Rafi MD Ashifujjman, 71330708.

*As testing proceeds more and more bugs are discovered.

How to know when to stop testing?

Test Coverage: When all planned test cases have been executed and the required coverage metrics (such as code coverage, functional coverage, etc.) have been met.

Bug Rate: If the rate at which new bugs are being discovered has significantly decreased and stabilizes over time, it indicates fewer defects remaining in the software.

Deadlines: When the project reaches the scheduled release date, and further delays are not feasible.

Risk Assessment: When the remaining risks are deemed acceptable by stakeholders, considering the context and impact of potential issues.

Quality Goals: When the software meets the predefined quality standards and acceptance criteria.

* Give examples of the types of bugs detected during:

Unit testing involves testing individual components or functions of a software application in isolation. The goal is to ensure that each unit performs as expected. We should do unit testing 40% of total testing.

Syntax Errors: Mistakes in the code's syntax. Like A missing semicolon or an incorrectly named variable.

Logic Errors: Incorrect implementation of logic within a function. A function designed to check if a number is even might return true for odd numbers due to an incorrect condition like if (number % 2 == 1) instead of if (number % 2 == 0).

Boundary Value Errors: Errors that occur at the edge of input ranges.

Null Pointer Exceptions: When the code tries to use an object reference that is null.

Integration testing focuses on verifying the interactions between integrated units or components. It ensures that combined components work together as intended.

Interface Mismatches: Incompatible interfaces between modules. Like, A module expecting a

date in DD/MM/YYYY format receives it in MM/DD/YYYY format from another module.

Data Format Errors: Issues arising from data not being correctly formatted when passed between modules.

Communication Errors: Failures in the communication protocols between integrated components.

Dependency Issues: Problems caused by incorrect handling of dependencies between modules. A module requires a certain version of a library, but an older version is used, causing compatibility issues.

System testing evaluates the complete and integrated software application to ensure it meets the specified requirements. This testing type assesses the system's overall behavior, performance, security, and compliance with the specified requirements.

Functional Errors: The software does not perform a function as specified. A shopping cart does not update correctly when items are added or removed.

Performance Bugs: Issues related to system performance, such as slow response times.

Security Vulnerabilities: Flaws that could be exploited for unauthorized access.

Usability Issues: Problems that affect the user experience and interface.

* What is functionality, performance, and acceptance testing?
These are system testing,

Functionality testing is a type of testing that verifies that each function of the software application operates in conformance with the requirement specification. This involves testing the user interface, APIs, databases, security, client/server applications, and functionality of the software under test. The goal is to ensure that the software behaves as expected and all functional requirements are met.

Verifying login, search, and payment processes on an e-commerce site are examples of functionality testing.

Performance testing is a type of testing that determines how a system performs in terms of responsiveness and stability under a particular workload. It determines whether a system or

subsystem fulfills its nonfunctional requirements. This is based on the SRS document functionality. It helps to identify the Response time, Throughput, Usability, speed, scalability, and reliability of the software. For example, checking how a streaming service handles thousands of simultaneous user's main target of performance testing.

Acceptance testing is a type of testing performed to determine whether the software system has met the requirement specifications. It is typically the final phase of testing before the software is released to the market or the client. This testing is usually done by the end-users or clients to ensure that the system meets their expectations and is ready for use.

* Why do we need several types of testing techniques?

Using multiple testing techniques is essential in software development to ensure the comprehensive evaluation and reliability of the software. Multiple testing techniques are necessary to ensure that software is robust, reliable, and meets all quality standards. Each type of testing provides unique insights and uncovers different issues like, Performance testing assesses how the software performs under various loads and conditions, Usability testing ensures the software is user-friendly and meets user expectations, Security testing identifies vulnerabilities to protect against threats and breaches. This testing approach ensures that the final product is functional, performs well, is secure, user-friendly, and meets all stakeholder and regulatory requirements.

* What is "test case"?

A test case is a set of conditions or variables under which a tester determines whether a system or software satisfies the requirements and functions correctly. Each test case is designed to verify a particular aspect of the software, ensuring that it behaves as expected.

* Why do we need carefully, not randomly, selected test cases?

Carefully selected test cases can focus on critical software aspects, optimizing resources, addressing high-risk areas, and provide clear documentation. In carefully selected test cases a test plan is developed first, and it consists of some feature or aim, like features to be tested or not tested, what would the test strategy, test suspension criteria, stopping criteria, effort

everything has been planned before. It covers comprehensive testing, ensuring efficiency, risk mitigation, quality assurance, stakeholder confidence, and regulatory compliance. For example, in an e-commerce platform, test cases would specifically target key functions like payment processing, user authentication, and load handling during peak times.

Randomly selected test cases can lead to incomplete, inefficient, and inconsistent testing, potentially missing critical defects and reducing overall software quality. If test cases are selected randomly,

- Many test cases would not contribute to the significance of the test suite.
- Would only detect errors that are already detected by other test cases in the suite.