Surprise Test 3

Student Name: Sahil Kaundal UID: 21BCS8197

Branch: CSE (Lateral Entry) Section/Group:20BCS-807\_B Semester: 4th Date of Performance: 10/05/2022

Subject Name: SE Subject Code: 20CSP-254

Q1.

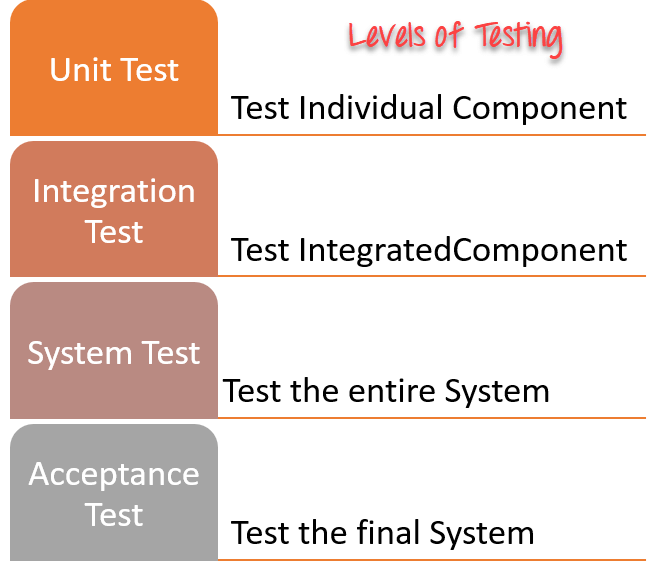
**Describe the two levels of testing? List various testing activities?**

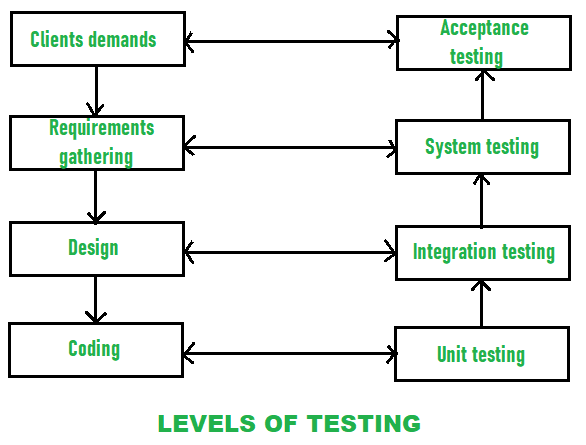
Answer:

[Software Testing](https://www.geeksforgeeks.org/software-testing-basics/) is an activity performed to identify errors so that errors can be removed to obtain a product with greater quality. To assure and maintain the quality of software and to represents the ultimate review of specification, design, and coding, Software testing is required.

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](https://www.geeksforgeeks.org/unit-testing-software-testing/) :  
   In this type of testing, errors are detected individually from every component or unit by individually testing the components or units of software to ensure that if they are fit for use by the developers. It is the smallest testable part of the software.

Unit testing is the first level of software testing, which is used to test if software modules are satisfying the given requirement or not.

The first level of testing involves analyzing each unit or an individual component of the software application.

Unit testing is also the first level of [functional testing](https://www.javatpoint.com/functional-testing). The primary purpose of executing unit testing is to validate unit components with their performance.

A unit component is an individual function or regulation of the application, or we can say that it is the smallest testable part of the software. The reason of performing the unit testing is to test the correctness of inaccessible code.

Unit testing will help the test engineer and developers in order to understand the base of code that makes them able to change defect causing code quickly. The developers implement the unit.

1. [Integration Testing](https://www.geeksforgeeks.org/software-engineering-integration-testing/) :  
   In this testing, two or more modules which are unit tested are integrated to test i.e., technique interacting components and are then verified if these integrated modules work as per the expectation or not and interface errors are also detected.

The second level of software testing is the integration testing. The integration testing process comes after unit testing.

It is mainly used to test the data flow from one module or component to other modules.

In integration testing, the test engineer tests the units or separate components or modules of the software in a group.

The primary purpose of executing the integration testing is to identify the defects at the interaction between integrated components or units.

When each component or module works separately, we need to check the data flow between the dependent modules, and this process is known as integration testing.

We only go for the integration testing when the functional testing has been completed successfully on each application module.

In simple words, we can say that integration testing aims to evaluate the accuracy of communication among all the modules.

1. [System Testing](https://www.geeksforgeeks.org/system-testing/) :  
   In system testing, complete and integrated Software’s are tested i.e. all the system elements forming the system is tested as a whole to meet the requirements of the system.
2. [Acceptance Testing](https://www.geeksforgeeks.org/acceptance-testing-software-testing/) :  
   It is a kind of testing conducted to ensure whether the requirement of the users is fulfilled prior to its delivery and the software works correctly in the user’s working environment.

Q2.

**Enlist three drawback of software testing.**

Answer:

1. Most testing types are time-consuming due to executing tests continuously. Thus, the failing tests should repeatedly run until fixing all the issues.
2. High initial expenses are required to align with the software testing process.
3. Lack of resources, skills, and understanding might affect the outcomes of software testing initiatives.
4. The continuous threat of competition in the market as a part of digital evolution.
5. Limited access to the code and features that might affect accuracy.
6. Lack of experienced software and QA testers who are aware of testing techniques.
7. The software testing team requires many members.
8. Increases the cost of the software and the budget.
9. Enhances the scope and increases the duration of the software development life cycle (SDLC).
10. Identifying the exact testing techniques and types (eg: [manual or automation testing](https://accelatest.com/manual-testing-vs-automation-testing-which-is-best-for-your-project/), [black box testing or white box testing](https://accelatest.com/black-and-white-box-testing-ultimate-guide/), etc.) is required depending on the actual requirements.
11. Many test management systems are expensive or clunky.

Q3.

**List and explain different types of testing done during the testing phase.**

Answer:

Involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. Unit testing involves the use of debugging technology and testing techniques at an application component level and is typically the responsibility of the developers, not the QA staff.

**Integration**

As the system is integrated, it is tested by the system developer for specification compliance. He is concerned with testing the system as it is integrated from its components. Integration testing is normally the most expensive activity in the systems integration process.

**It focuses on:**

• Interface testing where the interactions between sub-systems and components are tested

• Property testing where system properties such as reliability, performance and usability are tested

**System**

Testing the system as a whole to validate that it meets its specification and the objectives of its users. The testing of a complete system prior to delivery. The purpose of system testing is to identify defects that will only surface when a complete system is assembled. That is, defects that cannot be attributed to individual components or the interaction between two components. System testing includes testing of performance, security, configuration sensitivity, startup and recovery from failure modes. Involves test cases designed to validate that an application and its supporting hardware/software components are properly processing business data and transactions. System testing requires the use of regression testing techniques to validate that business functions are meeting defined requirements.

**Black Box**

This is testing without knowledge of the internal workings of the item being tested. For example, when black box testing is applied to software engineering, the tester would only know the "legal" inputs and what the expected outputs should be, but not how the program actually arrives at those outputs. It is because of this that black box testing can be considered testing with respect to the specifications, no other knowledge of the program is necessary. For this reason, the tester and the programmer can be independent of one another, avoiding programmer bias toward his own work.

**White Box**

Also known as glass box, structural, clear box and open box testing. White Box is a software testing technique whereby explicit knowledge of the internal workings of the item being tested are used to select the test data. Unlike Black Box testing, white box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do. He or she can then see if the program diverges from its intended goal.

White box testing does not account for errors caused by omission, and all visible code must also be readable.

There are two approaches to software testing; Manual Testing and Automated Testing.

Manual Testing: As the name implies, this involves manually clicking through an application or interacting with software and APIs with the goal of discovering bugs (defects) in software. This process is very expensive because it requires a human to setup the testing environment and execute all the tests themselves. It is also not very efficient because the tester is human, and the whole process is prone to omissions and errors.

Automated Testing: In software engineering, automated testing is more widely accepted as the correct way of running tests as manual tests are simply not sufficient. In automated testing, a machine is made to execute test scripts written in advance to detect possible defects in the software. These tests can vary a lot in complexity, from validating the results returned from a function to making sure that a sequence of complex actions on the front-end produce expected results.

The different types of tests

1. Unit testing: This is a very low-level type of testing that involves the testing of individual functions, classes, components or modules utilized by the software. When compared to the other forms of testing, Unit testing is relatively cheap to automate.
2. Integration testing: Integration testing is used to verify that the various units of your software work well together. This is relatively expensive because it requires multiple units of the software to be running at the time the testing is being done.
3. Functional testing: Functional testing focuses on ensuring that a given application meets its business requirements. A typical functional test will only check the output of an action without checking the intermediate stages of the system when performing that action.
4. End-to-end testing: The end-to-end testing is used to replicate a user behaviour with software in a complete application environment. it does this by verifying that the user flow functions as expected. These kinds of tests are particularly useful, but very hard to maintain when they are automated.
5. Acceptance testing: Acceptance testing is a more formal testing method executed to verify that a system meets its business requirements. This most times involves getting the end-users (customers) to interact with the system, to ensure that it meets their requirements before releasing the product into the market.
6. Performance testing: Performance testing is used to check the behaviour of a software system under significant load. These are non-functional test used to understand the reliability, stability and availability of the system. These tests are expensive to implement due to their complex nature.
7. Smoke testing: Smoke tests are very basic tests that ensure that the minimum functional requirement of a software system is met. They are quick to execute and the goal of this is to give you assurance that the major features of your system are working as expected.