**SOFTWARE TESTING ASSIGNMENT**

**MODULE-1(Fundamental)**

**Que.1 What is SDLC?**

SDLC stands for **S**oftware **D**evelopment **L**ife **C**ycle.

It is a process which is used for software development by software developer teams to develop, test and deploy the software at high quality.

It is essentially a series of phases that provide a model for the development and lifecycle management of an application or piece of software.

These phases are as listed below:  
 1) **Requirements Gathering and Analysis**: Gathering and defining the project requirements based on inputs from product owner or stakeholder.

2) **Design**: Creating a blueprint for the software that outlines its architecture, database schema, user interface, and other technical specifications.

3) **Implementation**: Writing code according to the design specifications. This phase includes actual programming, unit testing, and integration testing.

4) **Testing**: Verifying that the software functions correctly and meets the specified requirements. This includes various testing activities such as system testing, acceptance testing, and performance testing.

5) **Deployment**: Releasing the software for production use after successful testing. Deployment may involve installation, configuration, and migration of data.

6) **Maintenance**: Providing ongoing support and maintenance to the software. This phase includes fixing defects, enhancing features, and ensuring the software continues to meet user needs.

## Que.2 What is software testing?

Software testing is a process in which a software or its components are evaluated with the intent to find whether it satisfies the given requirements or not.

Testing is a very crucial part of the SDLC because it helps the software developers to find the actual result of the system and if the actual result does not match the expected result the system code needs to be fixed and by doing so the quality of the software or system can be increased.

**Que.3 What is agile methodology?**

Agile is a software development method which is used for developing a software in such condition where flexibility and speed is required.

Agile methodologies are designed to be more adaptive to change and to deliver working software in shorter cycles, typically known as sprints or iterations.

Key principles of Agile methodologies include:

1. **Iterative Development**: The project is divided into small sprints, usually 2-4 weeks long, where cross-functional teams work on specific sets of features.
2. **Collaboration and Communication**: There is a constant teamwork and communication between team members and stakeholders throughout the project.
3. **Adaptability to Change**: Agile methods are designed to accommodate changes in requirements and priorities, allowing teams to adjust their plans and deliver value quickly.
4. **Continuous Improvement**: Teams reflect on their processes and performance at the end of each iteration, identifying areas for improvement in terms of both the product and the development process.

**Que.4 What is SRS?**

SRS stands for **S**oftware **R**equirement **S**pecification.

It is a document that specifies all the functional and non-functional requirements of the system which is to be developed.

This document serves as a reference for all the software developers, software testers, project manager and client.

Key components provided in an SRS document are as listed below:

1. **Introduction**: It provides software developer team the overview of the software, its purpose, scope, and definitions of terms used.
2. **Functional Requirements**: It provides detailed descriptions of the functions and capabilities of system. This includes user interactions, data inputs and outputs, processing functions, etc.
3. **Non-Functional Requirements**: Quality attributes such as performance, reliability, security, usability, etc., that the software must adhere to.
4. **External Interface Requirements**: Specifications for interfaces with other systems, hardware, software components, and users.
5. **Constraints**: Any limitations or restrictions on the system's design or implementation, such as regulatory requirements, compatibility issues, or technological constraints.
6. **Appendices**: Supporting information such as diagrams, mockups, glossary of terms, etc., that aid in understanding and implementing the requirements.

**Que.5 What is oops?**

**Que.6 Write basic concepts of oops.**

**Que.7 What is object?**

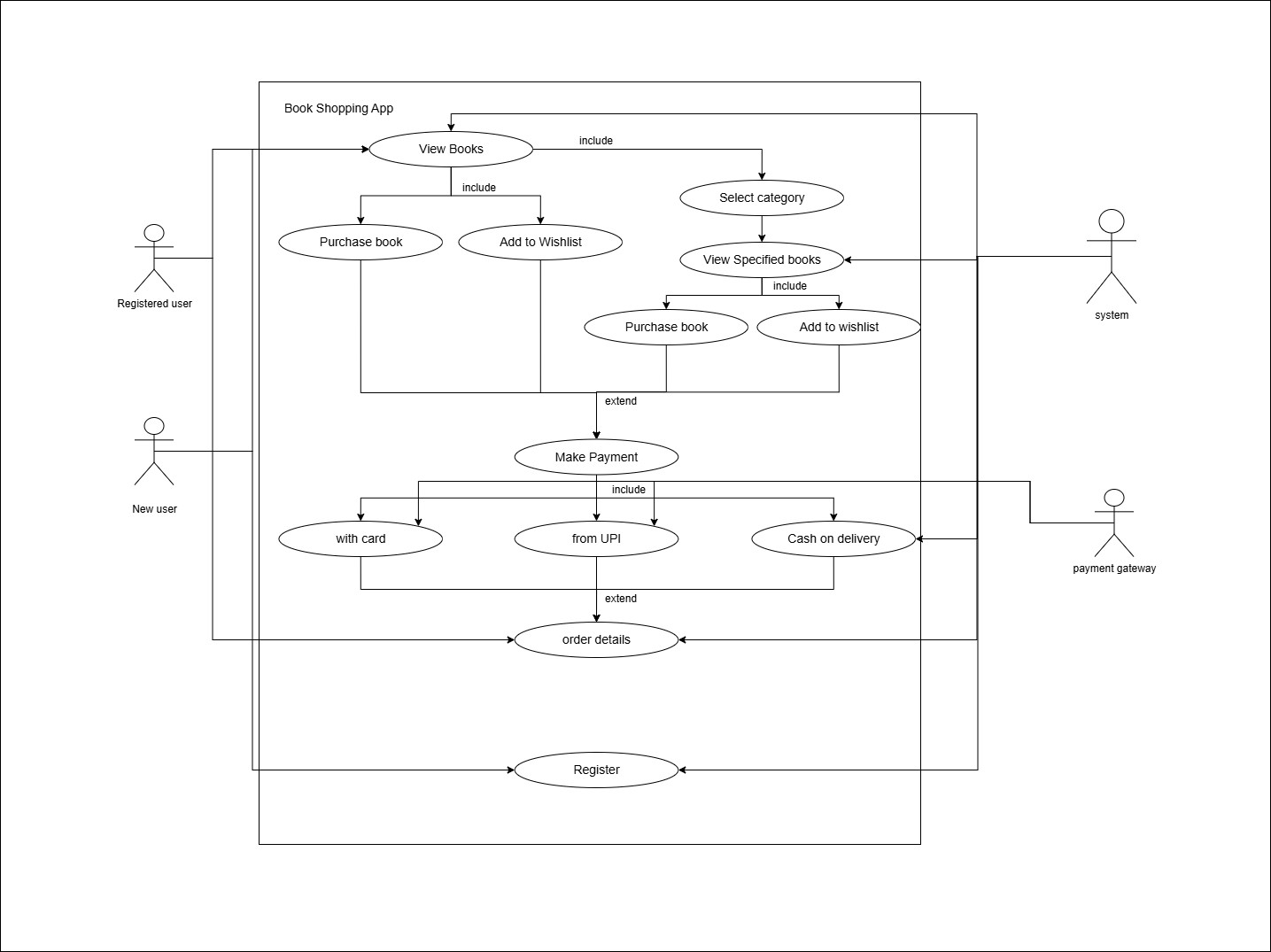
**Que.8 What is class?**

**Que. 9 What is encapsulation?**

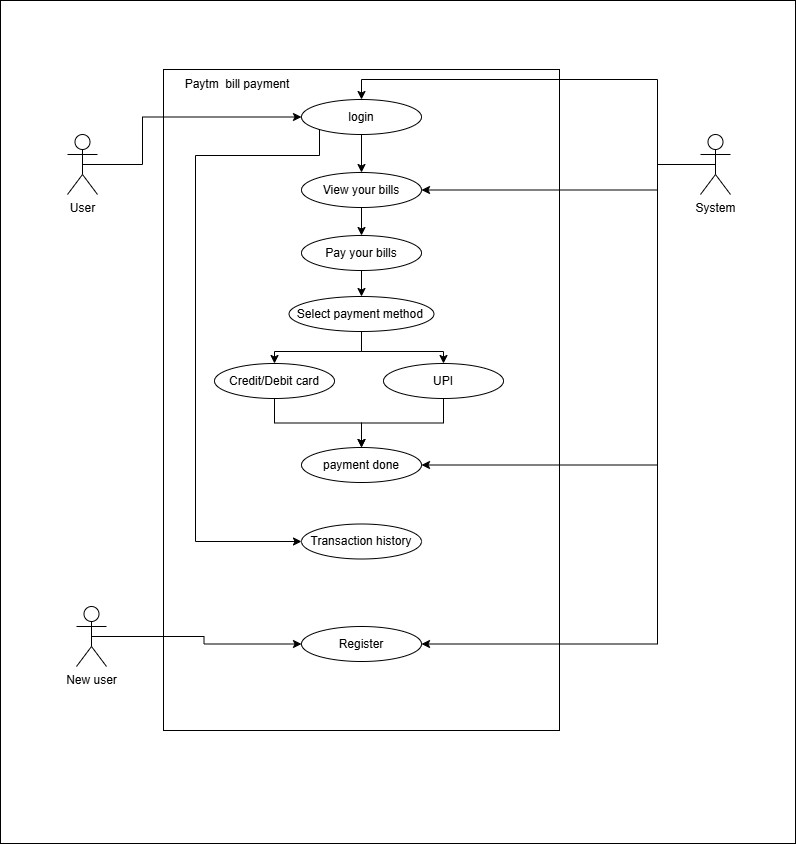
**Que.10 What is inheritance?**

**Que.11 What is polymorphism?**

**Que.12 Draw Usecase on Online book shopping.**



**Que.13 Draw Usecase on online bill payment system (paytm).**



**Que.14 Write SDLC phases with basic introduction.**

The SDLC phases are as listed below:

1) **Requirements Gathering and Analysis**: This is the first phase of SDLC.

In this phase all the project requirements are defined and gathered based on the input of the stakeholder or project owner.

Then the requirement can be divided into functional and non-functional requirement after doing analysis of the project.

2) **Design**: This phase is about creating a blueprint for the software that outlines its architecture, database schema, user interface, and other technical specifications.

3) **Implementation**: In this phase actual coding starts.

Writing code according to the design specifications. This phase includes actual programming, unit testing, and integration testing.

4) **Testing**: In this phase the tests are performed on the system and its components. The phase includes static and dynamic testing. The goal of this phase is to see actual and expected result of the system. If the results do not match then the bug is reported and needs to be fixed.

5) **Deployment**: Releasing the software for production use after successful testing. Deployment may involve installation, configuration, and migration of data.

6) **Maintenance**: Providing ongoing support and maintenance to the software. This phase includes fixing defects, enhancing features, and ensuring the software continues to meet user needs.

**Que.15 Explain Phases of the waterfall model.**

The Waterfall Model is a traditional software development methodology that proceeds sequentially through a series of phases. Each phase is performed after the previous phase has been completed. Here are the typical phases of the Waterfall Model:

**1) Requirements Gathering and Analysis:**

- In this phase, the project team gathers and documents all requirements for the software system.

- The goal is to create a comprehensive and unambiguous specification document that serves as a foundation for the entire project.

**2) System Design:**

- The design phase involves creating detailed specifications for the software system's components, interfaces, and data.

- It also includes planning for deployment, maintenance, and future upgrades.

**3) Implementation (Coding):**

- This is the phase where actual coding of the software system begins.

- Developers translate the design into executable code using programming languages and development tools.

- This phase is typically the longest as it involves writing, testing, and debugging the code.

**4) Testing:**

- Once the code is complete, it undergoes testing to identify and fix defects or bugs.

- In this phase the tests are performed on the system and its components. The phase includes static and dynamic testing. The goal of this phase is to see actual and expected result of the system. If the results do not match then the bug is reported and needs to be fixed.

- The goal is to ensure the software functions correctly and meets the user's expectations.

**5) Deployment (Installation):**

- After successful testing, the software is deployed to the production environment.

- Installation and configuration processes may be involved in this phase to set up the software for users.

- Training and documentation for users and support staff may also be provided.

**6) Maintenance:**

- Once the software is deployed, it enters the maintenance phase.

- During maintenance, any issues that arise in the operational software are addressed, including fixing bugs, making enhancements, and adapting the software to changes in the environment.

- Maintenance ensures the software continues to meet the needs of its users over time.

**Key Characteristics:**

- Sequential Flow

- Document-driven

- Rigid and Structured

**Que.16 Write phases of spiral model.**

The spiral method is a combination of iterative and the waterfall methodology. It is characterized by cycles of iterative development, where each cycle is divided into four phases.

These phases are:

1. **Planning**:

In this phase, project goals, alternatives, limitation, and risks are identified. This involves defining objectives for the current cycle, identifying alternative solutions, and evaluating these solutions against limitations like cost, schedule, and technical feasibility. Risk assessment is also a critical part of this phase.

1. **Risk Analysis**:

This phase involves evaluating identified risks and determining alternate solutions for risk mitigation. Risk analysis includes identifying potential risks, assessing their likelihood and impact, and planning for risk mitigation strategies.

1. **Engineering**:

In this phase, the actual development and testing of the software occur. The project is divided into smaller segments or spirals, and the team works on one segment at a time. Each segment includes requirements analysis, design, coding, testing, and review.

1. **Evaluation**:

The project undergoes a comprehensive review and evaluation of its progress and results. This phase involves reviewing project goals, plans, and risks. Based on the evaluation, decisions are made regarding whether to proceed to the next phase of the spiral or refine the current phase.

**Que.17 Write agile manifesto principles.**

**Que.18 Explain working methodology of agile model and also write pros and cons.**

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.

Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks.

Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is displayed to the customer and important stakeholders.

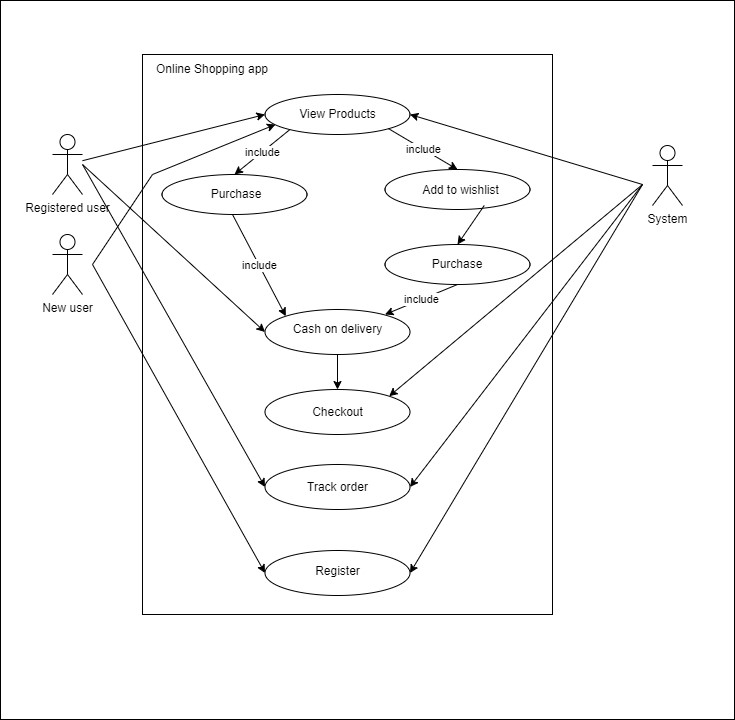
Pros are as listed below:

* Flexibility and Adaptability
* Improved Quality
* Enhanced Transparency
* Higher Stakeholder Engagement
* Predictable Costs and Schedule

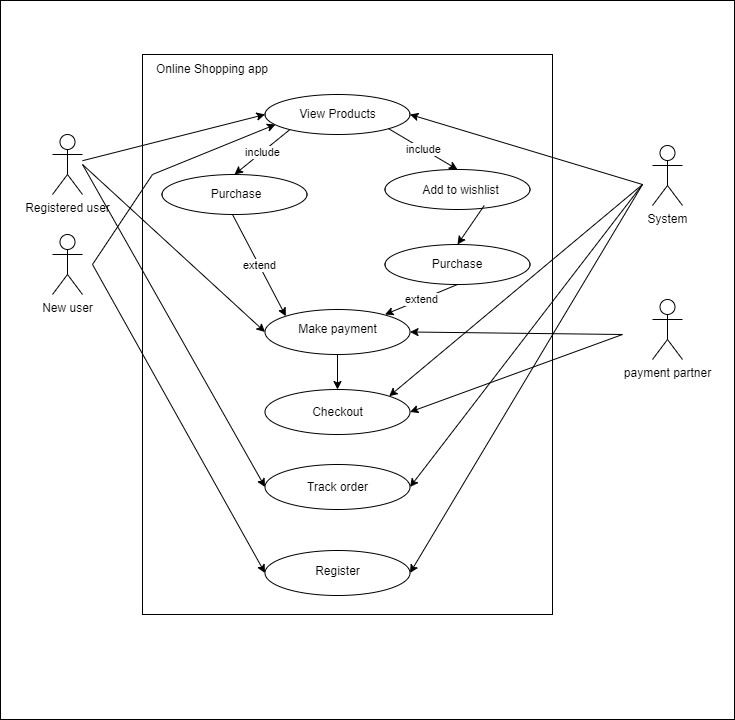
Cons are as listed below:

* Emphasis on Collaboration
* Lack of Predictability in Scope
* Not suitable for handling complex dependencies
* Depends heavily on customer interaction

**Que.19 Draw usecase on Online shopping product using COD.**



**Que.20 Draw usecase on Online shopping product using payment gateway.**



**MODULE-2 (Manual Testing)**

**Que.1 What is exploratory testing?**

**Que.2 What is traceability matrix?**

A traceability matrix is a structured document used primarily in software development and project management to ensure that requirements are aligned with the final product and that every requirement has been met during the testing phase.

To protect against changes, you should be able to trace back from every system component to the original requirement that caused its presence.

**Que.3 What is Boundary value testing?**

**Que.4 What is Equivalence partitioning testing?**

**Que.5 What is integration testing?**

Integration testing is a crucial phase in software development. In this testing the individual software modules are combined and tested as a group hence called integration testing.

Integration tests are performed to test the coexistence and communication of the internal modules within the system.

The purpose of this level of testing is to expose faults in the interaction between integrated units. test drivers and test stubs are used to assist in Integration Testing.

There are 2 levels of Integration Testing

* Component Integration Testing
* System Integration Testing

**Que.6 What determines the level of risk?**

**Que.7 What is alpha testing?**

alpha testing refers to the initial phase of software testing conducted before the product is released to a larger audience or the public.

This testing phase is typically carried out by the internal development team or a designated quality assurance team within the organization.

understanding alpha testing involves grasping its role as the initial phase of testing where internal teams rigorously test software for functionality and usability issues.

It's a critical step in ensuring that the software meets quality standards before it reaches a wider audience during beta testing and eventual release.

**Que.8 What is beta testing?**

Beta testing is the phase of software testing that occurs after alpha testing and before the official release of the software to the general public.

During beta testing, the software is made available to a selected group of external users who are not part of the development team. These users are often referred to as beta testers.

**Que.9 What is component(unit) testing?**

Component(unit) testing is a fundamental aspect of software testing that focuses on testing individual components or units of a software application.

Each unit is typically a small piece of code, such as a function, method, or class. Unit testing is conducted by developers to ensure that each unit behaves as expected and functions correctly in isolation before integrating it into the larger system.

**Que.10 What is functional system testing?**

Functional testing refers to a type of software testing that verifies that each function or feature of the software application operates in accordance with the requirements specified in its functional specifications.

This type of testing primarily focuses on what the software does (its functionalities) rather than how it achieves those functionalities (implementation details).

**Types of Functional Testing**:

**Unit Testing**: Testing individual units or components of the software in isolation to validate their functionalities.

**Integration Testing**: Testing interactions between different components or modules of the software to ensure they work together as expected.

**System Testing**: Testing the entire system as a whole to verify that all integrated components function correctly together.

**Acceptance Testing**: Also known as user acceptance testing (UAT), this involves testing the software with end users to validate whether it meets their expectations and business requirements.

**Que.11 What is non-functional system testing?**

Non-functional testing refers to a type of software testing that evaluates the performance, reliability, usability, and other aspects of a system beyond its basic functionality.

Unlike functional testing, which focuses on what the software does, non-functional testing focuses on how well the software performs under different conditions and constraints.

**Types of Non-Functional Testing**:

**Performance Testing**: Evaluates how the software performs in terms of speed, responsiveness, and stability under various workload conditions.

**Load Testing**: Checks the software's behavior when subjected to expected and peak loads to ensure it can handle the expected number of users and transactions.

**Stress Testing**: Tests the software's robustness by subjecting it to extreme conditions (e.g., high load, resource limitations) to identify its breaking points.

**Usability Testing**: Assesses how user-friendly and intuitive the software is for its intended users.

**Que.12 What is GUI Testing?**

**Que.13 What is Adhoc testing?**

**Que.14 What is load testing?**

**Que.15 What is stress Testing?**

**Que.16 What is white box testing and list the types of white box testing?**

**Que.17 What is black box testing? What are the different black box testing techniques?**

**Que.18 Mention what are the categories of defects?**

Defects in software development can be categorized based on various criteria. Here are some common categories of defects:

1. **Functional Defects**:

**Incorrect Calculation**: Errors in mathematical calculations or logic that lead to incorrect results.

**Functional Failure**: Features or functionalities not performing as specified in the requirements.

1. **Performance Defects**:

**Response Time**: Software taking too long to respond to user actions or requests.

**Throughput**: Issues with handling a large number of concurrent users or transactions.

**Resource Utilization**: Memory leaks, excessive CPU usage, or inefficient use of system resources.

1. **Compatibility Defects**:

**Hardware Compatibility**: Software not working properly on certain hardware configurations.

**Software Compatibility**: Issues with interoperability between different software versions, operating systems, browsers, or devices.

1. **Usability Defects**:

**User Interface Issues**: Poor layout, confusing navigation, or inconsistent design elements.

**Accessibility**: Software not meeting accessibility standards, making it difficult for users with disabilities to use.

1. **Security Defects**:

**Vulnerabilities**: Weaknesses in the software that could be exploited by attackers.

**Data Breaches**: Unauthorized access to sensitive data due to security flaws.

**Que.19** **Mention what bigbang testing is?**

Big Bang Testing is a way to test everything all at once, like a big explosion of tests!

Instead of testing parts of the software separately, you test everything together as a whole. This means testing all the features and functions at once.

It gives you quick feedback because you can see how everything interacts right away. If something goes wrong, you know exactly where to look.

Big Bang Testing is useful for small projects or when you're starting out and want to see how everything works together. However, for bigger projects, it might be better to test smaller parts first to catch issues early.

**Que.20** **What is the purpose of exit criteria?**

The purpose of exit criteria in any project is to set clear guidelines and standards that must be met before moving to the next phase or declaring the project complete.

In other word we can say that the purpose of exit criteria is to define when to STOP testing in the STLC.

**Que.21** **When should “regression testing” be performed?**

**Que.22** **What is the 7 key principles? explain in detail?**

**Testing shows presence of Defects:**

Testing only can show if there is any defect or error is present but can not show that there is no defect or error in the system.

Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

**Exhaustive testing is not possible:**

The truth is that you can't test everything.

What you need to do is assess risk and plan your tests around these risks so you can be sure you're testing the key functions.

That is, we must priorities our testing effort using a Risk Based Approach.

**Early testing:**

Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.

It's much easier and less expensive to fix bugs in the early stages of testing.

**Defect clustering:**

Defects are not evenly spread in a system or a software but are clustered in small modules which give the idea of defect clustering.

Understanding this can help in your testing because if you find one defect in a particular area, you'll likely find more in that same module.

**Pesticide paradox:**

This principle centers around the theory that if you repeatedly use a particular pesticide on your crops, the insects you're trying to kill or repel will eventually become immune to the pesticide and it will no longer be effective.

Likewise, if you continuously run the same tests, eventually they'll fail to find new defects, even though they'll probably confirm the software is working.

Consequently, you must continue to review your tests as well as add to your scenarios or modify them to help prevent this pesticide paradox. For example, maybe you could use a variety of testing techniques, methods, and approaches simultaneously.

**Absence of Errors Fallacy:**

If your software is 99% error-free but it doesn't follow your user's requirements, it's still not usable. That's why it's critical to run tests that pertain to the requirements of the system. Software testing isn't just about finding bugs, it's about ensuring that the software meets the user's needs and requirements.

**Testing is context dependent:**

Software testing is all about the context, which means that no one strategy will fit every scenario. The types of testing and the methods you use totally depend on the context of the systems or the software.

**Que.23 Difference between QA v/s QC v/s Tester.**

|  |  |  |
| --- | --- | --- |
| Quality assurance | Quality control | Tester |
| Process oriented activities. | Product oriented activities. | Product oriented activities. |
| It is a subset of Software Test Life Cycle (STLC). | QC can be considered as the subset of Quality Assurance. | Testing is the subset of Quality Control. |
| Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. | Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements. | Activities which ensure the identification of bugs/error/defects in the Software. |
| Preventive activities. | It is a corrective process. | It is a preventive process. |

**Que.24 Difference between Smoke and Sanity?**

**Que.25 Difference between verification and Validation.**

|  |  |
| --- | --- |
| Verification | validation |
| The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase. | The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements. |
| It includes checking the documents, designs, codes and programs. | It includes validating and testing the actual product. |
| Verification is a static testing. | Validation is a dynamic testing. |
| Hence does not need to execute code to perform tests. | Need to execute the code to perform tests. |
| It checks whether the software confirms to specification or not. | It checks whether the software meets the requirement and expectation of the customer or not. |

**Que.26 Explain types of Performance testing.**

**Que.27 What is Error, Defect, Bug and failure?**

**Error:**

When a system does not work as expected due to buggy code then it is called an error.

when the Development team or the developer fails to understand a requirement definition and hence that misunderstanding gets translated into buggy code.

This situation is referred to as an Error and is mainly a term coined by the developers.

**Defect:**

Defect is a situation where the actual and expected result of a system or a component does not match or are not in sync.

**Bug:**

A bug refers to defects which means that the software product or the application is not working as per the adhered requirements set. When we have any type of logical error, it causes our code to break, which results in a bug.

**Failure:**

Failure is the accumulation of several defects that ultimately lead to Software failure and results in the loss of information in critical modules thereby making the system unresponsive.

**Que.28 Difference between Priority and Severity.**

**Que.29 What is Bug Life Cycle?**

**Que.30 Explain the difference between Functional testing and Non-Functional testing.**

|  |  |
| --- | --- |
| Functional testing | Non-functional testing |
| It verifies the operations and actions of an application. | It verifies the behavior of an application. |
| Functional testing is easy to execute manually. | It is hard to perform non-functional testing manually. |
| Functional testing is based on the business requirement. | Non-functional testing is based on performance requirement. |
| It is based on the requirement of costumer. | It is based on the expectation of customer. |

Bottom of Form