual evidence for further analysis.
- **Step 8: Analyze Results**
- Once testing is complete, analyze the collected data. Look for patterns and common issues that participants faced during navigation. Pay attention to quantitative data (example: time taken for tasks) and qualitative feedback (example: user comments).
- **Step 9: Identify Improvement Areas & Implement Changes**
- Based on your analysis, identify specific areas where navigation can be enhanced. This may involve reworking navigation labels and restructuring menus based on user feedback. Ensure that all updates align with best practices in usability and user experience design.
- **Step 10: Retest and Iterate**

- After implementing changes, conduct follow-up navigation tests to evaluate the effectiveness of the updates. Continuous testing and iteration are essential for maintaining an optimal user experience. Regularly gather user feedback to inform future improvements.

- **ExampleExample : E-Commerce Navigation Testing**
- **Objective:** Improve the checkout process.
- **Task:** Find a product, add it to the cart, and complete the purchase.
- **Method:** Usability testing with diverse participants.
- **Observation:** Users struggle to locate the cart icon.
- **Improvement:** Make the cart icon more prominent.
- **Result:** Conduct follow-up testing to validate changes.

- **Expanded Example: Social Media Navigation Testing**
- **Objective:** Improve user engagement by optimizing navigation paths.
- **Task:** Post a photo and view notifications.
- **Method:** A/B testing of navigation menus.
- **Scenario A:** Users access "Post" via a dedicated menu button.
- **Scenario B:** Users access "Post" via a sub-menu.
- **Result:** Scenario A led to 25% faster task completion, indicating clearer navigation improves user satisfaction.

- **How to perform this test in Test Automation**
- **Step 1: Identify Essential User Tasks**
- Pinpoint the crucial tasks users should complete using your application's navigation. Whether finding information, purchasing, or connecting with others, these tasks are the heart of your navigation testing.
- **Step 2: Define Navigation Paths**
- Once you know the essential tasks, map out the paths users would take to achieve them. Visualize the journey users should ideally follow to complete each task. This clarity is the foundation for effective navigation testing.
- **Step 3: Crafting Clear Test Cases**
- Write test cases for each identified path. Could you keep it simple, clear, and concise? Your test cases should cover all scenarios users encounter, ensuring comprehensive testing.
- **Step 4: Choose the Right Test Automation Framework**
- Various **automation testing** frameworks like Testsigma, Selenium, Appium, and Cypress exist. Choose one that aligns with your application type and programming language. Think of it as picking the right tool for the job.
- **Step 5: Record Your Test Cases**

- Once your test cases are ready, record them using the chosen automation framework. This process turns your **manual testing** cases into automated scripts that can be executed consistently.
- **Step 6: Execute Tests Automatically**
- With your recorded test cases, you can execute them automatically using the chosen framework. This automated testing ensures a speedy and efficient evaluation of your application's navigation.

Configuration Testing

- Configuration Testing is the type of Software Testing that verifies the performance of the system under development against various combinations of software and hardware to find out the best configuration under which the system can work without any flaws or issues while matching its functional requirements.
- **What is Configuration Testing?**
- Configuration Testing is the process of testing the system under each configuration of the supported software and hardware. Here, the different configurations of hardware and software mean the multiple operating system versions, various browsers, various supported drivers, distinct memory sizes, different hard drive types, various types of CPU, etc.

- **Configuration Testing Example**
- Generally, Desktop applications will be of 2 tier or 3 tier, here we will consider a 3 tier Desktop application which is developed using Asp.Net and consists of Client, Business Logic Server and Database Server where each component supports below-mentioned platforms.

- **Client Platform** – Windows XP, Windows 7 OS, windows 8 OS , etc
- **Server Platform** – Windows Server 2008 R2,Windows Server 2008 R2, Windows Server 2012R2
- **Database** –SQL Server 2008, SQL Server 2008R2, SQL Server 2012, etc.

- A tester has to test the Combination of Client, Server and Database with combinations of the above-mentioned platforms and database versions to ensure that the application is functioning properly and does not fail.

- Configuration testing is not only restricted to Software but also applicable for Hardware which is why it is also referred as Hardware configuration testing, where we test different hardware devices like Printers, Scanners, Webcams, etc. that support the application under test.

- **Objectives of Configuration Testing:**
  1. **Adaptability to Different Configurations:** Check that the program's basic features work consistently and dependably in all configurations. Testing the behavior of the program with different setups and settings is part of this process.
  2. **Evaluation of Stability:** Examine the software's stability under various configurations. Find and fix any configuration-specific problems that might be causing crashes, unstable systems or strange behavior.
  3. **Testing the User Experience:** Assess the value and consistency of the user experience across various setups. Make that the graphical user interface (GUI) of the software adjusts to various screen sizes, resolutions and display settings.
  4. **Security Throughout Configurations:** To make sure that sensitive data is kept safe, test the software's security features in various setups. Determine and fix any vulnerabilities that might be configuration-specific.

5. **Compatibility of Networks:** Examine the software's behavior with various network setups. Evaluate its compatibility with various network types, speeds and latency.
6. **Data Compatibility:** Check if the programme can manage a range of data configurations, such as those from diverse sources, databases and file formats. Verify the consistency and integrity of the data across various setups.

- **Types of configuration Testing**
- there are two types of configuration testing as mentioned below
- **Software Configuration Testing**
- **Hardware Configuration Testing**

- **Software Configuration Testing**
- **Software configuration testing is testing the Application under test with multiple OS, different software updates, etc.** Software Configuration testing is very time consuming as it takes time to install and uninstall different software that is used for the testing.

- One of the approaches that is followed to test the software configuration is to test on Virtual Machines. Virtual Machine is an Environment that is installed on software and acts like a Physical Hardware and users will have the same feel as of a Physical Machine. Virtual Machines simulates real-time configurations.
- Instead of Installing and uninstalling the software in multiple physical machines which is time-consuming, it's always better to install the application/software in the virtual machine and continue testing. This process can be performed by having multiple virtual machines, which simplifies the job of a tester
- **Software configuration testing can typically begin when**
- Configurability requirements to be tested are specified
- Test Environment is ready
- Testing Team is well trained in configuration testing
- Build released is unit and Integration test passed
- Typical Test Strategy that is followed to test the software configuration test is to run the functional test suite across multiple software configurations to verify if the application under test is working as desired without any flaws or errors.

- Another strategy is to ensure the system is working fine by manually failing the test cases and verifying for efficiency.
- **Example:**
- There is a Banking Application, which has to be tested for its compatibility across multiple browsers when the application is hosted in an environment where all the prerequisites are present; it might pass the unit and Integration Testing in the test lab.
- But if the same application is installed in a client place and the machines are missing some software's updates or the versions on which the application is dependent directly or indirectly there is a chance that the application might fail. To avoid this kind of situation, it's always suggested to fail the tests manually by removing some of the configurability requirements and then proceed with the testing.
- **Hardware Configuration Testing**
- Hardware configuration testing is generally performed in labs, where we find physical machines with different hardware attached to them.
- Whenever a build is released, the software has to be installed in all the physical machines where the hardware is attached, and the test suite has to be run on each machine to ensure that the application is working fine.

- Also, while performing hardware configuration tests, we specify the type of hardware to be tested, and there are a lot of computer hardware and peripherals which make it quite impossible to run all of them. So it becomes the duty of the tester to analyze what hardware is mostly used by users and try to make the testing based on the prioritization.
- **Sample Test Cases**
- Consider a Banking Scenario to test for hardware compatibility. A Banking Application that is connected to a Note Counting Machine has to be tested with different models like Rolex, Strob, Maxsell, StoK, etc.
- Let's take some sample test cases to test the Note Counting Machine
- Verifying the connection of application with Rolex model when the prerequisites are NOT installed
- Verifying the connection of application with Rolex model when the prerequisites are installed
- Verify if the system is counting the notes correctly
- Verify if the system is counting the notes incorrectly
- Verifying the tampered notes
- Verifying the response times
- Verifying if the fake notes are detected and so on

Server-Side Issues in Configuration Testing

- **Server-side issues** are problems that arise in the backend infrastructure, where the application or service operates. The **server side is responsible for data processing, storage, API integration, and providing services to the client.**
- **Causes of Server-Side Issues:**
- Variations in server operating systems or middleware configurations.
- Incompatibility with database versions or APIs.
- Inadequate handling of network latency or user load.
- Misconfigured server settings or outdated libraries.

Common Server-Side Issues and Examples:

1. **Operating System Compatibility**:
    - **Issue**: A web application that runs smoothly on a Linux-based server encounters errors on a Windows Server due to unsupported scripts or platform-dependent configurations.
    - **Example**: A Java-based application uses file paths with Linux-specific syntax, which causes issues when deployed on a Windows Server.
    - **Solution**:
        - Test the application across various server operating systems (Linux, Windows, macOS, etc.).
        - Use environment-independent paths and configurations in the code.
2. **Database Version Mismatch**:
    - **Issue**: SQL queries that work in MySQL fail when executed in PostgreSQL due to differences in syntax or reserved keywords.

- **Example**: An e-commerce site uses LIMIT and OFFSET for pagination, which behaves differently in Oracle compared to PostgreSQL.
- **Solution**:
  - Test queries with all supported database systems (e.g., MySQL, Oracle, PostgreSQL).
  - Use abstraction layers like ORMs (e.g., Hibernate) to handle database differences.

3. **Middleware Configuration**:
   - **Issue**: A web server (e.g., Nginx, Apache) fails to serve a resource because of missing or misconfigured modules.
   - **Example**: An Apache server lacks the mod_rewrite module required for URL rewriting in an SEO-friendly website.
   - **Solution**:
     - Test middleware configurations for all required modules and ensure correct setup.
     - Create configuration guidelines for deployment teams.
4. **Load Handling and Scalability**:
   - **Issue**: An online ticket booking system crashes during peak traffic hours due to insufficient server resources or improper load balancing.
   - **Example**: The system cannot handle 10,000 concurrent users, causing delays and dropped transactions.
   - **Solution**:
     - Conduct load and stress testing to simulate high traffic.
     - Use techniques like auto-scaling, caching, and load balancers to handle increased loads.
5. **Network Latency and API Performance**:

- **Issue**: A real-time multiplayer game lags because the server API does not optimize responses for high-latency networks.
- **Example**: The server takes too long to respond to game state updates, leading to poor user experience.
- **Solution**:
  - Optimize server-side API calls for minimal latency.
  - Use techniques like edge servers or Content Delivery Networks (CDNs) for faster responses.

Client-Side Issues in Configuration Testing

- **Client-side issues arise from the environment where the end-user interacts with the application.** These issues are often related to the hardware or software configuration of the client's system, including browsers, operating systems, network conditions, and hardware limitations.
- **Causes of Client-Side Issues:**
- Variations in browsers and their versions.
- Differences in operating systems and updates.
- Limitations in hardware resources (e.g., RAM, CPU, GPU).
- Network speed and stability issues.

Common Client-Side Issues and Examples:

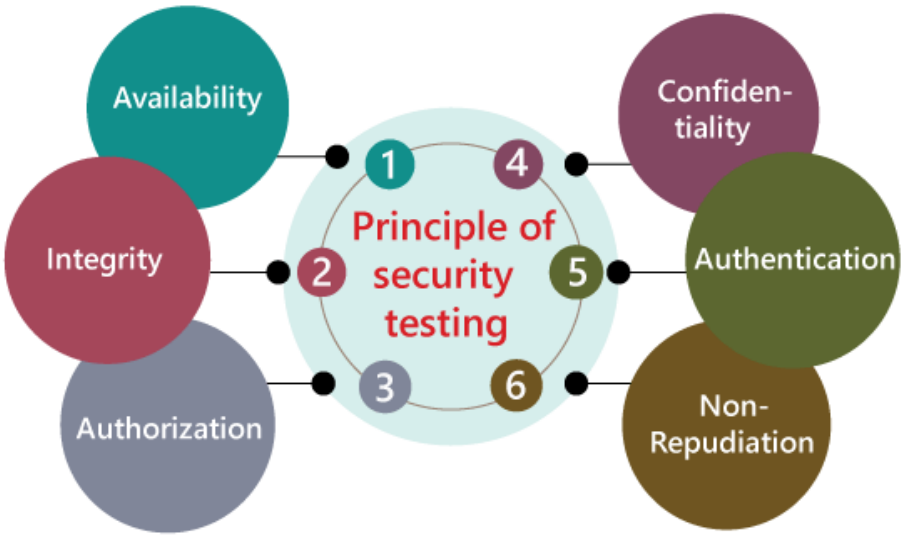
1. **Browser Compatibility**:
   - **Issue**: A web application renders correctly in Google Chrome but breaks in Safari due to CSS or JavaScript differences.

    - **Example**: Dropdown menus fail to function in older versions of Internet Explorer due to unsupported JavaScript features.
    - **Solution**:
        - Test the application on all target browsers (e.g., Chrome, Firefox, Edge, Safari) and their major versions.
        - Use browser compatibility tools like BrowserStack or Sauce Labs.
2. **Operating System Differences**:
    - **Issue**: A desktop application works on Windows 10 but fails on macOS Ventura because of API or library incompatibilities.
    - **Example**: A video editing tool crashes on macOS Ventura due to changes in the macOS file handling system.
    - **Solution**:
        - Test on all supported operating systems and ensure libraries and APIs are cross-platform compatible.
        - Use virtualization to simulate different OS environments.
3. **Hardware Limitations**:
    - **Issue**: An application requiring significant CPU or GPU resources runs slowly on low-end devices.
    - **Example**: A 3D rendering software freezes on a PC with a dual-core processor and integrated graphics.
    - **Solution**:
        - Define minimum and recommended hardware requirements.

        - Optimize performance by allowing users to adjust settings (e.g., quality vs. performance modes).
4. **Display and Resolution Issues**:
    - **Issue**: A mobile app's interface becomes unresponsive on devices with unconventional resolutions or aspect ratios.
    - **Example**: A banking app appears distorted on a foldable phone with a 21:9 aspect ratio.
    - **Solution**:
        - Test across various screen resolutions, aspect ratios, and device types (e.g., smartphones, tablets, desktops).
        - Use responsive design principles for UI development.
5. **Network Speed and Reliability**:
    - **Issue**: A streaming application buffers excessively on 3G networks or high-latency connections.
    - **Example**: A video conferencing tool struggles to maintain call quality on slow or unstable networks.
    - **Solution**:
        - Simulate different network conditions (3G, 4G, Wi-Fi, high latency) using network simulation tools.
        - Implement adaptive bitrate streaming or offline caching.

Security Testing

- **What is security testing?**
- Security testing is an integral part of software testing, which is used to discover the weaknesses, risks, or threats in the software application and also help us to stop the attack from the outsiders and ensure the security of our software applications.
- The primary objective of security testing is to find all the potential ambiguities and vulnerabilities of the application so that the software does not stop working. If we perform security testing, then it helps us to identify all the possible security threats and also help the programmer to fix those errors.
- It is a testing procedure, which is used to define that the data will be safe and also continue the working process of the software.
- **Principle of Security testing**
- Availability
- Integrity
- Authorization
- Confidentiality
- Authentication
- Non-repudiation



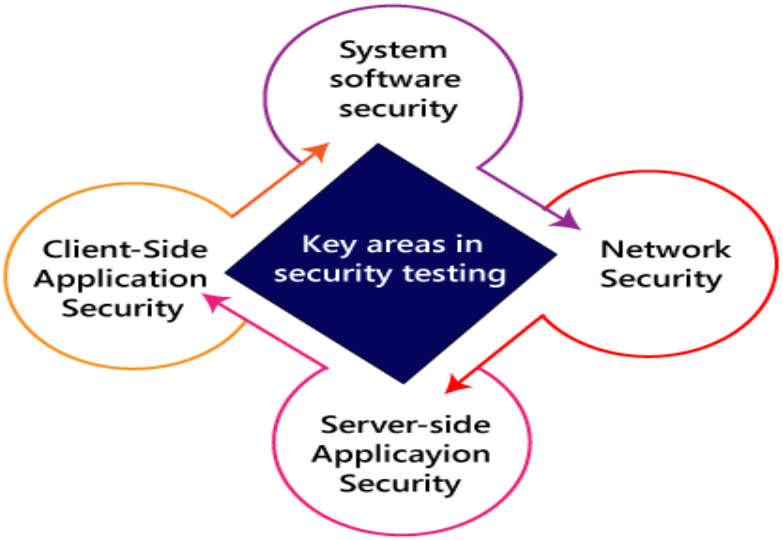
- **Availability**
- In this, the data must be retained by an official person, and they also guarantee that the data and statement services will be ready to use whenever we need it.
- ***Example***: An online food delivery service should remain operational during peak hours. If the server crashes due to a Denial-of-Service (DoS) attack, it violates availability.
- **Integrity**
- In this, we will secure those data which have been changed by the unofficial person. The primary objective of integrity is to permit the receiver to control the data that is given by the system.
- The integrity systems regularly use some of the similar fundamental approaches as confidentiality structures. Still, they generally include the data for the communication to create the source of an

algorithmic check rather than encrypting all of the communication. And also verify that correct data is conveyed from one application to another.

- ***Example***: In a stock trading app, if hackers alter stock prices, users could make decisions based on false data. Security testing prevents such manipulations.
- **Authorization**
- It is the process of defining that a client is permitted to perform an action and also receive the services. The example of authorization is Access control.
- ***Example***: In a corporate HR system, only managers should access performance reviews, not all employees. Security testing validates such access controls.
- **Confidentiality**
- It is a security process that prevents the leak of the data from the outsider's because it is the only way where we can ensure the security of our data.
- ***Example***: Encrypting credit card information on an e-commerce site ensures hackers can’t steal and misuse it.
- **Authentication**
- The authentication process comprises confirming the individuality of a person, tracing the source of a product that is necessary to allow access to the private information or the system.
- ***Example***: A social media app requiring both a password and a one-time code sent to your phone exemplifies strong authentication practices.
- **Non- repudiation**
- It is used as a reference to digital security, and it is a way of assurance that the sender of a message cannot disagree with having sent the message and that the recipient cannot repudiate having received the message.

- The non-repudiation is used to ensure that a conveyed message has been sent and received by the person who claims to have sent and received the message.
- ***Example***: In email systems, digital signatures ensure that the sender cannot deny having sent a message, and the receiver cannot deny receiving it.

- **Key Areas in Security Testing**
- While performing the security testing on the web application, we need to concentrate on the following areas to test the application:



- **System software security**
- In this, we will evaluate the vulnerabilities of the application based on different software such as **Operating system, Database system, etc.**
- **Network security**
- In this, we will check the **weakness of the network structure,** such as policies and resources.
- **Server-side application security**
- We will do server-side application security to ensure that the server encryption and its tools are sufficient to protect the software from any disturbance.
- **Client-side application security**
- In this, we will make sure that any intruders cannot operate on any browser or any tool which is used by customers.
- **Types of Security testing**
- As per Open Source Security Testing techniques, we have different types of security testing which as follows:
- Security Scanning
- Risk Assessment
- Vulnerability Scanning
- Security Auditing
- Ethical hacking
- Posture Assessment
- **Security Scanning**
- Security scanning can be done for both automation testing and manual testing. This scanning will be used to find the vulnerability or unwanted file modification in a web-based application, websites,

network, or the file system. After that, it will deliver the results which help us to decrease those threats. Security scanning is needed for those systems, which depends on the structure they use.

- **Risk Assessment**
- To moderate the risk of an application, we will go for risk assessment. In this, we will explore the security risk, which can be detected in the association. The risk can be further divided into three parts, and those are high, medium, and low. The primary purpose of the risk assessment process is to assess the vulnerabilities and control the significant threat.
- **Vulnerability Scanning**
- It is an application that is used to determine and generate a list of all the systems which contain the desktops, servers, laptops, virtual machines, printers, switches, and firewalls related to a network. The vulnerability scanning can be performed over the automated application and also identifies those software and systems which have acknowledged the security vulnerabilities.
- **Security Auditing**
- Security auditing is a structured method for evaluating the security measures of the organization. In this, we will do the inside review of the application and the control system for the security faults.
- **Ethical hacking**
- Ethical hacking is used to discover the weakness in the system and also helps the organization to fix those security loopholes before the nasty hacker exposes them. The ethical hacking will help us to increase the security position of the association because sometimes the ethical hackers use the same tricks, tools, and techniques that nasty hackers will use, but with the approval of the official person.
- The objective of ethical hacking is to enhance security and to protect the systems from malicious users' attacks.
- **Posture Assessment**

- It is a combination of ethical hacking, risk assessments, and security scanning, which helps us to display the complete security posture of an organization.
- **Advantages of Security Testing**
- Identifying vulnerabilities
- Improving system security
- Ensuring compliance
- Reducing risk
- Improving incident response
- **Disadvantages of Security Testing**
- Resource-intensive
- Complexity
- Limited testing scope
- False positives and negatives
- Time-consuming
- Difficulty in simulating real-world attacks

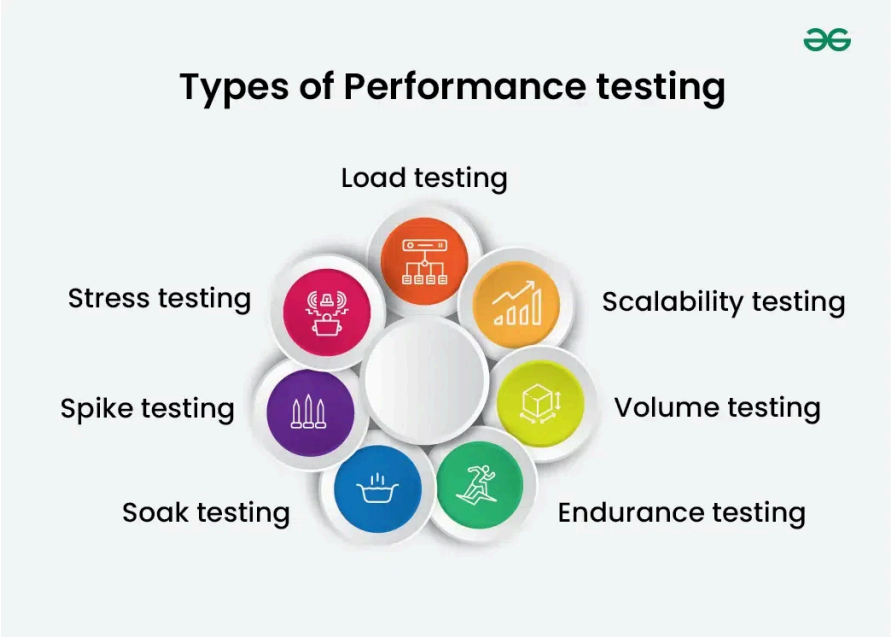
Performance Testing

What is performance testing?

Performance testing is a type of software testing that focuses on evaluating the performance and scalability of a system or application. Performance testing aims to identify bottlenecks, measure system performance under various loads and conditions, and ensure that the system can handle the expected number of users or transaction

Types of performance testing

Performance testing is also known as ***Perf Testing*** . The types of Performance testing are as follows:



1. Load testing

Load testing simulates a real-world load on the system to see how it performs under stress. It helps identify bottlenecks and determine the maximum number of users or transactions the system can handle. It checks the product's ability to perform under anticipated user loads. The objective is to identify performance congestion before the software product is launched in the market.

2. Stress testing

Stress testing is a type of load testing that tests the system's ability to handle a high load above normal usage levels. It helps identify the breaking point of the system and any potential issues that may occur under heavy load conditions. It involves testing a product under extreme workloads to see whether it handles high traffic or not. The objective is to identify the breaking point of a software product.

3. Spike testing

Spike testing is a type of load testing that tests the system's ability to handle sudden spikes in traffic. It helps identify any issues that may occur when the system is suddenly hit with a high number of requests. It tests the product's reaction to sudden large spikes in the load generated by users.

4. Soak testing

Soak testing is a type of load testing that tests the system's ability to handle a sustained load over a prolonged period. It helps identify any issues that may occur after prolonged usage of the system.

5. Endurance testing

Endurance testing is similar to soak testing, but it focuses on the long-term behavior of the system under a constant load. It is performed to ensure the software can handle the expected load over a long period.

6. Volume testing

In Volume testing , a large number of data is saved in a database and the overall software system's behavior is observed. The objective is to check the product's performance under varying database volumes.

7. Scalability testing

In Scalability testing , the software application's effectiveness is determined by scaling up to support an increase in user load. It helps in planning capacity additions to your software system.

Why use performance testing?

- The objective of performance testing is to eliminate performance congestion.
- It uncovers what needs to be improved before the product is launched in the market.
- The objective of performance testing is to **make software rapid.**
- The objective of performance testing is to **make software stable and reliable.**
- The objective of performance testing is to **evaluate the performance and scalability of a system or application under various loads and conditions.**

- It helps **measure system performance, and ensure that the system can handle the expected number of users or transactions.**
- It also **helps to ensure that the system is reliable, stable, and can handle the expected load in a production environment.**

Advantages of Performance Testing

- Identifying bottlenecks
- Improved scalability:
- Improved reliability
- Reduced risk
- Cost-effective
- Improved user experience

Disadvantages of Performance Testing

- Resource-intensive

Load Testing

- In software testing, load testing is an integral part of performance testing under non-functional testing.
- Load testing is testing where we check an application's performance by applying some load, which is either less than or equal to the desired load.
- Here, load means that N-number of users using the application simultaneously or sending the request to the server at a time.
- Load testing will help to detect the maximum operating capacity of an application and any blockages or bottlenecks.
- It governs how the software application performs while being accessed by several users at the same time.
- The load testing is mainly used to test the Client/Server's performance and applications that are web-based.
- In other words, we can say the load testing is used to find whether the organization used for comparing the application is necessary or not, and the performance of the application is maintained when it is at the maximum of its user load.
- Generally, load testing is used to signify how many concurrent users handle the application and the application's scale in terms of hardware, network capacity etc.

The Objective of Load Testing

- The load testing is used to perform the maximum quantity of software applications without important performance breakdown.
- It is used to govern the scalability of the application and allows various users to access it.
- It is used to identify the total count of users that can access the application simultaneously.
- The load testing is used to determine whether the latest infrastructure can run the software application or not and determine the sustainability of the application concerning extreme user load.

Types of Load Testing tools

Load Testing Tools [Open-Source]

- To perform the load testing, we can use the open-source load testing tools as we have various load testing tools available in the market for free of cost.
- If we have a limited budget for the particular project, we can use open-source tools. But not every time, as they may not be as advanced as the paid load testing tool.
- JMeter is the most frequently used open-source load testing tool.

Manual Load Testing

- One of the most effective load testing approaches is the manual process to perform the load testing. Still, we cannot rely on it as it does not produce repeatable outputs and assessable stress levels on an application.

- And if we perform the load testing manually, it requires a lot of workforces, which is quite expensive compared to paid tools.

Load Testing Tools [Enterprise-class]

- The paid load testing tools are used to support many protocols; therefore, it can implement various types of applications like Streaming Media, ERP/CRM, etc.
- LoadRunner is the most popular licensed load testing tool.

In house Developed Load Testing Tools

- The particular organization uses the in-house developed load testing tools approaches to build their tools to perform the load tests on their application for understanding the importance of load testing

Advantage of load testing

- **Identifying bottlenecks:** Load testing helps identify bottlenecks in the system such as slow database queries, insufficient memory, or network congestion. This helps developers optimize the system and ensure that it can handle the expected number of users or transactions.
- **Improved scalability:** By identifying the system's maximum capacity, load testing helps ensure that the system can handle an increasing number of users or transactions over time. This is particularly important for web-based systems and applications that are expected to handle a high volume of traffic.

- **Improved reliability:** Load testing helps identify any potential issues that may occur under heavy load conditions, such as increased error rates or slow response times. This helps ensure that the system is reliable and stable when it is deployed to production.
- **Reduced risk:** By identifying potential issues before deployment, load testing helps reduce the risk of system failure or poor performance in production.
- **Cost-effective:** Load testing is more cost-effective than fixing problems that occur in production. It is much cheaper to identify and fix issues during the testing phase than after deployment.
- **Improved user experience:** By identifying and addressing bottlenecks, load testing helps ensure that users have a positive experience when using the system. This can help improve customer satisfaction and loyalty.

Disadvantages of Load Testing

- **Resource-intensive:** Load testing can be resource-intensive, requiring significant hardware and software resources to simulate a large number of users or transactions. This can make load testing expensive and time-consuming.
- **Complexity:** Load testing can be complex, requiring specialized knowledge and expertise to set up and execute effectively. This can make it difficult for teams with limited resources or experience to perform load testing.

- **Limited testing scope:** Load testing is focused on the performance of the system under stress, and it may not be able to identify all types of issues or bugs. It's important to combine load testing with other types of testing such as functional testing, regression testing, and acceptance testing.
- **Inaccurate results:** If the load testing environment is not representative of the production environment or the load test scenarios do not accurately simulate real-world usage, the results of the test may not be accurate.
- **Difficulty in simulating real-world usage:** It's difficult to simulate real-world usage, and it's hard to predict how users will interact with the system. This makes it difficult to know if the system will handle the expected load.
- **Complexity in analyzing the results:** Load testing generates a large amount of data, and it can be difficult to analyze the results and determine the root cause of performance issues.

Stress Testing

- In software testing, stress testing is an important part of performance testing under non-functional testing.
- Stress Testing is testing used to check the accessibility and robustness of software beyond usual functional limits. It mainly considers critical software but it can also be used for all types of software applications.
- It is also known as *Endurance Testing, fatigue testing* or *Torture Testing*.
- The stress testing includes the testing beyond standard operational size, repeatedly to a breaking point, to get the outputs.
- It highlights the error handling and robustness under a heavy load instead of correct behavior under regular conditions.
- In other words, we can say that Stress testing is used to verify the constancy and
- dependability of the system and also make sure that the system would not crash under disaster circumstances.
- To analyze how the system works under extreme conditions, we perform stress testing outside the normal load.

The Objective of Stress Testing

- The primary purpose of executing the stress testing is to confirm that the software does not crash in lacking computational resources like disk space, memory, and network request.

- The implementation of stress testing certifies that the system fails and improves effortlessly, known as the recoverability process.
- We can use stress testing to discover hardware issues, data corruption issues.
- Stress testing will help us to identify the security weaknesses that might sneak-in throughout constant peak load.
- It helps determine the software application's data integrity throughout the extreme load, which implies that the data should be in a dependable state after a failure.

Features of Stress Testing

- Stress testing also makes sure that unpredicted failures do not cause security issues.
- It is used to analyze how the system works under rare circumstances and the system's behavior after a failure.
- Stress testing is used to check if the system has saved the data before crashing or not.
- Stress testing guarantees to display a suitable error message when the system is under stress.

Types of Stress Testing

Here are the following Types of Stress Testing

1. Server-client Stress Testing

Server-client stress testing also known as distributed stress testing is carried out across all clients from the server.

Also known as **Distributed Stress Testing** , this type involves testing the performance and robustness of a server by simulating multiple clients. The aim is to assess how well the server can handle numerous simultaneous requests from different clients.

- **Example** : Imagine a web application where thousands of users try to log in simultaneously. The server-client stress test would simulate these concurrent logins to ensure the server doesn't crash and can handle the load effectively.

2. Product Stress Testing

Product stress testing concentrates on discovering defects related to data locking and blocking, network issues, and performance congestion in a software product.

Product Stress Testing focuses on identifying issues related to data locking, network problems, and performance bottlenecks within a specific software product.

- **Example** : In a database application, product stress testing might involve simulating multiple transactions occurring simultaneously to check for issues like data locks or network slowdowns, ensuring the system can handle real-world usage without significant delays or crashes.

3. Transactional Stress Testing

Transaction stress testing is performed on one or more transactions between two or more applications. It is carried out for fine-tuning and optimizing the system.

Transactional Stress Testing is performed on transactions between two or more applications. This type of testing aims to optimize and fine-tune the system by simulating high volumes of transaction loads.

- **Example** : For an e-commerce platform, transactional stress testing could simulate thousands of transactions per minute between the payment gateway and the order management system to ensure the process remains smooth and error-free under peak loads.

4. Systematic Stress Testing

Systematic stress testing is integrated testing that is used to perform tests across multiple systems running on the same server. It is used to discover defects where one application data blocks another application.

Systematic Stress Testing involves integrated testing across multiple systems running on the same server. This approach helps identify defects where one application's data processing might interfere with another application on the same server.

- **Example** : On a shared server hosting multiple applications, systematic stress testing might reveal that a heavy load on a database application slows down a concurrent web application, highlighting the need for resource optimization or separation.

5. Analytical Stress Testing

Analytical or exploratory stress testing is performed to test the system with abnormal parameters or conditions that are unlikely to happen in a real scenario. It is carried out to find defects in unusual scenarios like a large number of users logged at the same time or a database going offline when it is accessed from a website.

Example : Analytical stress testing might involve simulating scenarios where the database goes offline while thousands of users are accessing the application, or where the application faces a sudden surge in traffic due to a viral event. This helps identify vulnerabilities that might not be apparent under normal conditions.

Advantages and disadvantages of Stress Testing

Advantages

Some of the vital benefits of performing Stress testing is as follows:

- Stress testing signifies the system's behavior after failure and makes sure that the system recovers quickly from the crashes.
- The most important advantage of executing the stress testing will make the system work in regular and irregular conditions in a suitable way.
- It determines the scalability and enhances the performance of the software.

Disadvantages

Some of the most common drawbacks of Stress testing are as follows:

- Even in open-source tools like JMeter, a load testing environment is required, which should be as close to the production environment setup as possible.
- If we are writing the Stress test script, the person should have enough scripting knowledge of the language supported by the particular tool.

- If we are using stress testing, it will require additional resources, which makes this testing a bit costlier.
- If we perform the Stress Testing manually, it becomes a tedious and complicated task to complete, and it may also not produce the expected results.

Difference between Load testing and Stress testing

Reference Link: https://www.javatpoint.com/load-testing-vs-stress-testing