Surprise Test:- 3

Student Name: Vivek Kumar
UID: 21BCS8129
Branch: BE-CSE (LEET)
Section/Group: 20BCS-809/A
Semester: 4th Sem
Date of Performance: 10/05/2022
Subject Name: Software Engineering
Subject Code: 20CSP-255

AIM: - Q1. List and explain different types of testing done during the testing phase.

There are many different types of testing. Here is a quick breakdown of the most common testing types:

- Accessibility testing
- Acceptance testing
- Black box testing
- End to end testing
- Functional testing
- Interactive testing
- Integration testing
- Load testing
- Nonfunctional testing
- Performance testing
- Regression testing
- Sanity testing
- Security testing
- Single user performance testing
- Smoke testing
- Stress testing
- Unit testing
- White-box testing
- And many more...

Many of these types of testing can be done manually — or they can be automated.

Accessibility Testing

Accessibility testing is the practice of ensuring your mobile and web apps are working and usable for users without and with disabilities such as vision impairment, hearing disabilities, and other physical or cognitive conditions.

Acceptance Testing

Acceptance testing ensures that the end-user (customers) can achieve the goals set in the business requirements, which determines whether the software is acceptable for delivery or not. It is also known as user acceptance testing (UAT).

Black Box Testing

Black box testing involves testing against a system where the code and paths are invisible.

End to End Testing

End to end testing is a technique that tests the application's workflow from beginning to end to make sure everything functions as expected.

Functional Testing

Functional testing checks an application, website, or system to ensure it's doing exactly what it's supposed to be doing.

Interactive Testing

Also known as manual testing, interactive testing enables testers to create and facilitate manual tests for those who do not use automation and collect results from external tests.

Integration Testing

Integration testing ensures that an entire, integrated system meets a set of requirements. It is performed in an integrated hardware and software environment to ensure that the entire system functions properly.

Load Testing

This type of non-functional software testing process determines how the software application behaves while being accessed by multiple users simultaneously.

Non-Functional Testing

Nonfunctional testing verifies the readiness of a system according to nonfunctional parameters (performance, accessibility, UX, etc.) which are never addressed by functional testing.

Performance Testing

Performance testing examines the speed, stability, reliability, scalability, and resource usage of a software application under a specified workload.

Regression Testing

Regression testing is performed to determine if code modifications break an application or consume resources.

Sanity Testing

Performed after bug fixes, sanity testing determines that the bugs are fixed and that no further issues are introduced to these changes.

Security Testing

Security testing unveils the vulnerabilities of the system to ensure that the software system and application are free from any threats or risks. These tests aim to find any potential flaws and weaknesses in the software system that could lead to a loss of data, revenue, or reputation per employees or outsides of a company.

Single User Performance Testing

Single user performance testing checks that the application under test performs fine according to specified threshold without any system load. This benchmark can be then used to define a realistic threshold when the system is under load.

Smoke Testing

This type of software testing validates the stability of a software application, it is performed on the initial software build to ensure that the critical functions of the program are working.

Stress Testing

Stress testing is a software testing activity that tests beyond normal operational capacity to test the results.

Unit Testing

Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own, speeding up testing strategies and reducing wasted tests.

White Box Testing

White box testing involves testing the product's underlying structure, architecture, and code to validate input-output flow and enhance design, usability, and security.

AIM: - Q2. Describe the two levels of testing? List various testing activities?

There are mainly four **Levels of Testing** in software testing:

1. **Unit Testing**: checks if software components are fulfilling functionalities or not.
2. **Integration Testing**: checks the data flow from one module to other modules.
3. **System Testing**: evaluates both functional and non-functional needs for the testing.
4. **Acceptance Testing**: checks the requirements of a specification or contract are met as per its delivery.

1. Unit testing:

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff. The main objective of unit testing is to isolate written code to test and determine if it works as intended.

Unit testing is an important step in the development process, because if done correctly, it can help detect early flaws in code which may be more difficult to find in later testing stages.

Unit testing is a component of test-driven development (TDD), a pragmatic methodology that takes a meticulous approach to building a product by means of continual testing and revision. This testing method is also the first level of software testing, which is performed before other testing methods such as integration testing. Unit tests are typically isolated to ensure a unit does not rely on any external code or functions. Testing can be done manually but is often automated.

How unit tests work

A unit test typically comprises of three stages: plan, cases and scripting and the unit test itself. In the first step, the unit test is prepared and reviewed. The next step is for the test cases and scripts to be made, then the code is tested.

Test-driven development requires that developers first write failing unit tests. Then they write code and refactor the application until the test passes. TDD typically results in an explicit and predictable code base.

Each test case is tested independently in an isolated environment, as to ensure a lack of dependencies in the code. The software developer should code criteria to verify each test case, and a testing framework can be used to report any failed tests. Developers should not make a test for every line of code, as this may take up too much time. Developers should then create tests focusing on code which could affect the behavior of the software being developed.

Unit testing involves only those characteristics that are vital to the performance of the unit under test. This encourages developers to modify the source code without immediate concerns about how such changes might affect the functioning of other units or the program as a whole. Once all of the units in a program have been found to be working in the most efficient and error-free manner possible, larger components of the program can be evaluated by means of integration testing. Unit tests should be performed frequently, and can be done manually or can be automated.

Types of unit testing

Unit tests can be performed manually or automated. Those employing a manual method may have an instinctual document made detailing each step in the process; however, automated testing is the more common method to unit tests. Automated approaches commonly use a testing framework to develop test cases. These frameworks are also set to flag and report any failed test cases while also providing a summary of test cases.

Advantages and disadvantages of unit testing

Advantages to unit testing include:

- The earlier a problem is identified; the fewer compound errors occur.

- Costs of fixing a problem early can quickly outweigh the cost of fixing it later.
- Debugging processes are made easier.
- Developers can quickly make changes to the code base.
- Developers can also re-use code, migrating it to new projects.

Disadvantages include:

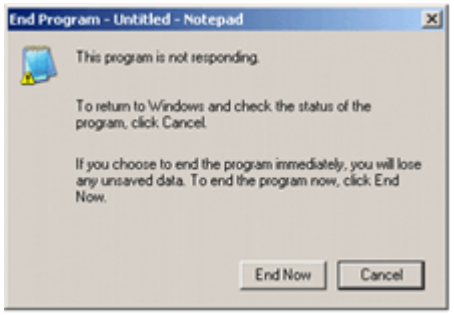
- Tests will not uncover every bug.
- Unit tests only test sets of data and its functionality—it will not catch errors in integration.
- More lines of test code may need to be written to test one line of code—creating a potential time investment.
- Unit testing may have a steep learning curve, for example, having to learn how to use specific automated software tools.

2. Stress testing:

Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions.

In Software Engineering, Stress Testing is also known as Endurance Testing. Under Stress Testing, AUT is be stressed for a short period of time to know its withstanding capacity. A most prominent use **of stress testing is to determine the limit, at which the system or software or hardware breaks**. It also checks whether the system demonstrates effective error management under extreme conditions.

The application under testing will be stressed when 5GB data is copied from the website and pasted in notepad. Notepad is under stress and gives 'Not Responded' error message.



Need for Stress Testing:

Consider the following scenarios –

- During festival time, an online shopping site may witness a spike in traffic, or when it announces a sale.
- When a blog is mentioned in a leading newspaper, it experiences a sudden surge in traffic.

It is imperative to perform Stress Testing to accommodate such abnormal traffic spikes. Failure to accommodate this sudden traffic may result in loss of revenue and repute.

Stress testing is also extremely valuable for the following reasons:

- To check whether the system works under abnormal conditions.
- Displaying appropriate error message when the system is under stress.
- System failure under extreme conditions could result in enormous revenue loss
- It is better to be prepared for extreme conditions by executing Stress Testing.

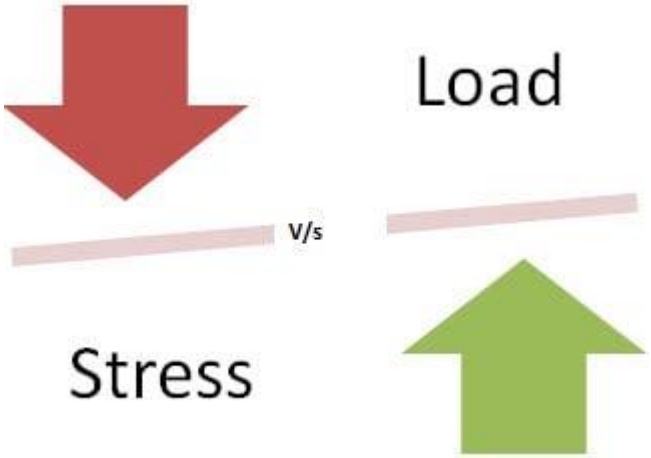
Goals of Stress Testing

The goal of stress testing is to analyze the behavior of the system after a failure. For stress testing to be successful, a system should display an appropriate error message while it is under extreme conditions.

To conduct Stress Testing, sometimes, massive data sets may be used which may get lost during Stress Testing. Testers should not lose this security-related data while doing stress testing.

The main purpose of stress testing is to make sure that the system recovers after failure which is called as **recoverability**.

Load Testing Vs Stress Testing

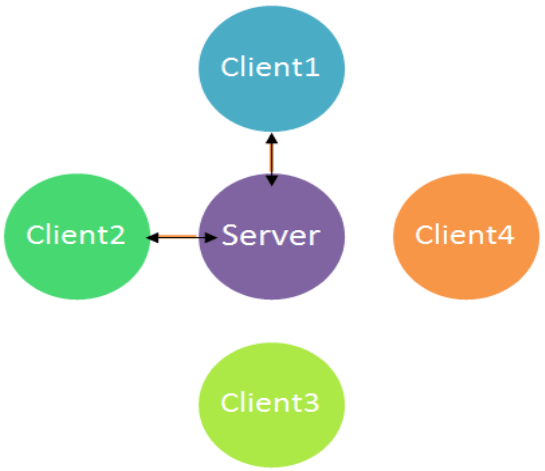


Load Testing	Stress Testing
Load Testing is to test the system behavior under normal workload conditions, and it is just testing or simulating with the actual workload	Stress testing is to test the system behavior under extreme conditions and is carried out till the system failure.
Load testing does not break the system	stress testing tries to break the system by testing with overwhelming data or resources.

Types of Stress Testing:

Following are the types of stress testing and are explained as follows:

Distributed Stress Testing:



STRESS SERVER AND CLIENT

In distributed client-server systems, testing is done across all clients from the server. The role of stress server is to distribute a set of stress tests to all stress clients and track on the status of the client. After the client contacts the server, the server adds the name of the client and starts sending data for testing.

Meanwhile, client machines send signal or heartbeat that it is connected with the server. If the server does not receive any signals from the client machine, it needs to be investigated further for debugging. From the figure, a server can connect with the 2 clients (Client1 and Client2), but it cannot send or receive a signal from Client 3 & 4.

Night run is the best option to run these stress testing scenarios. Large server farms need a more efficient method for determining which computers have had stress failures that need to be investigated.

Application Stress Testing:

This testing concentrates on finding defects related to data locking and blocking, network issues and performance bottlenecks in an application.

Transactional Stress Testing:

It does stress testing on one or more transactions between two or more applications. It is used for fine-tuning & optimizing the system.

Systemic Stress Testing:

This is integrated stress testing which can be tested across multiple systems running on the same server. It is used to find defects where one application data blocks another application.

Exploratory Stress Testing:

This is one of the types of stress testing which is used to test the system with unusual parameters or conditions that are unlikely to occur in a real scenario. It is used to find defects in unexpected scenarios like

1. A large number of users logged at the same time
2. If a virus scanner started in all machines simultaneously
3. If Database has gone offline when it is accessed from a website,
4. When a large volume of data is inserted to the database simultaneously

How to do Stress Testing?

Stress Testing process can be done in 5 major steps:

1. Planning the Stress Test. Here you gather the system data, analyze the system, define the stress test goals
2. Create Automation Scripts: In this phase, you create the Stress testing automation scripts, generate the test data for the stress scenarios.
3. Script Execution: In this stage, you run the Stress testing automation scripts and store the stress results.
4. Results Analysis: In this stage, you analyze the Stress Test results and identify bottlenecks.
5. Tweaking and Optimization: In this stage, you fine-tune the system, change configurations, optimize the code with goal meet the desired benchmark.

Lastly, you again run the entire cycle to determine that the tweaks have produced the desired results. For example, it's not unusual to have to 3 to 4 cycles of the Stress Testing process to achieve the performance goals

Tools recommended for Stress Testing:

LoadRunner

LoadRunner from HP is a widely-used Load Testing tool. Load Test Results shaped by LoadRunner are considered as a benchmark.

JMeter

JMeter is an Open-Source testing tool. It is a pure Java application for stress and Performance Testing. JMeter is intended to cover types of tests like load, functional, stress, etc. It needs JDK 5 or higher to function.

Stress Tester

This tool provides extensive analysis of the web application performance, provides results in graphical format, and it is extremely easy to use. No high-level scripting is required and gives a good return on investment.

Neo load

This is a popular tool available in the market to test the web and Mobile applications. This tool can simulate thousands of users in order to evaluate the application performance under load and analyze the response times. It also supports Cloud-integrated – performance, load and stress testing. It is easy to use, cost-effective and provides good scalability.

Metrics for Stress Testing

Metrics help in evaluating a System's performance and generally studied at the end of Stress Test. Commonly used metrics are –

Measuring Scalability & Performance

- Pages per Second: Measures how many pages have been requested / Second
- Throughput: Basic Metric – Response data size/Second
- Rounds: Number of times test scenarios have been planned Versus Number of times a client has executed

Application Response

- Hit time: Average time to retrieve an image or a page
- Time to the first byte: Time is taken to return the first byte of data or information
- Page Time: Time is taken to retrieve all the information in a page

Failures

- Failed Connections: Number of failed connections refused by the client (Weak Signal)
- Failed Rounds: Number of rounds it gets failed
- Failed Hits: Number of failed attempts done by the system (Broken links or unseen images)

AIM: - Q3. Give some design principles for maintainability.

Software Maintenance must be performed in order to:

· Correct faults.

· Improve the design.

· Implement enhancements.

· Interface with other systems.

· Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.

· Migrate legacy software.

· Retire software.

Categories of Software Maintenance –

Maintenance can be divided into the following:

1. **Corrective maintenance:**

Corrective maintenance of a software product may be essential either to rectify some bugs observed while the system is in use, or to enhance the performance of the system.

2. **Adaptive maintenance:**

This includes modifications and updating when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware and software.

3. **Perfective maintenance:**

A software product needs maintenance to support the new features that the users want or to change different types of functionalities of the system according to the customer demands.

4. **Preventive maintenance:**

This type of maintenance includes modifications and updating to prevent future problems of the software. It goals to attend problems, which are not significant at this moment but may cause serious issues in future.

Software maintenance is widely accepted part of SDLC now a days. It stands for all the modifications and updating done after the delivery of software product. There are number of reasons, why modifications are required, some of them are briefly mentioned below:

· **Market Conditions** - Policies, which changes over the time, such as taxation and newly introduced constraints like, how to maintain bookkeeping, may trigger need for modification.

· **Client Requirements** - Over the time, customer may ask for new features or functions in the software.

· **Host Modifications** - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.

· **Organization Changes** - If there is any business level change at client end, such as reduction of organization strength, acquiring another company, organization venturing into new business, need to modify in the original software may arise.

Types of maintenance

In a software lifetime, type of maintenance may vary based on its nature. It may be just a routine maintenance tasks as some bug discovered by some user or it may be a large event in itself based on maintenance size or nature. Following are some types of maintenance based on their characteristics:

· **Corrective Maintenance** - This includes modifications and updating done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.

· **Adaptive Maintenance** - This includes modifications and updating applied to keep the software product up-to date and tuned to the ever-changing world of technology and business environment.

· **Perfective Maintenance** - This includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.

· **Preventive Maintenance** - This includes modifications and updating to prevent future problems of the software. It aims to attend problems, which are not significant at this moment but may cause serious issues in future.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			

4.			